附录 A
相关命令使用说明

Appendix A
Manuals for Related Commands

南开大学软件学院
钟震宇

版本：1.02
日期：2021 年 6 月 27 日

目录

1 fdisk

```
FDISK(8)                    System Administration                    FDISK(8)
NAME         top
       fdisk - manipulate disk partition table
SYNOPSIS         top
       fdisk [options] device

       fdisk -l [device...]
DESCRIPTION         top
       fdisk is a dialog-driven program for creation and manipulation of
       partition tables. It understands GPT, MBR, Sun, SGI and BSD
       partition tables.

       Block devices can be divided into one or more logical disks
       called partitions. This division is recorded in the partition
       table, usually found in sector 0 of the disk. (In the BSD world
       one talks about `disk slices' and a `disklabel'.)

       All partitioning is driven by device I/O limits (the topology) by
       default. fdisk is able to optimize the disk layout for a
       4K-sector size and use an alignment offset on modern devices for
       MBR and GPT. It is always a good idea to follow fdisk's defaults
       as the default values (e.g., first and last partition sectors)
       and partition sizes specified by the +/-<size>{M,G,...} notation
       are always aligned according to the device properties.

       CHS (Cylinder-Head-Sector) addressing is deprecated and not used
       by default. Please, do not follow old articles and
       recommendations with fdisk -S <n> -H <n> advices for SSD or
       4K-sector devices.

       Note that partx(8) provides a rich interface for scripts to print
       disk layouts, fdisk is mostly designed for humans. Backward
       compatibility in the output of fdisk is not guaranteed. The input
       (the commands) should always be backward compatible.
OPTIONS         top
       -b, --sector-size sectorsize
           Specify the sector size of the disk. Valid values are 512,
           1024, 2048, and 4096. (Recent kernels know the sector size.
           Use this option only on old kernels or to override the
```

kernel’s ideas.) Since util-linux-2.17, fdisk differentiates between logical and physical sector size. This option changes both sector sizes to sectorsize.

-B, --protect-boot
Don’t erase the beginning of the first disk sector when creating a new disk label. This feature is supported for GPT and MBR.

-c, --compatibility[=mode]
Specify the compatibility mode, 'dos' or 'nondos'. The default is non-DOS mode. For backward compatibility, it is possible to use the option without the mode argument — then the default is used. Note that the optional mode argument cannot be separated from the -c option by a space, the correct form is for example -c=dos.

-h, --help
Display a help text and exit.

-L, --color[=when]
Colorize the output. The optional argument when can be auto, never or always. If the when argument is omitted, it defaults to auto. The colors can be disabled; for the current built-in default see the --help output. See also the COLORS section.

-l, --list
List the partition tables for the specified devices and then exit.

If no devices are given, the devices mentioned in /proc/partitions (if this file exists) are used. Devices are always listed in the order in which they are specified on the command-line, or by the kernel listed in /proc/partitions.

-x, --list-details
Like --list, but provides more details.

--lock[=mode]
Use exclusive BSD lock for device or file it operates. The optional argument mode can be yes, no (or 1 and 0) or

nonblock. If the mode argument is omitted, it defaults to "yes". This option overwrites environment variable $LOCK_BLOCK_DEVICE. The default is not to use any lock at all, but it's recommended to avoid collisions with udevd or other tools.

-n, --noauto-pt
Don't automatically create a default partition table on empty device. The partition table has to be explicitly created by user (by command like 'o', 'g', etc.).

-o, --output list
Specify which output columns to print. Use --help to get a list of all supported columns.

The default list of columns may be extended if list is specified in the format +list (e.g., -o +UUID).

-s, --getsz
Print the size in 512-byte sectors of each given block device. This option is DEPRECATED in favour of blockdev(8).

-t, --type type
Enable support only for disklabels of the specified type, and disable support for all other types.

-u, --units[=unit]
When listing partition tables, show sizes in 'sectors' or in 'cylinders'. The default is to show sizes in sectors. For backward compatibility, it is possible to use the option without the unit argument — then the default is used. Note that the optional unit argument cannot be separated from the -u option by a space, the correct form is for example '*-u=*cylinders'.

-C, --cylinders number
Specify the number of cylinders of the disk. I have no idea why anybody would want to do so.

-H, --heads number
Specify the number of heads of the disk. (Not the physical

number, of course, but the number used for partition tables.) Reasonable values are 255 and 16.

-S, --sectors number
Specify the number of sectors per track of the disk. (Not the physical number, of course, but the number used for partition tables.) A reasonable value is 63.

-w, --wipe when
Wipe filesystem, RAID and partition-table signatures from the device, in order to avoid possible collisions. The argument when can be auto, never or always. When this option is not given, the default is auto, in which case signatures are wiped only when in interactive mode. In all cases detected signatures are reported by warning messages before a new partition table is created. See also wipefs(8) command.

-W, --wipe-partitions when
Wipe filesystem, RAID and partition-table signatures from a newly created partitions, in order to avoid possible collisions. The argument when can be auto, never or always. When this option is not given, the default is auto, in which case signatures are wiped only when in interactive mode and after confirmation by user. In all cases detected signatures are reported by warning messages before a new partition is created. See also wipefs(8) command.

-V, --version
Display version information and exit.

DEVICES top

The device is usually /dev/sda, /dev/sdb or so. A device name refers to the entire disk. Old systems without libata (a library used inside the Linux kernel to support ATA host controllers and devices) make a difference between IDE and SCSI disks. In such cases the device name will be /dev/hd* (IDE) or /dev/sd* (SCSI).

The partition is a device name followed by a partition number. For example, /dev/sda1 is the first partition on the first hard disk in the system. See also Linux kernel documentation (the Documentation/admin-guide/devices.txt file).

SIZES top

The "last sector" dialog accepts partition size specified by number of sectors or by +/-<size>{K,B,M,G,...} notation.

If the size is prefixed by '+' then it is interpreted as relative to the partition first sector. If the size is prefixed by '-' then it is interpreted as relative to the high limit (last available sector for the partition).

In the case the size is specified in bytes than the number may be followed by the multiplicative suffixes KiB=1024, MiB=1024*1024, and so on for GiB, TiB, PiB, EiB, ZiB and YiB. The "iB" is optional, e.g., "K" has the same meaning as "KiB".

The relative sizes are always aligned according to device I/O limits. The +/-<size>{K,B,M,G,...} notation is recommended.

For backward compatibility fdisk also accepts the suffixes KB=1000, MB=1000*1000, and so on for GB, TB, PB, EB, ZB and YB. These 10^N suffixes are deprecated.

SCRIPT FILES top

fdisk allows reading (by 'I' command) sfdisk(8) compatible script files. The script is applied to in-memory partition table, and then it is possible to modify the partition table before you write it to the device.

And vice-versa it is possible to write the current in-memory disk layout to the script file by command 'O'.

The script files are compatible between cfdisk(8), sfdisk(8), fdisk and other libfdisk applications. For more details see sfdisk(8).

DISK LABELS top

GPT (GUID Partition Table)

GPT is modern standard for the layout of the partition table. GPT uses 64-bit logical block addresses, checksums, UUIDs and names for partitions and an unlimited number of partitions (although the number of partitions is usually restricted to 128 in many partitioning tools).

Note that the first sector is still reserved for a protective MBR in the GPT specification. It prevents MBR-only

partitioning tools from mis-recognizing and overwriting GPT disks.

GPT is always a better choice than MBR, especially on modern hardware with a UEFI boot loader.

DOS-type (MBR)

A DOS-type partition table can describe an unlimited number of partitions. In sector 0 there is room for the description of 4 partitions (called `primary'). One of these may be an extended partition; this is a box holding logical partitions, with descriptors found in a linked list of sectors, each preceding the corresponding logical partitions. The four primary partitions, present or not, get numbers 1-4. Logical partitions are numbered starting from 5.

In a DOS-type partition table the starting offset and the size of each partition is stored in two ways: as an absolute number of sectors (given in 32 bits), and as a Cylinders/Heads/Sectors triple (given in 10+8+6 bits). The former is OK — with 512-byte sectors this will work up to 2 TB. The latter has two problems. First, these C/H/S fields can be filled only when the number of heads and the number of sectors per track are known. And second, even if we know what these numbers should be, the 24 bits that are available do not suffice. DOS uses C/H/S only, Windows uses both, Linux never uses C/H/S. The C/H/S addressing is deprecated and may be unsupported in some later fdisk version.

Please, read the DOS-mode section if you want DOS-compatible partitions. fdisk does not care about cylinder boundaries by default.

BSD/Sun-type

A BSD/Sun disklabel can describe 8 partitions, the third of which should be a `whole disk' partition. Do not start a partition that actually uses its first sector (like a swap partition) at cylinder 0, since that will destroy the disklabel. Note that a BSD label is usually nested within a DOS partition.

IRIX/SGI-type

An IRIX/SGI disklabel can describe 16 partitions, the eleventh of which should be an entire `volume' partition, while the ninth should be labeled `volume header'. The volume header will also cover the partition table, i.e., it starts at block zero and extends by default over five cylinders. The remaining space in the volume header may be used by header directory entries. No partitions may overlap with the volume header. Also do not change its type or make some filesystem on it, since you will lose the partition table. Use this type of label only when working with Linux on IRIX/SGI machines or IRIX/SGI disks under Linux.

A sync() and an ioctl(BLKRRPART) (rereading the partition table from disk) are performed before exiting when the partition table has been updated.

DOS MODE AND DOS 6.X WARNING top

Note that all this is deprecated. You don’ t have to care about things like geometry and cylinders on modern operating systems. If you really want DOS-compatible partitioning then you have to enable DOS mode and cylinder units by using the '-c=dos -u=cylinders' fdisk command-line options.

The DOS 6.x FORMAT command looks for some information in the first sector of the data area of the partition, and treats this information as more reliable than the information in the partition table. DOS FORMAT expects DOS FDISK to clear the first 512 bytes of the data area of a partition whenever a size change occurs. DOS FORMAT will look at this extra information even if the /U flag is given — we consider this a bug in DOS FORMAT and DOS FDISK.

The bottom line is that if you use fdisk or cfdisk to change the size of a DOS partition table entry, then you must also use dd(1) to zero the first 512 bytes of that partition before using DOS FORMAT to format the partition. For example, if you were using fdisk to make a DOS partition table entry for /dev/sda1, then (after exiting fdisk and rebooting Linux so that the partition table information is valid) you would use the command dd if=/dev/zero of=/dev/sda1 bs=512 count=1 to zero the first 512 bytes of the partition.

fdisk usually obtains the disk geometry automatically. This is not necessarily the physical disk geometry (indeed, modern disks do not really have anything like a physical geometry, certainly not something that can be described in the simplistic Cylinders/Heads/Sectors form), but it is the disk geometry that MS-DOS uses for the partition table.

Usually all goes well by default, and there are no problems if Linux is the only system on the disk. However, if the disk has to be shared with other operating systems, it is often a good idea to let an fdisk from another operating system make at least one partition. When Linux boots it looks at the partition table, and tries to deduce what (fake) geometry is required for good cooperation with other systems.

Whenever a partition table is printed out in DOS mode, a consistency check is performed on the partition table entries. This check verifies that the physical and logical start and end points are identical, and that each partition starts and ends on a cylinder boundary (except for the first partition).

Some versions of MS-DOS create a first partition which does not begin on a cylinder boundary, but on sector 2 of the first cylinder. Partitions beginning in cylinder 1 cannot begin on a cylinder boundary, but this is unlikely to cause difficulty unless you have OS/2 on your machine.

For best results, you should always use an OS-specific partition table program. For example, you should make DOS partitions with the DOS FDISK program and Linux partitions with the Linux fdisk or Linux cfdisk(8) programs.

COLORS top

Implicit coloring can be disabled by an empty file /etc/terminal-colors.d/fdisk.disable.

See terminal-colors.d(5) for more details about colorization configuration. The logical color names supported by fdisk are:

header

The header of the output tables.

help-title
The help section titles.

warn
The warning messages.

welcome
The welcome message.

ENVIRONMENT top

FDISK_DEBUG=all
enables fdisk debug output.

LIBFDISK_DEBUG=all
enables libfdisk debug output.

LIBBLKID_DEBUG=all
enables libblkid debug output.

LIBSMARTCOLS_DEBUG=all
enables libsmartcols debug output.

LIBSMARTCOLS_DEBUG_PADDING=on
use visible padding characters. Requires enabled
LIBSMARTCOLS_DEBUG.

LOCK_BLOCK_DEVICE=<mode>
use exclusive BSD lock. The mode is "1" or "0". See --lock
for more details.

AUTHORS top

Karel Zak <kzak@redhat.com>, Davidlohr Bueso <dave@gnu.org>

The original version was written by Andries E. Brouwer, A. V. Le
Blanc and others.

SEE ALSO top

cfdisk(8), mkfs(8), partx(8), sfdisk(8)

REPORTING BUGS top

For bug reports, use the issue tracker at
https://github.com/karelzak/util-linux/issues.

AVAILABILITY top

The fdisk command is part of the util-linux package which can be

downloaded from Linux Kernel Archive <https://www.kernel.org/pub/linux/utils/util-linux/>. This page is part of the util-linux (a random collection of Linux utilities) project. Information about the project can be found at ⟨https://www.kernel.org/pub/linux/utils/util-linux/⟩. If you have a bug report for this manual page, send it to util-linux@vger.kernel.org. This page was obtained from the project's upstream Git repository ⟨git://git.kernel.org/pub/scm/utils/util-linux/util-linux.git⟩ on 2021-06-20. (At that time, the date of the most recent commit that was found in the repository was 2021-06-18.) If you discover any rendering problems in this HTML version of the page, or you believe there is a better or more up-to-date source for the page, or you have corrections or improvements to the information in this COLOPHON (which is not part of the original manual page), send a mail to man-pages@man7.org

util-linux 2.37.85-637cc 2021-04-02 FDISK(8)

2 mount

```
MOUNT(8)                    System Administration                    MOUNT(8)
NAME         top
       mount - mount a filesystem
SYNOPSIS         top
       mount [-h|-V]

       mount [-l] [-t fstype]

       mount -a [-fFnrsvw] [-t fstype] [-O optlist]

       mount [-fnrsvw] [-o options] device|mountpoint

       mount [-fnrsvw] [-t fstype] [-o options] device mountpoint

       mount --bind|--rbind|--move olddir newdir

       mount

       ↪ --make-[shared|slave|private|unbindable|rshared|rslave|rprivate|runbindable]
       mountpoint
DESCRIPTION         top
       All files accessible in a Unix system are arranged in one big
       tree, the file hierarchy, rooted at /. These files can be spread
       out over several devices. The mount command serves to attach the
       filesystem found on some device to the big file tree. Conversely,
       the umount(8) command will detach it again. The filesystem is
       used to control how data is stored on the device or provided in a
       virtual way by network or other services.

       The standard form of the mount command is:

          mount -t type device dir

       This tells the kernel to attach the filesystem found on device
       (which is of type type) at the directory dir. The option -t type
       is optional. The mount command is usually able to detect a
       filesystem. The root permissions are necessary to mount a
       filesystem by default. See section "Non-superuser mounts" below
       for more details. The previous contents (if any) and owner and
```

mode of dir become invisible, and as long as this filesystem remains mounted, the pathname dir refers to the root of the filesystem on device.

If only the directory or the device is given, for example:

```
mount /dir
```

then mount looks for a mountpoint (and if not found then for a device) in the /etc/fstab file. It' s possible to use the --target or --source options to avoid ambiguous interpretation of the given argument. For example:

```
mount --target /mountpoint
```

The same filesystem may be mounted more than once, and in some cases (e.g., network filesystems) the same filesystem may be mounted on the same mountpoint multiple times. The mount command does not implement any policy to control this behavior. All behavior is controlled by the kernel and it is usually specific to the filesystem driver. The exception is --all, in this case already mounted filesystems are ignored (see --all below for more details).

Listing the mounts

The listing mode is maintained for backward compatibility only.

For more robust and customizable output use findmnt(8), especially in your scripts. Note that control characters in the mountpoint name are replaced with '?'.

The following command lists all mounted filesystems (of type type):

```
mount [-l] [-t type]
```

The option -l adds labels to this listing. See below.

Indicating the device and filesystem

Most devices are indicated by a filename (of a block special device), like /dev/sda1, but there are other possibilities. For

example, in the case of an NFS mount, device may look like knuth.cwi.nl:/dir.

The device names of disk partitions are unstable; hardware reconfiguration, and adding or removing a device can cause changes in names. This is the reason why it' s strongly recommended to use filesystem or partition identifiers like UUID or LABEL. Currently supported identifiers (tags):

LABEL=label

Human readable filesystem identifier. See also -L.

UUID=uuid

Filesystem universally unique identifier. The format of the UUID is usually a series of hex digits separated by hyphens. See also -U.

Note that mount uses UUIDs as strings. The UUIDs from the command line or from fstab(5) are not converted to internal binary representation. The string representation of the UUID should be based on lower case characters.

PARTLABEL=label

Human readable partition identifier. This identifier is independent on filesystem and does not change by mkfs or mkswap operations It' s supported for example for GUID Partition Tables (GPT).

PARTUUID=uuid

Partition universally unique identifier. This identifier is independent on filesystem and does not change by mkfs or mkswap operations It' s supported for example for GUID Partition Tables (GPT).

ID=id

Hardware block device ID as generated by udevd. This identifier is usually based on WWN (unique storage identifier) and assigned by the hardware manufacturer. See ls /dev/disk/by-id for more details, this directory and running udevd is required. This identifier is not recommended for generic use as the identifier is not strictly defined and it

depends on udev, udev rules and hardware.

The command lsblk --fs provides an overview of filesystems, LABELs and UUIDs on available block devices. The command blkid -p <device> provides details about a filesystem on the specified device.

Don’t forget that there is no guarantee that UUIDs and labels are really unique, especially if you move, share or copy the device. Use lsblk -o +UUID,PARTUUID to verify that the UUIDs are really unique in your system.

The recommended setup is to use tags (e.g. UUID=uuid) rather than /dev/disk/by-{label,uuid,id,partuuid,partlabel} udev symlinks in the /etc/fstab file. Tags are more readable, robust and portable. The mount(8) command internally uses udev symlinks, so the use of symlinks in /etc/fstab has no advantage over tags. For more details see libblkid(3).

The proc filesystem is not associated with a special device, and when mounting it, an arbitrary keyword - for example, proc - can be used instead of a device specification. (The customary choice none is less fortunate: the error message 'none already mounted' from mount can be confusing.)

The files /etc/fstab, /etc/mtab and /proc/mounts

The file /etc/fstab (see fstab(5)), may contain lines describing what devices are usually mounted where, using which options. The default location of the fstab(5) file can be overridden with the --fstab path command-line option (see below for more details).

The command

mount -a [-t type] [-O optlist]

(usually given in a bootscript) causes all filesystems mentioned in fstab (of the proper type and/or having or not having the proper options) to be mounted as indicated, except for those whose line contains the noauto keyword. Adding the -F option will make mount fork, so that the filesystems are mounted in parallel.

When mounting a filesystem mentioned in fstab or mtab, it suffices to specify on the command line only the device, or only the mount point.

The programs mount and umount(8) traditionally maintained a list of currently mounted filesystems in the file /etc/mtab. The support for regular classic /etc/mtab is completely disabled at compile time by default, because on current Linux systems it is better to make /etc/mtab a symlink to /proc/mounts instead. The regular mtab file maintained in userspace cannot reliably work with namespaces, containers and other advanced Linux features. If the regular mtab support is enabled, then it' s possible to use the file as well as the symlink.

If no arguments are given to mount, the list of mounted filesystems is printed.

If you want to override mount options from /etc/fstab, you have to use the -o option:

```
mount device**|dir -o options
```

and then the mount options from the command line will be appended to the list of options from /etc/fstab. This default behaviour can be changed using the --options-mode command-line option. The usual behavior is that the last option wins if there are conflicting ones.

The mount program does not read the /etc/fstab file if both device (or LABEL, UUID, ID, PARTUUID or PARTLABEL) and dir are specified. For example, to mount device foo at /dir:

```
mount /dev/foo /dir
```

This default behaviour can be changed by using the --options-source-force command-line option to always read configuration from fstab. For non-root users mount always reads the fstab configuration.

Non-superuser mounts

Normally, only the superuser can mount filesystems. However, when

fstab contains the user option on a line, anybody can mount the corresponding filesystem.

Thus, given a line

```
/dev/cdrom /cd iso9660 ro,user,noauto,unhide
```

any user can mount the iso9660 filesystem found on an inserted CDROM using the command:

```
mount /cd
```

Note that mount is very strict about non-root users and all paths specified on command line are verified before fstab is parsed or a helper program is executed. It' s strongly recommended to use a valid mountpoint to specify filesystem, otherwise mount may fail. For example it' s a bad idea to use NFS or CIFS source on command line.

Since util-linux 2.35, mount does not exit when user permissions are inadequate according to libmount' s internal security rules. Instead, it drops suid permissions and continues as regular non-root user. This behavior supports use-cases where root permissions are not necessary (e.g., fuse filesystems, user namespaces, etc).

For more details, see fstab(5). Only the user that mounted a filesystem can unmount it again. If any user should be able to unmount it, then use users instead of user in the fstab line. The owner option is similar to the user option, with the restriction that the user must be the owner of the special file. This may be useful e.g. for /dev/fd if a login script makes the console user owner of this device. The group option is similar, with the restriction that the user must be a member of the group of the special file.

Bind mount operation

Remount part of the file hierarchy somewhere else. The call is:

```
mount --bind olddir newdir
```

or by using this fstab entry:

```
/olddir /newdir none bind
```

After this call the same contents are accessible in two places.

It is important to understand that "bind" does not create any second-class or special node in the kernel VFS. The "bind" is just another operation to attach a filesystem. There is nowhere stored information that the filesystem has been attached by a "bind" operation. The olddir and newdir are independent and the olddir may be unmounted.

One can also remount a single file (on a single file). It's also possible to use a bind mount to create a mountpoint from a regular directory, for example:

```
mount --bind foo foo
```

The bind mount call attaches only (part of) a single filesystem, not possible submounts. The entire file hierarchy including submounts can be attached a second place by using:

```
mount --rbind olddir newdir
```

Note that the filesystem mount options maintained by the kernel will remain the same as those on the original mount point. The userspace mount options (e.g., _netdev) will not be copied by mount and it's necessary to explicitly specify the options on the mount command line.

Since util-linux 2.27 mount permits changing the mount options by passing the relevant options along with --bind. For example:

```
mount -o bind,ro foo foo
```

This feature is not supported by the Linux kernel; it is implemented in userspace by an additional mount(2) remounting system call. This solution is not atomic.

The alternative (classic) way to create a read-only bind mount is

to use the remount operation, for example:

```
mount --bind olddir newdir mount -o remount,bind,ro
olddir newdir
```

Note that a read-only bind will create a read-only mountpoint (VFS entry), but the original filesystem superblock will still be writable, meaning that the olddir will be writable, but the newdir will be read-only.

It's also possible to change nosuid, nodev, noexec, noatime, nodiratime and relatime VFS entry flags via a "remount,bind" operation. The other flags (for example filesystem-specific flags) are silently ignored. It's impossible to change mount options recursively (for example with -o rbind,ro).

Since util-linux 2.31, mount ignores the bind flag from /etc/fstab on a remount operation (if "-o remount" is specified on command line). This is necessary to fully control mount options on remount by command line. In previous versions the bind flag has been always applied and it was impossible to re-define mount options without interaction with the bind semantic. This mount behavior does not affect situations when "remount,bind" is specified in the /etc/fstab file.

The move operation

Move a mounted tree to another place (atomically). The call is:

```
mount --move olddir newdir
```

This will cause the contents which previously appeared under olddir to now be accessible under newdir. The physical location of the files is not changed. Note that olddir has to be a mountpoint.

Note also that moving a mount residing under a shared mount is invalid and unsupported. Use findmnt -o TARGET,PROPAGATION to see the current propagation flags.

Shared subtree operations

Since Linux 2.6.15 it is possible to mark a mount and its

submounts as shared, private, slave or unbindable. A shared mount provides the ability to create mirrors of that mount such that mounts and unmounts within any of the mirrors propagate to the other mirror. A slave mount receives propagation from its master, but not vice versa. A private mount carries no propagation abilities. An unbindable mount is a private mount which cannot be cloned through a bind operation. The detailed semantics are documented in Documentation/filesystems/sharedsubtree.txt file in the kernel source tree; see also mount_namespaces(7).

Supported operations are:

```
mount --make-shared mountpoint
mount --make-slave mountpoint
mount --make-private mountpoint
mount --make-unbindable mountpoint
```

The following commands allow one to recursively change the type of all the mounts under a given mountpoint.

```
mount --make-rshared mountpoint
mount --make-rslave mountpoint
mount --make-rprivate mountpoint
mount --make-runbindable mountpoint
```

mount(8) does not read fstab(5) when a --make-* operation is requested. All necessary information has to be specified on the command line.

Note that the Linux kernel does not allow changing multiple propagation flags with a single mount(2) system call, and the flags cannot be mixed with other mount options and operations.

Since util-linux 2.23 the mount command can be used to do more propagation (topology) changes by one mount(8) call and do it also together with other mount operations. This feature is EXPERIMENTAL. The propagation flags are applied by additional mount(2) system calls when the preceding mount operations were successful. Note that this use case is not atomic. It is possible to specify the propagation flags in fstab(5) as mount options (private, slave, shared, unbindable, rprivate, rslave, rshared,

runbindable).

For example:

```
mount --make-private --make-unbindable /dev/sda1 /foo
```

is the same as:

```
mount /dev/sda1 /foo
mount --make-private /foo
mount --make-unbindable /foo
```

COMMAND-LINE OPTIONS top

The full set of mount options used by an invocation of mount is determined by first extracting the mount options for the filesystem from the fstab table, then applying any options specified by the -o argument, and finally applying a -r or -w option, when present.

The mount command does not pass all command-line options to the /sbin/mount.suffix mount helpers. The interface between mount and the mount helpers is described below in the section EXTERNAL HELPERS.

Command-line options available for the mount command are:

-a, --all

Mount all filesystems (of the given types) mentioned in fstab (except for those whose line contains the noauto keyword). The filesystems are mounted following their order in fstab. The mount command compares filesystem source, target (and fs root for bind mount or btrfs) to detect already mounted filesystems. The kernel table with already mounted filesystems is cached during mount --all. This means that all duplicated fstab entries will be mounted.

The option --all is possible to use for remount operation too. In this case all filters (-t and -O) are applied to the table of already mounted filesystems.

Since version 2.35 is possible to use the command line option -o to alter mount options from fstab (see also

--options-mode).

Note that it is a bad practice to use mount -a for fstab checking. The recommended solution is findmnt --verify.

-B, --bind
Remount a subtree somewhere else (so that its contents are available in both places). See above, under Bind mounts.

-c, --no-canonicalize
Don't canonicalize paths. The mount command canonicalizes all paths (from the command line or fstab) by default. This option can be used together with the -f flag for already canonicalized absolute paths. The option is designed for mount helpers which call mount -i. It is strongly recommended to not use this command-line option for normal mount operations.

Note that mount does not pass this option to the /sbin/mount.type helpers.

-F, --fork
(Used in conjunction with -a.) Fork off a new incarnation of mount for each device. This will do the mounts on different devices or different NFS servers in parallel. This has the advantage that it is faster; also NFS timeouts proceed in parallel. A disadvantage is that the order of the mount operations is undefined. Thus, you cannot use this option if you want to mount both /usr and /usr/spool.

-f, --fake
Causes everything to be done except for the actual system call; if it's not obvious, this "fakes" mounting the filesystem. This option is useful in conjunction with the -v flag to determine what the mount command is trying to do. It can also be used to add entries for devices that were mounted earlier with the -n option. The -f option checks for an existing record in /etc/mtab and fails when the record already exists (with a regular non-fake mount, this check is done by the kernel).

-i, --internal-only
Don’ t call the /sbin/mount.filesystem helper even if it exists.

-L, --label label
Mount the partition that has the specified label.

-l, --show-labels
Add the labels in the mount output. mount must have permission to read the disk device (e.g. be set-user-ID root) for this to work. One can set such a label for ext2, ext3 or ext4 using the e2label(8) utility, or for XFS using xfs_admin(8), or for reiserfs using reiserfstune(8).

-M, --move
Move a subtree to some other place. See above, the subsection The move operation.

-n, --no-mtab
Mount without writing in /etc/mtab. This is necessary for example when /etc is on a read-only filesystem.

-N, --namespace ns
Perform the mount operation in the mount namespace specified by ns. ns is either PID of process running in that namespace or special file representing that namespace.

mount switches to the mount namespace when it reads /etc/fstab, writes /etc/mtab: (or writes to _/run/mount) and calls the mount(2) system call, otherwise it runs in the original mount namespace. This means that the target namespace does not have to contain any libraries or other requirements necessary to execute the mount(2) call.

See mount_namespaces(7) for more information.

-O, --test-opts opts
Limit the set of filesystems to which the -a option applies. In this regard it is like the -t option except that -O is useless without -a. For example, the command

mount -a -O no_netdev

mounts all filesystems except those which have the option netdev specified in the options field in the /etc/fstab file.

It is different from -t in that each option is matched exactly; a leading no at the beginning of one option does not negate the rest.

The -t and -O options are cumulative in effect; that is, the command

mount -a -t ext2 -O _netdev

mounts all ext2 filesystems with the _netdev option, not all filesystems that are either ext2 or have the _netdev option specified.

-o, --options opts
Use the specified mount options. The opts argument is a comma-separated list. For example:

mount LABEL=mydisk -o noatime,nodev,nosuid

For more details, see the FILESYSTEM-INDEPENDENT MOUNT OPTIONS and FILESYSTEM-SPECIFIC MOUNT OPTIONS sections.

--options-mode mode
Controls how to combine options from fstab/mtab with options from the command line. mode can be one of ignore, append, prepend or replace. For example, append means that options from fstab are appended to options from the command line. The default value is prepend — it means command line options are evaluated after fstab options. Note that the last option wins if there are conflicting ones.

--options-source source
Source of default options. source is a comma-separated list of fstab, mtab and disable. disable disables fstab and mtab and disables --options-source-force. The default value is fstab,mtab.

--options-source-force
    Use options from fstab/mtab even if both device and dir are specified.

-R, --rbind
    Remount a subtree and all possible submounts somewhere else (so that its contents are available in both places). See above, the subsection Bind mounts.

-r, --read-only
    Mount the filesystem read-only. A synonym is -o ro.

    Note that, depending on the filesystem type, state and kernel behavior, the system may still write to the device. For example, ext3 and ext4 will replay the journal if the filesystem is dirty. To prevent this kind of write access, you may want to mount an ext3 or ext4 filesystem with the ro,noload mount options or set the block device itself to read-only mode, see the blockdev(8) command.

-s
    Tolerate sloppy mount options rather than failing. This will ignore mount options not supported by a filesystem type. Not all filesystems support this option. Currently it’s supported by the mount.nfs mount helper only.

--source device
    If only one argument for the mount command is given, then the argument might be interpreted as the target (mountpoint) or source (device). This option allows you to explicitly define that the argument is the mount source.

--target directory
    If only one argument for the mount command is given, then the argument might be interpreted as the target (mountpoint) or source (device). This option allows you to explicitly define that the argument is the mount target.

--target-prefix directory
    Prepend the specified directory to all mount targets. This

option can be used to follow fstab, but mount operations are done in another place, for example:

mount --all --target-prefix /chroot -o X-mount.mkdir

mounts all from system fstab to /chroot, all missing mountpoint are created (due to X-mount.mkdir). See also --fstab to use an alternative fstab.

-T, --fstab path
Specifies an alternative fstab file. If path is a directory, then the files in the directory are sorted by strverscmp(3); files that start with "." or without an .fstab extension are ignored. The option can be specified more than once. This option is mostly designed for initramfs or chroot scripts where additional configuration is specified beyond standard system configuration.

Note that mount does not pass the option --fstab to the /sbin/mount.type helpers, meaning that the alternative fstab files will be invisible for the helpers. This is no problem for normal mounts, but user (non-root) mounts always require fstab to verify the user' s rights.

-t, --types fstype
The argument following the -t is used to indicate the filesystem type. The filesystem types which are currently supported depend on the running kernel. See /proc/filesystems and /lib/modules/$(uname -r)/kernel/fs for a complete list of the filesystems. The most common are ext2, ext3, ext4, xfs, btrfs, vfat, sysfs, proc, nfs and cifs.

The programs mount and umount(8) support filesystem subtypes. The subtype is defined by a '.subtype' suffix. For example 'fuse.sshfs'. It' s recommended to use subtype notation rather than add any prefix to the mount source (for example 'sshfs#example.com' is deprecated).

If no -t option is given, or if the auto type is specified, mount will try to guess the desired type. mount uses the libblkid(3) library for guessing the filesystem type; if that

does not turn up anything that looks familiar, mount will try to read the file /etc/filesystems, or, if that does not exist, /proc/filesystems. All of the filesystem types listed there will be tried, except for those that are labeled "nodev" (e.g. devpts, proc and nfs). If /etc/filesystems ends in a line with a single *, mount will read /proc/filesystems afterwards. While trying, all filesystem types will be mounted with the mount option silent.

The auto type may be useful for user-mounted floppies. Creating a file /etc/filesystems can be useful to change the probe order (e.g., to try vfat before msdos or ext3 before ext2) or if you use a kernel module autoloader.

More than one type may be specified in a comma-separated list, for the -t option as well as in an /etc/fstab entry. The list of filesystem types for the -t option can be prefixed with no to specify the filesystem types on which no action should be taken. The prefix no has no effect when specified in an /etc/fstab entry.

The prefix no can be meaningful with the -a option. For example, the command

mount -a -t nomsdos,smbfs

mounts all filesystems except those of type msdos and smbfs.

For most types all the mount program has to do is issue a simple mount(2) system call, and no detailed knowledge of the filesystem type is required. For a few types however (like nfs, nfs4, cifs, smbfs, ncpfs) an ad hoc code is necessary. The nfs, nfs4, cifs, smbfs, and ncpfs filesystems have a separate mount program. In order to make it possible to treat all types in a uniform way, mount will execute the program /sbin/mount.type (if that exists) when called with type type. Since different versions of the smbmount program have different calling conventions, /sbin/mount.smbfs may have to be a shell script that sets up the desired call.

-U, --uuid uuid

Mount the partition that has the specified uuid.

-v, --verbose
Verbose mode.

-w, --rw, --read-write
Mount the filesystem read/write. Read-write is the kernel default and the mount default is to try read-only if the previous mount syscall with read-write flags on write-protected devices of filesystems failed.

A synonym is -o rw.

Note that specifying -w on the command line forces mount to never try read-only mount on write-protected devices or already mounted read-only filesystems.

-V, --version
Display version information and exit.

-h, --help
Display help text and exit.

FILESYSTEM-INDEPENDENT MOUNT OPTIONS top

Some of these options are only useful when they appear in the /etc/fstab file.

Some of these options could be enabled or disabled by default in the system kernel. To check the current setting see the options in /proc/mounts. Note that filesystems also have per-filesystem specific default mount options (see for example tune2fs -l output for ext_N_ filesystems).

The following options apply to any filesystem that is being mounted (but not every filesystem actually honors them - e.g., the sync option today has an effect only for ext2, ext3, ext4, fat, vfat, ufs and xfs):

async
All I/O to the filesystem should be done asynchronously. (See also the sync option.)

atime
    Do not use the noatime feature, so the inode access time is controlled by kernel defaults. See also the descriptions of the relatime and strictatime mount options.

noatime
    Do not update inode access times on this filesystem (e.g. for faster access on the news spool to speed up news servers). This works for all inode types (directories too), so it implies nodiratime.

auto
    Can be mounted with the -a option.

noauto
    Can only be mounted explicitly (i.e., the -a option will not cause the filesystem to be mounted).

context=context, fscontext=context, defcontext=context, and rootcontext=context
    The context= option is useful when mounting filesystems that do not support extended attributes, such as a floppy or hard disk formatted with VFAT, or systems that are not normally running under SELinux, such as an ext3 or ext4 formatted disk from a non-SELinux workstation. You can also use context= on filesystems you do not trust, such as a floppy. It also helps in compatibility with xattr-supporting filesystems on earlier 2.4.<x> kernel versions. Even where xattrs are supported, you can save time not having to label every file by assigning the entire disk one security context.

    A commonly used option for removable media is context="system_u:object_r:removable_t.

    Two other options are fscontext= and defcontext=, both of which are mutually exclusive of the context= option. This means you can use fscontext and defcontext with each other, but neither can be used with context.

    The fscontext= option works for all filesystems, regardless of their xattr support. The fscontext option sets the

overarching filesystem label to a specific security context. This filesystem label is separate from the individual labels on the files. It represents the entire filesystem for certain kinds of permission checks, such as during mount or file creation. Individual file labels are still obtained from the xattrs on the files themselves. The context option actually sets the aggregate context that fscontext provides, in addition to supplying the same label for individual files.

You can set the default security context for unlabeled files using defcontext= option. This overrides the value set for unlabeled files in the policy and requires a filesystem that supports xattr labeling.

The rootcontext= option allows you to explicitly label the root inode of a FS being mounted before that FS or inode becomes visible to userspace. This was found to be useful for things like stateless Linux.

Note that the kernel rejects any remount request that includes the context option, even when unchanged from the current context.

Warning: the context value might contain commas, in which case the value has to be properly quoted, otherwise mount will interpret the comma as a separator between mount options. Don’ t forget that the shell strips off quotes and thus double quoting is required. For example:

```
mount -t tmpfs none /mnt -o \
'context="system_u:object_r:tmp_t:s0:c127,c456",noexec'
```

For more details, see selinux(8).

defaults

Use the default options: rw, suid, dev, exec, auto, nouser, and async.

Note that the real set of all default mount options depends on the kernel and filesystem type. See the beginning of this section for more details.

dev
    Interpret character or block special devices on the filesystem.

nodev
    Do not interpret character or block special devices on the filesystem.

diratime
    Update directory inode access times on this filesystem. This is the default. (This option is ignored when noatime is set.)

nodiratime
    Do not update directory inode access times on this filesystem. (This option is implied when noatime is set.)

dirsync
    All directory updates within the filesystem should be done synchronously. This affects the following system calls: creat(2), link(2), unlink(2), symlink(2), mkdir(2), rmdir(2), mknod(2) and rename(2).

exec
    Permit execution of binaries.

noexec
    Do not permit direct execution of any binaries on the mounted filesystem.

group
    Allow an ordinary user to mount the filesystem if one of that user' s groups matches the group of the device. This option implies the options nosuid and nodev (unless overridden by subsequent options, as in the option line group,dev,suid).

iversion
    Every time the inode is modified, the i_version field will be incremented.

noiversion

Do not increment the i_version inode field.

mand
Allow mandatory locks on this filesystem. See fcntl(2).

nomand
Do not allow mandatory locks on this filesystem.

_netdev
The filesystem resides on a device that requires network access (used to prevent the system from attempting to mount these filesystems until the network has been enabled on the system).

nofail
Do not report errors for this device if it does not exist.

relatime
Update inode access times relative to modify or change time. Access time is only updated if the previous access time was earlier than the current modify or change time. (Similar to noatime, but it doesn' t break mutt(1) or other applications that need to know if a file has been read since the last time it was modified.)

Since Linux 2.6.30, the kernel defaults to the behavior provided by this option (unless noatime was specified), and the strictatime option is required to obtain traditional semantics. In addition, since Linux 2.6.30, the file' s last access time is always updated if it is more than 1 day old.

norelatime
Do not use the relatime feature. See also the strictatime mount option.

strictatime
Allows to explicitly request full atime updates. This makes it possible for the kernel to default to relatime or noatime but still allow userspace to override it. For more details about the default system mount options see /proc/mounts.

nostrictatime
    Use the kernel's default behavior for inode access time updates.

lazytime
    Only update times (atime, mtime, ctime) on the in-memory version of the file inode.

    This mount option significantly reduces writes to the inode table for workloads that perform frequent random writes to preallocated files.

    The on-disk timestamps are updated only when:

    - the inode needs to be updated for some change unrelated to file timestamps
    - the application employs fsync(2), syncfs(2), or sync(2)
    - an undeleted inode is evicted from memory
    - more than 24 hours have passed since the inode was written to disk.

nolazytime
    Do not use the lazytime feature.

suid
    Honor set-user-ID and set-group-ID bits or file capabilities when executing programs from this filesystem.

nosuid
    Do not honor set-user-ID and set-group-ID bits or file capabilities when executing programs from this filesystem. In addition, SELinux domain transitions require permission nosuid_transition, which in turn needs also policy capability nnp_nosuid_transition.

silent
    Turn on the silent flag.

loud
    Turn off the silent flag.

owner
    Allow an ordinary user to mount the filesystem if that user is the owner of the device. This option implies the options nosuid and nodev (unless overridden by subsequent options, as in the option line owner,dev,suid).

remount
    Attempt to remount an already-mounted filesystem. This is commonly used to change the mount flags for a filesystem, especially to make a readonly filesystem writable. It does not change device or mount point.

    The remount operation together with the bind flag has special semantics. See above, the subsection Bind mounts.

    The remount functionality follows the standard way the mount command works with options from fstab. This means that mount does not read fstab (or mtab) only when both device and dir are specified.

    mount -o remount,rw /dev/foo /dir

    After this call all old mount options are replaced and arbitrary stuff from fstab (or mtab) is ignored, except the loop= option which is internally generated and maintained by the mount command.

    mount -o remount,rw /dir

    After this call, mount reads fstab and merges these options with the options from the command line (-o). If no mountpoint is found in fstab, then a remount with unspecified source is allowed.

    mount allows the use of --all to remount all already mounted filesystems which match a specified filter (-O and -t). For example:

```
    mount --all -o remount,ro -t vfat

    remounts all already mounted vfat filesystems in read-only
    mode. Each of the filesystems is remounted by mount -o
    remount,ro /dir semantic. This means the mount command reads
    fstab or mtab and merges these options with the options from
    the command line.

ro
    Mount the filesystem read-only.

rw
    Mount the filesystem read-write.

sync
    All I/O to the filesystem should be done synchronously. In
    the case of media with a limited number of write cycles (e.g.
    some flash drives), sync may cause life-cycle shortening.

user
    Allow an ordinary user to mount the filesystem. The name of
    the mounting user is written to the mtab file (or to the
    private libmount file in /run/mount on systems without a
    regular mtab) so that this same user can unmount the
    filesystem again. This option implies the options noexec,
    nosuid, and nodev (unless overridden by subsequent options,
    as in the option line user,exec,dev,suid).

nouser
    Forbid an ordinary user to mount the filesystem. This is the
    default; it does not imply any other options.

users
    Allow any user to mount and to unmount the filesystem, even
    when some other ordinary user mounted it. This option implies
    the options noexec, nosuid, and nodev (unless overridden by
    subsequent options, as in the option line
    users,exec,dev,suid).

X-*
    All options prefixed with "X-" are interpreted as comments or
```

as userspace application-specific options. These options are not stored in user space (e.g., mtab file), nor sent to the mount.type helpers nor to the mount(2) system call. The suggested format is X-appname.option.

x-*
The same as X-* options, but stored permanently in user space. This means the options are also available for umount(8) or other operations. Note that maintaining mount options in user space is tricky, because it' s necessary use libmount-based tools and there is no guarantee that the options will be always available (for example after a move mount operation or in unshared namespace).

Note that before util-linux v2.30 the x-* options have not been maintained by libmount and stored in user space (functionality was the same as for X-* now), but due to the growing number of use-cases (in initrd, systemd etc.) the functionality has been extended to keep existing fstab configurations usable without a change.

X-mount.mkdir[=mode]
Allow to make a target directory (mountpoint) if it does not exit yet. The optional argument mode specifies the filesystem access mode used for mkdir(2) in octal notation. The default mode is 0755. This functionality is supported only for root users or when mount executed without suid permissions. The option is also supported as x-mount.mkdir, this notation is deprecated since v2.30.

nosymfollow
Do not follow symlinks when resolving paths. Symlinks can still be created, and readlink(1), readlink(2), realpath(1), and realpath(3) all still work properly.

FILESYSTEM-SPECIFIC MOUNT OPTIONS top

This section lists options that are specific to particular filesystems. Where possible, you should first consult filesystem-specific manual pages for details. Some of those pages are listed in the following table.

Filesystem(s)	Manual page
btrfs	btrfs(5)
cifs	mount.cifs(8)
ext2, ext3, ext4	ext4(5)
fuse	fuse(8)
nfs	nfs(5)
tmpfs	tmpfs(5)
xfs	xfs(5)

Note that some of the pages listed above might be available only after you install the respective userland tools.

The following options apply only to certain filesystems. We sort them by filesystem. All options follow the -o flag.

What options are supported depends a bit on the running kernel. Further information may be available in filesystem-specific files in the kernel source subdirectory Documentation/filesystems.

Mount options for adfs

uid=value and gid=value

Set the owner and group of the files in the filesystem (default: uid=gid=0).

ownmask=value and othmask=value

Set the permission mask for ADFS 'owner' permissions and 'other' permissions, respectively (default: 0700 and 0077, respectively). See also /usr/src/linux/Documentation/filesystems/adfs.rst.

Mount options for affs

uid=value and gid=value

Set the owner and group of the root of the filesystem (default: uid=gid=0, but with option uid or gid without specified value, the UID and GID of the current process are taken).

setuid=value and setgid=value

Set the owner and group of all files.

mode=value

Set the mode of all files to value & 0777 disregarding the original permissions. Add search permission to directories that have read permission. The value is given in octal.

protect

Do not allow any changes to the protection bits on the filesystem.

usemp

Set UID and GID of the root of the filesystem to the UID and GID of the mount point upon the first sync or umount, and then clear this option. Strange...

verbose

Print an informational message for each successful mount.

prefix=string

Prefix used before volume name, when following a link.

volume=string

Prefix (of length at most 30) used before '/' when following a symbolic link.

reserved=value

(Default: 2.) Number of unused blocks at the start of the

device.

root=value
Give explicitly the location of the root block.

bs=value
Give blocksize. Allowed values are 512, 1024, 2048, 4096.

grpquota|noquota|quota|usrquota
These options are accepted but ignored. (However, quota utilities may react to such strings in /etc/fstab.)

Mount options for debugfs
The debugfs filesystem is a pseudo filesystem, traditionally mounted on /sys/kernel/debug. As of kernel version 3.4, debugfs has the following options:

uid=n, gid=n
Set the owner and group of the mountpoint.

mode=value
Sets the mode of the mountpoint.

Mount options for devpts
The devpts filesystem is a pseudo filesystem, traditionally mounted on /dev/pts. In order to acquire a pseudo terminal, a process opens /dev/ptmx; the number of the pseudo terminal is then made available to the process and the pseudo terminal slave can be accessed as /dev/pts/<number>.

uid=value and gid=value
This sets the owner or the group of newly created pseudo terminals to the specified values. When nothing is specified, they will be set to the UID and GID of the creating process. For example, if there is a tty group with GID 5, then gid=5 will cause newly created pseudo terminals to belong to the tty group.

mode=value
Set the mode of newly created pseudo terminals to the specified value. The default is 0600. A value of mode=620 and

gid=5 makes "mesg y" the default on newly created pseudo terminals.

newinstance
Create a private instance of the devpts filesystem, such that indices of pseudo terminals allocated in this new instance are independent of indices created in other instances of devpts.

All mounts of devpts without this newinstance option share the same set of pseudo terminal indices (i.e., legacy mode). Each mount of devpts with the newinstance option has a private set of pseudo terminal indices.

This option is mainly used to support containers in the Linux kernel. It is implemented in Linux kernel versions starting with 2.6.29. Further, this mount option is valid only if CONFIG_DEVPTS_MULTIPLE_INSTANCES is enabled in the kernel configuration.

To use this option effectively, /dev/ptmx must be a symbolic link to pts/ptmx. See Documentation/filesystems/devpts.txt in the Linux kernel source tree for details.

ptmxmode=value
Set the mode for the new ptmx device node in the devpts filesystem.

With the support for multiple instances of devpts (see newinstance option above), each instance has a private ptmx node in the root of the devpts filesystem (typically /dev/pts/ptmx).

For compatibility with older versions of the kernel, the default mode of the new ptmx node is 0000. ptmxmode=value specifies a more useful mode for the ptmx node and is highly recommended when the newinstance option is specified.

This option is only implemented in Linux kernel versions starting with 2.6.29. Further, this option is valid only if CONFIG_DEVPTS_MULTIPLE_INSTANCES is enabled in the kernel

```
        configuration.

Mount options for fat
    (Note: fat is not a separate filesystem, but a common part of the
    msdos, umsdos and vfat filesystems.)

    blocksize={512|1024|2048}
        Set blocksize (default 512). This option is obsolete.

    uid=value and gid=value
        Set the owner and group of all files. (Default: the UID and
        GID of the current process.)

    umask=value
        Set the umask (the bitmask of the permissions that are not
        present). The default is the umask of the current process.
        The value is given in octal.

    dmask=value
        Set the umask applied to directories only. The default is the
        umask of the current process. The value is given in octal.

    fmask=value
        Set the umask applied to regular files only. The default is
        the umask of the current process. The value is given in
        octal.

    allow_utime=value
        This option controls the permission check of mtime/atime.

        20
            If current process is in group of file's group ID, you
            can change timestamp.

        2
            Other users can change timestamp.

    The default is set from 'dmask' option. (If the directory is
    writable, utime(2) is also allowed. I.e. ~dmask & 022)

    Normally utime(2) checks that the current process is owner of the
```

file, or that it has the CAP_FOWNER capability. But FAT filesystems don't have UID/GID on disk, so the normal check is too inflexible. With this option you can relax it.

check=value
    Three different levels of pickiness can be chosen:

    r[elaxed]
        Upper and lower case are accepted and equivalent, long name parts are truncated (e.g. verylongname.foobar becomes verylong.foo), leading and embedded spaces are accepted in each name part (name and extension).

    n[ormal]
        Like "relaxed", but many special characters (*, ?, <, spaces, etc.) are rejected. This is the default.

    s[trict]
        Like "normal", but names that contain long parts or special characters that are sometimes used on Linux but are not accepted by MS-DOS (+, =, etc.) are rejected.

codepage=value
    Sets the codepage for converting to shortname characters on FAT and VFAT filesystems. By default, codepage 437 is used.

conv=mode
    This option is obsolete and may fail or be ignored.

cvf_format=module
    Forces the driver to use the CVF (Compressed Volume File) module cvf__module_ instead of auto-detection. If the kernel supports kmod, the cvf_format=xxx option also controls on-demand CVF module loading. This option is obsolete.

cvf_option=option
    Option passed to the CVF module. This option is obsolete.

debug
    Turn on the debug flag. A version string and a list of filesystem parameters will be printed (these data are also

printed if the parameters appear to be inconsistent).

discard
If set, causes discard/TRIM commands to be issued to the block device when blocks are freed. This is useful for SSD devices and sparse/thinly-provisioned LUNs.

dos1xfloppy
If set, use a fallback default BIOS Parameter Block configuration, determined by backing device size. These static parameters match defaults assumed by DOS 1.x for 160 kiB, 180 kiB, 320 kiB, and 360 kiB floppies and floppy images.

errors={panic|continue|remount-ro}
Specify FAT behavior on critical errors: panic, continue without doing anything, or remount the partition in read-only mode (default behavior).

fat={12|16|32}
Specify a 12, 16 or 32 bit fat. This overrides the automatic FAT type detection routine. Use with caution!

iocharset=value
Character set to use for converting between 8 bit characters and 16 bit Unicode characters. The default is iso8859-1. Long filenames are stored on disk in Unicode format.

nfs={stale_rw|nostale_ro}
Enable this only if you want to export the FAT filesystem over NFS.

stale_rw: This option maintains an index (cache) of directory inodes which is used by the nfs-related code to improve look-ups. Full file operations (read/write) over NFS are supported but with cache eviction at NFS server, this could result in spurious ESTALE errors.

nostale_ro: This option bases the inode number and file handle on the on-disk location of a file in the FAT directory entry. This ensures that ESTALE will not be returned after a

file is evicted from the inode cache. However, it means that operations such as rename, create and unlink could cause file handles that previously pointed at one file to point at a different file, potentially causing data corruption. For this reason, this option also mounts the filesystem readonly.

To maintain backward compatibility, -o nfs is also accepted, defaulting to stale_rw.

tz=UTC

This option disables the conversion of timestamps between local time (as used by Windows on FAT) and UTC (which Linux uses internally). This is particularly useful when mounting devices (like digital cameras) that are set to UTC in order to avoid the pitfalls of local time.

time_offset=minutes

Set offset for conversion of timestamps from local time used by FAT to UTC. I.e., minutes will be subtracted from each timestamp to convert it to UTC used internally by Linux. This is useful when the time zone set in the kernel via settimeofday(2) is not the time zone used by the filesystem. Note that this option still does not provide correct time stamps in all cases in presence of DST - time stamps in a different DST setting will be off by one hour.

quiet

Turn on the quiet flag. Attempts to chown or chmod files do not return errors, although they fail. Use with caution!

rodir

FAT has the ATTR_RO (read-only) attribute. On Windows, the ATTR_RO of the directory will just be ignored, and is used only by applications as a flag (e.g. it’ s set for the customized folder).

If you want to use ATTR_RO as read-only flag even for the directory, set this option.

showexec

If set, the execute permission bits of the file will be

allowed only if the extension part of the name is .EXE, .COM, or .BAT. Not set by default.

sys_immutable
If set, ATTR_SYS attribute on FAT is handled as IMMUTABLE flag on Linux. Not set by default.

flush
If set, the filesystem will try to flush to disk more early than normal. Not set by default.

usefree
Use the "free clusters" value stored on FSINFO. It' ll be used to determine number of free clusters without scanning disk. But it' s not used by default, because recent Windows don' t update it correctly in some case. If you are sure the "free clusters" on FSINFO is correct, by this option you can avoid scanning disk.

dots, nodots, dotsOK=[yes|no]
Various misguided attempts to force Unix or DOS conventions onto a FAT filesystem.

Mount options for hfs
creator=cccc, type=cccc
Set the creator/type values as shown by the MacOS finder used for creating new files. Default values: '????'.

uid=n, gid=n
Set the owner and group of all files. (Default: the UID and GID of the current process.)

dir_umask=n, file_umask=n, umask=n
Set the umask used for all directories, all regular files, or all files and directories. Defaults to the umask of the current process.

session=n
Select the CDROM session to mount. Defaults to leaving that decision to the CDROM driver. This option will fail with anything but a CDROM as underlying device.

part=n
Select partition number n from the device. Only makes sense for CDROMs. Defaults to not parsing the partition table at all.

quiet
Don't complain about invalid mount options.

Mount options for hpfs

uid=value and gid=value
Set the owner and group of all files. (Default: the UID and GID of the current process.)

umask=value
Set the umask (the bitmask of the permissions that are not present). The default is the umask of the current process. The value is given in octal.

case={lower|asis}
Convert all files names to lower case, or leave them. (Default: case=lower.)

conv=mode
This option is obsolete and may fail or being ignored.

nocheck
Do not abort mounting when certain consistency checks fail.

Mount options for iso9660

ISO 9660 is a standard describing a filesystem structure to be used on CD-ROMs. (This filesystem type is also seen on some DVDs. See also the udf filesystem.)

Normal iso9660 filenames appear in an 8.3 format (i.e., DOS-like restrictions on filename length), and in addition all characters are in upper case. Also there is no field for file ownership, protection, number of links, provision for block/character devices, etc.

Rock Ridge is an extension to iso9660 that provides all of these

UNIX-like features. Basically there are extensions to each directory record that supply all of the additional information, and when Rock Ridge is in use, the filesystem is indistinguishable from a normal UNIX filesystem (except that it is read-only, of course).

norock
    Disable the use of Rock Ridge extensions, even if available. Cf. map.

nojoliet
    Disable the use of Microsoft Joliet extensions, even if available. Cf. map.

check={r[elaxed]|s[trict]}
    With check=relaxed, a filename is first converted to lower case before doing the lookup. This is probably only meaningful together with norock and map=normal. (Default: check=strict.)

uid=value and gid=value
    Give all files in the filesystem the indicated user or group id, possibly overriding the information found in the Rock Ridge extensions. (Default: uid=0,gid=0.)

map={n[ormal]|o[ff]|a[corn]}
    For non-Rock Ridge volumes, normal name translation maps upper to lower case ASCII, drops a trailing ';1', and converts ';' to '.'. With map=off no name translation is done. See norock. (Default: map=normal.) map=acorn is like map=normal but also apply Acorn extensions if present.

mode=value
    For non-Rock Ridge volumes, give all files the indicated mode. (Default: read and execute permission for everybody.) Octal mode values require a leading 0.

unhide
    Also show hidden and associated files. (If the ordinary files and the associated or hidden files have the same filenames, this may make the ordinary files inaccessible.)

block={512|1024|2048}
Set the block size to the indicated value. (Default: block=1024.)

conv=mode
This option is obsolete and may fail or being ignored.

cruft
If the high byte of the file length contains other garbage, set this mount option to ignore the high order bits of the file length. This implies that a file cannot be larger than 16 MB.

session=x
Select number of session on a multisession CD.

sbsector=xxx
Session begins from sector xxx.

The following options are the same as for vfat and specifying them only makes sense when using discs encoded using Microsoft' s Joliet extensions.

iocharset=value
Character set to use for converting 16 bit Unicode characters on CD to 8 bit characters. The default is iso8859-1.

utf8
Convert 16 bit Unicode characters on CD to UTF-8.

Mount options for jfs
iocharset=name
Character set to use for converting from Unicode to ASCII. The default is to do no conversion. Use iocharset=utf8 for UTF8 translations. This requires CONFIG_NLS_UTF8 to be set in the kernel .config file.

resize=value
Resize the volume to value blocks. JFS only supports growing a volume, not shrinking it. This option is only valid during

a remount, when the volume is mounted read-write. The resize keyword with no value will grow the volume to the full size of the partition.

nointegrity
Do not write to the journal. The primary use of this option is to allow for higher performance when restoring a volume from backup media. The integrity of the volume is not guaranteed if the system abnormally ends.

integrity
Default. Commit metadata changes to the journal. Use this option to remount a volume where the nointegrity option was previously specified in order to restore normal behavior.

errors={continue|remount-ro|panic}
Define the behavior when an error is encountered. (Either ignore errors and just mark the filesystem erroneous and continue, or remount the filesystem read-only, or panic and halt the system.)

noquota|quota|usrquota|grpquota
These options are accepted but ignored.

Mount options for msdos
See mount options for fat. If the msdos filesystem detects an inconsistency, it reports an error and sets the file system read-only. The filesystem can be made writable again by remounting it.

Mount options for ncpfs
Just like nfs, the ncpfs implementation expects a binary argument (a struct ncp_mount_data) to the mount system call. This argument is constructed by ncpmount(8) and the current version of mount (2.12) does not know anything about ncpfs.

Mount options for ntfs
iocharset=name
Character set to use when returning file names. Unlike VFAT, NTFS suppresses names that contain nonconvertible characters. Deprecated.

nls=name
New name for the option earlier called iocharset.

utf8
Use UTF-8 for converting file names.

uni_xlate={0|1|2}
For 0 (or 'no' or 'false'), do not use escape sequences for unknown Unicode characters. For 1 (or 'yes' or 'true') or 2, use vfat-style 4-byte escape sequences starting with ":". Here 2 gives a little-endian encoding and 1 a byteswapped bigendian encoding.

posix=[0|1]
If enabled (posix=1), the filesystem distinguishes between upper and lower case. The 8.3 alias names are presented as hard links instead of being suppressed. This option is obsolete.

uid=value, gid=value and umask=value
Set the file permission on the filesystem. The umask value is given in octal. By default, the files are owned by root and not readable by somebody else.

Mount options for overlay
Since Linux 3.18 the overlay pseudo filesystem implements a union mount for other filesystems.

An overlay filesystem combines two filesystems - an upper filesystem and a lower filesystem. When a name exists in both filesystems, the object in the upper filesystem is visible while the object in the lower filesystem is either hidden or, in the case of directories, merged with the upper object.

The lower filesystem can be any filesystem supported by Linux and does not need to be writable. The lower filesystem can even be another overlayfs. The upper filesystem will normally be writable and if it is it must support the creation of trusted.* extended attributes, and must provide a valid d_type in readdir responses, so NFS is not suitable.

A read-only overlay of two read-only filesystems may use any filesystem type. The options lowerdir and upperdir are combined into a merged directory by using:

```
mount -t overlay  overlay  \
  -olowerdir=/lower,upperdir=/upper,workdir=/work  /merged
```

lowerdir=directory
    Any filesystem, does not need to be on a writable filesystem.

upperdir=directory
    The upperdir is normally on a writable filesystem.

workdir=directory
    The workdir needs to be an empty directory on the same filesystem as upperdir.

userxattr
    Use the "user.overlay." xattr namespace instead of "trusted.overlay.". This is useful for unprivileged mounting of overlayfs.

redirect_dir={on|off|follow|nofollow}
    If the redirect_dir feature is enabled, then the directory will be copied up (but not the contents). Then the "{trusted|user}.overlay.redirect" extended attribute is set to the path of the original location from the root of the overlay. Finally the directory is moved to the new location.

    on
        Redirects are enabled.

    off
        Redirects are not created and only followed if "redirect_always_follow" feature is enabled in the kernel/module config.

    follow
        Redirects are not created, but followed.

nofollow

Redirects are not created and not followed (equivalent to "redirect_dir=off" if "redirect_always_follow" feature is not enabled).

index={on|off}

Inode index. If this feature is disabled and a file with multiple hard links is copied up, then this will "break" the link. Changes will not be propagated to other names referring to the same inode.

uuid={on|off}

Can be used to replace UUID of the underlying filesystem in file handles with null, and effectively disable UUID checks. This can be useful in case the underlying disk is copied and the UUID of this copy is changed. This is only applicable if all lower/upper/work directories are on the same filesystem, otherwise it will fallback to normal behaviour.

nfs_export={on|off}

When the underlying filesystems supports NFS export and the "nfs_export" feature is enabled, an overlay filesystem may be exported to NFS.

With the “nfs_export” feature, on copy_up of any lower object, an index entry is created under the index directory. The index entry name is the hexadecimal representation of the copy up origin file handle. For a non-directory object, the index entry is a hard link to the upper inode. For a directory object, the index entry has an extended attribute "{trusted|user}.overlay.upper" with an encoded file handle of the upper directory inode.

When encoding a file handle from an overlay filesystem object, the following rules apply

- For a non-upper object, encode a lower file handle from lower inode
- For an indexed object, encode a lower file handle from copy_up origin

- For a pure-upper object and for an existing non-indexed upper object, encode an upper file handle from upper inode

The encoded overlay file handle includes

- Header including path type information (e.g. lower/upper)
- UUID of the underlying filesystem
- Underlying filesystem encoding of underlying inode

This encoding format is identical to the encoding format file handles that are stored in extended attribute "{trusted|user}.overlay.origin". When decoding an overlay file handle, the following steps are followed

- Find underlying layer by UUID and path type information.
- Decode the underlying filesystem file handle to underlying dentry.
- For a lower file handle, lookup the handle in index directory by name.
- If a whiteout is found in index, return ESTALE. This represents an overlay object that was deleted after its file handle was encoded.
- For a non-directory, instantiate a disconnected overlay dentry from the decoded underlying dentry, the path type and index inode, if found.
- For a directory, use the connected underlying decoded dentry, path type and index, to lookup a connected overlay dentry.

Decoding a non-directory file handle may return a

disconnected dentry. copy_up of that disconnected dentry will create an upper index entry with no upper alias.

When overlay filesystem has multiple lower layers, a middle layer directory may have a "redirect" to lower directory. Because middle layer "redirects" are not indexed, a lower file handle that was encoded from the "redirect" origin directory, cannot be used to find the middle or upper layer directory. Similarly, a lower file handle that was encoded from a descendant of the "redirect" origin directory, cannot be used to reconstruct a connected overlay path. To mitigate the cases of directories that cannot be decoded from a lower file handle, these directories are copied up on encode and encoded as an upper file handle. On an overlay filesystem with no upper layer this mitigation cannot be used NFS export in this setup requires turning off redirect follow (e.g. "redirect_dir=nofollow").

The overlay filesystem does not support non-directory connectable file handles, so exporting with the subtree_check exportfs configuration will cause failures to lookup files over NFS.

When the NFS export feature is enabled, all directory index entries are verified on mount time to check that upper file handles are not stale. This verification may cause significant overhead in some cases.

Note: the mount options index=off,nfs_export=on are conflicting for a read-write mount and will result in an error.

xinfo={on|off|auto}

The "xino" feature composes a unique object identifier from the real object st_ino and an underlying fsid index. The "xino" feature uses the high inode number bits for fsid, because the underlying filesystems rarely use the high inode number bits. In case the underlying inode number does overflow into the high xino bits, overlay filesystem will fall back to the non xino behavior for that inode.

```
        For a detailed description of the effect of this option
        please refer to

        ↪  https://www.kernel.org/doc/html/latest/filesystems/overlayfs.html?highligh

    metacopy={on|off}
        When metadata only copy up feature is enabled, overlayfs will
        only copy up metadata (as opposed to whole file), when a
        metadata specific operation like chown/chmod is performed.
        Full file will be copied up later when file is opened for
        WRITE operation.

        In other words, this is delayed data copy up operation and
        data is copied up when there is a need to actually modify
        data.

    volatile
        Volatile mounts are not guaranteed to survive a crash. It is
        strongly recommended that volatile mounts are only used if
        data written to the overlay can be recreated without
        significant effort.

        The advantage of mounting with the "volatile" option is that
        all forms of sync calls to the upper filesystem are omitted.

        In order to avoid a giving a false sense of safety, the
        syncfs (and fsync) semantics of volatile mounts are slightly
        different than that of the rest of VFS. If any writeback
        error occurs on the upperdir' s filesystem after a volatile
        mount takes place, all sync functions will return an error.
        Once this condition is reached, the filesystem will not
        recover, and every subsequent sync call will return an error,
        even if the upperdir has not experience a new error since the
        last sync call.

        When overlay is mounted with "volatile" option, the directory
        "$workdir/work/incompat/volatile" is created. During next
        mount, overlay checks for this directory and refuses to mount
        if present. This is a strong indicator that user should throw
        away upper and work directories and create fresh one. In very
        limited cases where the user knows that the system has not
```

crashed and contents of upperdir are intact, The "volatile" directory can be removed.

Mount options for reiserfs

Reiserfs is a journaling filesystem.

conv

Instructs version 3.6 reiserfs software to mount a version 3.5 filesystem, using the 3.6 format for newly created objects. This filesystem will no longer be compatible with reiserfs 3.5 tools.

hash={rupasov|tea|r5|detect}

Choose which hash function reiserfs will use to find files within directories.

rupasov

A hash invented by Yury Yu. Rupasov. It is fast and preserves locality, mapping lexicographically close file names to close hash values. This option should not be used, as it causes a high probability of hash collisions.

tea

A Davis-Meyer function implemented by Jeremy Fitzhardinge. It uses hash permuting bits in the name. It gets high randomness and, therefore, low probability of hash collisions at some CPU cost. This may be used if EHASHCOLLISION errors are experienced with the r5 hash.

r5

A modified version of the rupasov hash. It is used by default and is the best choice unless the filesystem has huge directories and unusual file-name patterns.

detect

Instructs mount to detect which hash function is in use by examining the filesystem being mounted, and to write this information into the reiserfs superblock. This is only useful on the first mount of an old format filesystem.

hashed_relocation
    Tunes the block allocator. This may provide performance improvements in some situations.

no_unhashed_relocation
    Tunes the block allocator. This may provide performance improvements in some situations.

noborder
    Disable the border allocator algorithm invented by Yury Yu. Rupasov. This may provide performance improvements in some situations.

nolog
    Disable journaling. This will provide slight performance improvements in some situations at the cost of losing reiserfs' s fast recovery from crashes. Even with this option turned on, reiserfs still performs all journaling operations, save for actual writes into its journaling area. Implementation of nolog is a work in progress.

notail
    By default, reiserfs stores small files and 'file tails' directly into its tree. This confuses some utilities such as lilo(8). This option is used to disable packing of files into the tree.

replayonly
    Replay the transactions which are in the journal, but do not actually mount the filesystem. Mainly used by reiserfsck.

resize=number
    A remount option which permits online expansion of reiserfs partitions. Instructs reiserfs to assume that the device has number blocks. This option is designed for use with devices which are under logical volume management (LVM). There is a special resizer utility which can be obtained from ftp://ftp.namesys.com/pub/reiserfsprogs.

user_xattr
    Enable Extended User Attributes. See the attr(1) manual page.

acl
Enable POSIX Access Control Lists. See the acl(5) manual page.

barrier=none / barrier=flush
This disables / enables the use of write barriers in the journaling code. barrier=none disables, barrier=flush enables (default). This also requires an IO stack which can support barriers, and if reiserfs gets an error on a barrier write, it will disable barriers again with a warning. Write barriers enforce proper on-disk ordering of journal commits, making volatile disk write caches safe to use, at some performance penalty. If your disks are battery-backed in one way or another, disabling barriers may safely improve performance.

Mount options for ubifs
UBIFS is a flash filesystem which works on top of UBI volumes. Note that atime is not supported and is always turned off.

The device name may be specified as

ubiX_Y
UBI device number X, volume number Y

ubiY
UBI device number 0, volume number Y

ubiX:NAME
UBI device number X, volume with name NAME

ubi:NAME
UBI device number 0, volume with name NAME

Alternative ! separator may be used instead of :.

The following mount options are available:

bulk_read
Enable bulk-read. VFS read-ahead is disabled because it slows down the filesystem. Bulk-Read is an internal optimization.

Some flashes may read faster if the data are read at one go, rather than at several read requests. For example, OneNAND can do "read-while-load" if it reads more than one NAND page.

no_bulk_read
Do not bulk-read. This is the default.

chk_data_crc
Check data CRC-32 checksums. This is the default.

no_chk_data_crc
Do not check data CRC-32 checksums. With this option, the filesystem does not check CRC-32 checksum for data, but it does check it for the internal indexing information. This option only affects reading, not writing. CRC-32 is always calculated when writing the data.

compr={none|lzo|zlib}
Select the default compressor which is used when new files are written. It is still possible to read compressed files if mounted with the none option.

Mount options for udf
UDF is the "Universal Disk Format" filesystem defined by OSTA, the Optical Storage Technology Association, and is often used for DVD-ROM, frequently in the form of a hybrid UDF/ISO-9660 filesystem. It is, however, perfectly usable by itself on disk drives, flash drives and other block devices. See also iso9660.

uid=
Make all files in the filesystem belong to the given user. uid=forget can be specified independently of (or usually in addition to) uid=<user> and results in UDF not storing uids to the media. In fact the recorded uid is the 32-bit overflow uid -1 as defined by the UDF standard. The value is given as either <user> which is a valid user name or the corresponding decimal user id, or the special string "forget".

gid=
Make all files in the filesystem belong to the given group. gid=forget can be specified independently of (or usually in

addition to) gid=<group> and results in UDF not storing gids to the media. In fact the recorded gid is the 32-bit overflow gid -1 as defined by the UDF standard. The value is given as either <group> which is a valid group name or the corresponding decimal group id, or the special string "forget".

umask=
Mask out the given permissions from all inodes read from the filesystem. The value is given in octal.

mode=
If mode= is set the permissions of all non-directory inodes read from the filesystem will be set to the given mode. The value is given in octal.

dmode=
If dmode= is set the permissions of all directory inodes read from the filesystem will be set to the given dmode. The value is given in octal.

bs=
Set the block size. Default value prior to kernel version 2.6.30 was 2048. Since 2.6.30 and prior to 4.11 it was logical device block size with fallback to 2048. Since 4.11 it is logical block size with fallback to any valid block size between logical device block size and 4096.

For other details see the mkudffs(8) 2.0+ manpage, sections COMPATIBILITY and BLOCK SIZE.

unhide
Show otherwise hidden files.

undelete
Show deleted files in lists.

adinicb
Embed data in the inode. (default)

noadinicb

```
            Don't embed data in the inode.

        shortad
            Use short UDF address descriptors.

        longad
            Use long UDF address descriptors. (default)

        nostrict
            Unset strict conformance.

        iocharset=
            Set the NLS character set. This requires kernel compiled with
            CONFIG_UDF_NLS option.

        utf8
            Set the UTF-8 character set.

    Mount options for debugging and disaster recovery
        novrs
            Ignore the Volume Recognition Sequence and attempt to mount
            anyway.

        session=
            Select the session number for multi-session recorded optical
            media. (default= last session)

        anchor=
            Override standard anchor location. (default= 256)

        lastblock=
            Set the last block of the filesystem.

    Unused historical mount options that may be encountered and should be
        removed
        uid=ignore
            Ignored, use uid=<user> instead.

        gid=ignore
            Ignored, use gid=<group> instead.
```

```
        volume=
            Unimplemented and ignored.

        partition=
            Unimplemented and ignored.

        fileset=
            Unimplemented and ignored.

        rootdir=
            Unimplemented and ignored.

    Mount options for ufs
        ufstype=value
            UFS is a filesystem widely used in different operating
            systems. The problem are differences among implementations.
            Features of some implementations are undocumented, so its
            hard to recognize the type of ufs automatically. That' s why
            the user must specify the type of ufs by mount option.
            Possible values are:

            old
                Old format of ufs, this is the default, read only. (Don' t
                forget to give the -r option.)

            44bsd
                For filesystems created by a BSD-like system (NetBSD,
                FreeBSD, OpenBSD).

            ufs2
                Used in FreeBSD 5.x supported as read-write.

            5xbsd
                Synonym for ufs2.

            sun
                For filesystems created by SunOS or Solaris on Sparc.

            sunx86
                For filesystems created by Solaris on x86.
```

```
        hp
            For filesystems created by HP-UX, read-only.

        nextstep
            For filesystems created by NeXTStep (on NeXT station)
            (currently read only).

        nextstep-cd
            For NextStep CDROMs (block_size == 2048), read-only.

        openstep
            For filesystems created by OpenStep (currently read
            only). The same filesystem type is also used by Mac OS X.

    onerror=value
        Set behavior on error:

        panic
            If an error is encountered, cause a kernel panic.

        [lock|umount|repair]
            These mount options don't do anything at present; when an
            error is encountered only a console message is printed.

Mount options for umsdos
    See mount options for msdos. The dotsOK option is explicitly
    killed by umsdos.

Mount options for vfat
    First of all, the mount options for fat are recognized. The
    dotsOK option is explicitly killed by vfat. Furthermore, there
    are

    uni_xlate
        Translate unhandled Unicode characters to special escaped
        sequences. This lets you backup and restore filenames that
        are created with any Unicode characters. Without this option,
        a '?' is used when no translation is possible. The escape
        character is ':' because it is otherwise invalid on the vfat
        filesystem. The escape sequence that gets used, where u is
        the Unicode character, is: ':', (u & 0x3f), ((u>>6) & 0x3f),
```

```
        (u>>12).

    posix
        Allow two files with names that only differ in case. This
        option is obsolete.

    nonumtail
        First try to make a short name without sequence number,
        before trying name~num.ext.

    utf8
        UTF8 is the filesystem safe 8-bit encoding of Unicode that is
        used by the console. It can be enabled for the filesystem
        with this option or disabled with utf8=0, utf8=no or
        utf8=false. If uni_xlate gets set, UTF8 gets disabled.

    shortname=mode
        Defines the behavior for creation and display of filenames
        which fit into 8.3 characters. If a long name for a file
        exists, it will always be the preferred one for display.
        There are four modes:

        lower
            Force the short name to lower case upon display; store a
            long name when the short name is not all upper case.

        win95
            Force the short name to upper case upon display; store a
            long name when the short name is not all upper case.

        winnt
            Display the short name as is; store a long name when the
            short name is not all lower case or all upper case.

        mixed
            Display the short name as is; store a long name when the
            short name is not all upper case. This mode is the
            default since Linux 2.6.32.

Mount options for usbfs
    devuid=uid and devgid=gid and devmode=mode
```

Set the owner and group and mode of the device files in the usbfs filesystem (default: uid=gid=0, mode=0644). The mode is given in octal.

busuid=uid and busgid=gid and busmode=mode
Set the owner and group and mode of the bus directories in the usbfs filesystem (default: uid=gid=0, mode=0555). The mode is given in octal.

listuid=uid and listgid=gid and listmode=mode
Set the owner and group and mode of the file devices (default: uid=gid=0, mode=0444). The mode is given in octal.

DM-VERITY SUPPORT (EXPERIMENTAL) top

The device-mapper verity target provides read-only transparent integrity checking of block devices using kernel crypto API. The mount command can open the dm-verity device and do the integrity verification before on the device filesystem is mounted. Requires libcryptsetup with in libmount (optionally via dlopen(3)). If libcryptsetup supports extracting the root hash of an already mounted device, existing devices will be automatically reused in case of a match. Mount options for dm-verity:

verity.hashdevice=path
Path to the hash tree device associated with the source volume to pass to dm-verity.

verity.roothash=hex
Hex-encoded hash of the root of verity.hashdevice. Mutually exclusive with verity.roothashfile.

verity.roothashfile=path
Path to file containing the hex-encoded hash of the root of verity.hashdevice. Mutually exclusive with verity.roothash.

verity.hashoffset=offset
If the hash tree device is embedded in the source volume, offset (default: 0) is used by dm-verity to get to the tree.

verity.fecdevice=path
Path to the Forward Error Correction (FEC) device associated with the source volume to pass to dm-verity. Optional.

```
            Requires kernel built with CONFIG_DM_VERITY_FEC.

        verity.fecoffset=offset
            If the FEC device is embedded in the source volume, offset
            (default: 0) is used by dm-verity to get to the FEC area.
            Optional.

        verity.fecroots=value
            Parity bytes for FEC (default: 2). Optional.

        verity.roothashsig=path
            Path to pkcs7(1ssl) signature of root hash hex string.
            Requires crypt_activate_by_signed_key() from cryptsetup and
            kernel built with CONFIG_DM_VERITY_VERIFY_ROOTHASH_SIG. For
            device reuse, signatures have to be either used by all mounts
            of a device or by none. Optional.

        Supported since util-linux v2.35.

        For example commands:

            mksquashfs /etc /tmp/etc.squashfs
            dd if=/dev/zero of=/tmp/etc.hash bs=1M count=10
            veritysetup format /tmp/etc.squashfs /tmp/etc.hash
            openssl smime -sign -in <hash> -nocerts -inkey private.key \
            -signer private.crt -noattr -binary -outform der -out
            ↪  /tmp/etc.p7
            mount -o
            ↪  verity.hashdevice=/tmp/etc.hash,verity.roothash=<hash>,\
            verity.roothashsig=/tmp/etc.p7 /tmp/etc.squashfs /mnt

        create squashfs image from /etc directory, verity hash device and
        mount verified filesystem image to /mnt. The kernel will verify
        that the root hash is signed by a key from the kernel keyring if
        roothashsig is used.
LOOP-DEVICE SUPPORT         top
       One further possible type is a mount via the loop device. For
       example, the command

          mount /tmp/disk.img /mnt -t vfat -o loop=/dev/loop3
```

will set up the loop device /dev/loop3 to correspond to the file /tmp/disk.img, and then mount this device on /mnt.

If no explicit loop device is mentioned (but just an option '-o loop' is given), then mount will try to find some unused loop device and use that, for example

```
mount /tmp/disk.img /mnt -o loop
```

The mount command automatically creates a loop device from a regular file if a filesystem type is not specified or the filesystem is known for libblkid, for example:

```
mount /tmp/disk.img /mnt

mount -t ext4 /tmp/disk.img /mnt
```

This type of mount knows about three options, namely loop, offset and sizelimit, that are really options to losetup(8). (These options can be used in addition to those specific to the filesystem type.)

Since Linux 2.6.25 auto-destruction of loop devices is supported, meaning that any loop device allocated by mount will be freed by umount independently of /etc/mtab.

You can also free a loop device by hand, using losetup -d or umount -d.

Since util-linux v2.29, mount re-uses the loop device rather than initializing a new device if the same backing file is already used for some loop device with the same offset and sizelimit. This is necessary to avoid a filesystem corruption.

EXIT STATUS top

mount has the following exit status values (the bits can be ORed):

0
success

1

```
            incorrect invocation or permissions

       2
            system error (out of memory, cannot fork, no more loop
            devices)

       4
            internal mount bug

       8
            user interrupt

       16
            problems writing or locking /etc/mtab

       32
            mount failure

       64
            some mount succeeded

            The command mount -a returns 0 (all succeeded), 32 (all
            failed), or 64 (some failed, some succeeded).
EXTERNAL HELPERS         top
       The syntax of external mount helpers is:

       /sbin/mount.suffix spec dir [-sfnv] [-N namespace] [-o options]
       [-t type._subtype_]

       where the suffix is the filesystem type and the -sfnvoN options
       have the same meaning as the normal mount options. The -t option
       is used for filesystems with subtypes support (for example
       /sbin/mount.fuse -t fuse.sshfs).

       The command mount does not pass the mount options unbindable,
       runbindable, private, rprivate, slave, rslave, shared, rshared,
       auto, noauto, comment, x-*, loop, offset and sizelimit to the
       mount.<suffix> helpers. All other options are used in a
       comma-separated list as an argument to the -o option.
ENVIRONMENT         top
       LIBMOUNT_FSTAB=<path>
```

```
            overrides the default location of the fstab file (ignored for
            suid)

        LIBMOUNT_MTAB=<path>
            overrides the default location of the mtab file (ignored for
            suid)

        LIBMOUNT_DEBUG=all
            enables libmount debug output

        LIBBLKID_DEBUG=all
            enables libblkid debug output

        LOOPDEV_DEBUG=all
            enables loop device setup debug output
FILES         top
       See also "The files /etc/fstab, /etc/mtab and /proc/mounts"
       section above.

       /etc/fstab
            filesystem table

       /run/mount
            libmount private runtime directory

       /etc/mtab
            table of mounted filesystems or symlink to /proc/mounts

       /etc/mtab~
            lock file (unused on systems with mtab symlink)

       /etc/mtab.tmp
            temporary file (unused on systems with mtab symlink)

       /etc/filesystems
            a list of filesystem types to try
HISTORY         top
       A mount command existed in Version 5 AT&T UNIX.
BUGS         top
       It is possible for a corrupted filesystem to cause a crash.
```

Some Linux filesystems don't support -o sync and -o dirsync (the ext2, ext3, ext4, fat and vfat filesystems do support synchronous updates (a la BSD) when mounted with the sync option).

The -o remount may not be able to change mount parameters (all ext2fs-specific parameters, except sb, are changeable with a remount, for example, but you can't change gid or umask for the fatfs).

It is possible that the files /etc/mtab and /proc/mounts don't match on systems with a regular mtab file. The first file is based only on the mount command options, but the content of the second file also depends on the kernel and others settings (e.g. on a remote NFS server — in certain cases the mount command may report unreliable information about an NFS mount point and the /proc/mount file usually contains more reliable information.) This is another reason to replace the mtab file with a symlink to the /proc/mounts file.

Checking files on NFS filesystems referenced by file descriptors (i.e. the fcntl and ioctl families of functions) may lead to inconsistent results due to the lack of a consistency check in the kernel even if the noac mount option is used.

The loop option with the offset or sizelimit options used may fail when using older kernels if the mount command can't confirm that the size of the block device has been configured as requested. This situation can be worked around by using the losetup(8) command manually before calling mount with the configured loop device.

AUTHORS top

Karel Zak <kzak@redhat.com>

SEE ALSO top

mount(2), umount(2), filesystems(5), fstab(5), nfs(5), xfs(5), mount_namespaces(7), xattr(7), e2label(8), findmnt(8), losetup(8), lsblk(8), mke2fs(8), mountd(8), nfsd(8), swapon(8), tune2fs(8), umount(8), xfs_admin(8)

REPORTING BUGS top

For bug reports, use the issue tracker at https://github.com/karelzak/util-linux/issues.

AVAILABILITY top

The mount command is part of the util-linux package which can be downloaded from Linux Kernel Archive <https://www.kernel.org/pub/linux/utils/util-linux/>. This page is part of the util-linux (a random collection of Linux utilities) project. Information about the project can be found at ⟨https://www.kernel.org/pub/linux/utils/util-linux/⟩. If you have a bug report for this manual page, send it to util-linux@vger.kernel.org. This page was obtained from the project's upstream Git repository ⟨git://git.kernel.org/pub/scm/utils/util-linux/util-linux.git⟩ on 2021-06-20. (At that time, the date of the most recent commit that was found in the repository was 2021-06-18.) If you discover any rendering problems in this HTML version of the page, or you believe there is a better or more up-to-date source for the page, or you have corrections or improvements to the information in this COLOPHON (which is not part of the original manual page), send a mail to man-pages@man7.org

util-linux 2.37.109-b366e69 2021-06-20 MOUNT(8)

3 screen

```
SCREEN(1)                General Commands Manual                SCREEN(1)
NAME         top
       screen - screen manager with VT100/ANSI terminal emulation
SYNOPSIS         top
       screen [ -options ] [ cmd [ args ] ]
       screen -r [[pid.]tty[.host]]
       screen -r sessionowner/[[pid.]tty[.host]]
DESCRIPTION         top
       Screen is a full-screen window manager that multiplexes a
       physical terminal between several processes (typically
       interactive shells).  Each virtual terminal provides the
       functions of a DEC VT100 terminal and, in addition, several
       control functions from the ISO 6429 (ECMA 48, ANSI X3.64) and ISO
       2022 standards (e.g. insert/delete line and support for multiple
       character sets).  There is a scrollback history buffer for each
       virtual terminal and a copy-and-paste mechanism that allows
       moving text regions between windows.

       When screen is called, it creates a single window with a shell in
       it (or the specified command) and then gets out of your way so
       that you can use the program as you normally would.  Then, at any
       time, you can create new (full-screen) windows with other
       programs in them (including more shells), kill existing windows,
       view a list of windows, turn output logging on and off, copy-and-
       paste text between windows, view the scrollback history, switch
       between windows in whatever manner you wish, etc. All windows run
       their programs completely independent of each other. Programs
       continue to run when their window is currently not visible and
       even when the whole screen session is detached from the user's
       terminal.  When a program terminates, screen (per default) kills
       the window that contained it.  If this window was in the
       foreground, the display switches to the previous window; if none
       are left, screen exits. Shells usually distinguish between
       running as login-shell or sub-shell.  Screen runs them as sub-
       shells, unless told otherwise (See "shell" .screenrc command).

       Everything you type is sent to the program running in the current
       window.  The only exception to this is the one keystroke that is
       used to initiate a command to the window manager.  By default,
```

each command begins with a control-a (abbreviated C-a from now on), and is followed by one other keystroke. The command character and all the key bindings can be fully customized to be anything you like, though they are always two characters in length.

Screen does not understand the prefix "C-" to mean control, although this notation is used in this manual for readability. Please use the caret notation ("^A" instead of "C-a") as arguments to e.g. the escape command or the -e option. Screen will also print out control characters in caret notation.

The standard way to create a new window is to type "C-a c". This creates a new window running a shell and switches to that window immediately, regardless of the state of the process running in the current window. Similarly, you can create a new window with a custom command in it by first binding the command to a keystroke (in your .screenrc file or at the "C-a :" command line) and then using it just like the "C-a c" command. In addition, new windows can be created by running a command like:

```
screen emacs prog.c
```

from a shell prompt within a previously created window. This will not run another copy of screen, but will instead supply the command name and its arguments to the window manager (specified in the $STY environment variable) who will use it to create the new window. The above example would start the emacs editor (editing prog.c) and switch to its window. - Note that you cannot transport environment variables from the invoking shell to the application (emacs in this case), because it is forked from the parent screen process, not from the invoking shell.

If "/etc/utmp" is writable by screen, an appropriate record will be written to this file for each window, and removed when the window is terminated. This is useful for working with "talk", "script", "shutdown", "rsend", "sccs" and other similar programs that use the utmp file to determine who you are. As long as screen is active on your terminal, the terminal's own record is removed from the utmp file. See also "C-a L".

GETTING STARTED top

Before you begin to use screen you'll need to make sure you have correctly selected your terminal type, just as you would for any other termcap/terminfo program. (You can do this by using tset for example.)

If you're impatient and want to get started without doing a lot more reading, you should remember this one command: "C-a ?". Typing these two characters will display a list of the available screen commands and their bindings. Each keystroke is discussed in the section "DEFAULT KEY BINDINGS". The manual section "CUSTOMIZATION" deals with the contents of your .screenrc.

If your terminal is a "true" auto-margin terminal (it doesn't allow the last position on the screen to be updated without scrolling the screen) consider using a version of your terminal's termcap that has automatic margins turned off. This will ensure an accurate and optimal update of the screen in all circumstances. Most terminals nowadays have "magic" margins (automatic margins plus usable last column). This is the VT100 style type and perfectly suited for screen. If all you've got is a "true" auto-margin terminal screen will be content to use it, but updating a character put into the last position on the screen may not be possible until the screen scrolls or the character is moved into a safe position in some other way. This delay can be shortened by using a terminal with insert-character capability.

COMMAND-LINE OPTIONS top

Screen has the following command-line options:

-a include all capabilities (with some minor exceptions) in each window's termcap, even if screen must redraw parts of the display in order to implement a function.

-A Adapt the sizes of all windows to the size of the current terminal. By default, screen tries to restore its old window sizes when attaching to resizable terminals (those with "WS" in its description, e.g. suncmd or some xterm).

-c file
override the default configuration file from "$HOME/.screenrc" to file.

-d|-D [pid.tty.host]
does not start screen, but detaches the elsewhere running screen session. It has the same effect as typing "C-a d" from screen's controlling terminal. -D is the equivalent to the power detach key. If no session can be detached, this option is ignored. In combination with the -r/-R option more powerful effects can be achieved:

-d -r Reattach a session and if necessary detach it first.

-d -R Reattach a session and if necessary detach or even create it first.

-d -RR Reattach a session and if necessary detach or create it. Use the first session if more than one session is available.

-D -r Reattach a session. If necessary detach and logout remotely first.

-D -R Attach here and now. In detail this means: If a session is running, then reattach. If necessary detach and logout remotely first. If it was not running create it and notify the user. This is the author's favorite.

-D -RR Attach here and now. Whatever that means, just do it.

Note: It is always a good idea to check the status of your sessions by means of "screen -list".

-e xy
specifies the command character to be x and the character generating a literal command character to y (when typed after the command character). The default is "C-a" and `a', which can be specified as "-e^Aa". When creating a screen session, this option sets the default command character. In a multiuser session all users added will start off with this command character. But when attaching to an already running session, this option changes only the command character of the attaching user. This option is equivalent to either the commands "defescape" or "escape" respectively.

-f, -fn, and -fa
     turns flow-control on, off, or "automatic switching mode". This can also be defined through the "defflow" .screenrc command.

-h num
     Specifies the history scrollback buffer to be num lines high.

-i   will cause the interrupt key (usually C-c) to interrupt the display immediately when flow-control is on. See the "defflow" .screenrc command for details. The use of this option is discouraged.

-l and -ln
     turns login mode on or off (for /etc/utmp updating). This can also be defined through the "deflogin" .screenrc command.

-ls [match]
-list [match]
     does not start screen, but prints a list of pid.tty.host strings identifying your screen sessions. Sessions marked `detached' can be resumed with "screen -r". Those marked `attached' are running and have a controlling terminal. If the session runs in multiuser mode, it is marked `multi'. Sessions marked as `unreachable' either live on a different host or are `dead'. An unreachable session is considered dead, when its name matches either the name of the local host, or the specified parameter, if any. See the -r flag for a description how to construct matches. Sessions marked as `dead' should be thoroughly checked and removed. Ask your system administrator if you are not sure. Remove sessions with the -wipe option.

-L   tells screen to turn on automatic output logging for the windows.

-Logfile file
     By default logfile name is "screenlog.0". You can set new

logfile name with the "-Logfile" option.

-m causes screen to ignore the $STY environment variable. With "screen -m" creation of a new session is enforced, regardless whether screen is called from within another screen session or not. This flag has a special meaning in connection with the `-d' option:

-d -m Start screen in "detached" mode. This creates a new session but doesn't attach to it. This is useful for system startup scripts.

-D -m This also starts screen in "detached" mode, but doesn't fork a new process. The command exits if the session terminates.

-O selects an optimal output mode for your terminal rather than true VT100 emulation (only affects auto-margin terminals without `LP'). This can also be set in your .screenrc by specifying `OP' in a "termcap" command.

-p number_or_name|-|=|+
Preselect a window. This is useful when you want to reattach to a specific window or you want to send a command via the "-X" option to a specific window. As with screen's select command, "-" selects the blank window. As a special case for reattach, "=" brings up the windowlist on the blank window, while a "+" will create a new window. The command will not be executed if the specified window could not be found.

-q Suppress printing of error messages. In combination with "-ls" the exit value is as follows: 9 indicates a directory without sessions. 10 indicates a directory with running but not attachable sessions. 11 (or more) indicates 1 (or more) usable sessions. In combination with "-r" the exit value is as follows: 10 indicates that there is no session to resume. 12 (or more) indicates that there are 2 (or more) sessions to resume and you should specify which one to choose. In all other cases "-q" has no effect.

-Q Some commands now can be queried from a remote session using

this flag, e.g. "screen -Q windows". The commands will send the response to the stdout of the querying process. If there was an error in the command, then the querying process will exit with a non-zero status.

The commands that can be queried now are:
echo
info
lastmsg
number
select
time
title
windows

-r [pid.tty.host]
-r sessionowner/[pid.tty.host]
resumes a detached screen session. No other options (except combinations with -d/-D) may be specified, though an optional prefix of [pid.]tty.host may be needed to distinguish between multiple detached screen sessions. The second form is used to connect to another user's screen session which runs in multiuser mode. This indicates that screen should look for sessions in another user's directory. This requires setuid-root.

-R resumes screen only when it's unambiguous which one to attach, usually when only one screen is detached. Otherwise lists available sessions. -RR attempts to resume the first detached screen session it finds. If successful, all other command-line options are ignored. If no detached session exists, starts a new session using the specified options, just as if -R had not been specified. The option is set by default if screen is run as a login-shell (actually screen uses "-xRR" in that case). For combinations with the -d/-D option see there.

-s program
sets the default shell to the program specified, instead of the value in the environment variable $SHELL (or "/bin/sh" if not defined). This can also be defined through the

"shell" .screenrc command. See also there.

-S sessionname
When creating a new session, this option can be used to specify a meaningful name for the session. This name identifies the session for "screen -list" and "screen -r" actions. It substitutes the default [tty.host] suffix.

-t name
sets the title (a.k.a.) for the default shell or specified program. See also the "shelltitle" .screenrc command.

-T term
Set the $TERM environment variable using the specified term as opposed to the default setting of screen.

-U Run screen in UTF-8 mode. This option tells screen that your terminal sends and understands UTF-8 encoded characters. It also sets the default encoding for new windows to `utf8'.

-v Print version number.

-wipe [match]
does the same as "screen -ls", but removes destroyed sessions instead of marking them as `dead'. An unreachable session is considered dead, when its name matches either the name of the local host, or the explicitly given parameter, if any. See the -r flag for a description how to construct matches.

-x Attach to a not detached screen session. (Multi display mode). Screen refuses to attach from within itself. But when cascading multiple screens, loops are not detected; take care.

-X Send the specified command to a running screen session. You may use the -S option to specify the screen session if you have several screen sessions running. You can use the -d or -r option to tell screen to look only for attached or detached screen sessions. Note that this command doesn't work if the session is password protected.

-4 Resolve hostnames only to IPv4 addresses.

-6 Resolve hostnames only to IPv6 addresses.

DEFAULT KEY BINDINGS top

As mentioned, each screen command consists of a "C-a" followed by one other character. For your convenience, all commands that are bound to lower-case letters are also bound to their control character counterparts (with the exception of "C-a a"; see below), thus, "C-a c" as well as "C-a C-c" can be used to create a window. See section "CUSTOMIZATION" for a description of the command.

The following table shows the default key bindings. The trailing commas in boxes with multiple keystroke entries are separators, not part of the bindings.

C-a '	(select)	Prompt for a window name or number to switch to.
C-a "	(windowlist -b)	Present a list of all windows for selection.
C-a digit	(select 0-9)	Switch to window number 0 - 9
C-a -	(select -)	Switch to window number 0 - 9, or to the blank window.
C-a tab	(focus)	Switch the input focus to the next region. See also split, remove, only.

C-a C-a	(other)	Toggle to the window displayed previously. Note that this binding defaults to the command character typed twice, unless overridden. For instance, if you use the option "-e]x", this command becomes "]]".
C-a a	(meta)	Send the command character (C-a) to window. See escape command.
C-a A	(title)	Allow the user to enter a name for the current window.
C-a b, C-a C-b	(break)	Send a break to window.
C-a B	(pow_break)	Reopen the terminal line and send a break.
C-a c, C-a C-c	(screen)	Create a new window with a shell and switch to that window.
C-a C	(clear)	Clear the screen.
C-a d, C-a C-d	(detach)	Detach screen from this terminal.

C-a D D	(pow_detach)	Detach and logout.
C-a f, C-a C-f	(flow)	Toggle flow on, off or auto.
C-a F	(fit)	Resize the window to the current region size.
C-a C-g	(vbell)	Toggles screen's visual bell mode.
C-a h	(hardcopy)	Write a hardcopy of the current window to the file "hardcopy.n".
C-a H	(log)	Begins/ends logging of the current window to the file "screenlog.n".
C-a i, C-a C-i	(info)	Show info about this window.
C-a k, C-a C-k	(kill)	Destroy current window.
C-a l, C-a C-l	(redisplay)	Fully refresh current window.
C-a L	(login)	Toggle this windows login slot. Available only if screen is configured to update the utmp database.

C-a m, C-a C-m	(lastmsg)	Repeat the last message displayed in the message line.
C-a M	(monitor)	Toggles monitoring of the current window.
C-a space, C-a n, C-a C-n	(next)	Switch to the next window.
C-a N	(number)	Show the number (and title) of the current window.
C-a backspace, C-a C-h, C-a p, C-a C-p	(prev)	Switch to the previous window (opposite of C-a n).
C-a q, C-a C-q	(xon)	Send a control-q to the current window.
C-a Q	(only)	Delete all regions but the current one. See also split, remove, focus.
C-a r, C-a C-r	(wrap)	Toggle the current window's line-wrap setting (turn the current window's automatic margins on and off).
C-a s,	(xoff)	Send a control-s

C-a C-s;		to the current window.
C-a S	(split)	Split the current region horizontally into two new ones. See also only, remove, focus.
C-a t, C-a C-t	(time)	Show system information.
C-a u, C-a C-u	(parent)	Switch to the parent window.
C-a v	(version)	Display the version and compilation date.
C-a C-v	(digraph)	Enter digraph.
C-a w, C-a C-w	(windows)	Show a list of window.
C-a W	(width)	Toggle 80/132 columns.
C-a x or C-a C-x	(lockscreen)	Lock this terminal.
C-a X	(remove)	Kill the current region. See also split, only, focus.
C-a z, C-a C-z	(suspend)	Suspend screen. Your system must support BSD-style job-control.

C-a Z	(reset)	Reset the virtual terminal to its "power-on" values.
C-a .	(dumptermcap)	Write out a ".termcap" file.
C-a ?	(help)	Show key bindings.
C-a \	(quit)	Kill all windows and terminate screen.
C-a :	(colon)	Enter command line mode.
C-a [, C-a C-[, C-a esc	(copy)	Enter copy/scrollback mode.
C-a C-], C-a]	(paste .)	Write the contents of the paste buffer to the stdin queue of the current window.
C-a {, C-a }	(history)	Copy and paste a previous (command) line.
C-a >	(writebuf)	Write paste buffer to a file.
C-a <	(readbuf)	Reads the screen-exchange file into the paste buffer.
C-a =	(removebuf)	Removes the file used by C-a < and C-a >.

C-a ,	(license)	Shows where screen comes from, where it went to and why you can use it.
C-a _	(silence)	Start/stop monitoring the current window for inactivity.
C-a |	(split -v)	Split the current region vertically into two new ones.
C-a *	(displays)	Show a listing of all currently attached displays.

CUSTOMIZATION top

The "socket directory" defaults either to $HOME/.screen or simply to /tmp/screens or preferably to /usr/local/screens chosen at compile-time. If screen is installed setuid-root, then the administrator should compile screen with an adequate (not NFS mounted) socket directory. If screen is not running setuid-root, the user can specify any mode 700 directory in the environment variable $SCREENDIR.

When screen is invoked, it executes initialization commands from the files "/usr/local/etc/screenrc" and ".screenrc" in the user's home directory. These are the "programmer's defaults" that can be overridden in the following ways: for the global screenrc file screen searches for the environment variable $SYSTEM_SCREENRC (this override feature may be disabled at compile-time). The user specific screenrc file is searched in $SCREENRC, then $HOME/.screenrc. The command line option -c takes precedence over the above user screenrc files.

Commands in these files are used to set options, bind functions to keys, and to automatically establish one or more windows at the beginning of your screen session. Commands are listed one

per line, with empty lines being ignored. A command's arguments are separated by tabs or spaces, and may be surrounded by single or double quotes. A `#' turns the rest of the line into a comment, except in quotes. Unintelligible lines are warned about and ignored. Commands may contain references to environment variables. The syntax is the shell-like "$VAR " or "${VAR}". Note that this causes incompatibility with previous screen versions, as now the '$'-character has to be protected with '\' if no variable substitution shall be performed. A string in single-quotes is also protected from variable substitution.

Two configuration files are shipped as examples with your screen distribution: "etc/screenrc" and "etc/etcscreenrc". They contain a number of useful examples for various commands.

Customization can also be done 'on-line'. To enter the command mode type `C-a :'. Note that commands starting with "def" change default values, while others change current settings.

The following commands are available:

acladd usernames [crypted-pw]

addacl usernames

Enable users to fully access this screen session. Usernames can be one user or a comma separated list of users. This command enables to attach to the screen session and performs the equivalent of `aclchg usernames +rwx "#?"'. executed. To add a user with restricted access, use the `aclchg' command below. If an optional second parameter is supplied, it should be a crypted password for the named user(s). `Addacl' is a synonym to `acladd'. Multi user mode only.

aclchg usernames permbits list

chacl usernames permbits list

Change permissions for a comma separated list of users. Permission bits are represented as `r', `w' and `x'. Prefixing `+' grants the permission, `-' removes it. The third parameter is

a comma separated list of commands and/or windows (specified either by number or title). The special list `#' refers to all windows, `?' to all commands. if usernames consists of a single `*', all known users are affected.

A command can be executed when the user has the `x' bit for it. The user can type input to a window when he has its `w' bit set and no other user obtains a writelock for this window. Other bits are currently ignored. To withdraw the writelock from another user in window 2: `aclchg username -w+w 2'. To allow read-only access to the session: `aclchg username -w "#"'. As soon as a user's name is known to screen he can attach to the session and (per default) has full permissions for all command and windows. Execution permission for the acl commands, `at' and others should also be removed or the user may be able to regain write permission. Rights of the special username nobody cannot be changed (see the "su" command). `Chacl' is a synonym to `aclchg'. Multi user mode only.

acldel username

Remove a user from screen's access control list. If currently attached, all the user's displays are detached from the session. He cannot attach again. Multi user mode only.

aclgrp username [groupname]

Creates groups of users that share common access rights. The name of the group is the username of the group leader. Each member of the group inherits the permissions that are granted to the group leader. That means, if a user fails an access check, another check is made for the group leader. A user is removed from all groups the special value "none" is used for groupname. If the second parameter is omitted all groups the user is in are listed.

aclumask [[users] +bits | [users] -bits...]

umask [[users] +bits | [users] -bits...]

This specifies the access other users have to windows that will be created by the caller of the command. Users may be no, one or

a comma separated list of known usernames. If no users are specified, a list of all currently known users is assumed. Bits is any combination of access control bits allowed defined with the "aclchg" command. The special username "?" predefines the access that not yet known users will be granted to any window initially. The special username "??" predefines the access that not yet known users are granted to any command. Rights of the special username nobody cannot be changed (see the "su" command). `Umask' is a synonym to `aclumask'.

activity message

When any activity occurs in a background window that is being monitored, screen displays a notification in the message line. The notification message can be re-defined by means of the "activity" command. Each occurrence of `%' in message is replaced by the number of the window in which activity has occurred, and each occurrence of `^G' is replaced by the definition for bell in your termcap (usually an audible bell). The default message is

'Activity in window %n'

Note that monitoring is off for all windows by default, but can be altered by use of the "monitor" command (C-a M).

allpartial on|off

If set to on, only the current cursor line is refreshed on window change. This affects all windows and is useful for slow terminal lines. The previous setting of full/partial refresh for each window is restored with "allpartial off". This is a global flag that immediately takes effect on all windows overriding the "partial" settings. It does not change the default redraw behavior of newly created windows.

altscreen on|off

If set to on, "alternate screen" support is enabled in virtual terminals, just like in xterm. Initial setting is `off'.

at [identifier][#|*|%] command [args ...]

Execute a command at other displays or windows as if it had been entered there. "At" changes the context (the `current window' or `current display' setting) of the command. If the first parameter describes a non-unique context, the command will be executed multiple times. If the first parameter is of the form `identifier*' then identifier is matched against user names. The command is executed once for each display of the selected user(s). If the first parameter is of the form `identifier%' identifier is matched against displays. Displays are named after the ttys they attach. The prefix `/dev/' or `/dev/tty' may be omitted from the identifier. If identifier has a `#' or nothing appended it is matched against window numbers and titles. Omitting an identifier in front of the `#', `*' or `%'-character selects all users, displays or windows because a prefix-match is performed. Note that on the affected display(s) a short message will describe what happened. Permission is checked for initiator of the "at" command, not for the owners of the affected display(s). Note that the '#' character works as a comment introducer when it is preceded by whitespace. This can be escaped by prefixing a '\'. Permission is checked for the initiator of the "at" command, not for the owners of the affected display(s).

Caveat: When matching against windows, the command is executed at least once per window. Commands that change the internal arrangement of windows (like "other") may be called again. In shared windows the command will be repeated for each attached display. Beware, when issuing toggle commands like "login"! Some commands (e.g. "process") require that a display is associated with the target windows. These commands may not work correctly under "at" looping over windows.

attrcolor attrib [attribute/color-modifier]

This command can be used to highlight attributes by changing the color of the text. If the attribute attrib is in use, the specified attribute/color modifier is also applied. If no modifier is given, the current one is deleted. See the "STRING ESCAPES" chapter for the syntax of the modifier. Screen understands two pseudo-attributes, "i" stands for high-intensity

foreground color and "I" for high-intensity background color.

Examples:

attrcolor b "R"

Change the color to bright red if bold text is to be printed.

attrcolor u "-u b"

Use blue text instead of underline.

attrcolor b ".I"

Use bright colors for bold text. Most terminal emulators do this already.

attrcolor i "+b"

Make bright colored text also bold.

autodetach on|off

Sets whether screen will automatically detach upon hangup, which saves all your running programs until they are resumed with a screen -r command. When turned off, a hangup signal will terminate screen and all the processes it contains. Autodetach is on by default.

autonuke on|off

Sets whether a clear screen sequence should nuke all the output that has not been written to the terminal. See also "obuflimit".

backtick id lifespan autorefresh cmd args...

backtick id

Program the backtick command with the numerical id id. The output of such a command is used for substitution of the "%`" string escape. The specified lifespan is the number of seconds

the output is considered valid. After this time, the command is run again if a corresponding string escape is encountered. The autorefresh parameter triggers an automatic refresh for caption and hardstatus strings after the specified number of seconds. Only the last line of output is used for substitution.

If both the lifespan and the autorefresh parameters are zero, the backtick program is expected to stay in the background and generate output once in a while. In this case, the command is executed right away and screen stores the last line of output. If a new line gets printed screen will automatically refresh the hardstatus or the captions.

The second form of the command deletes the backtick command with the numerical id id.

bce [on|off]

Change background-color-erase setting. If "bce" is set to on, all characters cleared by an erase/insert/scroll/clear operation will be displayed in the current background color. Otherwise the default background color is used.

bell_msg [message]

When a bell character is sent to a background window, screen displays a notification in the message line. The notification message can be re-defined by this command. Each occurrence of `%' in message is replaced by the number of the window to which a bell has been sent, and each occurrence of `^G' is replaced by the definition for bell in your termcap (usually an audible bell). The default message is

'Bell in window %n'

An empty message can be supplied to the "bell_msg" command to suppress output of a message line (bell_msg ""). Without parameter, the current message is shown.

bind [class] key [command [args]]

Bind a command to a key. By default, most of the commands provided by screen are bound to one or more keys as indicated in the "DEFAULT KEY BINDINGS" section, e.g. the command to create a new window is bound to "C-c" and "c". The "bind" command can be used to redefine the key bindings and to define new bindings. The key argument is either a single character, a two-character sequence of the form "^x" (meaning "C-x"), a backslash followed by an octal number (specifying the ASCII code of the character), or a backslash followed by a second character, such as "\^" or "\\". The argument can also be quoted, if you like. If no further argument is given, any previously established binding for this key is removed. The command argument can be any command listed in this section.

If a command class is specified via the "-c" option, the key is bound for the specified class. Use the "command" command to activate a class. Command classes can be used to create multiple command keys or multi-character bindings.

Some examples:

```
bind ' ' windows
bind ^k
bind k
bind K kill
bind ^f screen telnet foobar
bind \033 screen -ln -t root -h 1000 9 su
```

would bind the space key to the command that displays a list of windows (so that the command usually invoked by "C-a C-w" would also be available as "C-a space"). The next three lines remove the default kill binding from "C-a C-k" and "C-a k". "C-a K" is then bound to the kill command. Then it binds "C-f" to the command "create a window with a TELNET connection to foobar", and bind "escape" to the command that creates an non-login window with a.k.a. "root" in slot #9, with a superuser shell and a scrollback buffer of 1000 lines.

```
bind -c demo1 0 select 10
bind -c demo1 1 select 11
bind -c demo1 2 select 12
```

```
bindkey "^B" command -c demo1
```

makes "C-b 0" select window 10, "C-b 1" window 11, etc.

```
bind -c demo2 0 select 10
bind -c demo2 1 select 11
bind -c demo2 2 select 12
bind - command -c demo2
```

makes "C-a - 0" select window 10, "C-a - 1" window 11, etc.

bindkey [-d] [-m] [-a] [[-k|-t] string [cmd-args]]

This command manages screen's input translation tables. Every entry in one of the tables tells screen how to react if a certain sequence of characters is encountered. There are three tables: one that should contain actions programmed by the user, one for the default actions used for terminal emulation and one for screen's copy mode to do cursor movement. See section "INPUT TRANSLATION" for a list of default key bindings.

If the -d option is given, bindkey modifies the default table, -m changes the copy mode table and with neither option the user table is selected. The argument string is the sequence of characters to which an action is bound. This can either be a fixed string or a termcap keyboard capability name (selectable with the -k option).

Some keys on a VT100 terminal can send a different string if application mode is turned on (e.g the cursor keys). Such keys have two entries in the translation table. You can select the application mode entry by specifying the -a option.

The -t option tells screen not to do inter-character timing. One cannot turn off the timing if a termcap capability is used.

Cmd can be any of screen's commands with an arbitrary number of args. If cmd is omitted the key-binding is removed from the table.

Here are some examples of keyboard bindings:

```
        bindkey -d

Show all of the default key bindings. The application mode
entries are marked with [A].

        bindkey -k k1 select 1

Make the "F1" key switch to window one.

        bindkey -t foo stuff barfoo

Make "foo" an abbreviation of the word "barfoo". Timeout is
disabled so that users can type slowly.

        bindkey "\024" mapdefault

This key-binding makes "^T" an escape character for key-bindings.
If you did the above "stuff barfoo" binding, you can enter the
word "foo" by typing "^Tfoo". If you want to insert a "^T" you
have to press the key twice (i.e., escape the escape binding).

        bindkey -k F1 command

Make the F11 (not F1!) key an alternative screen escape (besides
^A).

break[duration]

Send a break signal for duration*0.25 seconds to this window.
For non-Posix systems the time interval may be rounded up to full
seconds.  Most useful if a character device is attached to the
window rather than a shell process (See also chapter "WINDOW
TYPES"). The maximum duration of a break signal is limited to 15
seconds.

blanker

Activate the screen blanker. First the screen is cleared. If no
blanker program is defined, the cursor is turned off, otherwise,
the program is started and it's output is written to the screen.
```

The screen blanker is killed with the first keypress, the read key is discarded.

This command is normally used together with the "idle" command.

blankerprg [program-args]

Defines a blanker program. Disables the blanker program if an empty argument is given. Shows the currently set blanker program if no arguments are given.

breaktype [tcsendbreak|TIOCSBRK|TCSBRK]

Choose one of the available methods of generating a break signal for terminal devices. This command should affect the current window only. But it still behaves identical to "defbreaktype". This will be changed in the future. Calling "breaktype" with no parameter displays the break method for the current window.

bufferfile [exchange-file]

Change the filename used for reading and writing with the paste buffer. If the optional argument to the "bufferfile" command is omitted, the default setting ("/tmp/screen-exchange") is reactivated. The following example will paste the system's password file into the screen window (using the paste buffer, where a copy remains):

```
C-a : bufferfile /etc/passwd
C-a < C-a ]
C-a : bufferfile
```

bumpleft

Swaps window with previous one on window list.

bumpright

Swaps window with next one on window list.

c1 [on|off]

Change c1 code processing. "C1 on" tells screen to treat the input characters between 128 and 159 as control functions. Such an 8-bit code is normally the same as ESC followed by the corresponding 7-bit code. The default setting is to process c1 codes and can be changed with the "defc1" command. Users with fonts that have usable characters in the c1 positions may want to turn this off.

caption [top | bottom] always|splitonly[string]

caption string [string]

This command controls the display of the window captions. Normally a caption is only used if more than one window is shown on the display (split screen mode). But if the type is set to always screen shows a caption even if only one window is displayed. The default is splitonly.

The second form changes the text used for the caption. You can use all escapes from the "STRING ESCAPES" chapter. Screen uses a default of `%3n %t'.

You can mix both forms by providing a string as an additional argument.

You can have the caption displayed either at the top or bottom of the window. The default is bottom.

charset set

Change the current character set slot designation and charset mapping. The first four character of set are treated as charset designators while the fifth and sixth character must be in range '0' to '3' and set the GL/GR charset mapping. On every position a '.' may be used to indicate that the corresponding charset/mapping should not be changed (set is padded to six characters internally by appending '.' chars). New windows have "BBBB02" as default charset, unless a "encoding" command is active.
The current setting can be viewed with the "info" command.

chdir [directory]

Change the current directory of screen to the specified directory or, if called without an argument, to your home directory (the value of the environment variable $HOME). All windows that are created by means of the "screen" command from within ".screenrc" or by means of "C-a : screen ..." or "C-a c" use this as their default directory. Without a chdir command, this would be the directory from which screen was invoked.

Hardcopy and log files are always written to the window's default directory, not the current directory of the process running in the window. You can use this command multiple times in your .screenrc to start various windows in different default directories, but the last chdir value will affect all the windows you create interactively.

cjkwidth [on | off]

Treat ambiguous width characters as full/half width.

clear

Clears the current window and saves its image to the scrollback buffer.

collapse

Reorders window on window list, removing number gaps between them.

colon [prefix]

Allows you to enter ".screenrc" command lines. Useful for on-the-fly modification of key bindings, specific window creation and changing settings. Note that the "set" keyword no longer exists! Usually commands affect the current window rather than default settings for future windows. Change defaults with commands starting with 'def...'.

If you consider this as the `Ex command mode' of screen, you may regard "C-a esc" (copy mode) as its `Vi command mode'.

command [-c class]

This command has the same effect as typing the screen escape character (^A). It is probably only useful for key bindings. If the "-c" option is given, select the specified command class. See also "bind" and "bindkey".

compacthist [on|off]

This tells screen whether to suppress trailing blank lines when scrolling up text into the history buffer.

console [on|off]

Grabs or un-grabs the machines console output to a window. Note: Only the owner of /dev/console can grab the console output. This command is only available if the machine supports the ioctl TIOCCONS.

copy

Enter copy/scrollback mode. This allows you to copy text from the current window and its history into the paste buffer. In this mode a vi-like `full screen editor' is active:
The editor's movement keys are:

---

h, C-h, move the cursor left.
left arrow

---

j, C-n, move the cursor down.
down arrow

---

k, C-p, move the cursor up.
up arrow

---

l ('el'), move the cursor right.

right arrow	
0 (zero) C-a	move to the leftmost column.
+ and -	positions one line up and down.
H, M and L	move the cursor to the leftmost column of the top, center or bottom line of the window.
|	moves to the specified absolute column.
g or home	moves to the beginning of the buffer.
G or end	moves to the specified absolute line (default: end of buffer).
%	jumps to the specified percentage of the buffer.
^ or $	move to the leftmost column, to the first or last non-whitespace character on the line.
w, b, and e	move the cursor word by word.
B, E	move the cursor WORD by WORD (as in vi).
f/F, t/T	move the cursor forward/backward to the next occurence of the target. (eg, '3fy' will move the cursor to the 3rd 'y' to the right.)
; and ,	Repeat the last f/F/t/T command in the same/opposite direction.
C-e and C-y	scroll the display up/down by one line while preserving the cursor position.
C-u and C-d	scroll the display up/down by the specified amount of lines while preserving the cursor position. (Default: half screen-full).
C-b and C-f	scroll the display up/down a full screen.

Note: Emacs style movement keys can be customized by a .screenrc command. (E.g. markkeys "h=^B:l=^F:$=^E") There is no simple method for a full emacs-style keymap, as this involves multi-character codes.

Some keys are defined to do mark and replace operations.

The copy range is specified by setting two marks. The text between these marks will be highlighted. Press:

space or enter to set the first or second mark respectively. If mousetrack is set to `on', marks can also be set using left mouse click.

Y and y used to mark one whole line or to mark from start of line.

W marks exactly one word.

Any of these commands can be prefixed with a repeat count number by pressing digits

0..9 which is taken as a repeat count.

Example: "C-a C-[H 10 j 5 Y" will copy lines 11 to 15 into the paste buffer.

The folllowing search keys are defined:

/ Vi-like search forward.

? Vi-like search backward.

C-a s Emacs style incremental search forward.

C-r Emacs style reverse i-search.

n Find next search pattern.

N Find previous search pattern.

There are however some keys that act differently than in vi. Vi does not allow one to yank rectangular blocks of text, but screen does. Press: c or C to set the left or right margin respectively. If no repeat count is given, both default to the current cursor position.

Example: Try this on a rather full text screen:

"C-a [M 20 l SPACE c 10 l 5 j C SPACE".

This moves one to the middle line of the screen, moves in 20 columns left, marks the beginning of the paste buffer, sets the left column, moves 5 columns down, sets the right column, and then marks the end of the paste buffer. Now try:

"C-a [M 20 l SPACE 10 l 5 j SPACE"

and notice the difference in the amount of text copied.

J joins lines. It toggles between 4 modes: lines separated by a newline character (012), lines glued seamless, lines separated by a single whitespace and comma separated lines. Note that you can prepend the newline character with a carriage return character, by issuing a "crlf on".

v or V is for all the vi users with ":set numbers" - it toggles the left margin between column 9 and 1. Press

a before the final space key to toggle in append mode. Thus the contents of the paste buffer will not be overwritten, but is appended to.

A toggles in append mode and sets a (second) mark.

> sets the (second) mark and writes the contents of the paste buffer to the screen-exchange file (/tmp/screen-exchange per default) once copy-mode is finished.

This example demonstrates how to dump the whole scrollback buffer to that file: "C-A [g SPACE G $ >".

C-g gives information about the current line and column.

x or o exchanges the first mark and the current cursor position. You can use this to adjust an already placed mark.

C-l ('el') will redraw the screen.

@ does nothing. Does not even exit copy mode.

All keys not described here exit copy mode.

copy_reg [key]

No longer exists, use "readreg" instead.

crlf [on|off]

This affects the copying of text regions with the `C-a [' command. If it is set to `on', lines will be separated by the two character sequence `CR' - `LF'. Otherwise (default) only `LF' is used. When no parameter is given, the state is toggled.

defc1 on|off

Same as the c1 command except that the default setting for new windows is changed. Initial setting is `on'.

defautonuke on|off

Same as the autonuke command except that the default setting for new displays is changed. Initial setting is `off'. Note that you can use the special `AN' terminal capability if you want to have a dependency on the terminal type.

defbce on|off

Same as the bce command except that the default setting for new windows is changed. Initial setting is `off'.

defbreaktype [tcsendbreak|TIOCSBRK|TCSBRK]

Choose one of the available methods of generating a break signal for terminal devices. The preferred methods are tcsendbreak and TIOCSBRK. The third, TCSBRK, blocks the complete screen session for the duration of the break, but it may be the only way to generate long breaks. Tcsendbreak and TIOCSBRK may or may not produce long breaks with spikes (e.g. 4 per second). This is not only system-dependent, this also differs between serial board drivers. Calling "defbreaktype" with no parameter displays the current setting.

defcharset [set]

Like the charset command except that the default setting for new windows is changed. Shows current default if called without argument.

defdynamictitle on|off

Set default behaviour for new windows regarding if screen should change window title when seeing proper escape sequence. See also "TITLES (naming windows)" section.

defescape xy

Set the default command characters. This is equivalent to the "escape" except that it is useful multiuser sessions only. In a multiuser session "escape" changes the command character of the calling user, where "defescape" changes the default command characters for users that will be added later.

defflow on|off|auto [interrupt]

Same as the flow command except that the default setting for new windows is changed. Initial setting is `auto'. Specifying "defflow auto interrupt" is the same as the command-line options -fa and -i.

defgr on|off

Same as the gr command except that the default setting for new

windows is changed. Initial setting is `off'.

defhstatus [status]

The hardstatus line that all new windows will get is set to status. This command is useful to make the hardstatus of every window display the window number or title or the like. Status may contain the same directives as in the window messages, but the directive escape character is '^E' (octal 005) instead of '%'. This was done to make a misinterpretation of program generated hardstatus lines impossible. If the parameter status is omitted, the current default string is displayed. Per default the hardstatus line of new windows is empty.

defencoding enc

Same as the encoding command except that the default setting for new windows is changed. Initial setting is the encoding taken from the terminal.

deflog on|off

Same as the log command except that the default setting for new windows is changed. Initial setting is `off'.

deflogin on|off

Same as the login command except that the default setting for new windows is changed. This is initialized with `on' as distributed (see config.h.in).

defmode mode

The mode of each newly allocated pseudo-tty is set to mode. Mode is an octal number. When no "defmode" command is given, mode 0622 is used.

defmonitor on|off

Same as the monitor command except that the default setting for new windows is changed. Initial setting is `off'.

defmousetrack on|off

Same as the mousetrack command except that the default setting for new windows is changed. Initial setting is `off'.

defnonblock on|off|numsecs

Same as the nonblock command except that the default setting for displays is changed. Initial setting is `off'.

defobuflimit limit

Same as the obuflimit command except that the default setting for new displays is changed. Initial setting is 256 bytes. Note that you can use the special 'OL' terminal capability if you want to have a dependency on the terminal type.

defscrollback num

Same as the scrollback command except that the default setting for new windows is changed. Initial setting is 100.

defshell command

Synonym to the shell .screenrc command. See there.

defsilence on|off

Same as the silence command except that the default setting for new windows is changed. Initial setting is `off'.

defslowpaste msec

Same as the slowpaste command except that the default setting for new windows is changed. Initial setting is 0 milliseconds, meaning `off'.

defutf8 on|off

Same as the utf8 command except that the default setting for new

windows is changed. Initial setting is `on' if screen was started with "-U", otherwise `off'.

defwrap on|off

Same as the wrap command except that the default setting for new windows is changed. Initially line-wrap is on and can be toggled with the "wrap" command ("C-a r") or by means of "C-a : wrap on|off".

defwritelock on|off|auto

Same as the writelock command except that the default setting for new windows is changed. Initially writelocks will off.

detach [-h]

Detach the screen session (disconnect it from the terminal and put it into the background). This returns you to the shell where you invoked screen. A detached screen can be resumed by invoking screen with the -r option (see also section "COMMAND-LINE OPTIONS"). The -h option tells screen to immediately close the connection to the terminal ("hangup").

dinfo

Show what screen thinks about your terminal. Useful if you want to know why features like color or the alternate charset don't work.

displays

Shows a tabular listing of all currently connected user front-ends (displays). This is most useful for multiuser sessions. The following keys can be used in displays list:

k, C-p, or up	Move up one line.
j, C-n, or down	Move down one line.

C-a or home	Move to the first line.
C-e or end	Move to the last line.
C-u or C-d	Move one half page up or down.
C-b or C-f	Move one full page up or down.
mouseclick	Move to the selected line. Available when "mousetrack" is set to on.
space	Refresh the list
d	Detach that display
D	Power detach that display
C-g, enter, or escape	Exit the list

The following is an example of what "displays" could look like:

```
       xterm 80x42 jnweiger@/dev/ttyp4     0(m11)   &rWx
       facit 80x24 mlschroe@/dev/ttyhf nb 11(tcsh)   rwx
       xterm 80x42 jnhollma@/dev/ttyp5     0(m11)   &R.x
        (A)   (B)     (C)     (D)     (E) (F)(G)   (H)(I)
```

The legend is as follows:

(A) The terminal type known by screen for this display.

(B) Displays geometry as width x height.

(C) Username who is logged in at the display.

(D) Device name of the display or the attached device

(E) Display is in blocking or nonblocking mode. The available modes are "nb", "NB", "Z<", "Z>", and "BL".

(F) Number of the window

(G) Name/title of window

(H) Whether the window is shared

(I) Window permissions. Made up of three characters.

```
┌-----------------------------------------------------┐
|             Window permissions indicators           |

↪  ├-----------------┬------------------┬------------------┤
| 1st character    |  2nd character    |   3rd character
↪  |

↪  ├----┬------------┼-----┬------------┼-----┬-------------┤
|-   |no read     | -   |no write    | -   |no execute
↪  |

↪  ├----┼------------┼-----┼------------┼-----┼-------------┤
|r   |read        | w   |write       | x   |execute
↪  |

↪  ├----┼------------┼-----┼------------┼-----┼-------------┤
|    |            | W   |own wlock   |     |
↪  |

↪  ├----┴------------┴-----┴------------┴-----┴-------------┤
|Indicators of permissions suppressed by a foreign wlock  |

↪  ├----┬------------┬-----┬------------┬-----┬-------------┤
|R   |read only   | .   |no write    |     |
↪  |

↪  └----┴------------┴-----┴------------┴-----┴-------------┘
"displays" needs a region size of at least 10 characters
wide and 5 characters high in order to display.
```

digraph [preset[unicode-value]]

This command prompts the user for a digraph sequence. The next two characters typed are looked up in a builtin table and the

resulting character is inserted in the input stream. For example, if the user enters 'a"', an a-umlaut will be inserted. If the first character entered is a 0 (zero), screen will treat the following characters (up to three) as an octal number instead. The optional argument preset is treated as user input, thus one can create an "umlaut" key. For example the command "bindkey ^K digraph '"'" enables the user to generate an a-umlaut by typing CTRL-K a. When a non-zero unicode-value is specified, a new digraph is created with the specified preset. The digraph is unset if a zero value is provided for the unicode-value.

dumptermcap

Write the termcap entry for the virtual terminal optimized for the currently active window to the file ".termcap" in the user's "$HOME/.screen" directory (or wherever screen stores its sockets. See the "FILES" section below). This termcap entry is identical to the value of the environment variable $TERMCAP that is set up by screen for each window. For terminfo based systems you will need to run a converter like captoinfo and then compile the entry with tic.

dynamictitle on|off

Change behaviour for windows regarding if screen should change window title when seeing proper escape sequence. See also "TITLES (naming windows)" section.

echo [-n] message

The echo command may be used to annoy screen users with a 'message of the day'. Typically installed in a global /local/etc/screenrc. The option "-n" may be used to suppress the line feed. See also "sleep". Echo is also useful for online checking of environment variables.

encoding enc [enc]

Tell screen how to interpret the input/output. The first argument sets the encoding of the current window. Each window can emulate a different encoding. The optional second parameter overwrites

the encoding of the connected terminal. It should never be needed as screen uses the locale setting to detect the encoding. There is also a way to select a terminal encoding depending on the terminal type by using the "KJ" termcap entry.

Supported encodings are eucJP, SJIS, eucKR, eucCN, Big5, GBK, KOI8-R, KOI8-U, CP1251, UTF-8, ISO8859-2, ISO8859-3, ISO8859-4, ISO8859-5, ISO8859-6, ISO8859-7, ISO8859-8, ISO8859-9, ISO8859-10, ISO8859-15, jis.

See also "defencoding", which changes the default setting of a new window.

escape xy

Set the command character to x and the character generating a literal command character (by triggering the "meta" command) to y (similar to the -e option). Each argument is either a single character, a two-character sequence of the form "^x" (meaning "C-x"), a backslash followed by an octal number (specifying the ASCII code of the character), or a backslash followed by a second character, such as "\^" or "\\". The default is "^Aa".

eval command1[command2 ...]

Parses and executes each argument as separate command.

exec [[fdpat]newcommand [args ...]]

Run a unix subprocess (specified by an executable path newcommand and its optional arguments) in the current window. The flow of data between newcommands stdin/stdout/stderr, the process originally started in the window (let us call it "application-process") and screen itself (window) is controlled by the file descriptor pattern fdpat. This pattern is basically a three character sequence representing stdin, stdout and stderr of newcommand. A dot (.) connects the file descriptor to screen. An exclamation mark (!) causes the file descriptor to be connected to the application-process. A colon (:) combines both. User input will go to newcommand unless newcommand receives the application-process' output (fdpats first character is `!' or

`:') or a pipe symbol (|) is added (as a fourth character) to the end of fdpat.

Invoking `exec' without arguments shows name and arguments of the currently running subprocess in this window. Only one subprocess a time can be running in each window.

When a subprocess is running the `kill' command will affect it instead of the windows process.

Refer to the postscript file `doc/fdpat.ps' for a confusing illustration of all 21 possible combinations. Each drawing shows the digits 2,1,0 representing the three file descriptors of newcommand. The box marked `W' is the usual pty that has the application-process on its slave side. The box marked `P' is the secondary pty that now has screen at its master side.

Abbreviations: Whitespace between the word `exec' and fdpat and the command can be omitted. Trailing dots and a fdpat consisting only of dots can be omitted. A simple `|' is synonymous for the pattern `!..|'; the word exec can be omitted here and can always be replaced by `!'.

Examples:

exec ... /bin/sh

exec /bin/sh

!/bin/sh

Creates another shell in the same window, while the original shell is still running. Output of both shells is displayed and user input is sent to the new /bin/sh.

exec !.. stty 19200

exec ! stty 19200

!!stty 19200

Set the speed of the window's tty. If your stty command operates on stdout, then add another `!'.

exec !..| less

|less

This adds a pager to the window output. The special character `|' is needed to give the user control over the pager although it gets its input from the window's process. This works, because less listens on stderr (a behavior that screen would not expect without the `|') when its stdin is not a tty. Less versions newer than 177 fail miserably here; good old pg still works.

!:sed -n s/.*Error.*/\007/p

Sends window output to both, the user and the sed command. The sed inserts an additional bell character (oct. 007) to the window output seen by screen. This will cause "Bell in window x" messages, whenever the string "Error" appears in the window.

fit

Change the window size to the size of the current region. This command is needed because screen doesn't adapt the window size automatically if the window is displayed more than once.

flow [on|off|auto]

Sets the flow-control mode for this window. Without parameters it cycles the current window's flow-control setting from "automatic" to "on" to "off". See the discussion on "FLOW-CONTROL" later on in this document for full details and note, that this is subject to change in future releases. Default is set by `defflow'.

```
focus [next|prev|up|down|left|right|top|bottom]

Move the input focus to the next region. This is done in a cyclic
way so that the top left region is selected after the bottom
right one. If no option is given it defaults to `next'. The next
region to be selected is determined by how the regions are
layered.  Normally, the next region in the same layer would be
selected.  However, if that next region contains one or more
layers, the first region in the highest layer is selected first.
If you are at the last region of the current layer, `next' will
move the focus to the next region in the lower layer (if there is
a lower layer).  `Prev' cycles in the opposite order. See "split"
for more information about layers.

The rest of the options (`up', `down', `left', `right', `top',
and `bottom') are more indifferent to layers. The option `up'
will move the focus upward to the region that is touching the
upper left corner of the current region.  `Down' will move
downward to the region that is touching the lower left corner of
the current region. The option `left' will move the focus
leftward to the region that is touching the upper left corner of
the current region, while `right' will move rightward to the
region that is touching the upper right corner of the current
region. Moving left from a left most region or moving right from
a right most region will result in no action.

The option `top' will move the focus to the very first region in
the upper list corner of the screen, and `bottom' will move to
the region in the bottom right corner of the screen. Moving up
from a top most region or moving down from a bottom most region
will result in no action.

Useful bindings are (h, j, k, and l as in vi)
    bind h focus left
    bind j focus down
    bind k focus up
    bind l focus right
    bind t focus top
    bind b focus bottom
Note that k is traditionally bound to the kill command.
```

focusminsize [(width|max|_) (height|max|_)]

This forces any currently selected region to be automatically resized at least a certain width and height. All other surrounding regions will be resized in order to accommodate. This constraint follows everytime the "focus" command is used. The "resize" command can be used to increase either dimension of a region, but never below what is set with "focusminsize". The underscore `_' is a synonym for max. Setting a width and height of `0 0' (zero zero) will undo any constraints and allow for manual resizing. Without any parameters, the minimum width and height is shown.

gr [on|off]

Turn GR charset switching on/off. Whenever screen sees an input character with the 8th bit set, it will use the charset stored in the GR slot and print the character with the 8th bit stripped. The default (see also "defgr") is not to process GR switching because otherwise the ISO88591 charset would not work.

group [grouptitle]

Change or show the group the current window belongs to. Windows can be moved around between different groups by specifying the name of the destination group. Without specifying a group, the title of the current group is displayed.

hardcopy [-h] [file]

Writes out the currently displayed image to the file file, or, if no filename is specified, to hardcopy.n in the default directory, where n is the number of the current window. This either appends or overwrites the file if it exists. See below. If the option -h is specified, dump also the contents of the scrollback buffer.

hardcopy_append on|off

If set to "on", screen will append to the "hardcopy.n" files created by the command "C-a h", otherwise these files are overwritten each time. Default is `off'.

hardcopydir directory

Defines a directory where hardcopy files will be placed. If unset, hardcopys are dumped in screen's current working directory.

hardstatus [on|off]

hardstatus [always]firstline|lastline|message|ignore[string]

hardstatus string[string]

This command configures the use and emulation of the terminal's hardstatus line. The first form toggles whether screen will use the hardware status line to display messages. If the flag is set to `off', these messages are overlaid in reverse video mode at the display line. The default setting is `on'.

The second form tells screen what to do if the terminal doesn't have a hardstatus line (i.e. the termcap/terminfo capabilities "hs", "ts", "fs" and "ds" are not set). When "firstline/lastline" is used, screen will reserve the first/last line of the display for the hardstatus. "message" uses screen's message mechanism and "ignore" tells screen never to display the hardstatus. If you prepend the word "always" to the type (e.g., "alwayslastline"), screen will use the type even if the terminal supports a hardstatus.

The third form specifies the contents of the hardstatus line. '%h' is used as default string, i.e., the stored hardstatus of the current window (settable via "ESC]0;<string>^G" or "ESC_<string>ESC\") is displayed. You can customize this to any string you like including the escapes from the "STRING ESCAPES" chapter. If you leave out the argument string, the current string is displayed.

You can mix the second and third form by providing the string as additional argument.

height [-w|-d] [lines [cols]]

Set the display height to a specified number of lines. When no argument is given it toggles between 24 and 42 lines display. You can also specify a width if you want to change both values. The -w option tells screen to leave the display size unchanged and just set the window size, -d vice versa.

help[class]

Not really a online help, but displays a help screen showing you all the key bindings. The first pages list all the internal commands followed by their current bindings. Subsequent pages will display the custom commands, one command per key. Press space when you're done reading each page, or return to exit early. All other characters are ignored. If the "-c" option is given, display all bound commands for the specified command class. See also "DEFAULT KEY BINDINGS" section.

history

Usually users work with a shell that allows easy access to previous commands. For example csh has the command "!!" to repeat the last command executed. Screen allows you to have a primitive way of re-calling "the command that started ...": You just type the first letter of that command, then hit `C-a {' and screen tries to find a previous line that matches with the `prompt character' to the left of the cursor. This line is pasted into this window's input queue. Thus you have a crude command history (made up by the visible window and its scrollback buffer).

hstatus status

Change the window's hardstatus line to the string status.

idle [timeout[cmd-args]]

Sets a command that is run after the specified number of seconds inactivity is reached. This command will normally be the "blanker" command to create a screen blanker, but it can be any screen command. If no command is specified, only the timeout is

set. A timeout of zero (or the special timeout off) disables the timer. If no arguments are given, the current settings are displayed.

ignorecase [on|off]

Tell screen to ignore the case of characters in searches. Default is `off'. Without any options, the state of ignorecase is toggled.

info

Uses the message line to display some information about the current window: the cursor position in the form "(column,row)" starting with "(1,1)", the terminal width and height plus the size of the scrollback buffer in lines, like in "(80,24)+50", the current state of window XON/XOFF flow control is shown like this (See also section FLOW CONTROL):

+flow	automatic flow control, currently on.
-flow	automatic flow control, currently off.
+(+)flow	flow control enabled. Agrees with automatic control.
-(+)flow	flow control disabled. Disagrees with automatic control.
+(-)flow	flow control enabled. Disagrees with automatic control.

```
| -(-)flow  |  flow control disabled. Agrees with automatic control.
↪   |
↪   └---------┴-------------------------------------------------------------┘
```

The current line wrap setting (`+wrap' indicates enabled, `-wrap' not) is also shown. The flags `ins', `org', `app', `log', `mon' or `nored' are displayed when the window is in insert mode, origin mode, application-keypad mode, has output logging, activity monitoring or partial redraw enabled.

The currently active character set (G0, G1, G2, or G3) and in square brackets the terminal character sets that are currently designated as G0 through G3 is shown. If the window is in UTF-8 mode, the string "UTF-8" is shown instead.

Additional modes depending on the type of the window are displayed at the end of the status line (See also chapter "WINDOW TYPES").

If the state machine of the terminal emulator is in a non-default state, the info line is started with a string identifying the current state.

For system information use the "time" command.

ins_reg [key]

No longer exists, use "paste" instead.

kill

Kill current window.

If there is an `exec' command running then it is killed. Otherwise the process (shell) running in the window receives a HANGUP condition, the window structure is removed and screen (your display) switches to another window. When the last window is destroyed, screen exits. After a kill screen switches to the previously displayed window.

Note: Emacs users should keep this command in mind, when killing

a line. It is recommended not to use "C-a" as the screen escape key or to rebind kill to "C-a K".

lastmsg

Redisplay the last contents of the message/status line. Useful if you're typing when a message appears, because the message goes away when you press a key (unless your terminal has a hardware status line). Refer to the commands "msgwait" and "msgminwait" for fine tuning.

layout new [title]

Create a new layout. The screen will change to one whole region and be switched to the blank window. From here, you build the regions and the windows they show as you desire. The new layout will be numbered with the smallest available integer, starting with zero. You can optionally give a title to your new layout. Otherwise, it will have a default title of "layout". You can always change the title later by using the command layout title.

layout remove [n|title]

Remove, or in other words, delete the specified layout. Either the number or the title can be specified. Without either specification, screen will remove the current layout.

Removing a layout does not affect your set windows or regions.

layout next

Switch to the next layout available

layout prev

Switch to the previous layout available

layout select [n|title]

Select the desired layout. Either the number or the title can be specified. Without either specification, screen will prompt and

ask which screen is desired. To see which layouts are available, use the layout show command.

layout show

List on the message line the number(s) and title(s) of the available layout(s). The current layout is flagged.

layout title [title]

Change or display the title of the current layout. A string given will be used to name the layout. Without any options, the current title and number is displayed on the message line.

layout number [n]

Change or display the number of the current layout. An integer given will be used to number the layout. Without any options, the current number and title is displayed on the message line.

layout attach [title|:last]

Change or display which layout to reattach back to. The default is :last, which tells screen to reattach back to the last used layout just before detachment. By supplying a title, You can instruct screen to reattach to a particular layout regardless which one was used at the time of detachment. Without any options, the layout to reattach to will be shown in the message line.

layout save [n|title]

Remember the current arrangement of regions. When used, screen will remember the arrangement of vertically and horizontally split regions. This arrangement is restored when a screen session is reattached or switched back from a different layout. If the session ends or the screen process dies, the layout arrangements are lost. The layout dump command should help in this siutation. If a number or title is supplied, screen will remember the arrangement of that particular layout. Without any options, screen will remember the current layout.

Saving your regions can be done automatically by using the layout autosave command.

layout autosave [on|off]

Change or display the status of automatcally saving layouts. The default is on, meaning when screen is detached or changed to a different layout, the arrangement of regions and windows will be remembered at the time of change and restored upon return. If autosave is set to off, that arrangement will only be restored to either to the last manual save, using layout save, or to when the layout was first created, to a single region with a single window. Without either an on or off, the current status is displayed on the message line.

layout dump [filename]

Write to a file the order of splits made in the current layout. This is useful to recreate the order of your regions used in your current layout. Only the current layout is recorded. While the order of the regions are recorded, the sizes of those regions and which windows correspond to which regions are not. If no filename is specified, the default is layout-dump, saved in the directory that the screen process was started in. If the file already exists, layout dump will append to that file. As an example:

C-a : layout dump /home/user/.screenrc

will save or append the layout to the user's .screenrc file.

license

Display the disclaimer page. This is done whenever screen is started without options, which should be often enough. See also the "startup_message" command.

lockscreen

Lock this display. Call a screenlock program (/local/bin/lck or /usr/bin/lock or a builtin if no other is available). Screen does

not accept any command keys until this program terminates. Meanwhile processes in the windows may continue, as the windows are in the `detached' state. The screenlock program may be changed through the environment variable $LOCKPRG (which must be set in the shell from which screen is started) and is executed with the user's uid and gid.

Warning: When you leave other shells unlocked and you have no password set on screen, the lock is void: One could easily re-attach from an unlocked shell. This feature should rather be called `lockterminal'.

log [on|off]

Start/stop writing output of the current window to a file "screenlog.n" in the window's default directory, where n is the number of the current window. This filename can be changed with the `logfile' command. If no parameter is given, the state of logging is toggled. The session log is appended to the previous contents of the file if it already exists. The current contents and the contents of the scrollback history are not included in the session log.  Default is `off'.

logfile filename

logfile flush secs

Defines the name the log files will get. The default is "screenlog.%n". The second form changes the number of seconds screen will wait before flushing the logfile buffer to the file-system. The default value is 10 seconds.

login [on|off]

Adds or removes the entry in the utmp database file for the current window. This controls if the window is `logged in'. When no parameter is given, the login state of the window is toggled.  Additionally to that toggle, it is convenient having a `log in' and a `log out' key. E.g. `bind I login on' and `bind O login off' will map these keys to be C-a I and C-a O. The default setting (in config.h.in) should be "on" for a screen that

runs under suid-root. Use the "deflogin" command to change the default login state for new windows. Both commands are only present when screen has been compiled with utmp support.

logtstamp [on|off]

logtstamp after [secs]

logtstamp string
[string]

This command controls logfile time-stamp mechanism of screen. If time-stamps are turned "on", screen adds a string containing the current time to the logfile after two minutes of inactivity. When output continues and more than another two minutes have passed, a second time-stamp is added to document the restart of the output. You can change this timeout with the second form of the command. The third form is used for customizing the time-stamp string (`-- %n:%t -- time-stamp -- %M/%d/%y %c:%s --\n' by default).

mapdefault

Tell screen that the next input character should only be looked up in the default bindkey table. See also "bindkey".

mapnotnext

Like mapdefault, but don't even look in the default bindkey table.

maptimeout [timeout]

Set the inter-character timer for input sequence detection to a timeout of timeout ms. The default timeout is 300ms. Maptimeout with no arguments shows the current setting. See also "bindkey".

markkeys string

This is a method of changing the keymap used for copy/history mode. The string is made up of oldchar=newchar pairs which are

separated by `:'. Example: The string "B=^B:F=^F" will change the keys `C-b' and `C-f' to the vi style binding (scroll up/down fill page). This happens to be the default binding for `B' and `F'. The command "markkeys h=^B:l=^F:$=^E" would set the mode for an emacs-style binding. If your terminal sends characters, that cause you to abort copy mode, then this command may help by binding these characters to do nothing. The no-op character is `@' and is used like this: "markkeys @=L=H" if you do not want to use the `H' or `L' commands any longer. As shown in this example, multiple keys can be assigned to one function in a single statement.

maxwin num

Set the maximum window number screen will create. Doesn't affect already existing windows. The number can be increased only when there are no existing windows.

meta

Insert the command character (C-a) in the current window's input stream.

monitor [on|off]

Toggles activity monitoring of windows. When monitoring is turned on and an affected window is switched into the background, you will receive the activity notification message in the status line at the first sign of output and the window will also be marked with an `@' in the window-status display. Monitoring is initially off for all windows.

mousetrack [on|off]

This command determines whether screen will watch for mouse clicks. When this command is enabled, regions that have been split in various ways can be selected by pointing to them with a mouse and left-clicking them. Without specifying on or off, the current state is displayed. The default state is determined by the "defmousetrack" command.

msgminwait sec

Defines the time screen delays a new message when one message is currently displayed. The default is 1 second.

msgwait sec

Defines the time a message is displayed if screen is not disturbed by other activity. The default is 5 seconds.

multiuser on|off

Switch between singleuser and multiuser mode. Standard screen operation is singleuser. In multiuser mode the commands `acladd', `aclchg', `aclgrp' and `acldel' can be used to enable (and disable) other users accessing this screen session.

nethack on|off

Changes the kind of error messages used by screen. When you are familiar with the game "nethack", you may enjoy the nethack-style messages which will often blur the facts a little, but are much funnier to read. Anyway, standard messages often tend to be unclear as well.
This option is only available if screen was compiled with the NETHACK flag defined. The default setting is then determined by the presence of the environment variable $NETHACKOPTIONS and the file ~/.nethackrc - if either one is present, the default is on.

next

Switch to the next window. This command can be used repeatedly to cycle through the list of windows.

nonblock [on|off|numsecs]

Tell screen how to deal with user interfaces (displays) that cease to accept output. This can happen if a user presses ^S or a TCP/modem connection gets cut but no hangup is received. If nonblock is off (this is the default) screen waits until the display restarts to accept the output. If nonblock is on, screen

waits until the timeout is reached (on is treated as 1s). If the display still doesn't receive characters, screen will consider it "blocked" and stop sending characters to it. If at some time it restarts to accept characters, screen will unblock the display and redisplay the updated window contents.

number [[+|-]n]

Change the current window's number. If the given number n is already used by another window, both windows exchange their numbers. If no argument is specified, the current window number (and title) is shown. Using `+' or `-' will change the window's number by the relative amount specified.

obuflimit [limit]

If the output buffer contains more bytes than the specified limit, no more data will be read from the windows. The default value is 256. If you have a fast display (like xterm), you can set it to some higher value. If no argument is specified, the current setting is displayed.

only

Kill all regions but the current one.

other

Switch to the window displayed previously. If this window does no longer exist, other has the same effect as next.

partial on|off

Defines whether the display should be refreshed (as with redisplay) after switching to the current window. This command only affects the current window. To immediately affect all windows use the allpartial command. Default is `off', of course. This default is fixed, as there is currently no defpartial command.

password [crypted_pw]

Present a crypted password in your ".screenrc" file and screen will ask for it, whenever someone attempts to resume a detached. This is useful if you have privileged programs running under screen and you want to protect your session from reattach attempts by another user masquerading as your uid (i.e. any superuser.) If no crypted password is specified, screen prompts twice for typing a password and places its encryption in the paste buffer. Default is `none', this disables password checking.

paste [registers [dest_reg]]

Write the (concatenated) contents of the specified registers to the stdin queue of the current window. The register '.' is treated as the paste buffer. If no parameter is given the user is prompted for a single register to paste. The paste buffer can be filled with the copy, history and readbuf commands. Other registers can be filled with the register, readreg and paste commands. If paste is called with a second argument, the contents of the specified registers is pasted into the named destination register rather than the window. If '.' is used as the second argument, the displays paste buffer is the destination. Note, that "paste" uses a wide variety of resources: Whenever a second argument is specified no current window is needed. When the source specification only contains registers (not the paste buffer) then there need not be a current display (terminal attached), as the registers are a global resource. The paste buffer exists once for every user.

pastefont [on|off]

Tell screen to include font information in the paste buffer. The default is not to do so. This command is especially useful for multi character fonts like kanji.

pow_break

Reopen the window's terminal line and send a break condition. See `break'.

pow_detach

Power detach. Mainly the same as detach, but also sends a HANGUP signal to the parent process of screen. CAUTION: This will result in a logout, when screen was started from your login-shell.

pow_detach_msg [message]

The message specified here is output whenever a `Power detach' was performed. It may be used as a replacement for a logout message or to reset baud rate, etc. Without parameter, the current message is shown.

prev

Switch to the window with the next lower number. This command can be used repeatedly to cycle through the list of windows.

printcmd [cmd]

If cmd is not an empty string, screen will not use the terminal capabilities "po/pf" if it detects an ansi print sequence ESC [5 i, but pipe the output into cmd. This should normally be a command like "lpr" or "'cat > /tmp/scrprint'". printcmd without a command displays the current setting. The ansi sequence ESC \ ends printing and closes the pipe.

Warning: Be careful with this command! If other user have write access to your terminal, they will be able to fire off print commands.

process [key]

Stuff the contents of the specified register into screen's input queue. If no argument is given you are prompted for a register name. The text is parsed as if it had been typed in from the user's keyboard. This command can be used to bind multiple actions to a single key.

quit

Kill all windows and terminate screen. Note that on VT100-style terminals the keys C-4 and C-\ are identical. This makes the default bindings dangerous: Be careful not to type C-a C-4 when selecting window no. 4. Use the empty bind command (as in "bind '^\'") to remove a key binding.

readbuf [encoding] [filename]

Reads the contents of the specified file into the paste buffer. You can tell screen the encoding of the file via the -e option. If no file is specified, the screen-exchange filename is used. See also "bufferfile" command.

readreg [encoding] [register [filename]]

Does one of two things, dependent on number of arguments: with zero or one arguments it duplicates the paste buffer contents into the register specified or entered at the prompt. With two arguments it reads the contents of the named file into the register, just as readbuf reads the screen-exchange file into the paste buffer. You can tell screen the encoding of the file via the -e option. The following example will paste the system's password file into the screen window (using register p, where a copy remains):

```
C-a : readreg p /etc/passwd
C-a : paste p
```

redisplay

Redisplay the current window. Needed to get a full redisplay when in partial redraw mode.

register [-eencoding]key-string

Save the specified string to the register key. The encoding of the string can be specified via the -e option. See also the "paste" command.

remove

Kill the current region. This is a no-op if there is only one region.

removebuf

Unlinks the screen-exchange file used by the commands "writebuf" and "readbuf".

rendition bell | monitor | silence | so attr [color]

Change the way screen renders the titles of windows that have monitor or bell flags set in caption or hardstatus or windowlist. See the "STRING ESCAPES" chapter for the syntax of the modifiers. The default for monitor is currently "=b " (bold, active colors), for bell "=ub " (underline, bold and active colors), and "=u " for silence.

reset

Reset the virtual terminal to its "power-on" values. Useful when strange settings (like scroll regions or graphics character set) are left over from an application.

resize [-h|-v|-b|-l|-p] [[+|-] n[%] |=|max|min|_|0]

Resize the current region. The space will be removed from or added to the surrounding regions depending on the order of the splits. The available options for resizing are `-h'(horizontal), `-v'(vertical), `-b'(both), `-l'(local to layer), and `-p'(perpendicular). Horizontal resizes will add or remove width to a region, vertical will add or remove height, and both will add or remove size from both dimensions. Local and perpendicular are similar to horizontal and vertical, but they take in account of how a region was split. If a region's last split was horizontal, a local resize will work like a vertical resize. If a region's last split was vertical, a local resize will work like a horizontal resize. Perpendicular resizes work in opposite of local resizes. If no option is specified, local is the default.

The amount of lines to add or remove can be expressed a couple of

different ways. By specifying a number n by itself will resize the region by that absolute amount. You can specify a relative amount by prefixing a plus `+' or minus `-' to the amount, such as adding +n lines or removing -n lines. Resizing can also be expressed as an absolute or relative percentage by postfixing a percent sign `%'. Using zero `0' is a synonym for `min' and using an underscore `_' is a synonym for `max'.

Some examples are:

```
resize +N
       increase current region by N

resize -N
       decrease current region by N

resize  N
       set current region to N

resize 20%
       set current region to 20% of original size

resize +20%
       increase current region by 20%

resize -b =
       make all windows equally

resize  max
       maximize current region

resize  min
       minimize current region
```

Without any arguments, screen will prompt for how you would like to resize the current region.

See "focusminsize" if you want to restrict the minimun size a region can have.

```
screen [-opts] [n] [cmd [args]|//group]
```

Establish a new window. The flow-control options (-f, -fn and -fa), title (a.k.a.) option (-t), login options (-l and -ln) , terminal type option (-T <term>), the all-capability-flag (-a) and scrollback option (-h <num>) may be specified with each command. The option (-M) turns monitoring on for this window. The option (-L) turns output logging on for this window. If an optional number n in the range 0..MAXWIN-1 is given, the window number n is assigned to the newly created window (or, if this number is already in-use, the next available number). If a command is specified after "screen", this command (with the given arguments) is started in the window; otherwise, a shell is created. If //group is supplied, a container-type window is created in which other windows may be created inside it.

Thus, if your ".screenrc" contains the lines

```
# example for .screenrc:
screen 1
screen -fn -t foobar -L 2 telnet foobar
```

screen creates a shell window (in window #1) and a window with a TELNET connection to the machine foobar (with no flow-control using the title "foobar" in window #2) and will write a logfile ("screenlog.2") of the telnet session. Note, that unlike previous versions of screen no additional default window is created when "screen" commands are included in your ".screenrc" file. When the initialization is completed, screen switches to the last window specified in your .screenrc file or, if none, opens a default window #0.

Screen has built in some functionality of "cu" and "telnet". See also chapter "WINDOW TYPES".

scrollback num

Set the size of the scrollback buffer for the current windows to num lines. The default scrollback is 100 lines. See also the "defscrollback" command and use "info" to view the current setting. To access and use the contents in the scrollback buffer, use the "copy" command.

select [WindowID]

Switch to the window identified by WindowID. This can be a prefix of a window title (alphanumeric window name) or a window number. The parameter is optional and if omitted, you get prompted for an identifier. When a new window is established, the first available number is assigned to this window. Thus, the first window can be activated by "select 0". The number of windows is limited at compile-time by the MAXWIN configuration parameter (which defaults to 40). There are two special WindowIDs, "-" selects the internal blank window and "." selects the current window. The latter is useful if used with screen's "-X" option.

sessionname [name]

Rename the current session. Note, that for "screen -list" the name shows up with the process-id prepended. If the argument "name" is omitted, the name of this session is displayed. Caution: The $STY environment variables will still reflect the old name in pre-existing shells. This may result in confusion. Use of this command is generally discouraged. Use the "-S" command-line option if you want to name a new session. The default is constructed from the tty and host names.

setenv [var [string]]

Set the environment variable var to value string. If only var is specified, the user will be prompted to enter a value. If no parameters are specified, the user will be prompted for both variable and value. The environment is inherited by all subsequently forked shells.

setsid [on|off]

Normally screen uses different sessions and process groups for the windows. If setsid is turned off, this is not done anymore and all windows will be in the same process group as the screen backend process. This also breaks job-control, so be careful. The default is on, of course. This command is probably useful

only in rare circumstances.

shell command

Set the command to be used to create a new shell. This overrides the value of the environment variable $SHELL. This is useful if you'd like to run a tty-enhancer which is expecting to execute the program specified in $SHELL. If the command begins with a '-' character, the shell will be started as a login-shell. Typical shells do only minimal initialization when not started as a login-shell. E.g. Bash will not read your "~/.bashrc" unless it is a login-shell.

shelltitle title

Set the title for all shells created during startup or by the C-A C-c command. For details about what a title is, see the discussion entitled "TITLES (naming windows)".

silence [on|off|sec]

Toggles silence monitoring of windows. When silence is turned on and an affected window is switched into the background, you will receive the silence notification message in the status line after a specified period of inactivity (silence). The default timeout can be changed with the `silencewait' command or by specifying a number of seconds instead of `on' or `off'. Silence is initially off for all windows.

silencewait sec

Define the time that all windows monitored for silence should wait before displaying a message. Default 30 seconds.

sleep num

This command will pause the execution of a .screenrc file for num seconds. Keyboard activity will end the sleep. It may be used to give users a chance to read the messages output by "echo".

slowpaste msec

Define the speed at which text is inserted into the current window by the paste ("C-a]") command. If the slowpaste value is nonzero text is written character by character. screen will make a pause of msec milliseconds after each single character write to allow the application to process its input. Only use slowpaste if your underlying system exposes flow control problems while pasting large amounts of text.

sort

Sort the windows in alphabetical order of the window tiles.

source file

Read and execute commands from file file. Source commands may be nested to a maximum recursion level of ten. If file is not an absolute path and screen is already processing a source command, the parent directory of the running source command file is used to search for the new command file before screen's current directory.

Note that termcap/terminfo/termcapinfo commands only work at startup and reattach time, so they must be reached via the default screenrc files to have an effect.

sorendition [attr[color]]

This command is deprecated. See "rendition so" instead.

split[-v]

Split the current region into two new ones. All regions on the display are resized to make room for the new region. The blank window is displayed in the new region. The default is to create a horizontal split, putting the new regions on the top and bottom of each other. Using `-v' will create a vertical split, causing the new regions to appear side by side of each other. Use the "remove" or the "only" command to delete regions. Use "focus" to toggle between regions.

When a region is split opposite of how it was previously split (that is, vertical then horizontal or horizontal then vertical), a new layer is created. The layer is used to group together the regions that are split the same. Normally, as a user, you should not see nor have to worry about layers, but they will affect how some commands ("focus" and "resize") behave.

With this current implementation of screen, scrolling data will appear much slower in a vertically split region than one that is not. This should be taken into consideration if you need to use system commands such as "cat" or "tail -f".

startup_message on|off

Select whether you want to see the copyright notice during startup. Default is `on', as you probably noticed.

status [top|up|down|bottom] [left|right]

The status window by default is in bottom-left corner. This command can move status messages to any corner of the screen. top is the same as up, down is the same as bottom.

stuff [string]

Stuff the string string in the input buffer of the current window. This is like the "paste" command but with much less overhead. Without a parameter, screen will prompt for a string to stuff. You cannot paste large buffers with the "stuff" command. It is most useful for key bindings. See also "bindkey".

su [username [password [password2]]]

Substitute the user of a display. The command prompts for all parameters that are omitted. If passwords are specified as parameters, they have to be specified un-crypted. The first password is matched against the systems passwd database, the second password is matched against the screen password as set with the commands "acladd" or "password". "Su" may be useful for the screen administrator to test multiuser setups. When the identification fails, the user has access to the commands

available for user nobody. These are "detach", "license", "version", "help" and "displays".

suspend

Suspend screen. The windows are in the `detached' state, while screen is suspended. This feature relies on the shell being able to do job control.

term term

In each window's environment screen opens, the $TERM variable is set to "screen" by default. But when no description for "screen" is installed in the local termcap or terminfo data base, you set $TERM to - say - "vt100". This won't do much harm, as screen is VT100/ANSI compatible. The use of the "term" command is discouraged for non-default purpose. That is, one may want to specify special $TERM settings (e.g. vt100) for the next "screen rlogin othermachine" command. Use the command "screen -T vt100 rlogin othermachine" rather than setting and resetting the default.

termcap term terminal-tweaks[window-tweaks]

terminfo term terminal-tweaks[window-tweaks]

termcapinfo term terminal-tweaks[window-tweaks]

Use this command to modify your terminal's termcap entry without going through all the hassles involved in creating a custom termcap entry. Plus, you can optionally customize the termcap generated for the windows. You have to place these commands in one of the screenrc startup files, as they are meaningless once the terminal emulator is booted.

If your system uses the terminfo database rather than termcap, screen will understand the `terminfo' command, which has the same effects as the `termcap' command. Two separate commands are provided, as there are subtle syntactic differences, e.g. when parameter interpolation (using `%') is required. Note that termcap names of the capabilities have to be used with the

`terminfo' command.

In many cases, where the arguments are valid in both terminfo and termcap syntax, you can use the command `termcapinfo', which is just a shorthand for a pair of `termcap' and `terminfo' commands with identical arguments.

The first argument specifies which terminal(s) should be affected by this definition. You can specify multiple terminal names by separating them with `|'s. Use `*' to match all terminals and `vt*' to match all terminals that begin with "vt".

Each tweak argument contains one or more termcap defines (separated by `:'s) to be inserted at the start of the appropriate termcap entry, enhancing it or overriding existing values. The first tweak modifies your terminal's termcap, and contains definitions that your terminal uses to perform certain functions. Specify a null string to leave this unchanged (e.g. ''). The second (optional) tweak modifies all the window termcaps, and should contain definitions that screen understands (see the "VIRTUAL TERMINAL" section).

Some examples:

```
termcap xterm*  LP:hs@
```

Informs screen that all terminals that begin with `xterm' have firm auto-margins that allow the last position on the screen to be updated (LP), but they don't really have a status line (no 'hs' - append `@' to turn entries off). Note that we assume `LP' for all terminal names that start with "vt", but only if you don't specify a termcap command for that terminal.

```
termcap vt*  LP
```

```
termcap vt102|vt220  Z0=\E[?3h:Z1=\E[?3l
```

Specifies the firm-margined `LP' capability for all terminals that begin with `vt', and the second line will also add the escape-sequences to switch into (Z0) and back out of (Z1) 132-character-per-line mode if this is a VT102 or VT220. (You must specify Z0 and Z1 in your termcap to use the width-changing

commands.)

termcap vt100 "" l0=PF1:l1=PF2:l2=PF3:l3=PF4

This leaves your vt100 termcap alone and adds the function key labels to each window's termcap entry.

termcap h19|z19 am@:im=\E@:ei=\EO dc=\E[P

Takes a h19 or z19 termcap and turns off auto-margins (am@) and enables the insert mode (im) and end-insert (ei) capabilities (the `@' in the `im' string is after the `=', so it is part of the string). Having the `im' and `ei' definitions put into your terminal's termcap will cause screen to automatically advertise the character-insert capability in each window's termcap. Each window will also get the delete-character capability (dc) added to its termcap, which screen will translate into a line-update for the terminal (we're pretending it doesn't support character deletion).

If you would like to fully specify each window's termcap entry, you should instead set the $SCREENCAP variable prior to running screen. See the discussion on the "VIRTUAL TERMINAL" in this manual, and the termcap(5) man page for more information on termcap definitions.

title [windowtitle]

Set the name of the current window to windowtitle. If no name is specified, screen prompts for one. This command was known as `aka' in previous releases.

truecolor [on|off]

Enables truecolor support. Currently autodetection of truecolor support cannot be done reliably, as such it's left to user to enable. Default is off. Known terminals that may support it are: iTerm2, Konsole, st. Xterm includes support for truecolor escapes but converts them back to indexed 256 color space.

unbindall

Unbind all the bindings. This can be useful when screen is used solely for its detaching abilities, such as when letting a console application run as a daemon. If, for some reason, it is necessary to bind commands after this, use 'screen -X'.

unsetenv var

Unset an environment variable.

utf8 [on|off[on|off]]

Change the encoding used in the current window. If utf8 is enabled, the strings sent to the window will be UTF-8 encoded and vice versa. Omitting the parameter toggles the setting. If a second parameter is given, the display's encoding is also changed (this should rather be done with screen's "-U" option). See also "defutf8", which changes the default setting of a new window.

vbell [on|off]

Sets the visual bell setting for this window. Omitting the parameter toggles the setting. If vbell is switched on, but your terminal does not support a visual bell, a `vbell-message' is displayed in the status line when the bell character (^G) is received. Visual bell support of a terminal is defined by the termcap variable `vb' (terminfo: 'flash').

Per default, vbell is off, thus the audible bell is used. See also `bell_msg'.

vbell_msg [message]

Sets the visual bell message. message is printed to the status line if the window receives a bell character (^G), vbell is set to "on", but the terminal does not support a visual bell. The default message is "Wuff, Wuff!!". Without a parameter, the current message is shown.

vbellwait sec

Define a delay in seconds after each display of screen's visual bell message. The default is 1 second.

verbose [on|off]

If verbose is switched on, the command name is echoed, whenever a window is created (or resurrected from zombie state). Default is off. Without a parameter, the current setting is shown.

version

Print the current version and the compile date in the status line.

wall message

Write a message to all displays. The message will appear in the terminal's status line.

width [-w|-d] [cols [lines]]

Toggle the window width between 80 and 132 columns or set it to cols columns if an argument is specified. This requires a capable terminal and the termcap entries "Z0" and "Z1". See the "termcap" command for more information. You can also specify a new height if you want to change both values. The -w option tells screen to leave the display size unchanged and just set the window size, -d vice versa.

windowlist [-b] [-m] [-g]

windowlist string [string]

windowlist title [title]

Display all windows in a table for visual window selection. If screen was in a window group, screen will back out of the group and then display the windows in that group. If the -b option is given, screen will switch to the blank window before presenting the list, so that the current window is also selectable. The -m option changes the order of the windows, instead of sorting by

```
window numbers screen uses its internal most-recently-used list.
The -g option will show the windows inside any groups in that
level and downwards.

The following keys are used to navigate in "windowlist":

↪ ------------------------------------------------------------------------
k, C-p, or up      Move up one line.

↪ ------------------------------------------------------------------------
j, C-n, or down    Move down one line.

↪ ------------------------------------------------------------------------
C-g or escape      Exit windowlist.

↪ ------------------------------------------------------------------------
C-a or home        Move to the first line.

↪ ------------------------------------------------------------------------
C-e or end         Move to the last line.

↪ ------------------------------------------------------------------------
C-u or C-d         Move one half page up or down.

↪ ------------------------------------------------------------------------
C-b or C-f         Move one full page up or down.

↪ ------------------------------------------------------------------------
0..9               Using the number keys, move to the selected line.

↪ ------------------------------------------------------------------------
mouseclick         Move to the selected line. Available when
                   "mousetrack" is set to "on"

↪ ------------------------------------------------------------------------
/                  Search.

↪ ------------------------------------------------------------------------
n                  Repeat search in the forward direction.
```

```
↪ ------------------------------------------------------------------------
N                   Repeat search in the backward direction.

↪ ------------------------------------------------------------------------
m                   Toggle MRU.

↪ ------------------------------------------------------------------------
g                   Toggle group nesting.

↪ ------------------------------------------------------------------------
a                   All window view.

↪ ------------------------------------------------------------------------

C-h or backspace    Back out the group.

↪ ------------------------------------------------------------------------
,                   Switch numbers with the previous window.

↪ ------------------------------------------------------------------------
.                   Switch numbers with the next window.

↪ ------------------------------------------------------------------------
K                   Kill that window.

↪ ------------------------------------------------------------------------
space or enter      Select that window.

↪ ------------------------------------------------------------------------

The table format can be changed with the string and title option,
the title is displayed as table heading, while the lines are made
by using the string setting. The default setting is "Num
Name%=Flags" for the title and "%3n %t%=%f" for the lines.  See
the "STRING ESCAPES" chapter for more codes (e.g. color
settings).

"Windowlist" needs a region size of at least 10 characters wide
and 6 characters high in order to display.
```

windows [string]

Uses the message line to display a list of all the windows. Each window is listed by number with the name of process that has been started in the window (or its title); the current window is marked with a `*'; the previous window is marked with a `-'; all the windows that are "logged in" are marked with a `$'; a background window that has received a bell is marked with a `!'; a background window that is being monitored and has had activity occur is marked with an `@'; a window which has output logging turned on is marked with `(L)'; windows occupied by other users are marked with `&'; windows in the zombie state are marked with `Z'. If this list is too long to fit on the terminal's status line only the portion around the current window is displayed. The optional string parameter follows the "STRING ESCAPES" format. If string parameter is passed, the output size is unlimited. The default command without any parameter is limited to a size of 1024 bytes.

wrap [on|off]

Sets the line-wrap setting for the current window. When line-wrap is on, the second consecutive printable character output at the last column of a line will wrap to the start of the following line. As an added feature, backspace (^H) will also wrap through the left margin to the previous line. Default is `on'. Without any options, the state of wrap is toggled.

writebuf [-e encoding] [filename]

Writes the contents of the paste buffer to the specified file, or the public accessible screen-exchange file if no filename is given. This is thought of as a primitive means of communication between screen users on the same host. If an encoding is specified the paste buffer is recoded on the fly to match the encoding. The filename can be set with the bufferfile command and defaults to "/tmp/screen-exchange".

writelock [on|off|auto]

In addition to access control lists, not all users may be able to

write to the same window at once. Per default, writelock is in `auto' mode and grants exclusive input permission to the user who is the first to switch to the particular window. When he leaves the window, other users may obtain the writelock (automatically). The writelock of the current window is disabled by the command "writelock off". If the user issues the command "writelock on" he keeps the exclusive write permission while switching to other windows.

xoff

xon

Insert a CTRL-s / CTRL-q character to the stdin queue of the current window.

zmodem [off|auto|catch|pass]

zmodem sendcmd [string]

zmodem recvcmd [string]

Define zmodem support for screen. Screen understands two different modes when it detects a zmodem request: "pass" and "catch". If the mode is set to "pass", screen will relay all data to the attacher until the end of the transmission is reached. In "catch" mode screen acts as a zmodem endpoint and starts the corresponding rz/sz commands. If the mode is set to "auto", screen will use "catch" if the window is a tty (e.g. a serial line), otherwise it will use "pass".

You can define the templates screen uses in "catch" mode via the second and the third form.

Note also that this is an experimental feature.

zombie [keys[onerror]]

Per default screen windows are removed from the window list as soon as the windows process (e.g. shell) exits. When a string of two keys is specified to the zombie command, `dead' windows will

remain in the list. The kill command may be used to remove such a window. Pressing the first key in the dead window has the same effect. When pressing the second key, screen will attempt to resurrect the window. The process that was initially running in the window will be launched again. Calling zombie without parameters will clear the zombie setting, thus making windows disappear when their process exits.

As the zombie-setting is manipulated globally for all windows, this command should probably be called defzombie, but it isn't.

Optionally you can put the word "onerror" after the keys. This will cause screen to monitor exit status of the process running in the window. If it exits normally ('0'), the window disappears. Any other exit value causes the window to become a zombie.

zombie_timeout[seconds]

Per default screen windows are removed from the window list as soon as the windows process (e.g. shell) exits. If zombie keys are defined (compare with above zombie command), it is possible to also set a timeout when screen tries to automatically reconnect a dead screen window.

THE MESSAGE LINE top

Screen displays informational messages and other diagnostics in a message line. While this line is distributed to appear at the bottom of the screen, it can be defined to appear at the top of the screen during compilation. If your terminal has a status line defined in its termcap, screen will use this for displaying its messages, otherwise a line of the current screen will be temporarily overwritten and output will be momentarily interrupted. The message line is automatically removed after a few seconds delay, but it can also be removed early (on terminals without a status line) by beginning to type.

The message line facility can be used by an application running in the current window by means of the ANSI Privacy message control sequence. For instance, from within the shell, try something like:

```
echo '<esc>^Hello world from window '$WINDOW'<esc>\\'
```

where '<esc>' is an escape, '^' is a literal up-arrow, and '\\' turns into a single backslash.

WINDOW TYPES top

Screen provides three different window types. New windows are created with screen's screen command (see also the entry in chapter "CUSTOMIZATION"). The first parameter to the screen command defines which type of window is created. The different window types are all special cases of the normal type. They have been added in order to allow screen to be used efficiently as a console multiplexer with 100 or more windows.

- The normal window contains a shell (default, if no parameter is given) or any other system command that could be executed from a shell (e.g. slogin, etc...)

- If a tty (character special device) name (e.g. "/dev/ttya") is specified as the first parameter, then the window is directly connected to this device. This window type is similar to "screen cu -l /dev/ttya". Read and write access is required on the device node, an exclusive open is attempted on the node to mark the connection line as busy. An optional parameter is allowed consisting of a comma separated list of flags in the notation used by stty(1):

  <baud_rate>
  Usually 300, 1200, 9600 or 19200. This affects transmission as well as receive speed.

  cs8 or cs7
  Specify the transmission of eight (or seven) bits per byte.

  ixon or -ixon
  Enables (or disables) software flow-control (CTRL-S/CTRL-Q) for sending data.

  ixoff or -ixoff
  Enables (or disables) software flow-control for receiving data.

istrip or -istrip
Clear (or keep) the eight bit in each received byte.

You may want to specify as many of these options as applicable. Unspecified options cause the terminal driver to make up the parameter values of the connection. These values are system dependent and may be in defaults or values saved from a previous connection.

For tty windows, the info command shows some of the modem control lines in the status line. These may include `RTS', `CTS', 'DTR', `DSR', `CD' and more. This depends on the available ioctl()'s and system header files as well as the on the physical capabilities of the serial board. Signals that are logical low (inactive) have their name preceded by an exclamation mark (!), otherwise the signal is logical high (active). Signals not supported by the hardware but available to the ioctl() interface are usually shown low.

When the CLOCAL status bit is true, the whole set of modem signals is placed inside curly braces ({ and }). When the CRTSCTS or TIOCSOFTCAR bit is set, the signals `CTS' or `CD' are shown in parenthesis, respectively.

For tty windows, the command break causes the Data transmission line (TxD) to go low for a specified period of time. This is expected to be interpreted as break signal on the other side. No data is sent and no modem control line is changed when a break is issued.

- If the first parameter is "//telnet", the second parameter is expected to be a host name, and an optional third parameter may specify a TCP port number (default decimal 23). Screen will connect to a server listening on the remote host and use the telnet protocol to communicate with that server.

For telnet windows, the command info shows details about the connection in square brackets ([and]) at the end of the status line.

b BINARY. The connection is in binary mode.

e ECHO. Local echo is disabled.

c SGA. The connection is in `character mode' (default: `line mode').

t TTYPE. The terminal type has been requested by the remote host. Screen sends the name "screen" unless instructed otherwise (see also the command `term').

w NAWS. The remote site is notified about window size changes.

f LFLOW. The remote host will send flow control information. (Ignored at the moment.)

Additional flags for debugging are x, t and n (XDISPLOC, TSPEED and NEWENV).

For telnet windows, the command break sends the telnet code IAC BREAK (decimal 243) to the remote host.

This window type is only available if screen was compiled with the ENABLE_TELNET option defined.

STRING ESCAPES top

Screen provides an escape mechanism to insert information like the current time into messages or file names. The escape character is '%' with one exception: inside of a window's hardstatus '^%' ('^E') is used instead.

Here is the full list of supported escapes:

% the escape character itself

C The count of screen windows. Prefix with '-' to limit to current window group.

E sets %? to true if the escape character has been pressed.

f flags of the window, see "windows" for meanings of the various flags

F sets %? to true if the window has the focus

h hardstatus of the window

H hostname of the system

n window number

P sets %? to true if the current region is in copy/paste mode

S session name

s window size

t window title

u all other users on this window

w all window numbers and names. With '-' qualifier: up to the current window; with '+' qualifier: starting with the window after the current one.

W all window numbers and names except the current one

x the executed command including arguments running in this windows

X the executed command without arguments running in this windows

? the part to the next '%?' is displayed only if a '%' escape inside the part expands to a non-empty string

: else part of '%?'

= pad the string to the display's width (like TeX's hfill). If a number is specified, pad to the percentage of the window's width. A '0' qualifier tells screen to treat the number as absolute position. You can specify to pad

relative to the last absolute pad position by adding a '+' qualifier or to pad relative to the right margin by using '-'. The padding truncates the string if the specified position lies before the current position. Add the 'L' qualifier to change this.

< same as '%=' but just do truncation, do not fill with spaces

> mark the current text position for the next truncation. When screen needs to do truncation, it tries to do it in a way that the marked position gets moved to the specified percentage of the output area. (The area starts from the last absolute pad position and ends with the position specified by the truncation operator.) The 'L' qualifier tells screen to mark the truncated parts with '...'.

{ attribute/color modifier string terminated by the next "}"

` Substitute with the output of a 'backtick' command. The length qualifier is misused to identify one of the commands.

The 'c' and 'C' escape may be qualified with a '0' to make screen use zero instead of space as fill character. The '0' qualifier also makes the '=' escape use absolute positions. The 'n' and '=' escapes understand a length qualifier (e.g. '%3n'), 'D' and 'M' can be prefixed with 'L' to generate long names, 'w' and 'W' also show the window flags if 'L' is given.

An attribute/color modifier is used to change the attributes or the color settings. Its format is "[attribute modifier] [color description]". The attribute modifier must be prefixed by a change type indicator if it can be confused with a color description. The following change types are known:

+ add the specified set to the current attributes

- remove the set from the current attributes

! invert the set in the current attributes

```
=       change the current attributes to the specified set

The attribute set can either be specified as a hexadecimal number
or a combination of the following letters:

d       dim
u       underline
b       bold
r       reverse
s       standout
B       blinking

The old format of specifying colors by letters (k,r,g,y,b,m,c,w)
is now deprecated. Colors are coded as 0-7 for basic ANSI, 0-255
for 256 color mode, or for truecolor, either a hexadecimal code
starting with x, or HTML notation as either 3 or 6 hexadecimal
digits. Foreground and background are specified by putting a
semicolon between them. Ex: "#FFF;#000" or "i7;0" is white on a
black background.

The following numbers are for basic ANSI:

0       black
1       red
2       green
3       yellow
4       blue
5       magenta
6       cyan
7       white

You can also use the pseudo-color 'i' to set just the brightness
and leave the color unchanged.
As a special case, "%{-}" restores the attributes and colors that
were set before the last change was made (i.e., pops one level of
the color-change stack).

Examples:

"i2"    set color to bright green
```

"+b r" use bold red

"#F00;FFA"
write in bright red color on a pale yellow background.

%-Lw%{#AAA;#006}%50>%n%f* %t%{-}%+Lw%<
The available windows centered at the current window and truncated to the available width. The current window is displayed white on blue. This can be used with "hardstatus alwayslastline".

%?%F%{;2}%?%3n %t%? [%h]%?
The window number and title and the window's hardstatus, if one is set. Also use a red background if this is the active focus. Useful for "caption string".

FLOW-CONTROL top

Each window has a flow-control setting that determines how screen deals with the XON and XOFF characters (and perhaps the interrupt character). When flow-control is turned off, screen ignores the XON and XOFF characters, which allows the user to send them to the current program by simply typing them (useful for the emacs editor, for instance). The trade-off is that it will take longer for output from a "normal" program to pause in response to an XOFF. With flow-control turned on, XON and XOFF characters are used to immediately pause the output of the current window. You can still send these characters to the current program, but you must use the appropriate two-character screen commands (typically "C-a q" (xon) and "C-a s" (xoff)). The xon/xoff commands are also useful for typing C-s and C-q past a terminal that intercepts these characters.

Each window has an initial flow-control value set with either the -f option or the "defflow" .screenrc command. Per default the windows are set to automatic flow-switching. It can then be toggled between the three states 'fixed on', 'fixed off' and 'automatic' interactively with the "flow" command bound to "C-a f".

The automatic flow-switching mode deals with flow control using the TIOCPKT mode (like "rlogin" does). If the tty driver does not

support TIOCPKT, screen tries to find out the right mode based on the current setting of the application keypad - when it is enabled, flow-control is turned off and visa versa. Of course, you can still manipulate flow-control manually when needed.

If you're running with flow-control enabled and find that pressing the interrupt key (usually C-c) does not interrupt the display until another 6-8 lines have scrolled by, try running screen with the "interrupt" option (add the "interrupt" flag to the "flow" command in your .screenrc, or use the -i command-line option). This causes the output that screen has accumulated from the interrupted program to be flushed. One disadvantage is that the virtual terminal's memory contains the non-flushed version of the output, which in rare cases can cause minor inaccuracies in the output. For example, if you switch screens and return, or update the screen with "C-a l" you would see the version of the output you would have gotten without "interrupt" being on. Also, you might need to turn off flow-control (or use auto-flow mode to turn it off automatically) when running a program that expects you to type the interrupt character as input, as it is possible to interrupt the output of the virtual terminal to your physical terminal when flow-control is enabled. If this happens, a simple refresh of the screen with "C-a l" will restore it. Give each mode a try, and use whichever mode you find more comfortable.

TITLES (naming windows) top

You can customize each window's name in the window display (viewed with the "windows" command (C-a w)) by setting it with one of the title commands. Normally the name displayed is the actual command name of the program created in the window. However, it is sometimes useful to distinguish various programs of the same name or to change the name on-the-fly to reflect the current state of the window.

The default name for all shell windows can be set with the "shelltitle" command in the .screenrc file, while all other windows are created with a "screen" command and thus can have their name set with the -t option. Interactively, there is the title-string escape-sequence (<esc>kname<esc>\) and the "title" command (C-a A). The former can be output from an application to control the window's name under software control, and the latter will prompt for a name when typed. You can also bind pre-defined

names to keys with the "title" command to set things quickly without prompting. Changing title by this escape sequence can be controlled by defdynamictitle and dynamictitle commands.

Finally, screen has a shell-specific heuristic that is enabled by setting the window's name to "search|name" and arranging to have a null title escape-sequence output as a part of your prompt. The search portion specifies an end-of-prompt search string, while the name portion specifies the default shell name for the window. If the name ends in a `:' screen will add what it believes to be the current command running in the window to the end of the window's shell name (e.g. "name:cmd"). Otherwise the current command name supersedes the shell name while it is running.

Here's how it works: you must modify your shell prompt to output a null title-escape-sequence (<esc>k<esc>\) as a part of your prompt. The last part of your prompt must be the same as the string you specified for the search portion of the title. Once this is set up, screen will use the title-escape-sequence to clear the previous command name and get ready for the next command. Then, when a newline is received from the shell, a search is made for the end of the prompt. If found, it will grab the first word after the matched string and use it as the command name. If the command name begins with either '!', '%', or '^' screen will use the first word on the following line (if found) in preference to the just-found name. This helps csh users get better command names when using job control or history recall commands.

Here's some .screenrc examples:

```
screen -t top 2 nice top
```

Adding this line to your .screenrc would start a nice-d version of the "top" command in window 2 named "top" rather than "nice".

```
shelltitle '> |csh'
screen 1
```

These commands would start a shell with the given shelltitle.

The title specified is an auto-title that would expect the prompt and the typed command to look something like the following:

```
/usr/joe/src/dir> trn
```

(it looks after the '> ' for the command name). The window status would show the name "trn" while the command was running, and revert to "csh" upon completion.

```
bind R screen -t '% |root:' su
```

Having this command in your .screenrc would bind the key sequence "C-a R" to the "su" command and give it an auto-title name of "root:". For this auto-title to work, the screen could look something like this:

```
% !em
emacs file.c
```

Here the user typed the csh history command "!em" which ran the previously entered "emacs" command. The window status would show "root:emacs" during the execution of the command, and revert to simply "root:" at its completion.

```
bind o title
bind E title ""
bind u title (unknown)
```

The first binding doesn't have any arguments, so it would prompt you for a title when you type "C-a o". The second binding would clear an auto-title's current setting (C-a E). The third binding would set the current window's title to "(unknown)" (C-a u).

One thing to keep in mind when adding a null title-escape-sequence to your prompt is that some shells (like the csh) count all the non-control characters as part of the prompt's length. If these invisible characters aren't a multiple of 8 then backspacing over a tab will result in an incorrect display. One way to get around this is to use a prompt like this:

```
set prompt='^[[0000m^[k^[\% '
```

The escape-sequence "<esc>[0000m" not only normalizes the character attributes, but all the zeros round the length of the invisible characters up to 8. Bash users will probably want to echo the escape sequence in the PROMPT_COMMAND:

```
PROMPT_COMMAND='printf "\033k\033\134"'
```

(I used "\134" to output a `\' because of a bug in bash v1.04).

THE VIRTUAL TERMINAL top

Each window in a screen session emulates a VT100 terminal, with some extra functions added. The VT100 emulator is hard-coded, no other terminal types can be emulated.

Usually screen tries to emulate as much of the VT100/ANSI standard as possible. But if your terminal lacks certain capabilities, the emulation may not be complete. In these cases screen has to tell the applications that some of the features are missing. This is no problem on machines using termcap, because screen can use the $TERMCAP variable to customize the standard screen termcap.

But if you do a rlogin on another machine or your machine supports only terminfo this method fails. Because of this, screen offers a way to deal with these cases. Here is how it works:

When screen tries to figure out a terminal name for itself, it first looks for an entry named "screen.<term>", where <term> is the contents of your $TERM variable. If no such entry exists, screen tries "screen" (or "screen-w" if the terminal is wide (132 cols or more)). If even this entry cannot be found, "vt100" is used as a substitute.

The idea is that if you have a terminal which doesn't support an important feature (e.g. delete char or clear to EOS) you can build a new termcap/terminfo entry for screen (named "screen.<dumbterm>") in which this capability has been disabled. If this entry is installed on your machines you are able to do a rlogin and still keep the correct termcap/terminfo entry. The terminal name is put in the $TERM variable of all new windows. Screen also sets the $TERMCAP variable reflecting the capabilities of the virtual terminal emulated. Notice that,

however, on machines using the terminfo database this variable has no effect. Furthermore, the variable $WINDOW is set to the window number of each window.

The actual set of capabilities supported by the virtual terminal depends on the capabilities supported by the physical terminal. If, for instance, the physical terminal does not support underscore mode, screen does not put the `us' and `ue' capabilities into the window's $TERMCAP variable, accordingly. However, a minimum number of capabilities must be supported by a terminal in order to run screen; namely scrolling, clear screen, and direct cursor addressing (in addition, screen does not run on hardcopy terminals or on terminals that over-strike).

Also, you can customize the $TERMCAP value used by screen by using the "termcap" .screenrc command, or by defining the variable $SCREENCAP prior to startup. When the latter is defined, its value will be copied verbatim into each window's $TERMCAP variable. This can either be the full terminal definition, or a filename where the terminal "screen" (and/or "screen-w") is defined.

Note that screen honors the "terminfo" .screenrc command if the system uses the terminfo database rather than termcap.

When the boolean `G0' capability is present in the termcap entry for the terminal on which screen has been called, the terminal emulation of screen supports multiple character sets. This allows an application to make use of, for instance, the VT100 graphics character set or national character sets. The following control functions from ISO 2022 are supported: lock shift G0 (SI), lock shift G1 (SO), lock shift G2, lock shift G3, single shift G2, and single shift G3. When a virtual terminal is created or reset, the ASCII character set is designated as G0 through G3. When the `G0' capability is present, screen evaluates the capabilities `S0', `E0', and `C0' if present. `S0' is the sequence the terminal uses to enable and start the graphics character set rather than SI. `E0' is the corresponding replacement for SO. `C0' gives a character by character translation string that is used during semi-graphics mode. This string is built like the `acsc' terminfo capability.

When the `po' and `pf' capabilities are present in the terminal's termcap entry, applications running in a screen window can send output to the printer port of the terminal. This allows a user to have an application in one window sending output to a printer connected to the terminal, while all other windows are still active (the printer port is enabled and disabled again for each chunk of output). As a side-effect, programs running in different windows can send output to the printer simultaneously. Data sent to the printer is not displayed in the window. The info command displays a line starting `PRIN' while the printer is active.

Screen maintains a hardstatus line for every window. If a window gets selected, the display's hardstatus will be updated to match the window's hardstatus line. If the display has no hardstatus the line will be displayed as a standard screen message. The hardstatus line can be changed with the ANSI Application Program Command (APC): "ESC_<string>ESC\". As a convenience for xterm users the sequence "ESC]0..2;<string>^G" is also accepted.

Some capabilities are only put into the $TERMCAP variable of the virtual terminal if they can be efficiently implemented by the physical terminal. For instance, `dl' (delete line) is only put into the $TERMCAP variable if the terminal supports either delete line itself or scrolling regions. Note that this may provoke confusion, when the session is reattached on a different terminal, as the value of $TERMCAP cannot be modified by parent processes.

The "alternate screen" capability is not enabled by default. Set the altscreen .screenrc command to enable it.

The following is a list of control sequences recognized by screen. "(V)" and "(A)" indicate VT100-specific and ANSI- or ISO-specific functions, respectively.

ESC E Next Line

ESC D Index

```
ESC M  Reverse Index

ESC H  Horizontal Tab Set

ESC Z  Send VT100 Identification String

ESC 7                  (V)
       Save Cursor and Attributes

ESC 8                  (V)
       Restore Cursor and Attributes

ESC [s                 (A)
       Save Cursor and Attributes

ESC [u                 (A)
       Restore Cursor and Attributes

ESC c  Reset to Initial State

ESC g  Visual Bell

ESC Pn p
       Cursor Visibility (97801)

       Pn = 6 Invisible

       Pn = 7 Visible

ESC =                  (V)
       Application Keypad Mode

ESC >                  (V)
       Numeric Keypad Mode

ESC # 8                (V)
       Fill Screen with E's

ESC \                  (A)
       String Terminator
```

```
ESC ^                   (A)
       Privacy Message String (Message Line)

ESC !  Global Message String (Message Line)

ESC k  A.k.a. Definition String

ESC P                   (A)
       Device Control String.  Outputs a string directly to the
       host terminal without interpretation.

ESC _                   (A)
       Application Program Command (Hardstatus)

ESC ] 0 ; string ^G   (A)
       Operating System Command (Hardstatus, xterm title hack)

ESC ] 83 ; cmd ^G     (A)
       Execute screen command. This only works if multi-user
       support is compiled into screen. The pseudo-user
       ":window:" is used to check the access control list. Use
       "addacl :window: -rwx #?" to create a user with no rights
       and allow only the needed commands.

Control-N             (A)
       Lock Shift G1 (SO)

Control-O             (A)
       Lock Shift G0 (SI)

ESC n                   (A)
       Lock Shift G2

ESC o                   (A)
       Lock Shift G3

ESC N                   (A)
       Single Shift G2

ESC O                   (A)
       Single Shift G3
```

```
ESC ( Pcs             (A)
       Designate character set as G0

ESC ) Pcs             (A)
       Designate character set as G1

ESC * Pcs             (A)
       Designate character set as G2

ESC + Pcs             (A)
       Designate character set as G3

ESC [ Pn ; Pn H
       Direct Cursor Addressing

ESC [ Pn ; Pn f
       same as above

ESC [ Pn J
       Erase in Display

       Pn = None or 0
              From Cursor to End of Screen

       Pn = 1 From Beginning of Screen to Cursor

       Pn = 2 Entire Screen

ESC [ Pn K
       Erase in Line

       Pn = None or 0
              From Cursor to End of Line

       Pn = 1 From Beginning of Line to Cursor

       Pn = 2 Entire Line

ESC [ Pn X
       Erase character
```

```
ESC [ Pn A
       Cursor Up

ESC [ Pn B
       Cursor Down

ESC [ Pn C
       Cursor Right

ESC [ Pn D
       Cursor Left

ESC [ Pn E
       Cursor next line

ESC [ Pn F
       Cursor previous line

ESC [ Pn G
       Cursor horizontal position

ESC [ Pn `
       same as above

ESC [ Pn d
       Cursor vertical position

ESC [ Ps ;...; Ps m
       Select Graphic Rendition

       Ps = None or 0
              Default Rendition

       Ps = 1 Bold

       Ps = 2                (A)
              Faint

       Ps = 3                (A)
              Standout Mode (ANSI: Italicized)
```

Ps = 4 Underlined

Ps = 5 Blinking

Ps = 7 Negative Image

Ps = 22 (A)
Normal Intensity

Ps = 23 (A)
Standout Mode off (ANSI: Italicized off)

Ps = 24 (A)
Not Underlined

Ps = 25 (A)
Not Blinking

Ps = 27 (A)
Positive Image

Ps = 30 (A)
Foreground Black

Ps = 31 (A)
Foreground Red

Ps = 32 (A)
Foreground Green

Ps = 33 (A)
Foreground Yellow

Ps = 34 (A)
Foreground Blue

Ps = 35 (A)
Foreground Magenta

Ps = 36 (A)

Foreground Cyan

Ps = 37 (A)
Foreground White

Ps = 39 (A)
Foreground Default

Ps = 40 (A)
Background Black

Ps = ...

Ps = 49 (A)
Background Default

ESC [Pn g
Tab Clear

Pn = None or 0
Clear Tab at Current Position

Pn = 3 Clear All Tabs

ESC [Pn ; Pn r (V)
Set Scrolling Region

ESC [Pn I (A)
Horizontal Tab

ESC [Pn Z (A)
Backward Tab

ESC [Pn L (A)
Insert Line

ESC [Pn M (A)
Delete Line

ESC [Pn @ (A)
Insert Character

```
ESC [ Pn P            (A)
       Delete Character

ESC [ Pn S
       Scroll Scrolling Region Up

ESC [ Pn T
       Scroll Scrolling Region Down

ESC [ Pn ^
       same as above

ESC [ Ps ;...; Ps h
       Set Mode

ESC [ Ps ;...; Ps l
       Reset Mode

       Ps = 4                (A)
              Insert Mode

       Ps = 20               (A)
              Automatic Linefeed Mode

       Ps = 34
              Normal Cursor Visibility

       Ps = ?1               (V)
              Application Cursor Keys

       Ps = ?3               (V)
              Change Terminal Width to 132 columns

       Ps = ?5               (V)
              Reverse Video

       Ps = ?6               (V)
              Origin Mode

       Ps = ?7               (V)
```

```
                Wrap Mode

        Ps = ?9
                X10 mouse tracking

        Ps = ?25                (V)
                Visible Cursor

        Ps = ?47
                Alternate Screen (old xterm code)

        Ps = ?1000              (V)
                VT200 mouse tracking

        Ps = ?1047
                Alternate Screen (new xterm code)

        Ps = ?1049
                Alternate Screen (new xterm code)

ESC [ 5 i               (A)
        Start relay to printer (ANSI Media Copy)

ESC [ 4 i               (A)
        Stop relay to printer (ANSI Media Copy)

ESC [ 8 ; Ph ; Pw t
        Resize the window to `Ph' lines and `Pw' columns (SunView
        special)

ESC [ c
        Send VT100 Identification String

ESC [ x
        Send Terminal Parameter Report

ESC [ > c
        Send VT220 Secondary Device Attributes String

ESC [ 6 n
        Send Cursor Position Report
```

INPUT TRANSLATION top

In order to do a full VT100 emulation screen has to detect that a sequence of characters in the input stream was generated by a keypress on the user's keyboard and insert the VT100 style escape sequence. Screen has a very flexible way of doing this by making it possible to map arbitrary commands on arbitrary sequences of characters. For standard VT100 emulation the command will always insert a string in the input buffer of the window (see also command stuff in the command table). Because the sequences generated by a keypress can change after a reattach from a different terminal type, it is possible to bind commands to the termcap name of the keys. Screen will insert the correct binding after each reattach. See the bindkey command for further details on the syntax and examples.

Here is the table of the default key bindings. The fourth is what command is executed if the keyboard is switched into application mode.

Key name	Termcap name	Command	App mode
Cursor up	ku	\033[A	\033OA
Cursor down	kd	\033[B	\033OB
Cursor right	kr	\033[C	\033OC
Cursor left	kl	\033[D	\033OD
Function key 0	k0	\033[10~	
Function key 1	k1	\033OP	
Function key 2	k2	\033OQ	
Function key 3	k3	\033OR	
Function key 4	k4	\033OS	
Function key 5	k5	\033[15~	

Function key 6	k6	\033[17~	
Function key 7	k7	\033[18~	
Function key 8	k8	\033[19~	
Function key 9	k9	\033[20~	
Function key 10	k;	\033[21~	
Function key 11	F1	\033[23~	
Function key 12	F2	\033[24~	
Home	kh	\033[1~	
End	kH	\033[4~	
Insert	kI	\033[2~	
Delete	kD	\033[3~	
Page up	kP	\033[5~	
Page down	kN	\033[6~	
Keypad 0	f0	0	\033Op
Keypad 1	f1	1	\033Oq
Keypad 2	f2	2	\033Or
Keypad 3	f3	3	\033Os
Keypad 4	f4	4	\033Ot
Keypad 5	f5	5	\033Ou
Keypad 6	f6	6	\033Ov

Keypad 7	f7	7	\033Ow
Keypad 8	f8	8	\033Ox
Keypad 9	f9	9	\033Oy
Keypad +	f+	+	\033Ok
Keypad -	f-	-	\033Om
Keypad *	f*	*	\033Oj
Keypad /	f/	/	\033Oo
Keypad =	fq	=	\033OX
Keypad .	f.	.	\033On
Keypad ,	f,	,	\033Ol
Keypad enter	fe	\015	\033OM

SPECIAL TERMINAL CAPABILITIES top

The following table describes all terminal capabilities that are recognized by screen and are not in the termcap(5) manual. You can place these capabilities in your termcap entries (in `/etc/termcap') or use them with the commands `termcap', `terminfo' and `termcapinfo' in your screenrc files. It is often not possible to place these capabilities in the terminfo database.

LP (bool)
Terminal has VT100 style margins (`magic margins'). Note that this capability is obsolete because screen uses the standard 'xn' instead.

Z0 (str)
Change width to 132 columns.

Z1 (str)
Change width to 80 columns.

```
WS   (str)
       Resize display. This capability has the desired width and
       height as arguments. SunView(tm) example: '\E[8;%d;%dt'.

NF   (bool)
       Terminal doesn't need flow control. Send ^S and ^Q direct
       to the application. Same as 'flow off'. The opposite of
       this capability is 'nx'.

G0   (bool)
       Terminal can deal with ISO 2022 font selection sequences.

S0   (str)
       Switch charset 'G0' to the specified charset. Default is
       '\E(%.'.

E0   (str)
       Switch charset 'G0' back to standard charset. Default is
       '\E(B'.

C0   (str)
       Use the string as a conversion table for font '0'. See the
       'ac' capability for more details.

CS   (str)
       Switch cursor-keys to application mode.

CE   (str)
       Switch cursor-keys back to normal mode.

AN   (bool)
       Turn on autonuke. See the 'autonuke' command for more
       details.

OL   (num)
       Set the output buffer limit. See the 'obuflimit' command
       for more details.

KJ   (str)
       Set the encoding of the terminal. See the 'encoding'
```

command for valid encodings.

AF (str)
Change character foreground color in an ANSI conform way. This capability will almost always be set to '\E[3%dm' ('\E[3%p1%dm' on terminfo machines).

AB (str)
Same as 'AF', but change background color.

AX (bool)
Does understand ANSI set default fg/bg color (\E[39m / \E[49m).

XC (str)
Describe a translation of characters to strings depending on the current font. More details follow in the next section.

XT (bool)
Terminal understands special xterm sequences (OSC, mouse tracking).

C8 (bool)
Terminal needs bold to display high-intensity colors (e.g. Eterm).

TF (bool)
Add missing capabilities to the termcap/info entry. (Set by default).

CHARACTER TRANSLATION top

Screen has a powerful mechanism to translate characters to arbitrary strings depending on the current font and terminal type. Use this feature if you want to work with a common standard character set (say ISO8851-latin1) even on terminals that scatter the more unusual characters over several national language font pages.

Syntax:

```
XC=<charset-mapping>{,,<charset-mapping>}
<charset-mapping> := <designator><template>{,<mapping>}
```

```
    <mapping> := <char-to-be-mapped><template-arg>
```

The things in braces may be repeated any number of times.

A <charset-mapping> tells screen how to map characters in font <designator> ('B': Ascii, 'A': UK, 'K': German, etc.) to strings. Every <mapping> describes to what string a single character will be translated. A template mechanism is used, as most of the time the codes have a lot in common (for example strings to switch to and from another charset). Each occurrence of '%' in <template> gets substituted with the <template-arg> specified together with the character. If your strings are not similar at all, then use '%' as a template and place the full string in <template-arg>. A quoting mechanism was added to make it possible to use a real '%'. The '\' character quotes the special characters '\', '%', and ','.

Here is an example:

```
    termcap hp700 'XC=B\E(K%\E(B,\304[,\326\\\\,\334]'
```

This tells screen how to translate ISOlatin1 (charset 'B') upper case umlaut characters on a hp700 terminal that has a German charset. '\304' gets translated to '\E(K[\E(B' and so on. Note that this line gets parsed *three* times before the internal lookup table is built, therefore a lot of quoting is needed to create a single '\'.

Another extension was added to allow more emulation: If a mapping translates the unquoted '%' char, it will be sent to the terminal whenever screen switches to the corresponding <designator>. In this special case the template is assumed to be just '%' because the charset switch sequence and the character mappings normally haven't much in common.

This example shows one use of the extension:

```
    termcap xterm 'XC=K%,%\E(B,[\304,\\\\\326,]\334'
```

Here, a part of the German ('K') charset is emulated on an xterm. If screen has to change to the 'K' charset, '\E(B' will be sent

```
       to the terminal, i.e. the ASCII charset is used instead. The
       template is just '%', so the mapping is straightforward: '[' to
       '\304', '\' to '\326', and ']' to '\334'.
ENVIRONMENT         top
       COLUMNS        Number of columns on the terminal (overrides
                      termcap entry).
       HOME           Directory in which to look for .screenrc.
       LINES          Number of lines on the terminal (overrides termcap
                      entry).
       LOCKPRG        Screen lock program.
       NETHACKOPTIONS Turns on nethack option.
       PATH           Used for locating programs to run.
       SCREENCAP      For customizing a terminal's TERMCAP value.
       SCREENDIR      Alternate socket directory.
       SCREENRC       Alternate user screenrc file.
       SHELL          Default shell program for opening windows (default
                      "/bin/sh").  See also "shell" .screenrc command.
       STY            Alternate socket name.
       SYSTEM_SCREENRC
                      Alternate system screenrc file.
       TERM           Terminal name.
       TERMCAP        Terminal description.
       WINDOW         Window number of a window (at creation time).
FILES          top
       .../screen-4.?.??/etc/screenrc
       .../screen-4.?.??/etc/etcscreenrc Examples in the screen
                                         distribution package for
                                         private and global
                                         initialization files.
       $SYSTEM_SCREENRC
       /usr/local/etc/screenrc           screen initialization commands
       $SCREENRC
       $HOME/.screenrc                   Read in after
                                         /usr/local/etc/screenrc
       $SCREENDIR/S-<login>
       /local/screens/S-<login>          Socket directories (default)
       /usr/tmp/screens/S-<login>        Alternate socket directories.
       <socket directory>/.termcap       Written by the "termcap" output
                                         function
       /usr/tmp/screens/screen-exchange  or
       /tmp/screen-exchange              screen `interprocess
```

```
                                        communication buffer'
       hardcopy.[0-9]                   Screen images created by the
                                        hardcopy function
       screenlog.[0-9]                  Output log files created by the
                                        log function
       /usr/lib/terminfo/?/*            or
       /etc/termcap                     Terminal capability databases
       /etc/utmp                        Login records
       $LOCKPRG                         Program that locks a terminal.
SEE ALSO         top
       termcap(5), utmp(5), vi(1), captoinfo(1), tic(1)
AUTHORS          top
       Originally created by Oliver Laumann. For a long time maintained
       and developed by Juergen Weigert, Michael Schroeder, Micah Cowan
       and Sadrul Habib Chowdhury. Since 2015 maintained and developed
       by Amadeusz Slawinski <amade@asmblr.net> and Alexander Naumov
       <alexander_naumov@opensuse.org>.
COPYLEFT         top
       Copyright (c) 2018
           Alexander Naumov <alexander_naumov@opensuse.org>
           Amadeusz Slawinski <amade@asmblr.net>
       Copyright (c) 2015-2017
            Juergen Weigert <jnweiger@immd4.informatik.uni-erlangen.de>
            Alexander Naumov <alexander_naumov@opensuse.org>
            Amadeusz Slawinski <amade@asmblr.net>
       Copyright (c) 2010-2015
            Juergen Weigert <jnweiger@immd4.informatik.uni-erlangen.de>
            Sadrul Habib Chowdhury <sadrul@users.sourceforge.net>
       Copyright (c) 2008, 2009
            Juergen Weigert <jnweiger@immd4.informatik.uni-erlangen.de>
            Michael Schroeder <mlschroe@immd4.informatik.uni-erlangen.de>
            Micah Cowan <micah@cowan.name>
            Sadrul Habib Chowdhury <sadrul@users.sourceforge.net>
       Copyright (C) 1993-2003
            Juergen Weigert <jnweiger@immd4.informatik.uni-erlangen.de>
            Michael Schroeder <mlschroe@immd4.informatik.uni-erlangen.de>
       Copyright (C) 1987 Oliver Laumann
       This program is free software; you can redistribute it and/or
       modify it under the terms of the GNU General Public License as
       published by the Free Software Foundation; either version 3, or
       (at your option) any later version.
```

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
General Public License for more details.
You should have received a copy of the GNU General Public License
along with this program (see the file COPYING); if not, write to
the Free Software Foundation, Inc., 59 Temple Place - Suite 330,
Boston, MA 02111-1307, USA

CONTRIBUTORS top

Eric S. Raymond <esr@thyrsus.com>,
Thomas Renninger <treen@suse.com>,
Axel Beckert <abe@deuxchevaux.org>,
Ken Beal <kbeal@amber.ssd.csd.harris.com>,
Rudolf Koenig <rfkoenig@immd4.informatik.uni-erlangen.de>,
Toerless Eckert <eckert@immd4.informatik.uni-erlangen.de>,
Wayne Davison <davison@borland.com>,
Patrick Wolfe <pat@kai.com, kailand!pat>,
Bart Schaefer <schaefer@cse.ogi.edu>,
Nathan Glasser <nathan@brokaw.lcs.mit.edu>,
Larry W. Virden <lvirden@cas.org>,
Howard Chu <hyc@hanauma.jpl.nasa.gov>,
Tim MacKenzie <tym@dibbler.cs.monash.edu.au>,
Markku Jarvinen <mta@{cc,cs,ee}.tut.fi>,
Marc Boucher <marc@CAM.ORG>,
Doug Siebert <dsiebert@isca.uiowa.edu>,
Ken Stillson <stillson@tsfsrv.mitre.org>,
Ian Frechett <frechett@spot.Colorado.EDU>,
Brian Koehmstedt <bpk@gnu.ai.mit.edu>,
Don Smith <djs6015@ultb.isc.rit.edu>,
Frank van der Linden <vdlinden@fwi.uva.nl>,
Martin Schweikert <schweik@cpp.ob.open.de>,
David Vrona <dave@sashimi.lcu.com>,
E. Tye McQueen <tye%spillman.UUCP@uunet.uu.net>,
Matthew Green <mrg@eterna.com.au>,
Christopher Williams <cgw@pobox.com>,
Matt Mosley <mattm@access.digex.net>,
Gregory Neil Shapiro <gshapiro@wpi.WPI.EDU>,
Johannes Zellner <johannes@zellner.org>,
Pablo Averbuj <pablo@averbuj.com>.

VERSION top

This is version 4.3.1. Its roots are a merge of a custom version

2.3PR7 by Wayne Davison and several enhancements to Oliver Laumann's version 2.0. Note that all versions numbered 2.x are copyright by Oliver Laumann.

AVAILABILITY top

The latest official release of screen available via anonymous ftp from ftp.gnu.org/gnu/screen/ or any other GNU distribution site. The home site of screen is savannah.gnu.org/projects/screen/. If you want to help, send a note to screen-devel@gnu.org.

BUGS top

- `dm' (delete mode) and `xs' are not handled correctly (they are ignored). `xn' is treated as a magic-margin indicator.

- Screen has no clue about double-high or double-wide characters. But this is the only area where vttest is allowed to fail.

- It is not possible to change the environment variable $TERMCAP when reattaching under a different terminal type.

- The support of terminfo based systems is very limited. Adding extra capabilities to $TERMCAP may not have any effects.

- Screen does not make use of hardware tabs.

- Screen must be installed as set-uid with owner root on most systems in order to be able to correctly change the owner of the tty device file for each window. Special permission may also be required to write the file "/etc/utmp".

- Entries in "/etc/utmp" are not removed when screen is killed with SIGKILL. This will cause some programs (like "w" or "rwho") to advertise that a user is logged on who really isn't.

- Screen may give a strange warning when your tty has no utmp entry.

- When the modem line was hung up, screen may not automatically detach (or quit) unless the device driver is configured to send a HANGUP signal. To detach a screen session use the -D or -d command line option.

- If a password is set, the command line options -d and -D still detach a session without asking.

- Both "breaktype" and "defbreaktype" change the break generating method used by all terminal devices. The first should change a window specific setting, where the latter should change only the default for new windows.

- When attaching to a multiuser session, the user's .screenrc file is not sourced. Each user's personal settings have to be included in the .screenrc file from which the session is booted, or have to be changed manually.

- A weird imagination is most useful to gain full advantage of all the features.

- Send bug-reports, fixes, enhancements, t-shirts, money, beer & pizza to screen-devel@gnu.org.

COLOPHON top

This page is part of the screen (screen manager) project. Information about the project can be found at ⟨http://www.gnu.org/software/screen/⟩. If you have a bug report for this manual page, see ⟨https://savannah.gnu.org/bugs/?func=additem&group=screen⟩. This page was obtained from the project's upstream Git repository ⟨https://savannah.gnu.org/git/?group=screen⟩ on 2021-06-20. (At that time, the date of the most recent commit that was found in the repository was 2020-12-17.) If you discover any rendering problems in this HTML version of the page, or you believe there is a better or more up-to-date source for the page, or you have corrections or improvements to the information in this COLOPHON (which is not part of the original manual page), send a mail to man-pages@man7.org

4 find

```
FIND(1)                  General Commands Manual                 FIND(1)
NAME         top
       find - search for files in a directory hierarchy
SYNOPSIS         top
       find [-H] [-L] [-P] [-D debugopts] [-Olevel] [starting-point...]
       [expression]
DESCRIPTION         top
       This manual page documents the GNU version of find.  GNU find
       searches the directory tree rooted at each given starting-point
       by evaluating the given expression from left to right, according
       to the rules of precedence (see section OPERATORS), until the
       outcome is known (the left hand side is false for and operations,
       true for or), at which point find moves on to the next file name.
       If no starting-point is specified, `.' is assumed.

       If you are using find in an environment where security is
       important (for example if you are using it to search directories
       that are writable by other users), you should read the `Security
       Considerations' chapter of the findutils documentation, which is
       called Finding Files and comes with findutils.  That document
       also includes a lot more detail and discussion than this manual
       page, so you may find it a more useful source of information.
OPTIONS         top
       The -H, -L and -P options control the treatment of symbolic
       links.  Command-line arguments following these are taken to be
       names of files or directories to be examined, up to the first
       argument that begins with `-', or the argument `(' or `!'.  That
       argument and any following arguments are taken to be the
       expression describing what is to be searched for.  If no paths
       are given, the current directory is used.  If no expression is
       given, the expression -print is used (but you should probably
       consider using -print0 instead, anyway).

       This manual page talks about `options' within the expression
       list.  These options control the behaviour of find but are
       specified immediately after the last path name.  The five `real'
       options -H, -L, -P, -D and -O must appear before the first path
       name, if at all.  A double dash -- could theoretically be used to
       signal that any remaining arguments are not options, but this
```

does not really work due to the way find determines the end of the following path arguments: it does that by reading until an expression argument comes (which also starts with a `-'). Now, if a path argument would start with a `-', then find would treat it as expression argument instead. Thus, to ensure that all start points are taken as such, and especially to prevent that wildcard patterns expanded by the calling shell are not mistakenly treated as expression arguments, it is generally safer to prefix wildcards or dubious path names with either `./' or to use absolute path names starting with '/'. Alternatively, it is generally safe though non-portable to use the GNU option -files0-from to pass arbitrary starting points to find.

-P Never follow symbolic links. This is the default behaviour. When find examines or prints information about files, and the file is a symbolic link, the information used shall be taken from the properties of the symbolic link itself.

-L Follow symbolic links. When find examines or prints information about files, the information used shall be taken from the properties of the file to which the link points, not from the link itself (unless it is a broken symbolic link or find is unable to examine the file to which the link points). Use of this option implies -noleaf. If you later use the -P option, -noleaf will still be in effect. If -L is in effect and find discovers a symbolic link to a subdirectory during its search, the subdirectory pointed to by the symbolic link will be searched.

When the -L option is in effect, the -type predicate will always match against the type of the file that a symbolic link points to rather than the link itself (unless the symbolic link is broken). Actions that can cause symbolic links to become broken while find is executing (for example -delete) can give rise to confusing behaviour. Using -L causes the -lname and -ilname predicates always to return false.

-H Do not follow symbolic links, except while processing the

command line arguments. When find examines or prints information about files, the information used shall be taken from the properties of the symbolic link itself. The only exception to this behaviour is when a file specified on the command line is a symbolic link, and the link can be resolved. For that situation, the information used is taken from whatever the link points to (that is, the link is followed). The information about the link itself is used as a fallback if the file pointed to by the symbolic link cannot be examined. If -H is in effect and one of the paths specified on the command line is a symbolic link to a directory, the contents of that directory will be examined (though of course -maxdepth 0 would prevent this).

If more than one of -H, -L and -P is specified, each overrides the others; the last one appearing on the command line takes effect. Since it is the default, the -P option should be considered to be in effect unless either -H or -L is specified.

GNU find frequently stats files during the processing of the command line itself, before any searching has begun. These options also affect how those arguments are processed. Specifically, there are a number of tests that compare files listed on the command line against a file we are currently considering. In each case, the file specified on the command line will have been examined and some of its properties will have been saved. If the named file is in fact a symbolic link, and the -P option is in effect (or if neither -H nor -L were specified), the information used for the comparison will be taken from the properties of the symbolic link. Otherwise, it will be taken from the properties of the file the link points to. If find cannot follow the link (for example because it has insufficient privileges or the link points to a nonexistent file) the properties of the link itself will be used.

When the -H or -L options are in effect, any symbolic links listed as the argument of -newer will be dereferenced, and the timestamp will be taken from the file to which the symbolic link points. The same consideration applies to -newerXY, -anewer and -cnewer.

The -follow option has a similar effect to -L, though it takes effect at the point where it appears (that is, if -L is not used but -follow is, any symbolic links appearing after -follow on the command line will be dereferenced, and those before it will not).

-D debugopts

Print diagnostic information; this can be helpful to diagnose problems with why find is not doing what you want. The list of debug options should be comma separated. Compatibility of the debug options is not guaranteed between releases of findutils. For a complete list of valid debug options, see the output of find -D help. Valid debug options include

exec Show diagnostic information relating to -exec, -execdir, -ok and -okdir

opt Prints diagnostic information relating to the optimisation of the expression tree; see the -O option.

rates Prints a summary indicating how often each predicate succeeded or failed.

search Navigate the directory tree verbosely.

stat Print messages as files are examined with the stat and lstat system calls. The find program tries to minimise such calls.

tree Show the expression tree in its original and optimised form.

all Enable all of the other debug options (but help).

help Explain the debugging options.

-Olevel

Enables query optimisation. The find program reorders tests to speed up execution while preserving the overall

effect; that is, predicates with side effects are not reordered relative to each other. The optimisations performed at each optimisation level are as follows.

0 Equivalent to optimisation level 1.

1 This is the default optimisation level and corresponds to the traditional behaviour. Expressions are reordered so that tests based only on the names of files (for example -name and -regex) are performed first.

2 Any -type or -xtype tests are performed after any tests based only on the names of files, but before any tests that require information from the inode. On many modern versions of Unix, file types are returned by readdir() and so these predicates are faster to evaluate than predicates which need to stat the file first. If you use the -fstype FOO predicate and specify a filesystem type FOO which is not known (that is, present in `/etc/mtab') at the time find starts, that predicate is equivalent to -false.

3 At this optimisation level, the full cost-based query optimiser is enabled. The order of tests is modified so that cheap (i.e. fast) tests are performed first and more expensive ones are performed later, if necessary. Within each cost band, predicates are evaluated earlier or later according to whether they are likely to succeed or not. For -o, predicates which are likely to succeed are evaluated earlier, and for -a, predicates which are likely to fail are evaluated earlier.

The cost-based optimiser has a fixed idea of how likely any given test is to succeed. In some cases the probability takes account of the specific nature of the test (for example, -type f is assumed to be more likely to succeed than -type c). The cost-based optimiser is

currently being evaluated. If it does not actually improve the performance of find, it will be removed again. Conversely, optimisations that prove to be reliable, robust and effective may be enabled at lower optimisation levels over time. However, the default behaviour (i.e. optimisation level 1) will not be changed in the 4.3.x release series. The findutils test suite runs all the tests on find at each optimisation level and ensures that the result is the same.

EXPRESSION top

The part of the command line after the list of starting points is the expression. This is a kind of query specification describing how we match files and what we do with the files that were matched. An expression is composed of a sequence of things:

Tests Tests return a true or false value, usually on the basis of some property of a file we are considering. The -empty test for example is true only when the current file is empty.

Actions

Actions have side effects (such as printing something on the standard output) and return either true or false, usually based on whether or not they are successful. The -print action for example prints the name of the current file on the standard output.

Global options

Global options affect the operation of tests and actions specified on any part of the command line. Global options always return true. The -depth option for example makes find traverse the file system in a depth-first order.

Positional options

Positional options affect only tests or actions which follow them. Positional options always return true. The -regextype option for example is positional, specifying the regular expression dialect for regular expressions occurring later on the command line.

Operators

Operators join together the other items within the expression. They include for example -o (meaning logical OR) and -a (meaning logical AND). Where an operator is missing, -a is assumed.

The -print action is performed on all files for which the whole expression is true, unless it contains an action other than -prune or -quit. Actions which inhibit the default -print are -delete, -exec, -execdir, -ok, -okdir, -fls, -fprint, -fprintf, -ls, -print and -printf.

The -delete action also acts like an option (since it implies -depth).

POSITIONAL OPTIONS

Positional options always return true. They affect only tests occurring later on the command line.

-daystart

Measure times (for -amin, -atime, -cmin, -ctime, -mmin, and -mtime) from the beginning of today rather than from 24 hours ago. This option only affects tests which appear later on the command line.

-follow

Deprecated; use the -L option instead. Dereference symbolic links. Implies -noleaf. The -follow option affects only those tests which appear after it on the command line. Unless the -H or -L option has been specified, the position of the -follow option changes the behaviour of the -newer predicate; any files listed as the argument of -newer will be dereferenced if they are symbolic links. The same consideration applies to -newerXY, -anewer and -cnewer. Similarly, the -type predicate will always match against the type of the file that a symbolic link points to rather than the link itself. Using -follow causes the -lname and -ilname predicates always to return false.

-regextype type

Changes the regular expression syntax understood by -regex

and -iregex tests which occur later on the command line. To see which regular expression types are known, use -regextype help. The Texinfo documentation (see SEE ALSO) explains the meaning of and differences between the various types of regular expression.

-warn, -nowarn
Turn warning messages on or off. These warnings apply only to the command line usage, not to any conditions that find might encounter when it searches directories. The default behaviour corresponds to -warn if standard input is a tty, and to -nowarn otherwise. If a warning message relating to command-line usage is produced, the exit status of find is not affected. If the POSIXLY_CORRECT environment variable is set, and -warn is also used, it is not specified which, if any, warnings will be active.

GLOBAL OPTIONS
Global options always return true. Global options take effect even for tests which occur earlier on the command line. To prevent confusion, global options should specified on the command-line after the list of start points, just before the first test, positional option or action. If you specify a global option in some other place, find will issue a warning message explaining that this can be confusing.

The global options occur after the list of start points, and so are not the same kind of option as -L, for example.

-d A synonym for -depth, for compatibility with FreeBSD, NetBSD, MacOS X and OpenBSD.

-depth Process each directory's contents before the directory itself. The -delete action also implies -depth.

-files0-from file
Read the starting points from file instead of getting them on the command line. In contrast to the known limitations of passing starting points via arguments on the command line, namely the limitation of the amount of file names, and the inherent ambiguity of file names clashing with

option names, using this option allows to safely pass an arbitrary number of starting points to find.

Using this option and passing starting points on the command line is mutually exclusive, and is therefore not allowed at the same time.

The file argument is mandatory. One can use -files0-from - to read the list of starting points from the standard input stream, and e.g. from a pipe. In this case, the actions -ok and -okdir are not allowed, because they would obviously interfere with reading from standard input in order to get a user confirmation.

The starting points in file have to be separated by ASCII NUL characters. Two consecutive NUL characters, i.e., a starting point with a Zero-length file name is not allowed and will lead to an error diagnostic followed by a non-Zero exit code later. The given file has to contain at least one starting point, i.e., an empty input file will be diagnosed as well.

The processing of the starting points is otherwise as usual, e.g. find will recurse into subdirectories unless otherwise prevented. To process only the starting points, one can additionally pass -maxdepth 0.

Further notes: if a file is listed more than once in the input file, it is unspecified whether it is visited more than once. If the file is mutated during the operation of find, the result is unspecified as well. Finally, the seek position within the named file at the time find exits, be it with -quit or in any other way, is also unspecified. By "unspecified" here is meant that it may or may not work or do any specific thing, and that the behavior may change from platform to platform, or from findutils release to release.

-help, --help
Print a summary of the command-line usage of find and exit.

-ignore_readdir_race
       Normally, find will emit an error message when it fails to stat a file. If you give this option and a file is deleted between the time find reads the name of the file from the directory and the time it tries to stat the file, no error message will be issued. This also applies to files or directories whose names are given on the command line. This option takes effect at the time the command line is read, which means that you cannot search one part of the filesystem with this option on and part of it with this option off (if you need to do that, you will need to issue two find commands instead, one with the option and one without it).

       Furthermore, find with the -ignore_readdir_race option will ignore errors of the -delete action in the case the file has disappeared since the parent directory was read: it will not output an error diagnostic, and the return code of the -delete action will be true.

-maxdepth levels
       Descend at most levels (a non-negative integer) levels of directories below the starting-points. Using -maxdepth 0 means only apply the tests and actions to the starting-points themselves.

-mindepth levels
       Do not apply any tests or actions at levels less than levels (a non-negative integer). Using -mindepth 1 means process all files except the starting-points.

-mount Don't descend directories on other filesystems. An
       alternate name for -xdev, for compatibility with some other versions of find.

-noignore_readdir_race
       Turns off the effect of -ignore_readdir_race.

-noleaf
       Do not optimize by assuming that directories contain 2

fewer subdirectories than their hard link count. This option is needed when searching filesystems that do not follow the Unix directory-link convention, such as CD-ROM or MS-DOS filesystems or AFS volume mount points. Each directory on a normal Unix filesystem has at least 2 hard links: its name and its `.' entry. Additionally, its subdirectories (if any) each have a `..' entry linked to that directory. When find is examining a directory, after it has statted 2 fewer subdirectories than the directory's link count, it knows that the rest of the entries in the directory are non-directories (`leaf' files in the directory tree). If only the files' names need to be examined, there is no need to stat them; this gives a significant increase in search speed.

-version, --version
Print the find version number and exit.

-xdev Don't descend directories on other filesystems.

TESTS

Some tests, for example -newerXY and -samefile, allow comparison between the file currently being examined and some reference file specified on the command line. When these tests are used, the interpretation of the reference file is determined by the options -H, -L and -P and any previous -follow, but the reference file is only examined once, at the time the command line is parsed. If the reference file cannot be examined (for example, the stat(2) system call fails for it), an error message is issued, and find exits with a nonzero status.

A numeric argument n can be specified to tests (like -amin, -mtime, -gid, -inum, -links, -size, -uid and -used) as

+n for greater than n,

-n for less than n,

n for exactly n.

Supported tests:

-amin n
       File was last accessed less than, more than or exactly n minutes ago.

-anewer reference
       Time of the last access of the current file is more recent than that of the last data modification of the reference file. If reference is a symbolic link and the -H option or the -L option is in effect, then the time of the last data modification of the file it points to is always used.

-atime n
       File was last accessed less than, more than or exactly n*24 hours ago. When find figures out how many 24-hour periods ago the file was last accessed, any fractional part is ignored, so to match -atime +1, a file has to have been accessed at least two days ago.

-cmin n
       File's status was last changed less than, more than or exactly n minutes ago.

-cnewer reference
       Time of the last status change of the current file is more recent than that of the last data modification of the reference file. If reference is a symbolic link and the -H option or the -L option is in effect, then the time of the last data modification of the file it points to is always used.

-ctime n
       File's status was last changed less than, more than or exactly n*24 hours ago. See the comments for -atime to understand how rounding affects the interpretation of file status change times.

-empty File is empty and is either a regular file or a directory.

-executable
       Matches files which are executable and directories which

are searchable (in a file name resolution sense) by the current user. This takes into account access control lists and other permissions artefacts which the -perm test ignores. This test makes use of the access(2) system call, and so can be fooled by NFS servers which do UID mapping (or root-squashing), since many systems implement access(2) in the client's kernel and so cannot make use of the UID mapping information held on the server. Because this test is based only on the result of the access(2) system call, there is no guarantee that a file for which this test succeeds can actually be executed.

-false Always false.

-fstype type
File is on a filesystem of type type. The valid filesystem types vary among different versions of Unix; an incomplete list of filesystem types that are accepted on some version of Unix or another is: ufs, 4.2, 4.3, nfs, tmp, mfs, S51K, S52K. You can use -printf with the %F directive to see the types of your filesystems.

-gid n File's numeric group ID is less than, more than or exactly n.

-group gname
File belongs to group gname (numeric group ID allowed).

-ilname pattern
Like -lname, but the match is case insensitive. If the -L option or the -follow option is in effect, this test returns false unless the symbolic link is broken.

-iname pattern
Like -name, but the match is case insensitive. For example, the patterns `fo*' and `F??' match the file names `Foo', `FOO', `foo', `fOo', etc. The pattern `*foo*` will also match a file called '.foobar'.

-inum n
File has inode number smaller than, greater than or

exactly n. It is normally easier to use the -samefile test instead.

-ipath pattern
Like -path. but the match is case insensitive.

-iregex pattern
Like -regex, but the match is case insensitive.

-iwholename pattern
See -ipath. This alternative is less portable than -ipath.

-links n
File has less than, more than or exactly n hard links.

-lname pattern
File is a symbolic link whose contents match shell pattern pattern. The metacharacters do not treat `/' or `.' specially. If the -L option or the -follow option is in effect, this test returns false unless the symbolic link is broken.

-mmin n
File's data was last modified less than, more than or exactly n minutes ago.

-mtime n
File's data was last modified less than, more than or exactly n*24 hours ago. See the comments for -atime to understand how rounding affects the interpretation of file modification times.

-name pattern
Base of file name (the path with the leading directories removed) matches shell pattern pattern. Because the leading directories are removed, the file names considered for a match with -name will never include a slash, so `-name a/b' will never match anything (you probably need to use -path instead). A warning is issued if you try to do this, unless the environment variable POSIXLY_CORRECT

is set. The metacharacters (`*', `?', and `[]') match a `.' at the start of the base name (this is a change in findutils-4.2.2; see section STANDARDS CONFORMANCE below). To ignore a directory and the files under it, use -prune rather than checking every file in the tree; see an example in the description of that action. Braces are not recognised as being special, despite the fact that some shells including Bash imbue braces with a special meaning in shell patterns. The filename matching is performed with the use of the fnmatch(3) library function. Don't forget to enclose the pattern in quotes in order to protect it from expansion by the shell.

-newer reference

Time of the last data modification of the current file is more recent than that of the last data modification of the reference file. If reference is a symbolic link and the -H option or the -L option is in effect, then the time of the last data modification of the file it points to is always used.

-newerXY reference

Succeeds if timestamp X of the file being considered is newer than timestamp Y of the file reference. The letters X and Y can be any of the following letters:

a The access time of the file reference
B The birth time of the file reference
c The inode status change time of reference
m The modification time of the file reference
t reference is interpreted directly as a time

Some combinations are invalid; for example, it is invalid for X to be t. Some combinations are not implemented on all systems; for example B is not supported on all systems. If an invalid or unsupported combination of XY is specified, a fatal error results. Time specifications are interpreted as for the argument to the -d option of GNU date. If you try to use the birth time of a reference file, and the birth time cannot be determined, a fatal error message results. If you specify a test which refers

to the birth time of files being examined, this test will fail for any files where the birth time is unknown.

-nogroup
No group corresponds to file's numeric group ID.

-nouser
No user corresponds to file's numeric user ID.

-path pattern
File name matches shell pattern pattern. The metacharacters do not treat `/' or `.' specially; so, for example,
find . -path "./sr*sc"
will print an entry for a directory called ./src/misc (if one exists). To ignore a whole directory tree, use -prune rather than checking every file in the tree. Note that the pattern match test applies to the whole file name, starting from one of the start points named on the command line. It would only make sense to use an absolute path name here if the relevant start point is also an absolute path. This means that this command will never match anything:
find bar -path /foo/bar/myfile -print
Find compares the -path argument with the concatenation of a directory name and the base name of the file it's examining. Since the concatenation will never end with a slash, -path arguments ending in a slash will match nothing (except perhaps a start point specified on the command line). The predicate -path is also supported by HP-UX find and is part of the POSIX 2008 standard.

-perm mode
File's permission bits are exactly mode (octal or symbolic). Since an exact match is required, if you want to use this form for symbolic modes, you may have to specify a rather complex mode string. For example `-perm g=w' will only match files which have mode 0020 (that is, ones for which group write permission is the only permission set). It is more likely that you will want to use the `/' or `-' forms, for example `-perm -g=w', which

matches any file with group write permission. See the EXAMPLES section for some illustrative examples.

-perm -mode
All of the permission bits mode are set for the file. Symbolic modes are accepted in this form, and this is usually the way in which you would want to use them. You must specify `u', `g' or `o' if you use a symbolic mode. See the EXAMPLES section for some illustrative examples.

-perm /mode
Any of the permission bits mode are set for the file. Symbolic modes are accepted in this form. You must specify `u', `g' or `o' if you use a symbolic mode. See the EXAMPLES section for some illustrative examples. If no permission bits in mode are set, this test matches any file (the idea here is to be consistent with the behaviour of -perm -000).

-perm +mode
This is no longer supported (and has been deprecated since 2005). Use -perm /mode instead.

-readable
Matches files which are readable by the current user. This takes into account access control lists and other permissions artefacts which the -perm test ignores. This test makes use of the access(2) system call, and so can be fooled by NFS servers which do UID mapping (or root-squashing), since many systems implement access(2) in the client's kernel and so cannot make use of the UID mapping information held on the server.

-regex pattern
File name matches regular expression pattern. This is a match on the whole path, not a search. For example, to match a file named ./fubar3, you can use the regular expression `.*bar.' or `.*b.*3', but not `f.*r3'. The regular expressions understood by find are by default Emacs Regular Expressions (except that `.' matches newline), but this can be changed with the -regextype

option.

-samefile name
File refers to the same inode as name. When -L is in effect, this can include symbolic links.

-size n[cwbkMG]
File uses less than, more than or exactly n units of space, rounding up. The following suffixes can be used:

`b' for 512-byte blocks (this is the default if no suffix is used)

`c' for bytes

`w' for two-byte words

`k' for kibibytes (KiB, units of 1024 bytes)

`M' for mebibytes (MiB, units of 1024 * 1024 = 1048576 bytes)

`G' for gibibytes (GiB, units of 1024 * 1024 * 1024 = 1073741824 bytes)

The size is simply the st_size member of the struct stat populated by the lstat (or stat) system call, rounded up as shown above. In other words, it's consistent with the result you get for ls -l. Bear in mind that the `%k' and `%b' format specifiers of -printf handle sparse files differently. The `b' suffix always denotes 512-byte blocks and never 1024-byte blocks, which is different to the behaviour of -ls.

The + and - prefixes signify greater than and less than, as usual; i.e., an exact size of n units does not match. Bear in mind that the size is rounded up to the next unit. Therefore -size -1M is not equivalent to -size -1048576c. The former only matches empty files, the latter matches files from 0 to 1,048,575 bytes.

```
-true  Always true.

-type c
       File is of type c:

       b      block (buffered) special

       c      character (unbuffered) special

       d      directory

       p      named pipe (FIFO)

       f      regular file

       l      symbolic link; this is never true if the -L option
              or the -follow option is in effect, unless the
              symbolic link is broken.  If you want to search for
              symbolic links when -L is in effect, use -xtype.

       s      socket

       D      door (Solaris)

       To search for more than one type at once, you can supply
       the combined list of type letters separated by a comma `,'
       (GNU extension).

-uid n File's numeric user ID is less than, more than or exactly
       n.

-used n
       File was last accessed less than, more than or exactly n
       days after its status was last changed.

-user uname
       File is owned by user uname (numeric user ID allowed).

-wholename pattern
       See -path.  This alternative is less portable than -path.
```

-writable
Matches files which are writable by the current user. This takes into account access control lists and other permissions artefacts which the -perm test ignores. This test makes use of the access(2) system call, and so can be fooled by NFS servers which do UID mapping (or root-squashing), since many systems implement access(2) in the client's kernel and so cannot make use of the UID mapping information held on the server.

-xtype c
The same as -type unless the file is a symbolic link. For symbolic links: if the -H or -P option was specified, true if the file is a link to a file of type c; if the -L option has been given, true if c is `l'. In other words, for symbolic links, -xtype checks the type of the file that -type does not check.

-context pattern
(SELinux only) Security context of the file matches glob pattern.

ACTIONS
-delete
Delete files; true if removal succeeded. If the removal failed, an error message is issued. If -delete fails, find's exit status will be nonzero (when it eventually exits). Use of -delete automatically turns on the `-depth' option.

Warnings: Don't forget that the find command line is evaluated as an expression, so putting -delete first will make find try to delete everything below the starting points you specified. When testing a find command line that you later intend to use with -delete, you should explicitly specify -depth in order to avoid later surprises. Because -delete implies -depth, you cannot usefully use -prune and -delete together.

Together with the -ignore_readdir_race option, find will ignore errors of the -delete action in the case the file

has disappeared since the parent directory was read: it will not output an error diagnostic, and the return code of the -delete action will be true.

-exec command ;

Execute command; true if 0 status is returned. All following arguments to find are taken to be arguments to the command until an argument consisting of `;' is encountered. The string `{}' is replaced by the current file name being processed everywhere it occurs in the arguments to the command, not just in arguments where it is alone, as in some versions of find. Both of these constructions might need to be escaped (with a `\') or quoted to protect them from expansion by the shell. See the EXAMPLES section for examples of the use of the -exec option. The specified command is run once for each matched file. The command is executed in the starting directory. There are unavoidable security problems surrounding use of the -exec action; you should use the -execdir option instead.

-exec command {} +

This variant of the -exec action runs the specified command on the selected files, but the command line is built by appending each selected file name at the end; the total number of invocations of the command will be much less than the number of matched files. The command line is built in much the same way that xargs builds its command lines. Only one instance of `{}' is allowed within the command, and it must appear at the end, immediately before the `+'; it needs to be escaped (with a `\') or quoted to protect it from interpretation by the shell. The command is executed in the starting directory. If any invocation with the `+' form returns a non-zero value as exit status, then find returns a non-zero exit status. If find encounters an error, this can sometimes cause an immediate exit, so some pending commands may not be run at all. For this reason -exec my-command ... {} + -quit may not result in my-command actually being run. This variant of -exec always returns true.

-execdir command ;

-execdir command {} +

Like -exec, but the specified command is run from the subdirectory containing the matched file, which is not normally the directory in which you started find. As with -exec, the {} should be quoted if find is being invoked from a shell. This a much more secure method for invoking commands, as it avoids race conditions during resolution of the paths to the matched files. As with the -exec action, the `+' form of -execdir will build a command line to process more than one matched file, but any given invocation of command will only list files that exist in the same subdirectory. If you use this option, you must ensure that your PATH environment variable does not reference `.'; otherwise, an attacker can run any commands they like by leaving an appropriately-named file in a directory in which you will run -execdir. The same applies to having entries in PATH which are empty or which are not absolute directory names. If any invocation with the `+' form returns a non-zero value as exit status, then find returns a non-zero exit status. If find encounters an error, this can sometimes cause an immediate exit, so some pending commands may not be run at all. The result of the action depends on whether the + or the ; variant is being used; -execdir command {} + always returns true, while -execdir command {} ; returns true only if command returns 0.

-fls file

True; like -ls but write to file like -fprint. The output file is always created, even if the predicate is never matched. See the UNUSUAL FILENAMES section for information about how unusual characters in filenames are handled.

-fprint file

True; print the full file name into file file. If file does not exist when find is run, it is created; if it does exist, it is truncated. The file names /dev/stdout and

/dev/stderr are handled specially; they refer to the standard output and standard error output, respectively. The output file is always created, even if the predicate is never matched. See the UNUSUAL FILENAMES section for information about how unusual characters in filenames are handled.

-fprint0 file
True; like -print0 but write to file like -fprint. The output file is always created, even if the predicate is never matched. See the UNUSUAL FILENAMES section for information about how unusual characters in filenames are handled.

-fprintf file format
True; like -printf but write to file like -fprint. The output file is always created, even if the predicate is never matched. See the UNUSUAL FILENAMES section for information about how unusual characters in filenames are handled.

-ls True; list current file in ls -dils format on standard output. The block counts are of 1 KB blocks, unless the environment variable POSIXLY_CORRECT is set, in which case 512-byte blocks are used. See the UNUSUAL FILENAMES section for information about how unusual characters in filenames are handled.

-ok command ;
Like -exec but ask the user first. If the user agrees, run the command. Otherwise just return false. If the command is run, its standard input is redirected from /dev/null. This action may not be specified together with the -files0-from option.

The response to the prompt is matched against a pair of regular expressions to determine if it is an affirmative or negative response. This regular expression is obtained from the system if the POSIXLY_CORRECT environment variable is set, or otherwise from find's message translations. If the system has no suitable definition,

find's own definition will be used. In either case, the interpretation of the regular expression itself will be affected by the environment variables LC_CTYPE (character classes) and LC_COLLATE (character ranges and equivalence classes).

-okdir command ;
Like -execdir but ask the user first in the same way as for -ok. If the user does not agree, just return false. If the command is run, its standard input is redirected from /dev/null. This action may not be specified together with the -files0-from option.

-print True; print the full file name on the standard output, followed by a newline. If you are piping the output of find into another program and there is the faintest possibility that the files which you are searching for might contain a newline, then you should seriously consider using the -print0 option instead of -print. See the UNUSUAL FILENAMES section for information about how unusual characters in filenames are handled.

-print0
True; print the full file name on the standard output, followed by a null character (instead of the newline character that -print uses). This allows file names that contain newlines or other types of white space to be correctly interpreted by programs that process the find output. This option corresponds to the -0 option of xargs.

-printf format
True; print format on the standard output, interpreting `\' escapes and `%' directives. Field widths and precisions can be specified as with the printf(3) C function. Please note that many of the fields are printed as %s rather than %d, and this may mean that flags don't work as you might expect. This also means that the `-' flag does work (it forces fields to be left-aligned). Unlike -print, -printf does not add a newline at the end of the string. The escapes and directives are:

\a Alarm bell.

\b Backspace.

\c Stop printing from this format immediately and flush the output.

\f Form feed.

\n Newline.

\r Carriage return.

\t Horizontal tab.

\v Vertical tab.

\0 ASCII NUL.

\\ A literal backslash (`\').

\NNN The character whose ASCII code is NNN (octal).

A `\' character followed by any other character is treated as an ordinary character, so they both are printed.

%% A literal percent sign.

%a File's last access time in the format returned by the C ctime(3) function.

%Ak File's last access time in the format specified by k, which is either `@' or a directive for the C strftime(3) function. The following shows an incomplete list of possible values for k. Please refer to the documentation of strftime(3) for the full list. Some of the conversion specification characters might not be available on all systems, due to differences in the implementation of the strftime(3) library function.

```
@       seconds since Jan. 1, 1970, 00:00 GMT, with
        fractional part.

Time fields:

H       hour (00..23)

I       hour (01..12)

k       hour ( 0..23)

l       hour ( 1..12)

M       minute (00..59)

p       locale's AM or PM

r       time, 12-hour (hh:mm:ss [AP]M)

S       Second (00.00 .. 61.00).  There is a
        fractional part.

T       time, 24-hour (hh:mm:ss.xxxxxxxxxx)

+       Date and time, separated by `+', for example
        `2004-04-28+22:22:05.0'.  This is a GNU
        extension.  The time is given in the current
        timezone (which may be affected by setting
        the TZ environment variable).  The seconds
        field includes a fractional part.

X       locale's time representation (H:M:S).  The
        seconds field includes a fractional part.

Z       time zone (e.g., EDT), or nothing if no time
        zone is determinable

Date fields:

a       locale's abbreviated weekday name (Sun..Sat)
```

A locale's full weekday name, variable length (Sunday..Saturday)

b locale's abbreviated month name (Jan..Dec)

B locale's full month name, variable length (January..December)

c locale's date and time (Sat Nov 04 12:02:33 EST 1989). The format is the same as for ctime(3) and so to preserve compatibility with that format, there is no fractional part in the seconds field.

d day of month (01..31)

D date (mm/dd/yy)

F date (yyyy-mm-dd)

h same as b

j day of year (001..366)

m month (01..12)

U week number of year with Sunday as first day of week (00..53)

w day of week (0..6)

W week number of year with Monday as first day of week (00..53)

x locale's date representation (mm/dd/yy)

y last two digits of year (00..99)

Y year (1970...)

%b The amount of disk space used for this file in 512-byte blocks. Since disk space is allocated in multiples of the filesystem block size this is usually greater than %s/512, but it can also be smaller if the file is a sparse file.

%c File's last status change time in the format returned by the C ctime(3) function.

%Ck File's last status change time in the format specified by k, which is the same as for %A.

%d File's depth in the directory tree; 0 means the file is a starting-point.

%D The device number on which the file exists (the st_dev field of struct stat), in decimal.

%f Print the basename; the file's name with any leading directories removed (only the last element). For /, the result is `/'. See the EXAMPLES section for an example.

%F Type of the filesystem the file is on; this value can be used for -fstype.

%g File's group name, or numeric group ID if the group has no name.

%G File's numeric group ID.

%h Dirname; the Leading directories of the file's name (all but the last element). If the file name contains no slashes (since it is in the current directory) the %h specifier expands to `.'. For files which are themselves directories and contain a slash (including /), %h expands to the empty string. See the EXAMPLES section for an example.

%H Starting-point under which file was found.

%i File's inode number (in decimal).

%k The amount of disk space used for this file in 1 KB blocks. Since disk space is allocated in multiples of the filesystem block size this is usually greater than %s/1024, but it can also be smaller if the file is a sparse file.

%l Object of symbolic link (empty string if file is not a symbolic link).

%m File's permission bits (in octal). This option uses the `traditional' numbers which most Unix implementations use, but if your particular implementation uses an unusual ordering of octal permissions bits, you will see a difference between the actual value of the file's mode and the output of %m. Normally you will want to have a leading zero on this number, and to do this, you should use the # flag (as in, for example, `%#m').

%M File's permissions (in symbolic form, as for ls). This directive is supported in findutils 4.2.5 and later.

%n Number of hard links to file.

%p File's name.

%P File's name with the name of the starting-point under which it was found removed.

%s File's size in bytes.

%S File's sparseness. This is calculated as (BLOCKSIZE*st_blocks / st_size). The exact value you will get for an ordinary file of a certain length is system-dependent. However, normally sparse files will have values less than 1.0, and files which use indirect blocks may have a value which is greater than 1.0. In general the number

of blocks used by a file is file system dependent. The value used for BLOCKSIZE is system-dependent, but is usually 512 bytes. If the file size is zero, the value printed is undefined. On systems which lack support for st_blocks, a file's sparseness is assumed to be 1.0.

%t File's last modification time in the format returned by the C ctime(3) function.

%Tk File's last modification time in the format specified by k, which is the same as for %A.

%u File's user name, or numeric user ID if the user has no name.

%U File's numeric user ID.

%y File's type (like in ls -l), U=unknown type (shouldn't happen)

%Y File's type (like %y), plus follow symbolic links: `L'=loop, `N'=nonexistent, `?' for any other error when determining the type of the target of a symbolic link.

%Z (SELinux only) file's security context.

%{ %[%(
Reserved for future use.

A `%' character followed by any other character is discarded, but the other character is printed (don't rely on this, as further format characters may be introduced). A `%' at the end of the format argument causes undefined behaviour since there is no following character. In some locales, it may hide your door keys, while in others it may remove the final page from the novel you are reading.

The %m and %d directives support the #, 0 and + flags, but the other directives do not, even if they print numbers.

Numeric directives that do not support these flags include G, U, b, D, k and n. The `-' format flag is supported and changes the alignment of a field from right-justified (which is the default) to left-justified.

See the UNUSUAL FILENAMES section for information about how unusual characters in filenames are handled.

-prune True; if the file is a directory, do not descend into it. If -depth is given, then -prune has no effect. Because -delete implies -depth, you cannot usefully use -prune and -delete together. For example, to skip the directory src/emacs and all files and directories under it, and print the names of the other files found, do something like this:

```
find . -path ./src/emacs -prune -o -print
```

-quit Exit immediately (with return value zero if no errors have occurred). This is different to -prune because -prune only applies to the contents of pruned directories, while -quit simply makes find stop immediately. No child processes will be left running. Any command lines which have been built by -exec ... + or -execdir ... + are invoked before the program is exited. After -quit is executed, no more files specified on the command line will be processed. For example,
`find /tmp/foo /tmp/bar -print -quit` will print only `/tmp/foo`.
One common use of -quit is to stop searching the file system once we have found what we want. For example, if we want to find just a single file we can do this:

```
find / -name needle -print -quit
```

OPERATORS

Listed in order of decreasing precedence:

(expr)

Force precedence. Since parentheses are special to the shell, you will normally need to quote them. Many of the examples in this manual page use backslashes for this purpose: `\(...\)' instead of `(...)'.

```
! expr True if expr is false.  This character will also usually
       need protection from interpretation by the shell.

-not expr
       Same as ! expr, but not POSIX compliant.

expr1 expr2
       Two expressions in a row are taken to be joined with an
       implied -a; expr2 is not evaluated if expr1 is false.

expr1 -a expr2
       Same as expr1 expr2.

expr1 -and expr2
       Same as expr1 expr2, but not POSIX compliant.

expr1 -o expr2
       Or; expr2 is not evaluated if expr1 is true.

expr1 -or expr2
       Same as expr1 -o expr2, but not POSIX compliant.

expr1 , expr2
       List; both expr1 and expr2 are always evaluated.  The
       value of expr1 is discarded; the value of the list is the
       value of expr2.  The comma operator can be useful for
       searching for several different types of thing, but
       traversing the filesystem hierarchy only once.  The
       -fprintf action can be used to list the various matched
       items into several different output files.

Please note that -a when specified implicitly (for example by two
tests appearing without an explicit operator between them) or
explicitly has higher precedence than -o.  This means that find .
-name afile -o -name bfile -print will never print afile.
UNUSUAL FILENAMES         top
Many of the actions of find result in the printing of data which
is under the control of other users.  This includes file names,
sizes, modification times and so forth.  File names are a
potential problem since they can contain any character except
```

`\0' and `/'. Unusual characters in file names can do unexpected and often undesirable things to your terminal (for example, changing the settings of your function keys on some terminals). Unusual characters are handled differently by various actions, as described below.

-print0, -fprint0
Always print the exact filename, unchanged, even if the output is going to a terminal.

-ls, -fls
Unusual characters are always escaped. White space, backslash, and double quote characters are printed using C-style escaping (for example `\f', `\"'). Other unusual characters are printed using an octal escape. Other printable characters (for -ls and -fls these are the characters between octal 041 and 0176) are printed as-is.

-printf, -fprintf
If the output is not going to a terminal, it is printed as-is. Otherwise, the result depends on which directive is in use. The directives %D, %F, %g, %G, %H, %Y, and %y expand to values which are not under control of files' owners, and so are printed as-is. The directives %a, %b, %c, %d, %i, %k, %m, %M, %n, %s, %t, %u and %U have values which are under the control of files' owners but which cannot be used to send arbitrary data to the terminal, and so these are printed as-is. The directives %f, %h, %l, %p and %P are quoted. This quoting is performed in the same way as for GNU ls. This is not the same quoting mechanism as the one used for -ls and -fls. If you are able to decide what format to use for the output of find then it is normally better to use `\0' as a terminator than to use newline, as file names can contain white space and newline characters. The setting of the LC_CTYPE environment variable is used to determine which characters need to be quoted.

-print, -fprint
Quoting is handled in the same way as for -printf and -fprintf. If you are using find in a script or in a

situation where the matched files might have arbitrary names, you should consider using -print0 instead of -print.

The -ok and -okdir actions print the current filename as-is. This may change in a future release.

STANDARDS CONFORMANCE top

For closest compliance to the POSIX standard, you should set the POSIXLY_CORRECT environment variable. The following options are specified in the POSIX standard (IEEE Std 1003.1-2008, 2016 Edition):

-H This option is supported.

-L This option is supported.

-name This option is supported, but POSIX conformance depends on the POSIX conformance of the system's fnmatch(3) library function. As of findutils-4.2.2, shell metacharacters (`*', `?' or `[]' for example) match a leading `.', because IEEE PASC interpretation 126 requires this. This is a change from previous versions of findutils.

-type Supported. POSIX specifies `b', `c', `d', `l', `p', `f' and `s'. GNU find also supports `D', representing a Door, where the OS provides these. Furthermore, GNU find allows multiple types to be specified at once in a comma-separated list.

-ok Supported. Interpretation of the response is according to the `yes' and `no' patterns selected by setting the LC_MESSAGES environment variable. When the POSIXLY_CORRECT environment variable is set, these patterns are taken system's definition of a positive (yes) or negative (no) response. See the system's documentation for nl_langinfo(3), in particular YESEXPR and NOEXPR. When POSIXLY_CORRECT is not set, the patterns are instead taken from find's own message catalogue.

-newer Supported. If the file specified is a symbolic link, it is always dereferenced. This is a change from previous

behaviour, which used to take the relevant time from the symbolic link; see the HISTORY section below.

-perm Supported. If the POSIXLY_CORRECT environment variable is not set, some mode arguments (for example +a+x) which are not valid in POSIX are supported for backward-compatibility.

Other primaries

The primaries -atime, -ctime, -depth, -exec, -group, -links, -mtime, -nogroup, -nouser, -ok, -path, -print, -prune, -size, -user and -xdev are all supported.

The POSIX standard specifies parentheses `(', `)', negation `!' and the logical AND/OR operators -a and -o.

All other options, predicates, expressions and so forth are extensions beyond the POSIX standard. Many of these extensions are not unique to GNU find, however.

The POSIX standard requires that find detects loops:

> The find utility shall detect infinite loops; that is, entering a previously visited directory that is an ancestor of the last file encountered. When it detects an infinite loop, find shall write a diagnostic message to standard error and shall either recover its position in the hierarchy or terminate.

GNU find complies with these requirements. The link count of directories which contain entries which are hard links to an ancestor will often be lower than they otherwise should be. This can mean that GNU find will sometimes optimise away the visiting of a subdirectory which is actually a link to an ancestor. Since find does not actually enter such a subdirectory, it is allowed to avoid emitting a diagnostic message. Although this behaviour may be somewhat confusing, it is unlikely that anybody actually depends on this behaviour. If the leaf optimisation has been turned off with -noleaf, the directory entry will always be examined and the diagnostic message will be issued where it is appropriate. Symbolic links cannot be used to create filesystem

cycles as such, but if the -L option or the -follow option is in use, a diagnostic message is issued when find encounters a loop of symbolic links. As with loops containing hard links, the leaf optimisation will often mean that find knows that it doesn't need to call stat() or chdir() on the symbolic link, so this diagnostic is frequently not necessary.

The -d option is supported for compatibility with various BSD systems, but you should use the POSIX-compliant option -depth instead.

The POSIXLY_CORRECT environment variable does not affect the behaviour of the -regex or -iregex tests because those tests aren't specified in the POSIX standard.

ENVIRONMENT VARIABLES top

LANG Provides a default value for the internationalization variables that are unset or null.

LC_ALL If set to a non-empty string value, override the values of all the other internationalization variables.

LC_COLLATE
The POSIX standard specifies that this variable affects the pattern matching to be used for the -name option. GNU find uses the fnmatch(3) library function, and so support for LC_COLLATE depends on the system library. This variable also affects the interpretation of the response to -ok; while the LC_MESSAGES variable selects the actual pattern used to interpret the response to -ok, the interpretation of any bracket expressions in the pattern will be affected by LC_COLLATE.

LC_CTYPE
This variable affects the treatment of character classes used in regular expressions and also with the -name test, if the system's fnmatch(3) library function supports this. This variable also affects the interpretation of any character classes in the regular expressions used to interpret the response to the prompt issued by -ok. The LC_CTYPE environment variable will also affect which characters are considered to be unprintable when filenames

are printed; see the section UNUSUAL FILENAMES.

LC_MESSAGES
Determines the locale to be used for internationalised messages. If the POSIXLY_CORRECT environment variable is set, this also determines the interpretation of the response to the prompt made by the -ok action.

NLSPATH
Determines the location of the internationalisation message catalogues.

PATH Affects the directories which are searched to find the executables invoked by -exec, -execdir, -ok and -okdir.

POSIXLY_CORRECT
Determines the block size used by -ls and -fls. If POSIXLY_CORRECT is set, blocks are units of 512 bytes. Otherwise they are units of 1024 bytes.

Setting this variable also turns off warning messages (that is, implies -nowarn) by default, because POSIX requires that apart from the output for -ok, all messages printed on stderr are diagnostics and must result in a non-zero exit status.

When POSIXLY_CORRECT is not set, -perm +zzz is treated just like -perm /zzz if +zzz is not a valid symbolic mode. When POSIXLY_CORRECT is set, such constructs are treated as an error.

When POSIXLY_CORRECT is set, the response to the prompt made by the -ok action is interpreted according to the system's message catalogue, as opposed to according to find's own message translations.

TZ Affects the time zone used for some of the time-related format directives of -printf and -fprintf.

EXAMPLES top

Simple `find|xargs` approach

- Find files named core in or below the directory /tmp and

delete them.

```
$ find /tmp -name core -type f -print | xargs /bin/rm -f
```

Note that this will work incorrectly if there are any filenames containing newlines, single or double quotes, or spaces.

Safer `find -print0 | xargs -0` approach

- Find files named core in or below the directory /tmp and delete them, processing filenames in such a way that file or directory names containing single or double quotes, spaces or newlines are correctly handled.

```
$ find /tmp -name core -type f -print0 | xargs -0 /bin/rm
↪ -f
```

The -name test comes before the -type test in order to avoid having to call stat(2) on every file.

Note that there is still a race between the time find traverses the hierarchy printing the matching filenames, and the time the process executed by xargs works with that file.

Processing arbitrary starting points

- Given that another program proggy pre-filters and creates a huge NUL-separated list of files, process those as starting points, and find all regular, empty files among them:

```
$ proggy | find -files0-from - -maxdepth 0 -type f -empty
```

The use of `-files0-from -` means to read the names of the starting points from standard input, i.e., from the pipe; and -maxdepth 0 ensures that only explicitly those entries are examined without recursing into directories (in the case one of the starting points is one).

Executing a command for each file

- Run file on every file in or below the current directory.

```
$ find . -type f -exec file '{}' \;
```

Notice that the braces are enclosed in single quote marks to protect them from interpretation as shell script punctuation. The semicolon is similarly protected by the use of a backslash, though single quotes could have been used in that case also.

In many cases, one might prefer the `-exec ... +` or better the `-execdir ... +` syntax for performance and security reasons.

Traversing the filesystem just once - for 2 different actions

- Traverse the filesystem just once, listing set-user-ID files and directories into /root/suid.txt and large files into /root/big.txt.

```
$ find / \
    \( -perm -4000 -fprintf /root/suid.txt '%#m %u %p\n'
    ↪  \) , \
    \( -size +100M -fprintf /root/big.txt '%-10s %p\n' \)
```

This example uses the line-continuation character '\' on the first two lines to instruct the shell to continue reading the command on the next line.

Searching files by age

- Search for files in your home directory which have been modified in the last twenty-four hours.

```
$ find $HOME -mtime 0
```

This command works this way because the time since each file was last modified is divided by 24 hours and any remainder is discarded. That means that to match -mtime 0, a file will have to have a modification in the past which is less than 24 hours ago.

Searching files by permissions

- Search for files which are executable but not readable.

```
$ find /sbin /usr/sbin -executable \! -readable -print
```

- Search for files which have read and write permission for their owner, and group, but which other users can read but not write to.

  ```
  $ find . -perm 664
  ```

  Files which meet these criteria but have other permissions bits set (for example if someone can execute the file) will not be matched.

- Search for files which have read and write permission for their owner and group, and which other users can read, without regard to the presence of any extra permission bits (for example the executable bit).

  ```
  $ find . -perm -664
  ```

  This will match a file which has mode 0777, for example.

- Search for files which are writable by somebody (their owner, or their group, or anybody else).

  ```
  $ find . -perm /222
  ```

- Search for files which are writable by either their owner or their group.

  ```
  $ find . -perm /220
  $ find . -perm /u+w,g+w
  $ find . -perm /u=w,g=w
  ```

  All three of these commands do the same thing, but the first one uses the octal representation of the file mode, and the other two use the symbolic form. The files don't have to be writable by both the owner and group to be matched; either will do.

- Search for files which are writable by both their owner and their group.

```
$ find . -perm -220
$ find . -perm -g+w,u+w
```

Both these commands do the same thing.

- A more elaborate search on permissions.

```
$ find . -perm -444 -perm /222 \! -perm /111
$ find . -perm -a+r -perm /a+w \! -perm /a+x
```

These two commands both search for files that are readable for everybody (-perm -444 or -perm -a+r), have at least one write bit set (-perm /222 or -perm /a+w) but are not executable for anybody (! -perm /111 or ! -perm /a+x respectively).

Pruning - omitting files and subdirectories

- Copy the contents of /source-dir to /dest-dir, but omit files and directories named .snapshot (and anything in them). It also omits files or directories whose name ends in `~', but not their contents.

```
$ cd /source-dir
$ find . -name .snapshot -prune -o \( \! -name '*~'
↪ -print0 \) \
    | cpio -pmd0 /dest-dir
```

The construct -prune -o \(... -print0 \) is quite common. The idea here is that the expression before -prune matches things which are to be pruned. However, the -prune action itself returns true, so the following -o ensures that the right hand side is evaluated only for those directories which didn't get pruned (the contents of the pruned directories are not even visited, so their contents are irrelevant). The expression on the right hand side of the -o is in parentheses only for clarity. It emphasises that the -print0 action takes place only for things that didn't have -prune applied to them. Because the default `and' condition between tests binds more tightly than -o, this is the default anyway, but the parentheses help to show what is going on.

- Given the following directory of projects and their associated SCM administrative directories, perform an efficient search for the projects' roots:

```
$ find repo/ \
    \( -exec test -d '{}/.svn' \; \
    -or -exec test -d '{}/.git' \; \
    -or -exec test -d '{}/CVS' \; \
    \) -print -prune
```

Sample output:

```
repo/project1/CVS
repo/gnu/project2/.svn
repo/gnu/project3/.svn
repo/gnu/project3/src/.svn
repo/project4/.git
```

In this example, -prune prevents unnecessary descent into directories that have already been discovered (for example we do not search project3/src because we already found project3/.svn), but ensures sibling directories (project2 and project3) are found.

Other useful examples

- Search for several file types.

```
$ find /tmp -type f,d,l
```

Search for files, directories, and symbolic links in the directory /tmp passing these types as a comma-separated list (GNU extension), which is otherwise equivalent to the longer, yet more portable:

```
$ find /tmp \( -type f -o -type d -o -type l \)
```

- Search for files with the particular name needle and stop immediately when we find the first one.

```
$ find / -name needle -print -quit
```

- Demonstrate the interpretation of the %f and %h format directives of the -printf action for some corner-cases. Here is an example including some output.

```
$ find . .. / /tmp /tmp/TRACE compile
↪  compile/64/tests/find -maxdepth 0 -printf
↪  '[%h][%f]\n'
[.][.]
[.][..]
[][/]
[][tmp]
[/tmp][TRACE]
[.][compile]
[compile/64/tests][find]
```

EXIT STATUS top

find exits with status 0 if all files are processed successfully, greater than 0 if errors occur. This is deliberately a very broad description, but if the return value is non-zero, you should not rely on the correctness of the results of find.

When some error occurs, find may stop immediately, without completing all the actions specified. For example, some starting points may not have been examined or some pending program invocations for -exec ... {} + or -execdir ... {} + may not have been performed.

HISTORY top

As of findutils-4.2.2, shell metacharacters (`*', `?' or `[]' for example) used in filename patterns match a leading `.', because IEEE POSIX interpretation 126 requires this.

As of findutils-4.3.3, -perm /000 now matches all files instead of none.

Nanosecond-resolution timestamps were implemented in findutils-4.3.3.

As of findutils-4.3.11, the -delete action sets find's exit status to a nonzero value when it fails. However, find will not exit immediately. Previously, find's exit status was unaffected by the failure of -delete.

Feature	Added in	Also occurs in
-files0-from	4.9.0	
-newerXY	4.3.3	BSD
-D	4.3.1	
-O	4.3.1	
-readable	4.3.0	
-writable	4.3.0	
-executable	4.3.0	
-regextype	4.2.24	
-exec ... +	4.2.12	POSIX
-execdir	4.2.12	BSD
-okdir	4.2.12	
-samefile	4.2.11	
-H	4.2.5	POSIX
-L	4.2.5	POSIX
-P	4.2.5	BSD
-delete	4.2.3	
-quit	4.2.3	
-d	4.2.3	BSD
-wholename	4.2.0	
-iwholename	4.2.0	
-ignore_readdir_race	4.2.0	
-fls	4.0	
-ilname	3.8	
-iname	3.8	
-ipath	3.8	
-iregex	3.8	

The syntax -perm +MODE was removed in findutils-4.5.12, in favour of -perm /MODE. The +MODE syntax had been deprecated since findutils-4.2.21 which was released in 2005.

NON-BUGS top

Operator precedence surprises

The command find . -name afile -o -name bfile -print will never print afile because this is actually equivalent to find . -name afile -o \(-name bfile -a -print \). Remember that the precedence of -a is higher than that of -o and when there is no operator specified between tests, -a is assumed.

"paths must precede expression" error message

```
$ find . -name *.c -print
find: paths must precede expression
find: possible unquoted pattern after predicate `-name'?
```

This happens when the shell could expand the pattern *.c to more than one file name existing in the current directory, and passing the resulting file names in the command line to find like this:

```
find . -name frcode.c locate.c word_io.c -print
```

That command is of course not going to work, because the -name predicate allows exactly only one pattern as argument. Instead of doing things this way, you should enclose the pattern in quotes or escape the wildcard, thus allowing find to use the pattern with the wildcard during the search for file name matching instead of file names expanded by the parent shell:

```
$ find . -name '*.c' -print
$ find . -name \*.c -print
```

BUGS top

There are security problems inherent in the behaviour that the POSIX standard specifies for find, which therefore cannot be fixed. For example, the -exec action is inherently insecure, and -execdir should be used instead.

The environment variable LC_COLLATE has no effect on the -ok action.

REPORTING BUGS top

GNU findutils online help:
<https://www.gnu.org/software/findutils/#get-help>
Report any translation bugs to
<https://translationproject.org/team/>

Report any other issue via the form at the GNU Savannah bug tracker:

<https://savannah.gnu.org/bugs/?group=findutils>

General topics about the GNU findutils package are discussed at the bug-findutils mailing list:

<https://lists.gnu.org/mailman/listinfo/bug-findutils>

COPYRIGHT top

This is free software: you are free to change and redistribute it. There is NO WARRANTY, to the extent permitted by law.

SEE ALSO top

chmod(1), locate(1), ls(1), updatedb(1), xargs(1), lstat(2), stat(2), ctime(3) fnmatch(3), printf(3), strftime(3), locatedb(5), regex(7)

Full documentation <https://www.gnu.org/software/findutils/find> or available locally via: info find

COLOPHON top

This page is part of the findutils (find utilities) project. Information about the project can be found at ⟨http://www.gnu.org/software/findutils/⟩. If you have a bug report for this manual page, see ⟨https://savannah.gnu.org/bugs/?group=findutils⟩. This page was obtained from the project's upstream Git repository ⟨git://git.savannah.gnu.org/findutils.git⟩ on 2021-06-20. (At that time, the date of the most recent commit that was found in the repository was 2021-05-08.) If you discover any rendering problems in this HTML version of the page, or you believe there is a better or more up-to-date source for the page, or you have corrections or improvements to the information in this COLOPHON (which is not part of the original manual page), send a mail to man-pages@man7.org

FIND(1)

5 wc

```
WC(1)                          User Commands                          WC(1)
NAME         top
       wc - print newline, word, and byte counts for each file
SYNOPSIS         top
       wc [OPTION]... [FILE]...
       wc [OPTION]... --files0-from=F
DESCRIPTION         top
       Print newline, word, and byte counts for each FILE, and a total
       line if more than one FILE is specified.  A word is a
       non-zero-length sequence of characters delimited by white space.

       With no FILE, or when FILE is -, read standard input.

       The options below may be used to select which counts are printed,
       always in the following order: newline, word, character, byte,
       maximum line length.

       -c, --bytes
              print the byte counts

       -m, --chars
              print the character counts

       -l, --lines
              print the newline counts

       --files0-from=F
              read input from the files specified by NUL-terminated
              names in file F; If F is - then read names from standard
              input

       -L, --max-line-length
              print the maximum display width

       -w, --words
              print the word counts

       --help display this help and exit
```

--version
output version information and exit

AUTHOR top

Written by Paul Rubin and David MacKenzie.

REPORTING BUGS top

GNU coreutils online help:
<https://www.gnu.org/software/coreutils/>
Report any translation bugs to
<https://translationproject.org/team/>

COPYRIGHT top

SEE ALSO top

Full documentation <https://www.gnu.org/software/coreutils/wc>
or available locally via: info '(coreutils) wc invocation'

COLOPHON top

This page is part of the coreutils (basic file, shell and text manipulation utilities) project. Information about the project can be found at ⟨http://www.gnu.org/software/coreutils/⟩. If you have a bug report for this manual page, see ⟨http://www.gnu.org/software/coreutils/⟩. This page was obtained from the tarball coreutils-8.32.tar.xz fetched from ⟨http://ftp.gnu.org/gnu/coreutils/⟩ on 2021-06-20. If you discover any rendering problems in this HTML version of the page, or you believe there is a better or more up-to-date source for the page, or you have corrections or improvements to the information in this COLOPHON (which is not part of the original manual page), send a mail to man-pages@man7.org

GNU coreutils 8.32 March 2020 WC(1)