**LINUX+ PREPARATION (103)**

**BIOS –** basic input output services

**UEFI** – universal extensible firmware interface

Може да се използват и двете в една система.

**MASS STORAGE DEVICES**

**ATA –** AT Attachment interface

**SATA** – Serial ATA, след появата и, ATA се превръща в Parallel ATA (PATA)

**IDE** – Intergrated Drive Electonics вътре е включен контролера в кутията на хард диска

**SSD** – Solid state Drives новите варианти са на PCIe (NVM express)

**SCSI** – Small Computer System Interface, появавява се 1978, за да се прикачат различни устройства, хард диск или принтери.

**ATAPI** – Оригиналната АТА е била създадена само за хардове но след това се разширява в ATA Packet Interface (ATAPI) за свързване на различни устройства (хард,флопи,сд-ром).

**Hot plug and cold plug**

Hot plug са компоненти които могат да се извадяt или монтират в компютъра по време на работа. Това са най-вече всички периферни устройства или хард дискове.

Cold plug са тези които не могат да се пипат по време на работа като процесор или рам.

**VIRTUAL FILES FOR HARDWARE AND SYSTEM INFORMATION**

Виртуалните фаилови системи в линукс са две: PROCFS и SYSFS.

/proc – съдържа специални фаилове които извличат или изпращат информация към ядрото. Например командата Free извлича информацията от файла /proc/meminfo или информация за процесора от /proc/cpuinfo. Съдържа информация също и за мрежите като: активните интерфейси /proc/net/dev, маршрутизирането /proc/ мрежова статистика /proc/net/netstat.

В проц може да се видят папки с номера на процесите които са стартирани в система в момента. Всички фаилове може да са с големина нула но те не са празни и могат да се прочетат. Procfs е по старата фаилове система и тъй като започва да се пълни с най различни системни файлове и да става доста омазана се разработва SYSFS която е значително по добре структурирана и поема голяма част от нещата които са били в /proc.

Sysctl е команда с която може да се манипулира procfs.

**директният достъп до паметта** (на [английски](https://bg.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8_%D0%B5%D0%B7%D0%B8%D0%BA): *direct memory access*, съкратено **DMA**) е метод за пренос на данни между [оперативната памет](https://bg.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D0%B0_%D0%BF%D0%B0%D0%BC%D0%B5%D1%82) и вътрешните устройства без участието на [централния процесор](https://bg.wikipedia.org/wiki/%D0%A6%D0%B5%D0%BD%D1%82%D1%80%D0%B0%D0%BB%D0%B5%D0%BD_%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D0%BE%D1%80), който се използва за бързо въвеждане и извеждане на информацията и разтоварване на централния процесор от [входно-изходни](https://bg.wikipedia.org/wiki/%D0%92%D1%85%D0%BE%D0%B4-%D0%B8%D0%B7%D1%85%D0%BE%D0%B4) операции.

Sysfs е по-новата и по-добре структурирана виртуална фаилова система. В /sys се съдържа информация за устройствата, драйвърите (модулите на ядрото), UDEV и HAL.

**lsdev** – команда която се инсталира с пакета PROCINFO. Изкарва информация от проц за DMA, I/O Ports и IRQ за всяко устройство в табличен вид.

В пакета cpuinfo се съдържа и командата cpuinfo но трябва да се прегледа МАН страницата.

**UDEV** е демон, който се занимава с dynamic device management необходим за хот плуг устройствата. Когато устройство е добавено, премахното или е със сменен статус, ядрото изпраща сигнал към system-udevd.service който управлява udev.

Hot plugging се постига от комбинация на 3 компонента – UDEV, HAL, DBUS.

Udev съсдава възли на устройствата в /dev, тоест прави устройствата видими за потребителя и задава имена на устройствата така, че те да бъдат лесно разпознаваеми.

**TOOLS AND UTILITIES**

**lspci** – изкарва списък на всички PCI Devices

PCI, which stands for Peripheral Component Interconnect

**setpci** е команда с която може да се настройват pci devices.

**lsusb** – изкарва списък на всички USB Devices

**D-BUS** – demon който предава съобщения или нотификации на потребителя от ядрото, примери: когато е открит нов хардуер, свършила е хартията в принтера или сме поставили диск в двд-то се отваря някоя програма.

**KERNEL MODULES**

**Lsmod** – извежда списък за текущия статус на активните модулите които са заредени в системата, чете ги от /proc/modules. Няма допълнителни параметри за командата.

**Modinfo** – дава информация за модула. Може да се въведе пълното име на файла или името на модула. Показва пътя към модула, създател, зависимуст към друг модул, алиас и др.

**Uname** – показва информация за системата. Важни параметри са: –а извежда цялата информация за системата (име,ядро,ос и др.) –r извежда само kernel realese.

**Modprobe** – използва за монтиране или премахване на модули от системата. Повечето модули имат зависимусти, затова е важно зареждането на правилния модул в правилния ред. Важни параметри:

-a – зарежда модулите

-r – премахва модулите

-v – verbose

-n – dry run (използва се в комбинация с V)

modprobe –nav vfat – зарежда модула за vfat

## Линукс ядрото – Linux Kernel

Линукс ядрото е сърцето на операционната система. Всъщност, то е самата операционна система. Зарежда се със стартирането на компютъра и се заключва в паметта до изключването на машината. Има две основни задачи да обслужва хардуера на ниско ниво и да създава средата, необходима за работата на процесите.

Линукс ядрото се характеризира като монолитно – един голям изпълним файл (максимално няколко мегабайта), съдържащ множество логически разделени компоненти.

Модулите дават възможност на ядрото да придобива временни пълномощия, каквито не са му вродени при компилацията. Ядрото може да зарежда динамично допълнителни парчета код, за да изпълни конкретна задача и след това да изчисти модула от паметта.

Има някои драйвери или функции, които не могат да работят като модули. Те трябва или да се вградят в ядрото, или напълно да се изключат. Повечето неща се копилират като модули, но някои, от които зависи пряко бързодействието на системата, задължително трябва да бъдат част от ядрото.

**101.2 BOOT THE SYSTEM**

**BOOTING A COMPUTER**

Биоса се съхранява в РОМ или флаш памет на дъното. След стартиране на компютъра, биоса извършва power-on self test (POST) за проверка на машината.Открива кое е буутващото устройство и зарежда първия сектор от MBR на това устройство. Изпълнимия код е не по-голям от 512 байта в MBR. Всяко устройство съдържа такъв код в неговия MBR (USB,CD,SSD и т.н.). Този код се зарежда от биоса и се нарича **first stage boot loader** или **stage 1 boot loader.**

При стандартния MBR хард драйв уиндолса проверява partition table за да открие primary partition на буутващо то устройството което е активно и да зареди първия сектор. Тази част е позната като partition boot record ili stage 2.

Зареждане на няколко системи

Стари варианти са:LOADLIN, OS/2 BOOT MANAGER

Новите linux loader са:SMART BOOT LOADER:

LOLO – LInux Loader

GRUB – Grand Unified Boot Loader

GRUB2 – the newer boot loader

**CHAIN LOADING**

Когато буут мениджъра вземе контрола той може да зареди едно нещо, друг буут лодер. Това се нарича chain loading и най-често се проявява, когато boot manager които се намира на MBR зареди boot loader който се намира на partition boot record.

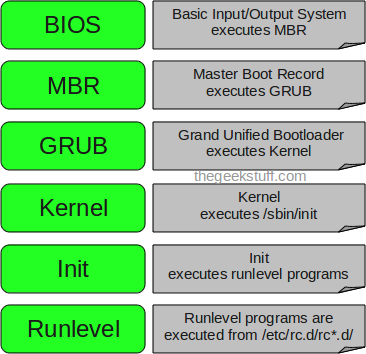
Как зарежда системата с GRUB2:

1. When a PC is turned on, the Basic Input Output Service (BIOS) performs a self test.
2. When the machine passes its self test, the BIOS loads the MBR, usually from the first 512-byte sector of the boot drive. The boot drive is most often the first hard drive on the system, but might also be a diskette, CD, or USB key.
3. For a hard drive, the MBR loads a stage 1 boot loader, which is typically either the LILO or GRUB stage1 boot loader on a Linux system. This is another 512-byte, single-sector record.
4. The stage 1 boot loader usually loads a sequence of records called the stage 2 boot loader (or sometimes the stage 1.5 loader).
5. The stage 2 loader loads the operating system. For Linux, this is the kernel and possibly an initial RAM disk (initrd or initramfs).

**The Init Procces**

Когато ядрото приключи с зареждане се стартира /sbin/init. Това е за системите които използват system V, за по-новите системи /sbim/init е символна връзка към /lib/systemd/systemd.

Systemd юнитите се съхраняват в /usr/lib/systemd/system/ (poweroff.target, reboot.target, multi-user.target и т.н.).

****

1. BIOS

* BIOS stands for Basic Input/Output System
* Performs some system integrity checks
* Searches, loads, and executes the boot loader program.
* It looks for boot loader in floppy, cd-rom, or hard drive. You can press a key (typically F12 of F2, but it depends on your system) during the BIOS startup to change the boot sequence.
* Once the boot loader program is detected and loaded into the memory, BIOS gives the control to it.
* So, in simple terms BIOS loads and executes the MBR boot loader.

2. MBR

* MBR stands for Master Boot Record.
* It is located in the 1st sector of the bootable disk. Typically /dev/hda, or /dev/sda
* MBR is less than 512 bytes in size. This has three components 1) primary boot loader info in 1st 446 bytes 2) partition table info in next 64 bytes 3) mbr validation check in last 2 bytes.
* It contains information about GRUB (or LILO in old systems).
* So, in simple terms MBR loads and executes the GRUB boot loader.

3. GRUB

* GRUB stands for Grand Unified Bootloader.
* If you have multiple kernel images installed on your system, you can choose which one to be executed.
* GRUB displays a splash screen, waits for few seconds, if you don’t enter anything, it loads the default kernel image as specified in the grub configuration file.
* GRUB has the knowledge of the filesystem (the older Linux loader LILO didn’t understand filesystem).
* Grub configuration file is /boot/grub/grub.conf (/etc/grub.conf is a link to this). The following is sample grub.conf of CentOS.

#boot=/dev/sda default=0 timeout=5 splashimage=(hd0,0)/boot/grub/splash.xpm.gz hiddenmenu title CentOS (2.6.18-194.el5PAE) root (hd0,0) kernel /boot/vmlinuz-2.6.18-194.el5PAE ro root=LABEL=/ initrd /boot/initrd-2.6.18-194.el5PAE.img

* As you notice from the above info, it contains kernel and initrd image.
* So, in simple terms GRUB just loads and executes Kernel and initrd images.

4. Kernel

* Mounts the root file system as specified in the “root=” in grub.conf
* Kernel executes the /sbin/init program
* Since init was the 1st program to be executed by Linux Kernel, it has the process id (PID) of 1. Do a ‘ps -ef | grep init’ and check the pid.
* initrd stands for Initial RAM Disk.
* initrd is used by kernel as temporary root file system until kernel is booted and the real root file system is mounted. It also contains necessary drivers compiled inside, which helps it to access the hard drive partitions, and other hardware.

5. Init

* Looks at the /etc/inittab file to decide the Linux run level.
* Following are the available run levels
  + 0 – halt
  + 1 – Single user mode
  + 2 – Multiuser, without NFS
  + 3 – Full multiuser mode
  + 4 – unused
  + 5 – X11
  + 6 – reboot
* Init identifies the default initlevel from /etc/inittab and uses that to load all appropriate program.
* Execute ‘grep initdefault /etc/inittab’ on your system to identify the default run level
* If you want to get into trouble, you can set the default run level to 0 or 6. Since you know what 0 and 6 means, probably you might not do that.
* Typically you would set the default run level to either 3 or 5.

6. Runlevel programs

* When the Linux system is booting up, you might see various services getting started. For example, it might say “starting sendmail …. OK”. Those are the runlevel programs, executed from the run level directory as defined by your run level.
* Depending on your default init level setting, the system will execute the programs from one of the following directories.
  + Run level 0 – /etc/rc.d/rc0.d/
  + Run level 1 – /etc/rc.d/rc1.d/
  + Run level 2 – /etc/rc.d/rc2.d/
  + Run level 3 – /etc/rc.d/rc3.d/
  + Run level 4 – /etc/rc.d/rc4.d/
  + Run level 5 – /etc/rc.d/rc5.d/
  + Run level 6 – /etc/rc.d/rc6.d/
* Please note that there are also symbolic links available for these directory under /etc directly. So, /etc/rc0.d is linked to /etc/rc.d/rc0.d.
* Under the /etc/rc.d/rc\*.d/ directories, you would see programs that start with S and K.
* Programs starts with S are used during startup. S for startup.
* Programs starts with K are used during shutdown. K for kill.
* There are numbers right next to S and K in the program names. Those are the sequence number in which the programs should be started or killed.
* For example, S12syslog is to start the syslog deamon, which has the sequence number of 12. S80sendmail is to start the sendmail daemon, which has the sequence number of 80. So, syslog program will be started before sendmail.

**INITRD**

In a nutshell, when your bootloader (GRUB?) loads your Linux kernel, it is of course the kernel's job to finish the boot process. But to do so, it might require particular drivers to be able to work with, say, hardware RAID controllers, or a network, and so on. And depending on where those critically important drivers are, the kernel might not have the ability to load them; hence, the creation of a *preliminary* root file system that would contain just enough in the way of loadable modules to give the kernel access to the rest of the hardware.

Quite simply, it's the bootloader's job to pass control to the kernel, hand it the "initrd" (initial ram disk), let the kernel mount it and get what it needs, whereupon the kernel can toss the initrd and replace it with the *real* root filesystem. With me so far?

**INITRAMFS**

The only purpose of an initramfs is to mount the root filesystem. The initramfs is a complete set of directories that you would find on a normal root filesystem. It is bundled into a single cpio archive and compressed with one of several compression algorithms.

**Kernel Parameter**

INIT

This sets the initial command to be executed by the kernel. Default is to use /sbin/init, which is the parent of all processes.  
To boot system without password pass /bin/bash or /bin/sh as argument to init  
init=/bin/bash

SINGLE

The most common argument that is passed to the init process is the word ‘single’ which instructs init to boot the computer in single user mode, and not launch all the usual daemons

#### ROOT=/DEV/DEVICE

This argument tells the kernel what device (hard disk, floppy disk) to be used as the root filesystem while booting. For example following boot parameter use /dev/sda1 as the root file system:  
root=/dev/sda1

If you copy entire partition from /dev/sda1 to /dev/sdb1 then use  
root=/dev/sdb1

#### RO

This argument tells the kernel to mount root file system as read-only. This is done so that fsck program can check and repair a Linux file system. Please note that you should never ever run fsck on read/write file system.

#### RW

This argument tells the kernel to mount root file system as read and write mode.

#### PANIC=SECOND

Specify kernel behavior on panic. By default, the kernel will not reboot after a panic, but this option will cause a kernel reboot after N seconds. For example following boot parameter will force to reboot Linux after 10 seconds  
panic=10

#### MAXCPUS=NUMBER

Specify maximum number of processors that an SMP kernel should make use of. For example if you have four cpus and would like to use 2 CPU then pass 2 as a number to maxcpus (useful to test different software performances and configurations).  
maxcpus=2

#### DEBUG

Enable kernel debugging. This option is useful for kernel hackers and developers who wish to troubleshoot problem

#### SELINUX [0|1]

Disable or enable SELinux at boot time.

* Value 0 : Disable selinux
* Value 1 : Enable selinux

#### RAID=/DEV/MDN

This argument tells kernel howto assembly of RAID arrays at boot time. Please note that When md is compiled into the kernel (not as module), partitions of type 0xfd are scanned and automatically assembled into RAID arrays. This autodetection may be suppressed with the kernel parameter “raid=noautodetect”. As of kernel 2.6.9, only drives with a type 0 superblock can be autodetected and run at boot time.

#### MEM=MEMEORY\_SIZE

This is a classic parameter. Force usage of a specific amount of memory to be used when the kernel is not able to see the whole system memory or for test. For example:  
mem=1024M

The kernel command line is a null-terminated string currently up to 255 characters long, plus the final null. A string that is too long will be automatically truncated by the kernel, a boot loader may allow a longer command line to be passed to permit future kernels to extend this limit (H. Peter Anvin ).

### How to begin the enter parameters mode?

You need to enter all this parameter at Grub or Lilo boot prompt. For example if you are using Grub as a boot loader, at Grub prompt press ‘e’ to edit command before booting.

1) Select second line  
2) Again, press ‘e’ to edit selected command  
3) Type any of above parameters.

See an example of “[recovering grub boot loader password](https://www.cyberciti.biz/tips/howto-recovering-grub-boot-loader-password.html)“, for more information. Another option is to type above parameters in grub.conf or lilo.conf file itself.

Run Levels, Boot Targets

System V Runlevels

Допълнителна информация в файла за Linux kurs.

Init процеса при използване на shutdown първо изпраща SIGTERM сигнал за да може програмите да се изключат нормално в рамките на 5секунди, след това изпраща сигнал SIGKILL.

По дифолт shutdown превключва в runlevel 1 и за това се използват параметри като –h за спиране и –r за рестартиране. Тук може да се задава с точност времето в което искаме да рестартираме или спрем машината, както и да изпратим съобщение към всички които са се логнали в системата.

[root@attic4-cent ~]# shutdown 5 File system recovery needed

Broadcast message from ian@attic4-cent

    (/dev/pts/1) at 18:11 ...

The system is going down for maintenance in 5 minutes!

File system recovery needed

^Cshutdown: Shutdown cancelled

[root@attic4-cent ~]# shutdown -r 10 Reloading updated kernel&

[1] 5667

[root@attic4-cent ~]#

Broadcast message from ian@attic4-cent

    (/dev/pts/1) at 18:11 ...

The system is going down for reboot in 10 minutes!

Reloading updated kernel

fg

shutdown -r 10 Reloading updated kernel

^Cshutdown: Shutdown cancelled

[root@attic4-cent ~]# shutdown -h 23:59&

[1] 5669

[root@attic4-cent ~]#

Broadcast message from ian@attic4-cent

    (/dev/pts/1) at 18:11 ...

The system is going down for halt in 348 minutes!

[root@attic4-cent ~]# shutdown -c

shutdown: Shutdown cancelled

[1]+  Done                    shutdown -h 23:59

**-c използва се за да се канселира заявката за shutdown**

**-t използва се за да се увеличи времето между сигналите SIGTERM и SIGKILL.**

root@attic4:~# date;shutdown -t60 17 Time to do backups&

Tue Jul 14 18:27:08 EDT 2015

[1] 2240

root@attic4:~#

Broadcast message from root (tty1) (Tue Jul 14 18:29:08 2015):

Time to do backups

The system is going DOWN to maintenance mode in 15 minutes!

Други варианти са: Poweroff , Halt , Reboot

**Notifying Users With WALL**

##### **Using wall to warn users**

[root@atticf22 ~]# wall Scheduled outage at 23:59 has been canceled

Broadcast message from ian@atticf22 (pts/1) (Tue Jul 14 21:07:05 2015):

                                                                           Scheduled outage at 23:59 has been canceled

                                                                           [root@atticf22 ~]# echo -e "We are experiencing system problemsOutage rescheduled to 02:30" | wall

                                                                           Broadcast message from ian@atticf22 (pts/1) (Tue Jul 14 21:07:36 2015):

                                                                           We are experiencing system problems

Outage rescheduled to 02:30

## System V /etc/inittab

As usual, lines starting with # are comments. Other lines have several fields with the following format:

id:runlevels:action:process

**id**

is a unique identifier of one to four characters. Older versions limited this to two characters, so you may see only two characters used.

**runlevels**

lists the runlevels for which the action for this id should be taken. If no runlevels are listed, do the action for all runlevels.

**action**

describes which of several possible actions should be taken.

**process**

tells which process, if any, should be run when the action on this line is performed.

Some of the common actions that may be specified in /etc/inittab are shown in Table 3. See the man pages for inittab for other possibilities.

| **Action** | **Purpose** |
| --- | --- |
| respawn | Restart the process whenever it terminates. Usually used for getty processes, which monitor for logins. |
| wait | Start the process once when the specified runlevel is entered and wait for its termination before init proceeds. |
| once | Start the process once when the specified runlevel is entered. |
| initdefault | Specifies the runlevel to enter after system boot. |
| ctrlaltdel | Execute the associated process when init receives the SIGINT signal, for example, when someone on the system console presses CTRL-ALT-DEL. |

Our beloved /etc/inittab is no more. Instead, we have a /etc/systemd/system/ directory chock-full of symlinks to files in /usr/lib/systemd/system/. /usr/lib/systemd/system/ contains init scripts; to start a service at boot it must be linked to /etc/systemd/system/.

Systemd tasks are organized as **units**. The most common units are services (.service), mount points (.mount), devices (.device), sockets (.socket), or timers (.timer). For instance, starting the secure shell daemon is done by the unit ssh.service.

# ****Topic 102.1: Linux Installation and Package Management****

Основните системни директории са:

/bin – съдържа основните програми на операционната система, без които тя не би функционирала правилно. Повечето програми от тази директория може да се изпълняват от всички потребители. Тя е присъства и в пътя по подразбиране на всички потребители.

/boot – съдържа ядрото на операционната система, както и други файлове, необходими за правилното функциониране на ядрото.

/dev – съдържа файловете отговарящи за хардуерните устройства на компютъра. Файловете в тази директория са достъпни само за четене от потребителите.

/etc – съдържа глобалните конфигурационни файлове на операционната система и нейните програми. Някои от програмите може да пазят конфигурационни файлове и на други места.

/home – съдържа домашните директории на потребителите. Тук за всеки потребител се създава отделна директория в която може да се съхраняват конфигурационни файлове или документи от различен тип. Специфичното в тези директории е начина на разграничаването на конфигурационните файлове и поддиректории от обикновените, а именно чрез добавяне на точка в началото им. Повечето файлови мениджъри скриват файл или директория, който започва с точка.

/lib – директорията съдържа най-важните споделени библиотеки на операционната система – glibc-solibs. Тези библиотеки се наричат споделени (shared object), защото повечето програми ги използват при изпълнението си. /lib/modules – също много важна директория. Тук се съхраняват модулите на ядрото, т.е. драйверите на операционната система. Туй като всяка Линукс система може да има повече от едно ядро, то в тази директория се създават поддиректории с номера на всяко едно инсталирано ядро.

/mnt – незадължителна системна директория. Тя се създава автоматично и нейната цел е обединяването на всички монтирани устройства на едно място. Линукс обаче не ви ограничава да монтирате устройства в други директории. 8 /opt – тук повечето дистрибуции инсталират графичните среди като KDE и GNOME. /proc – това е специална директория в която няма реални файлове. Посредством тази директория вие имате достъп до различни параметри на ядрото, както и до различна информация пряко предоставяна от ядрото на операционната система.

/root – домашната директория на свръхпотребителя. За обикновения потребител тази директория е заключена и той няма никакъв достъп до файловете в нея.

/sbin – важна системна директория с множество програми за управление на операционната и файловата система. Тази директория не е включена в пътя по подразбиране на обикновения потребител. Всички програми в тази директория искат root привилегии, за да работят.

/tmp – директория в която програмите записват временните си файлове. Тя има неограничен достъп за всички потребители. В нея обаче може да има поддиректории създадени от програми, които да са недостъпни за обикновените потребители.

/usr – директория с множество поддиректории. Тук се пазят повечето инсталирани програми, помощната информация и др. /usr/X11R6, /usr/X11 – в тази директория се пазят файловете на графичния сървър, шрифтовете му и неговите библиотеки.

/usr/bin – съдържа бинарните файлове на инсталираните програми, които не са важни за нормалното функциониране на операционната система.

/usr/doc – част от документацията на програмите. Тук се съхранява Linux-HOWTO.

/usr/include – тази директория е необходима само ако ще се инсталират програми от изходен код. Тук се пазят заглавните C файлове на инсталираните библиотеки и програми, които са необходими за инсталиране на други програми.

/usr/info – част от документацията на Линукс. Достъпа до нея се осъществява чрез специална програма – info. /usr/local – директория в която по подразбиране се инсталират програмите 9 компилирани от изходен код. Тази директория копира структурата на /usr.

/usr/man – най-мащабната част от Линукс документацията. Тук се съхранява ръководство (manual page) за почти всички инсталирани програми. Достъпа до тази информация се осъществява с програмата man.

/usr/lib – важна директория, в която се съхраняват повечето инсталирани библиотеки. Тези библиотеки не са необходими за работата на операционната система, а за работата на допълнително инсталирания софтуер.

/usr/share – също важна директория. Тук програмите инсталират файловете, необходими за тяхната правилна работа.

/usr/src – в тази директория се пази изходният код на Линукс ядрото. Той е необходим само когато ядрото ще се прекомпилира. И тук подобно на /lib/modules може да има код на повече от едно ядро. Те се съхраняват в директории, различаващи се с номера на версията. В директорията /usr/src присъства и линк linux, който сочи към една от директориите съдържащи изходен код.

/var – системна директория съдържаща предимно логове на програмите и система, както и опашка с отложените или предстоящи задачи за автоматично изпълнение. Директория съдържа и друга информация, като пристигнали писма, файлове заключващи дадени процеси и сокети на работещи в момента програми. В повечето директории обикновения потребител има права за четене, но не и за за

Disks formatted with the GUID Partition Table (GPT) do not have the concept of geometry as MBR disks do. For these disks you use the gdisk (GPT fdisk) command or parted. If you use gdisk on an MBR formatted disk it will do an in-memory conversion of the disk to GPT format. You **can** write this to convert the disk, but this is a very risky operation, so make sure you have a good backup before you try it.

**LVM**

LVM manages disk space using:

* Physical Volumes (PVs)
* Volume Groups (VGs)
* Logical Volumes (LVs)

A Physical Volume is either a whole drive or a partition on a drive. Although LVM can use the whole drive without having a partition defined, this is not usually a good idea as you will see in the section [MBR, EBR, GPT, and LVM internals](https://www.ibm.com/developerworks/library/l-lpic1-102-1/#2-internals).

A Volume Group is a collection of one or more PVs. The space in a VG is managed as if it were one large disk, even if the underlying PVs are spread across multiple partitions or multiple disks. The underlying PVs can be different sizes and on different kinds of disks as is illustrated shortly.

A Logical Volume is analogous to a physical GPT or MBR partition in the sense that it is the unit of space that is formatted with a particular filesystem type, such as ext4 or XFS, and is then mounted as part of your Linux filesystem. An LV resides entirely within a VG.

Think of a PV as the unit of physical space that is aggregated into an abstraction called a VG, which is rather like a virtual drive. The VG or virtual drive is then partitioned into LVs for use by filesystems.

**Повече информация в семинара за файлови системи.**

# 102.2 Install a boot manager

**GRUB Legacy**

GRUB (Legacy) has a configuration file that is usually stored in /boot/grub/grub.conf. If your file system supports symbolic links, as most Linux file systems do, you probably have /boot/grub/menu.lst as a symbolic link to /boot/grub/grub.conf.

The grub command (/sbin/grub, or, on some systems, /usr/sbin/grub) is a small but reasonably powerful shell that supports several commands for installing GRUB, booting systems, locating and displaying configuration files, and similar tasks. This shell shares much code with the second stage GRUB boot loader, so it is useful to learn about GRUB without having to boot to a second stage GRUB environment. The GRUB stage 2 runs either in menu mode, so that you can choose an operating system from a menu, or in command mode, where you specify individual commands to load a system. There are also several other commands, such as grub-install, that use the grub shell and help automate tasks such as installing GRUB.

Listing 1 shows a fairly complex GRUB configuration file. As you look through it, remember one important thing: GRUB, at least GRUB Legacy, counts drives, partitions, and things that need to be counted, starting at 0 rather than 1. The second entry for CentOS has a kernel line that is very long. Listing 1 shows it with a backslash (\) indicating where it was broken for publication.

##### **Listing 1. /boot/grub/menu.lst GRUB configuration example**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59 | # grub.conf generated by anaconda  #  # You do not have to rerun grub after making changes to this file  # NOTICE:  You do not have a /boot partition.  This means that  #          all kernel and initrd paths are relative to /, eg.  #          root (hd0,5)  #          kernel /boot/vmlinuz-version ro root=/dev/hda6  #          initrd /boot/initrd-version.img  #boot=/dev/hda  default=0  timeout=60  splashimage=(hd0,0)/boot/grub/splash.xpm.gz  #password --md5 $1$y.uQRs1W$Sqs30hDB3GtE957PoiDWO.    title Fedora 22 64-bit (sda5)      root (hd0,4)          kernel /boot/grub2/i386-pc/core.img    title Fedora 18 64-bit (sda7)      root (hd0,6)          kernel /boot/grub2/i386-pc/core.img    title CentOS 6 64-bit (sda11)          root (hd0,10)          configfile /boot/grub/menu.lst    title CentOS (2.6.32-504.23.4.el6.x86\_64)      root (hd0,10)      kernel /boot/vmlinuz-2.6.32-504.23.4.el6.x86\_64 ro \             root=UUID=2f60a3b4-ef6c-4d4c-9ef4-50d7f75124a2 rd\_NO\_LUKS rd\_NO\_LVM \             LANG=en\_US.UTF-8 rd\_NO\_MD SYSFONT=latarcyrheb-sun16 crashkernel=128M \             KEYBOARDTYPE=pc KEYTABLE=us rd\_NO\_DM rhgb quiet      initrd /boot/initramfs-2.6.32-504.23.4.el6.x86\_64.img    title Fedora 20 64-bit (sda10)      root (hd0,9)          configfile /boot/grub/menu.lst    title Ubuntu 12.04-LTS 64-bit (sda9)      root (hd0,8)          kernel /boot/grub/core.img    title Ubuntu 14.04 32-bit (sda12)      root (hd0,11)          kernel /boot/grub/core.img    title Slackware 13.37 64-bit (sda6)      root (hd0,5)          chainloader +1          boot    title Open SUSE 11.4 64-bit (sda8)      root (hd0,7)          configfile /boot/grub/menu.lst    title Windows Example      rootnoverify (hd0,0)      chainloader +1  ##### |

The first set of options in [Listing 1](https://www.ibm.com/developerworks/library/l-lpic1-102-2/#grub-menu-lst) control how GRUB operates. For GRUB, these are called menu commands, and they must appear before other commands. The remaining sections give per-image options for the operating systems that you want to allow GRUB to boot. "Title" is considered a menu command. Each instance of title is followed by one or more general or menu entry commands.

The menu commands that apply to all other sections in [Listing 1](https://www.ibm.com/developerworks/library/l-lpic1-102-2/#grub-menu-lst) are:

**#**

Any line starting with a # is a comment and is ignored by GRUB. This particular configuration file was originally generated by anaconda, the Red Hat installer. You will probably find comments added to your GRUB configuration file if you install GRUB when you install Linux. The comments often serve as an aid to the system upgrade program so that you can keep your GRUB configuration current with upgraded kernels. Pay attention to any markers that are left for this purpose if you edit the configuration yourself.

**default**

Specifies which system to load if the user does not make a choice within a timeout. In [Listing 1](https://www.ibm.com/developerworks/library/l-lpic1-102-2/#grub-menu-lst), default=0 means to load the first entry. Remember that GRUB counts from 0 rather than 1. If not specified, then the default is to boot the first entry, entry number 0.

**timeout**

Specifies a timeout in seconds before booting the default entry. Note that LILO uses tenths of a second for timeouts, while GRUB uses whole seconds.

**splashimage**

Specifies the background, or splash, image to be displayed with the boot menu. GRUB Legacy refers to the first hard drive as (hd0) and the first partition on that drive as (hd0,0), so the specification of splashimage=(hd0,0)/boot/grub/splash.xpm.gz means to use the file /boot/grub/splash.xpm.gz located on partition 1 of the first hard drive. Remember to count from 0. The image is an XPM file compressed with gzip. Support for splashimage is a patch that might or might not be included in your distribution.

**password**

Specifies a password that you must enter before you can unlock the menu and either edit a configuration line or enter GRUB commands. The password can be in clear text. GRUB also permits passwords to be stored as an MD5 digest, as in the commented out example in [Listing 1](https://www.ibm.com/developerworks/library/l-lpic1-102-2/#grub-menu-lst). This is somewhat more secure, and most administrators set a password. Without a password, you have complete access to the GRUB command line.

[Listing 1](https://www.ibm.com/developerworks/library/l-lpic1-102-2/#grub-menu-lst) shows a CentOS kernel, /boot/vmlinuz-2.6.32-504.23.4.el6.x86\_64, on /dev/sda11 (hd0,10), plus several systems that are configured to chain load. [Listing 1](https://www.ibm.com/developerworks/library/l-lpic1-102-2/#grub-menu-lst) also has examples of loading GRUB 2 via /boot/grub2/i386-pc/core.img and an example of a typical Windows XP chain loading entry, although this system does not actually have Windows installed. The commands used in these sections are:

**Title**

Is a descriptive title that is shown as the menu item when Grub boots. You use the arrow keys to move up and down through the title list and then press Enter to select a particular entry.

**root**

Specifies the partition that will be booted. As with splashimage, remember that counting starts at 0, so the first Red Hat system that is specified as root (hd0,6) is actually on partition 7 of the first hard drive (/dev/hda7 in this case), while the first Ubuntu system, which is specified as root (hd1,10), is on the second hard drive (/dev/hdb11). GRUB attempts to mount this partition to check it and provide values to the booted operating system in some cases.

**kernel**

Specifies the kernel image to be loaded and any required kernel parameters. A kernel value like /boot/grub2/i386-pc/core.img usually means loading a GRUB 2 boot loader from the named root partition.

**initrd**

Is the name of the initial RAM disk, which contains modules needed by the kernel before your file systems are mounted.

**savedefault**

Is not used in this example. If the menu command default=saved is specified and the savedefaultcommand is specified for an operating system, then booting that operating system causes it to become the default until another operating system with savedefault specified is booted. In [Listing 1](https://www.ibm.com/developerworks/library/l-lpic1-102-2/#grub-menu-lst), the specification of default=0 overrides any saved default.

**boot**

Is an optional parameter that instructs GRUB to boot the selected operating system. This is the default action when all commands for a selection have been processed.

**lock**

Is not used in [Listing 1](https://www.ibm.com/developerworks/library/l-lpic1-102-2/#grub-menu-lst). This does not boot the specified entry until a password is entered. If you use this, then you should also specify a password in the initial options; otherwise, a user can edit out your lock option and boot the system or add "single" to one of the other entries. It is possible to specify a different password for individual entries if you want.

**rootnoverify**

Is similar to root, except that GRUB does not attempt to mount the file system or verify its parameters. This is usually used for file systems such as NTFS that are not supported by GRUB. You might also use this if you want GRUB to load the master boot record on a hard drive (for example, to access a different configuration file or to reload your previous boot loader).

**chainloader**

Specifies that another file will be loaded as a stage 1 file. The value "+1" is equivalent to 0+1, which means to load one sector starting at sector 0; that is, load the first sector from the device specified by root or rootnoverify.

**configfile**

Specifies that the running copy of GRUB replaces its configuration file with one loaded from the target location. For this to work, it is advisable that the version of GRUB that is loading the new configfile is as current as the version that built it.

I'll install GRUB in the partition boot record of the partition containing my CentOS distribution. I use the grub-install command and specify the device where the 512-byte stage1 boot loader should go. In my example, that's /dev/sda11 or (hd0,10) using GRUB notation. See [Listing 3](https://www.ibm.com/developerworks/library/l-lpic1-102-2/#grub-install-ex). You need to have root authority to write the partition boot record. If you have added or deleted devices you might have to remove your /boot/grub/device.map file and allow grub-install to rebuild is as shown in our example. This won't happen often, but if grub-install throws some odd error that you don’t understand, you might find deleting the device.map file helpful.

##### **Listing 3. Install GRUB Legacy in a partition boot record**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | [root@attic4-cent ~]# rm /boot/grub/device.map  rm: remove regular file `/boot/grub/device.map'? y  [root@attic4-cent ~]# grub-install /dev/sda11  Probing devices to guess BIOS drives. This might take a long time.  Installation finished. No error reported.  This is the contents of the device map /boot/grub/device.map.  Check if this is correct or not. If any of the lines is incorrect,  fix it and re-run the script `grub-install'.    (fd0)   /dev/fd0  (hd0)   /dev/sda  (hd1)   /dev/sdb  (hd2)   /dev/sdc  (hd3)   /dev/sdd |

As you already learned the standard DOS MBR can't boot a logical partition, so you'll need something else to get this system booted. One option would be to install GRUB in the MBR by doing grub-install /dev/sda which would also install GRUB in the MBR of our disk (/dev/sda). I'll also show you how to do it with GRUB 2 in a moment, but before you commit to either approach step, you might want to test out your setup using a GRUB boot CD.

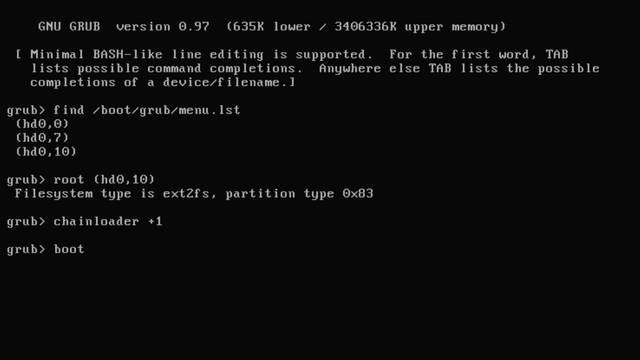
##### **Listing 4. Creating a GRUB bootable CD image**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | [ian@attic4-cent ~]$ mkdir mkdir -p grubcd/boot/grub  [ian@attic4-cent ~]$ ls /usr/share/grub/  x86\_64-redhat  [ian@attic4-cent ~]$ ls /usr/share/grub/x86\_64-redhat/stage2\_eltorito  /usr/share/grub/x86\_64-redhat/stage2\_eltorito  [ian@attic4-cent ~]$ locate stage2\_eltorito  /usr/share/grub/x86\_64-redhat/stage2\_eltorito  [ian@attic4-cent ~]$ cp /usr/share/grub/x86\_64-redhat/stage2\_eltorito grubcd/boot/grub  [ian@attic4-cent ~]$ genisoimage -R -b boot/grub/stage2\_eltorito -no-emul-boot \  > -boot-load-size 4 -boot-info-table -o grubcd.iso grubcd  I: -input-charset not specified, using utf-8 (detected in locale settings)  Size of boot image is 4 sectors -> No emulation  Total translation table size: 2048  Total rockridge attributes bytes: 760  Total directory bytes: 4576  Path table size(bytes): 34  Max brk space used 22000  241 extents written (0 MB) |

### Booting with GRUB legacy

Now you are ready to reboot your system using the GRUB CD that you just built. If your BIOS is not set up to boot automatically from a CD or DVD if present, then you might need to press some system-specific key (F8 on my BIOS) to choose a boot device other than your hard drive. The CD boots to a GRUB prompt as shown in [Figure 1](https://www.ibm.com/developerworks/library/l-lpic1-102-2/#boot-grub-cd).

##### **Figure 1. Booting your GRUB CD**



In this example, I used the find command to find GRUB config files called menu.lst and found 3, including my CentOS GRUB configuration file on device (hd0,10) or /dev/sda11. I then used the rootcommand to set )hd0,10) as the root for further file operations. I installed GRUB in the partition boot record of (hd0,10), So I use the chainloader command to tell grub to boot whatever boot loader is in the first sector of (hd0,10). Finally I use the boot command to boot this new loader

**GRUB-2**

RUB 2 is the successor to GRUB. It was rewritten from scratch to make it significantly more modular and portable. It targets different architectures and boot methods and has many new features, including the ability to handle UEFI firmware and GPT formatted drives.

The GRUB 2 packages contain several programs, normally in /usr/bin or /usr/sbin. The actual package name has changed over time and is not the same for all distributions. The binaries also have names that usually start with grub- or grub2-. For example, on Ubuntu 14, you will find grub-image provided by package grub-common, while on Fedora 22, you will find grub2-mkimage provided by package grub2-tools.

The heart of GRUB 2 is a multiboot kernel (/boot/grub/core.img) along with a configuration file (/boot/grub/grub.cfg). These will be generated for you if you run grub-install and set the target as your MBR (for example: grub-install /dev/sda). Run grub-install --help as shown in [Listing 7](https://www.ibm.com/developerworks/library/l-lpic1-102-2/#listing14) to get an idea of the programs that get called to do all the work. Some things are similar to Grub Legacy, but there are a number of new items, such as --modules, --grub-setup, --grub-image, and so on.

If you run grub-install /dev/sda, the process builds a core image file for you, builds a configuration file, and installs GRUB 2 in your MBR. If you're not ready to commit to GRUB 2 for your whole setup, you can build these parts yourself and then boot the GRUB 2 core image from GRUB Legacy or LILO.

### Building the GRUB 2 configuration file

The GRUB 2 configuration file is normally /boot/grub/grub.cfg. Unlike GRUB Legacy, you should normally not edit this file yourself because it will be overwritten the next time your GRUB 2 installation is updated. You should build it using grub-mkconfig. On some systems, such as Ubuntu, the update-grub command is a front-end to grub-mkconfig that saves its output in /boot/grub/grub.cfg. These commands look for general settings (such as background or timeouts) in /etc/default/grub and then run executables from /etc/grub.d/ to build various parts of the configuration file, such as the header, a section for the current Linux distribution, sections for other operating systems, and your own custom additions. If you need to customize the GRUB 2 menu, you add your changes to a file in /etc/grub.d/ such as 40\_custom, or add your own file. Remember that it needs to be executable. I show you an example of customization late when I show you how to chain load GRUB legacy from GRUB 2.

##### **Building a GRUB 2 configuration file with grub-mkconfig**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | ian@attic-u14:~$ sudo grub-mkconfig -o /boot/grub/grub.cfg  Generating grub configuration file ...  Found linux image: /boot/vmlinuz-3.16.0-43-generic  Found initrd image: /boot/initrd.img-3.16.0-43-generic  Found linux image: /boot/vmlinuz-3.16.0-30-generic  Found initrd image: /boot/initrd.img-3.16.0-30-generic  Found memtest86+ image: /boot/memtest86+.elf  Found memtest86+ image: /boot/memtest86+.bin  Found Fedora release 20 (Heisenbug) on /dev/sda10  Found CentOS release 6.6 (Final) on /dev/sda11  Found Fedora release 22 (Twenty Two) on /dev/sda5  Found Slackware Linux (Slackware 13.37.0) on /dev/sda6  Found Fedora release 18 (Spherical Cow) on /dev/sda7  Found openSUSE 11.4 (x86\_64) on /dev/sda8  Found Ubuntu 12.04 LTS (12.04) on /dev/sda9  done |

Notice that the menuentry stanzas look more like shell scripts than the plain commands without logic of GRUB Legacy. Another important change from GRUB Legacy is that partition numbering now starts at 1, although disk numbering still starts at 0. So /dev/sda7 is (hd0,7) in GRUB 2 where it would be (hd0,6) in GRUB Legacy. GRUB 2 can also use an optional partition name as well as a number. So (hd0,7) can also be referred to as (hd0,msdos7) to make clear that it is on an MBR formatted disk. On a GPT formatted disk, you would use (hd0,gpt7).

### Building a bootable GRUB 2 rescue CD

GRUB 2 comes with a grub-mkrescue or grub2-mkrescue command to help you create a rescue CD image. Recent versions of grub-mkrescue use the xorriso package rather than the mkisofs package or genisoimage package to create the ISO image, so you need to install if your first attempt fails with an error message indicating that Xorriso is not found or is at the wrong level. [Listing 12](https://www.ibm.com/developerworks/library/l-lpic1-102-2/#grub2-rescue) shows how to create a GRUB 2 rescue image in file rescue.iso. You don't have to be root to create the rescue ISO.

##### **Listing 12. Creating a GRUB 2 rescue image**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | [root@echidna ~]# /usr/bin/grub2-mkrescue -o rescue.iso  Enabling BIOS support ...  xorriso 1.2.4 : RockRidge filesystem manipulator, libburnia project.    Drive current: -outdev 'stdio:rescue.iso'  Media current: stdio file, overwriteable  Media status : is blank  Media summary: 0 sessions, 0 data blocks, 0 data, 7177m free  Added to ISO image: directory '/'='/tmp/tmp.Dw4KSbpoIx'  xorriso : UPDATE : 196 files added in 1 seconds  xorriso : UPDATE : 196 files added in 1 seconds  xorriso : NOTE : Copying to System Area: 29191 bytes from file '/tmp/tmp.LepCeiJPZM'  ISO image produced: 1094 sectors  Written to medium : 1094 sectors at LBA 0  Writing to 'stdio:rescue.iso' completed successfully. |

Once you have created the ISO image, you can burn it to a CD (or DVD) using your favorite burning tool. If you prefer, you can also copy it to a USB flash drive and boot from that assuming your BIOS supports booting from such devices. [Listing 13](https://www.ibm.com/developerworks/library/l-lpic1-102-2/#grub2-rescue-flash) shows how to use the dd command to copy the ISO image to the USB flash drive /dev/sde.

**Warning:** Make sure that you copy the image to the correct device. Copying it to the wrong device can destroy a lot of your data.

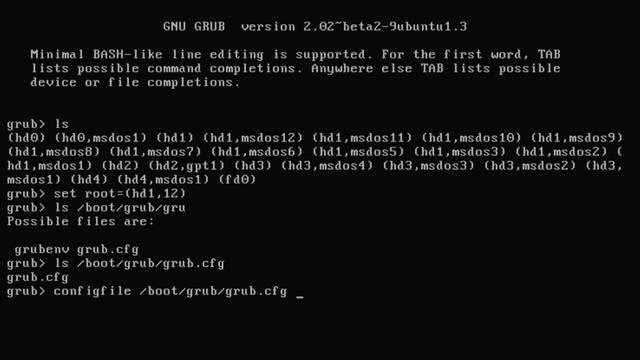
##### **Listing 13. Writing a GRUB 2 rescue image to a USB flash drive**

|  |  |
| --- | --- |
| 1  2  3  4  5 | ian@attic-u14:~$ # Burn .iso image to USB stick /dev/sde  ian@attic-u14:~$ sudo dd if=rescue.iso of=/dev/sde  9864+0 records in  9864+0 records out  5050368 bytes (5.1 MB) copied, 3.95946 s, 1.3 MB/s |

### Booting with GRUB 2

You'll boot the USB flash drive to see how it works. As with GRUB legacy, this rescue disk boots to a GRUB prompt where you enter commands. I show you a few that you can use to start the Ubuntu 14 system whose configuration file I built earlier. [Figure 5](https://www.ibm.com/developerworks/library/l-lpic1-102-2/#grub2-rescue-boot) shows the screen after you boot and enter some commands.

##### **Figure 5. Booting the GRUB 2 rescue flash drive**



The commands entered were:

**ls**

With no arguments, lists the devices that were found. This can take some time to run. Flash drives are not normally BIOS drives, but if you boot from one, it likely shows up as hd0 and displaces other drives, causing them not to be numbered as you expect. Using a bootable CD or DVD avoids this problem.

**set**

Sets variable values. In this case, you set the root variable. Compare with the GRUB legacy rootcommand. You use hd1 instead of hd0, because the previous command told you that hd0 is now the USB flash drive from which you booted.

**ls**

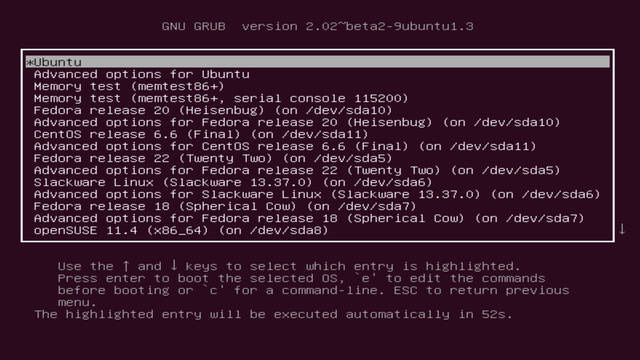
With a path, displays files or directories. Use Bash-like tab completion to either complete the path component you are typing or to get a list of possible completions as you see here (I pressed **tab** after entering /boot/grub/gru).

**configfile**

As with GRUB legacy, you can load a configuration file using the configfile command. You load the one you built earlier.

Once you load the configuration file, your screen looks like [Figure 6](https://www.ibm.com/developerworks/library/l-lpic1-102-2/#ubuntu-grub2).

##### **Figure 6. Ubuntu 14 configuration file for GRUB2**



As with GRUB legacy, you can edit configuration entries or enter commands before booting a system.

You can explore the many GRUB 2 commands further in the GRUB manual. Try typing info grub or info grub2 in a Linux terminal window to open the manual.

# 102.3 Manage shared libraries

Виж книгата за Убунто за допълнителна информация.

Linux supports two classes of libraries, namely:

* Static libraries – are bound to a program statically at compile time.
* Dynamic or shared libraries – are loaded when a program is launched and loaded into memory and binding occurs at run time.

Dynamic or shared libraries can further be categorized into:

* Dynamically linked libraries – here a program is linked with the shared library and the kernel loads the library (in case it’s not in memory) upon execution.
* Dynamically loaded libraries – the program takes full control by calling functions with the library.

Shared libraries are loaded by **ld.so** (or **ld.so.x**) and **ld-linux.so** (or **ld-linux.so.x**) programs, where **x** is the version. In Linux, **/lib/ld-linux.so.x** searches and loads all shared libraries used by a program.

A program can call a library using its library name or filename, and a library path stores directories where libraries can be found in the filesystem. By default, libraries are located in **/usr/local/lib**, **/usr/local/lib64**, **/usr/lib** and **/usr/lib64**; system startup libraries are in **/lib** and **/lib64**. Programmers can, however, install libraries in custom locations.

### The ldd command

Apart from knowing that a statically linked program is likely to be large, how can you tell whether a program is statically linked? And if it is dynamically linked, how do you know what libraries it needs? The ldd command can answer both questions. If you are running a system such as Debian or Ubuntu, you probably don't have the sln executable, so you might also want to check the /sbin/ldconfig executable. Listing 2 shows the output of the ldd command for the ln and sln executables and also the ldconfig executable. The example is from a Fedora 22 64-bit system (atticf20). For comparison, the output from Ubuntu 14 32-bit system (attic-u14) is shown for /bin/ln.

**ldd -v /path/to/program/executable**

##### **Listing 2. Output of ldd for sln and ln**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | [ian@atticf20 ~]$ # Fedora 22 64-bit  [ian@atticf20 ~]$ ldd /sbin/sln /sbin/ldconfig /bin/ln  /sbin/sln:      not a dynamic executable  /sbin/ldconfig:      not a dynamic executable  /bin/ln:      linux-vdso.so.1 (0x00007ffedd31e000)      libc.so.6 => /lib64/libc.so.6 (0x00007f2d3bd5d000)      /lib64/ld-linux-x86-64.so.2 (0x00007f2d3c11d000)    ian@attic-u14:~/data/lpic-1$ # Ubuntu 14 32-bit  ian@attic-u14:~/data/lpic-1$ ldd /bin/ln      linux-gate.so.1 =>  (0xb779d000)      libc.so.6 => /lib/i386-linux-gnu/libc.so.6 (0xb75d7000)      /lib/ld-linux.so.2 (0xb77a0000) |

Because ldd is actually concerned with dynamic linking, it tells us that both sln and ldconfig are statically linked by telling us that they are "not a dynamic executable," while it tells us the names of three shared libraries (linux-vdso.so.1, libc.so.6, and /lib64/ld-linux-x86-64.so.2) that the lncommand needs. Note that .so indicates that these are shared objects or dynamic libraries. This output also illustrates three different types of information you are likely to see.

**linux-vdso.so.1**

is the Linux Virtual Dynamic Shared Object, which I discuss in a moment. You can also see linux-gate.so.1 as in the Ubuntu 14 example.

**libc.so.6**

has a pointer to /lib64/libc.so.6 or /lib/i386-linux-gnu/libc.so.6. You can also see this pointing to /lib/libc.so.6 on older 32-bit systems.

**/lib64/ld-linux-x86-64.so.2**

is the absolute path to another library.

The library that you see as linux-vdso.so.1 is a virtual library, or Virtual Dynamic Shared Object, that is located only in each program's address space. Some systems call this linux-gate.so.1. This virtual library provides the necessary logic to allow user programs to access system functions through the fastest means available on the particular processor, either interrupt, or with most newer processors, fast system call.

Because shared libraries can exist in many different directories, searching through all of these directories when a program is launched would be greatly inefficient: which is one of the likely disadvantages of dynamic libraries. Therefore a mechanism of caching employed, performed by a the program **ldconfig**.

## What is ldconfig?

ldconfig is used to create, udpate and remove symbolic links for the current shared libraries based on the lib directories present in the /etc/ld.so.conf

By default, **ldconfig** reads the content of **/etc/ld.so.conf**, creates the appropriate symbolic links in the dynamic link directories, and then writes a cache to **/etc/ld.so.cache** which is then easily used by other programs.

This is very important especially when you have just installed new shared libraries or created your own, or created new library directories. You need to run **ldconfig** command to effect the changes.

Ldconfig –p | head -5

## Loading specific libraries

If you're running an older application that needs a specific older version of a shared library, or if you're developing a new shared library or version of a shared library, you might want to override the default search paths used by the loader. This might also be needed by scripts that use product-specific shared libraries that might be installed in the /opt tree.

Just as you can set the PATH variable to specify a search path for executables, you can set the LD\_LIBRARY\_PATH variable to a colon-separated list of directories that should be searched for shared libraries before the system ones specified in ld.so.cache. For example, you might use a command like:

|  |  |
| --- | --- |
| 1 | export LD\_LIBRARY\_PATH=/usr/lib/oldstuff:/opt/IBM/AgentController/lib |

LD command - combines a number of object and archive [files](https://www.lifewire.com/practical-examples-of-the-zip-command-2201158), relocates their data and ties up symbol references. Usually the last step in compiling a program is to run **ld**.

# 102.4 Debian package management

Suppose that, instead of installing a package, you want to find out whether the package depends on other packages. You can use the -s (for simulate) option on apt-get. There are several other options with equivalent function, such as --just-print and \--dry-run. Check the man pages for full details. Listing 3 shows what happens for a simulation of installing the gfortran-doc package.

##### **Listing 3. Simulated or dry-run install of gfortran-doc**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | ian@attic-u14:~$ sudo apt-get install -s gfortran-doc  Reading package lists... Done  Building dependency tree  Reading state information... Done  The following extra packages will be installed:    gfortran-4.8-doc  The following NEW packages will be installed:    gfortran-4.8-doc gfortran-doc  0 upgraded, 2 newly installed, 0 to remove and 23 not upgraded.  Inst gfortran-4.8-doc (4.8.4-2ubuntu1~14.04 Ubuntu:14.04/trusty-updates [all])  Inst gfortran-doc (4:4.8.2-1ubuntu6 Ubuntu:14.04/trusty [i386])  Conf gfortran-4.8-doc (4.8.4-2ubuntu1~14.04 Ubuntu:14.04/trusty-updates [all])  Conf gfortran-doc (4:4.8.2-1ubuntu6 Ubuntu:14.04/trusty [i386]) |

As you see, the gfortran-doc documentation package requires the gfortran-4.8-doc packages. The reverse is not the case.

## Package locations

In the previous section, you learned how to install a Debian package. But where do the packages come from? How does apt-get know where to download packages from? I mentioned that apt-get reads a package list from somewhere. The starting point for that somewhere is /etc/apt/sources.list. The list tells apt-get where to look for packages, including from a CD-ROM, from your local file system, or over a network using HTTP or FTP. You can add more sources in the /etc/apt/sources.list.d directory.

Apt-get and similar tools use a local database to determine what packages are installed. They can check installed levels against available levels. To do this, information on available levels is retrieved from the sources that are listed in /etc/apt/sources.list and stored on your local system. You use the command apt-get update to synchronize the information in your local database with the sources specified in /etc/apt/sources.list. You should do this before installing or updating any package, and always after modifying /etc/apt/sources.list or adding files to /etc/apt/sources.list.d.

## Removing Debian packages

Notice that the gfortran-4.8, libgfortran-4.8-dev, and libgfortran3 packages that we installed as a prerequisites for gfortran are not actually removed automatically, although the output tells you they are no longer needed. The autoremove function of apt-get (or the equivalent remove function and the --auto-remove option) removes the requested packages, along with any packages that were installed as dependencies but are no longer required by any installed packages. This can include dependencies installed by packages other than the one or ones you are trying to remove. Listing 6 shows a simulated removal of gfortran and its dependencies. I show how to remove just gfortran, and then use apt-get autoremove to clean up the newly orphaned dependencies.

##### **Listing 6. Removing gfortran and dependencies**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44 | ian@attic-u14:~$ # Simulate removal of gfortran and dependencies  ian@attic-u14:~$ sudo apt-get autoremove -s gfortran  Reading package lists... Done  Building dependency tree  Reading state information... Done  The following packages will be REMOVED:    gfortran gfortran-4.8 libgfortran-4.8-dev libgfortran3  0 upgraded, 0 newly installed, 4 to remove and 23 not upgraded.  Remv gfortran [4:4.8.2-1ubuntu6]  Remv gfortran-4.8 [4.8.4-2ubuntu1~14.04]  Remv libgfortran-4.8-dev [4.8.4-2ubuntu1~14.04]  Remv libgfortran3 [4.8.4-2ubuntu1~14.04]  ian@attic-u14:~$ # Remove just gfortran  ian@attic-u14:~$ sudo apt-get remove gfortran  Reading package lists... Done  Building dependency tree  Reading state information... Done  The following packages were automatically installed and are no longer required:    gfortran-4.8 libgfortran-4.8-dev libgfortran3  Use 'apt-get autoremove' to remove them.  The following packages will be REMOVED:    gfortran  0 upgraded, 0 newly installed, 1 to remove and 23 not upgraded.  After this operation, 33.8 kB disk space will be freed.  Do you want to continue? [Y/n] y  (Reading database ... 202421 files and directories currently installed.)  Removing gfortran (4:4.8.2-1ubuntu6) ...  Processing triggers for man-db (2.6.7.1-1ubuntu1) ...  ian@attic-u14:~$ # Autoremove unneeded packages  ian@attic-u14:~$ sudo apt-get autoremove  Reading package lists... Done  Building dependency tree  Reading state information... Done  The following packages will be REMOVED:    gfortran-4.8 libgfortran-4.8-dev libgfortran3  0 upgraded, 0 newly installed, 3 to remove and 23 not upgraded.  After this operation, 17.6 MB disk space will be freed.  Do you want to continue? [Y/n] y  (Reading database ... 202416 files and directories currently installed.)  Removing gfortran-4.8 (4.8.4-2ubuntu1~14.04) ...  Removing libgfortran-4.8-dev:i386 (4.8.4-2ubuntu1~14.04) ...  Removing libgfortran3:i386 (4.8.4-2ubuntu1~14.04) ...  Processing triggers for man-db (2.6.7.1-1ubuntu1) ...  Processing triggers for libc-bin (2.19-0ubuntu6.6) ... |

As you see, you use the autoremove function of apt-get without any package name, to remove all unused packages that were installed but are no longer required on your system. You can also use the apt-get purge option to remove configuration information. See the man page for more information.

### Updating all packages or upgrading to a new distribution

Rather than updating individual packages, you can update all packages on your system using the apt-get upgrade command. Similarly, apt-get dist-upgrade helps you migrate to a new level of your distribution.

For more information on other capabilities and options for apt-get, see the man page.

### APT configuration—the apt.conf file

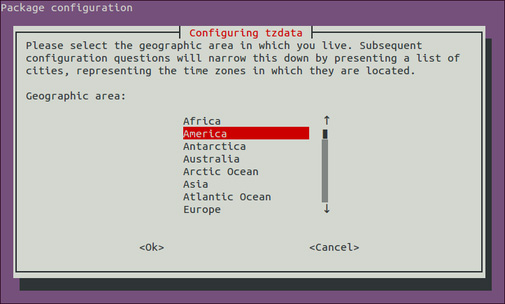
The man page for apt-get shows that there are many options. If you use the apt-get command a lot and find the default options are not to your liking, you can set new defaults in /etc/apt/apt.conf.

## Reconfiguring Debian packages

APT includes a capability called debconf, which configures packages after they are installed. Packages that use this capability (and not all do) can be reconfigured after they are installed. The easiest way to do this is to use the dpkg-reconfigure command. For example, the adduser command might create home directories that are readable by all system users. You might not want this for privacy reasons. Similarly, the tzdata package supports changing the time zone using dpkg-reconfigure tzdata. You must run dpkg-reconfigure with root authority.

Figure 1 shows the first question that you are asked if you run sudo dpkg-reconfigure tzdata. Your preset default might not be America; it reflects your own system. Navigate around this text-mode screen using the Tab key and the cursor movement keys.

##### **Figure 1. Using dpkg-reconfigure to reconfigure time zone**



You can also use debconf-show to view the current configuration for a package as shown in Listing 8.

##### **Listing 8. Displaying tzdata configuration using debconf-show**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | ian@attic-u14:~$ sudo debconf-show tzdata    tzdata/Zones/Pacific:    tzdata/Zones/Europe:    tzdata/Zones/Indian:    tzdata/Zones/Australia:  \* tzdata/Areas: America    tzdata/Zones/Arctic:    tzdata/Zones/Atlantic:  \* tzdata/Zones/Etc: UTC  \* tzdata/Zones/America: New\_York    tzdata/Zones/US:    tzdata/Zones/Africa:    tzdata/Zones/SystemV:    tzdata/Zones/Antarctica:    tzdata/Zones/Asia: |

## Debian package information

Now let's look at some tools for getting information about packages. Some of these tools do other things as well, but the focus here is on how to get information.

### Package status with dpkg

Another tool that is part of the APT system is the dpkg tool. This is a medium-level package management tool that can install and remove packages and display status information. You can control configuration of dpkg by /etc/dpkg/dpkg.cfg, and you might also have a .dpkg.cfg file in your home directory to provide further configuration.

The dpkg tool uses many files in the /var/lib/dpkg tree in your filesystem. In particular, the file /var/lib/dpkg/status contains status information about packages on your system. Listing 9 shows the use of dpkg -s to display the status of the tzdata package after we updated it and the gfortran package after we removed it. If configuration remains, which it can in some cases, you can use the purge option to purge downloaded package files from the cache and remove configuration information.

##### **Listing 9. Tzdata package status**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25 | ian@attic-u14:~$ dpkg -s gfortran tzdata  dpkg-query: package 'gfortran' is not installed and no information is available    Package: tzdata  Status: install ok installed  Priority: required  Section: libs  Installed-Size: 1538  Maintainer: Ubuntu Developers <ubuntu-devel-discuss@lists.ubuntu.com>  Architecture: all  Multi-Arch: foreign  Version: 2015d-0ubuntu0.14.04  Replaces: libc0.1, libc0.3, libc6, libc6.1  Provides: tzdata-jessie  Depends: debconf (>= 0.5) | debconf-2.0  Description: time zone and daylight-saving time data   This package contains data required for the implementation of   standard local time for many representative locations around the   globe. It is updated periodically to reflect changes made by   political bodies to time zone boundaries, UTC offsets, and   daylight-saving rules.  Homepage: http://www.iana.org/time-zones  Original-Maintainer: GNU Libc Maintainers <debian-glibc@lists.debian.org>  Use dpkg --info (= dpkg-deb --info) to examine archive files,  and dpkg --contents (= dpkg-deb --contents) to list their contents. |

### Packages and the files in them

You might want to know what is in a package or what package a particular file came from. These are both tasks for dpkg. Listing 10 illustrates the use of dpkg -L to list the files (including directories) installed by the libparted package. For most packages, you can just give the package name and not worry about specifying a particular version. However, some packages might be available in multiple versions, so you might need to specify a more detailed package name when using dpkg to interrogate the package information.

-L – показва файловете и директориите които са инсталирани с пакета.

Dpkg –L libreoffice

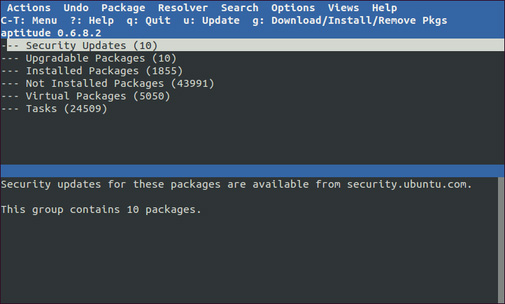
-S – показва даден файл към кой пакет принадлежи.

Dpkg –S /lib/i386-linux/libreoffice.so.0

## Using aptitude

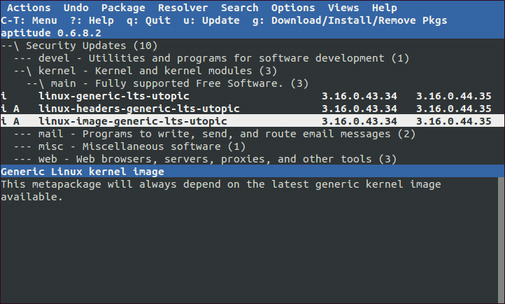
Earlier, I mentioned that the status for packages is kept in /var/lib/dpkg/status. I also mentioned that dpkg could do more than just display package information. Now, let's look at the aptitude command, which provides a text-based full-screen interface (using ncurses) to the APT package management functions. You can install aptitude using apt-get if it is not already installed. You can use aptitude to install or remove packages and to control status flags that indicate whether packages should be kept up-to-date or held in their present state, for example. If you run the aptitude command (as root), you see a screen similar to Figure 2.

##### **Figure 2. Running aptitude**



Press Enter to expand or collapse the various selections, then use **ctrl-t** to access the menu bar. Figure 3 shows that a new kernel version, 3.16.0.43.34, is available for my system, among other available updates. The 'i' in the left column indicates that the current status is to install the package. The Help menu item explains the various options you have, including holding a package at its current level rather than updating it, removing it, or marking it as being automatically installed and thus eligible for automatic removal. Remember the autoremove option of apt-get? Now you know how to examine or control which packages are eligible for automatic removal. Use the keyboard shortcuts described in the Help or use the **Package** menu item to change the flags.

##### **Figure 3. Running aptitude and examining package flags**



You can use the slash ("/") key to search for packages. For example, if you wanted to reinstall the gfortran package that we removed earlier, simply type "/gfortran" to search for it. If the search takes you to something else with gfortran in it, such as gfortran-doc, press the n key to advance to the next match. Then use the **Package** menu to mark the package for installation.

When you are finished, select **Actions->Install/remove packages** (or press G) to apply your selections to the system. You can also click the quit option if you do not want to apply the changes.

For help at any time, use the menu bar or type "?" (question mark) for help, and then press the Q key to exit the help.

## Upgrading Debian with other tools

You see that aptitude can help you install or remove individual packages and upgrade all the packages on your system to the latest level.

In addition to aptitude, there are several other interactive package management interfaces for Debian systems, including dselect, synaptic, update-manager, gnome-apt, and wajig. Synaptic is a graphical application for use with the X Window System. Figure 4 shows the synaptic user interface with our old friend, the gfortran package, marked for installation.

## Finding Debian packages

In the final topic on Debian package management, I look at ways to find packages. Usually, apt-get and the other tools discussed here already know about any Debian package you might need from the list of available packages. A command that we haven't used yet is apt-cache, which is useful for searching package information on your system. apt-cache can search using regular expressions (see "[Learn Linux, 101: Search text files using regular expressions](https://www.ibm.com/developerworks/linux/library/l-lpic1-v3-103-7/index.html)" for more information on regular expressions). Suppose you wanted to find the name of the package containing the Linux loader. Listing 13 shows how you can accomplish this.

##### **Listing 13. Searching for the Linux loader with apt-cache**

|  |  |
| --- | --- |
| 1  2  3 | ian@attic-u14:~$ apt-cache search "linux loader"  lilo - LInux LOader - the classic OS boot loader  lilo-doc - LInux LOader - Documentation for the classic OS boot loader |

You saw earlier that aptitude and synaptic also offer search tools. If you use synaptic, note that you have options on the search menu for searching only package names or package descriptions as well.

If you still can't find the package, you can find it among the list of packages on the Debian site (see [Related topics](https://www.ibm.com/developerworks/library/l-lpic1-102-4/#artrelatedtopics) for a link) or elsewhere on the Internet.

Most of the package tools can tell you a lot more about an installed package than about one that you do not yet have installed, such as the list of files within a package. If you need to find what package contains a program that you do not have installed, there are a few ways:

* You can guess what package might contain it and download the package without installing. Once you have the package, you can interrogate it.
* You can search the Internet.
* You can try the command-not-found capability, which is described under [Command not found](https://www.ibm.com/developerworks/library/l-lpic1-102-4/#cnf) later in this tutorial.

The apt-get command has a -d option to download a package and not install it. There is also a --print-uris option to show where a package would be downloaded from and what its checksum would be. Current checksums are likely to be SHA256 checksums, so you can check the integrity of the downloaded package using the sha256sum command. Note that the URI and checksum information are not displayed if you have already downloaded the package, so you should get this information before downloading the package.

# 102.5 RPM and YUM package management

### RPM

Red Hat introduced RPM in 1995. RPM is now the package management system used for packaging in the Linux Standard Base (LSB). The rpm command options are grouped into three subgroups for:

* Querying and verifying packages
* Installing, upgrading, and removing packages
* Performing miscellaneous functions

### YUM

YUM adds automatic updates and package management, including dependency management, to RPM systems. In addition to understanding the installed packages on a system, YUM is like the Debian Advanced Packaging Tool (APT) in that it works with repositories, which are collections of packages and are typically accessible over a network connection.

## Package locations

In the previous section, you learned how to install an RPM package. But where do the packages come from? How does yum know where to download packages from? The starting point is the /etc/yum.repos.d/ directory, which usually contains several repo files. This is the default location for repository information, but other locations may be specified in the YUM configuration file, normally /etc/yum.conf. shows the fedora-updates.repo corresponding to the location from which we installed gcc-gfortran on our Fedora 21 system.

## Removing RPM packages

If you want to remove a package, you can use the remove option of yum, or the -e option of rpm. A test run to remove gcc-gfortran using rpm -e is shown in . If the package can be removed, there is no output.

##### **Test removal of gcc-gfortran**

|  |  |
| --- | --- |
| 1  2 | [root@attic-f21 ~]# <**strong**>rpm -e --test gcc-gfortran</**strong**>  [ |

Unlike the simulated removal of Debian packages using apt-get, the RPM system does not maintain information on packages that were automatically added, so there is no trivial way to find out which dependencies might also be removed. However, if you specify multiple packages for removal on a single command, then packages without dependencies will be removed before packages that have dependencies.

When you remove packages using rpm, there is no prompt before the packages are removed, unlike when you install packages. However, if you attempt to remove a package that is required for some other package, the operation is not performed and you get an error message as shown in .

If you use yum remove instead, then you will be prompted after the transaction tests are performed. If the package you are trying to remove is a dependent package for some other installed packages, then YUM will offer to remove those as well as the dependent package, as shown in .

## Upgrading RPM packages

Now that you know how to install and remove an RPM, let's look at upgrading RPM packages to a newer level. You can use yum update to update your entire system, or you can specify a single package or a wildcard specification. shows how to update all the packages whose names start with "pop". Note the use of apostrophes to prevent shell expansion of the "\*".

If you know where the RPM files are located, or have downloaded them, you can also update them by using the rpm command. This is similar to installing, except that you use the -U or the -F option instead of the -i option. The difference between these two options is that the -U option will upgrade an existing package **or** install the package if it is not already installed, while the -F option will only upgrade or *freshen* a package that is already installed. Because of this, the -U option is frequently used, particularly when the command line contains a list of RPMs. This way, uninstalled packages are installed, while installed packages are upgraded. Two other options, -v (verbose) and -h (hash marks), are often used to give progress indication. shows how to update the cairo package and its cairo-gobject dependency using the rpm command. We have the cairo rpm already downloaded in root's home directory, while we retrieve the cairo-gobject package from one of the update mirrors.

## Querying RPM packages

In our examples you saw that installing an rpm with the rpm command requires the full name of the package file (or URL), such as gcc-gfortran-4.9.2-6.fc21.x86\_64.rpm. On the other hand, installing with yum, or removing an rpm with either command requires only the package name, such as gcc-gfortran. As with APT, RPM maintains an internal database of your installed packages, allowing you to manipulate installed packages using the package name. In this section, we look at some of the information that is available to you from this database using the -q (for query) option of the rpmcommand, or the associated yum queries.

We'll reinstall the gcc-gfortran package to provide our examples.

The basic query simply asks if a package is installed, and, if so, what version. Add the -i option and you get information about the package. Note that you need to have root authority to install, upgrade, or remove packages, but non-root users can perform queries against the rpm database.

##### **Displaying information about gcc-gfortran**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62 | [ian@attic-f21 ~]$ <**strong**>yum list gcc-gfortran</**strong**>  Loaded plugins: langpacks  Installed Packages  gcc-gfortran.x86\_64                    4.9.2-6.fc21                     @updates  Available Packages  gcc-gfortran.i686                      4.9.2-6.fc21                     updates    [ian@attic-f21 ~]$ <**strong**>rpm -q gcc-gfortran</**strong**>  gcc-gfortran-4.9.2-6.fc21.x86\_64    [ian@attic-f21 ~]$ <**strong**>yum info gcc-gfortran</**strong**>  Loaded plugins: langpacks  Installed Packages  Name        : gcc-gfortran  Arch        : x86\_64  Version     : 4.9.2  Release     : 6.fc21  Size        : 18 M  Repo        : installed  From repo   : updates  Summary     : Fortran support  URL         : http://gcc.gnu.org  License     : GPLv3+ and GPLv3+ with exceptions and GPLv2+ with exceptions and              : LGPLv2+ and BSD  Description : The gcc-gfortran package provides support for compiling Fortran              : programs with the GNU Compiler Collection.    Available Packages  Name        : gcc-gfortran  Arch        : i686  Version     : 4.9.2  Release     : 6.fc21  Size        : 7.5 M  Repo        : updates/21/x86\_64  Summary     : Fortran support  URL         : http://gcc.gnu.org  License     : GPLv3+ and GPLv3+ with exceptions and GPLv2+ with exceptions and              : LGPLv2+ and BSD  Description : The gcc-gfortran package provides support for compiling Fortran              : programs with the GNU Compiler Collection.    [ian@attic-f21 ~]$ <**strong**>rpm -qi gcc-gfortran</**strong**>  Name        : gcc-gfortran  Version     : 4.9.2  Release     : 6.fc21  Architecture: x86\_64  Install Date: Mon 27 Jul 2015 09:36:14 PM EDT  Group       : Development/Languages  Size        : 19126083  License     : GPLv3+ and GPLv3+ with exceptions and GPLv2+ with exceptions and LGPLv2+ and BSD  Signature   : RSA/SHA256, Fri 13 Feb 2015 09:02:15 PM EST, Key ID 89ad4e8795a43f54  Source RPM  : gcc-4.9.2-6.fc21.src.rpm  Build Date  : Thu 12 Feb 2015 07:40:58 AM EST  Build Host  : buildhw-08.phx2.fedoraproject.org  Relocations : (not relocatable)  Packager    : Fedora Project  Vendor      : Fedora Project  URL         : http://gcc.gnu.org  Summary     : Fortran support  Description :  The gcc-gfortran package provides support for compiling Fortran  programs with the GNU Compiler Collection. |

### RPM packages and files in them

You will often want to know what is in a package or what package a particular file came from. To list the files in the gcc-gfortran package, use the -ql option as shown in . There are many files in this package, so we've only shown part of the output.

You can restrict the files listed to just configuration files by adding the -c option to your query. Similarly, the -d option limits the display to just documentation files.

### Querying package files

The above package commands query the RPM database for installed packages. If you just downloaded a package and want the same kind of information, you can get this using the -p option (for package file) on your query along with specifying the package file name (as used for installing the package). shows this for the two vim packages that we downloaded earlier. We run it as root only because the files were in root's home directory. You can add other query options, such as -l to list files or -i to list information.

### Querying all installed packages

The -a option applies your query to all installed packages. This can generate a lot of output, so you will usually use it in conjunction with one or more filters, such as sort to sort the listing, more or less to page through it, wc to obtain package or file counts, or grep to search for packages if you aren't sure of the name. shows the following queries:

1. A sorted list of all packages on the system
2. A count of all packages on the system
3. A count of all files in all packages on the system
4. A count of all documentation files installed with RPMs
5. A search for all packages with "fortran" (case-insensitive) as part of their name

### Which package owns a file?

Given that you can list all packages and all files in a package, you now have all the information you need to find which package owns a file. However, the rpm command provides a -f (or --file) option to help you locate the package that owns an installed file. Suppose you want to know which of the vim packages we saw earlier actually provides the vim command. You will need to provide the full path to the file. shows how to use the which command to get the full path to the vim command, and a handy tip for using this output as input to the rpm -qf command. Note that the tick marks surrounding `which vim` are back-ticks. Another way of using this in the Bash shell is to use $(which vim).

##### **Which package supplies the vim executable**

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | [ian@attic-f21 ~]$ <**strong**>which vim</**strong**>  /usr/bin/vim  [ian@attic-f21 ~]$ <**strong**>rpm -qf `which vim`</**strong**>  vim-enhanced-7.4.475-2.fc21.x86\_64  [ian@attic-f21 ~]$ <**strong**>rpm -qf $(which vim)</**strong**>  vim-enhanced-7.4.475-2.fc21.x86\_64 |

### RPM dependencies

The rpm command provides an option to interrogate installed packages or package files to find out what capabilities they depend on or *require*.This is the --requires option, which may be abbreviated to -R. shows the capabilities required by gcc-gfortran. Add the -p option and use the full RPM file name if you want to query the package file instead of the RPM database.

## RPM package file integrity

To ensure their integrity, RPM packages include a digest, such as MD5 or SHA1, and are usually digitally signed. Packages that are digitally signed need a public key for verification. To check the integrity of an RPM package file, use the --checksig (abbreviated to -K) option of rpm. You will usually find it useful to add the -v option for more verbose output. shows an example for the vim-enhanced RPM.

##### **Checking the integrity of the vim-enhanced package file**

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | [ian@attic-f21 ~]$ <**strong**>rpm -vK vim-enhanced-7.4.475-2.fc21.x86\_64.rpm</**strong**>  vim-enhanced-7.4.475-2.fc21.x86\_64.rpm:      Header V3 RSA/SHA256 Signature, key ID 95a43f54: OK      Header SHA1 digest: OK (696e492a4ee7a672cb3851d220de804dce0c9484)      V3 RSA/SHA256 Signature, key ID 95a43f54: OK      MD5 digest: OK (6bc225b37f43e7e7075668d04a73b9ea) |

## Using rpm2cpio

If you download an RPM and need to examine its contents, rather than install it, you can use the rpm2cpiocommand to convert the contents to a cpio archive and then filter that through the cpiocommand to extract individual files or all the files in the package. shows how to do this for the gcc-gfortran package and then shows some of the files (and directories) that were unpacked. See the man pages for rpm2cpio and cpio for additional details on these commands.

##### **Unpacking the gcc-gfortran package with rpm2cpio**

[[ian@attic-f21 ~]$ <**strong**>mkdir gcc-gfortran</**strong**>

[ian@attic-f21 ~]$ <**strong**>cd gcc-gfortran</**strong**>

[ian@attic-f21 <**strong**>gcc-gfortran]$ rpm2cpio ../gcc-gfortran-4.9.2-6.fc21.x86\_64.rpm | cpio -idv</**strong**>

./usr/bin/f95

./usr/bin/gfortran

./usr/lib/gcc

./usr/lib/gcc/x86\_64-redhat-linux

./usr/lib/gcc/x86\_64-redhat-linux/4.9.2

./usr/lib/gcc/x86\_64-redhat-linux/4.9.2/32

./usr/lib/gcc/x86\_64-redhat-linux/4.9.2/32/libcaf\_single.a

./usr/lib/gcc/x86\_64-redhat-linux/4.9.2/32/libgfortran.a

./usr/lib/gcc/x86\_64-redhat-linux/4.9.2/32/libgfortran.so

./usr/lib/gcc/x86\_64-redhat-linux/4.9.2/32/libgfortranbegin.a

./usr/lib/gcc/x86\_64-redhat-linux/4.9.2/finclude

## Downloading RPMs from repositories

Although yum will automatically retrieve packages from repositories, you may want to download RPMs and save them, perhaps to install them on a non-networked system, or to examine their contents, or for some other reason. You can use the yumdownloader command to do this as shown in . In our case, the gcc-gfortran.x86\_64 package is already installed, so there are no additional packages to download.

##### **Downloading the gcc-gfortran package**

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | [ian@attic-f21 ~]$ <**strong**>yumdownloader --resolve gcc-gfortran.x86\_64</**strong**>  Loaded plugins: langpacks  --> Running transaction check  ---> Package gcc-gfortran.x86\_64 0:4.9.2-6.fc21 will be reinstalled  --> Finished Dependency Resolution  gcc-gfortran-4.9.2-6.fc21.x86\_64.rpm                       | 7.7 MB  00:04 |

The --resolve option of yumdownloader will cause other required packages to be downloaded, too. To illustrate this also shows the files downloaded using the --resolve option when we download gcc-gfortran. Note that we did not specify an architecture (x86\_64 or i686), so the default download is for the i686 version.

## Topic 103: GNU and UNIX commands

# 103.1 The Linux command line

## Commands and sequences

The shell's main function is to interpret your commands so you can interact with your Linux system. On Linux (and UNIX®) systems, commands have a *command name*, and then *options* and *parameters*. Some commands have neither options nor parameters, and some have one but not the other.

### Echo

The echo command prints (or echos) its arguments to the terminal as shown in Listing 3.

##### **Listing 3. Echo examples**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | [ian@atticf20 ~]$ echo Word  Word  [ian@atticf20 ~]$ echo A phrase  A phrase  [ian@atticf20 ~]$ echo Where     are   my   spaces?  Where are my spaces?  [ian@atticf20 ~]$ echo "Here     are   my   spaces." # plus comment  Here     are   my   spaces. |

Както се вижда от примера има голяма разлика със и без кавичките. Може да се използват единични или двойни кавички.

### Bash shell metacharacters and control operators

Bash has several metacharacters, which when not quoted, also serve to divide input into words. Besides a blank, these are:

* |
* &
* ;
* (
* )
* <
* >

We will discuss some of these in more detail in other parts of this tutorial. For now, note that if you want to include a metacharacter as part of your text, it must be either quoted or escaped using a backslash (\) as shown in Listing 4.

The new line and certain metacharacters or pairs of metacharacters also serve as control operators. These are:

* ||
* &&
* &
* ;
* ;;
* |
* (
* )

echo $? – показва дали е изпълнена правилно предходната команда.

0 – ок

1- грешка

echo $! – показва PID на последния процес в бекграунд.

## Environment variables

When you are running in a bash shell, many things constitute your environment, such as the form of your prompt, your home directory, your working directory, the name of your shell, files that you have opened, functions that you have defined, and so on. Your environment includes many variables that may have been set by bash or by you. The bash shell also allows you to have shell variables, which you may export to your environment for use by other processes running in the shell or by other shells that you may spawn from the current shell.

Both environment variables and shell variables have a name. You reference the value of a variable by prefixing its name with '$'. Some of the common bash environment variables that you will encounter are shown in Table 2.

##### **Table 2. Some common bash environment variables**

| **Name** | **Function** |
| --- | --- |
| USER | The name of the logged-in user |
| UID | The numeric user id of the logged-in user |
| HOME | The user's home directory |
| PWD | The current working directory |
| SHELL | The name of the shell |
| $ | The process id (or PIDof the running bash shell (or other) process |
| PPID | The process id of the process that started this process (that is, the id of the parent process) |
| ? | The exit code of the last command |

Listing 7 shows what you might see in some of these common bash variables.

##### **Listing 7. Environment and shell variables**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | [ian@atticf20 ~]$ echo $USER $UID  ian 1000  [ian@atticf20 ~]$ echo $SHELL $HOME $PWD  /bin/bash /home/ian /home/ian  [ian@atticf20 ~]$ (exit 0);echo $?;(exit 4);echo $?  0  4  [ian@atticf20 ~]$ echo $$ $PPID  3175 2457 |

В shell **променливите** имат **име**, **значение** и понякога **атрибути**.  
**Името** е идентификатор на променливата. То е символен низ, който може да се състои от букви, цифри и символа за подчертаване ("\_"). Името на променлива трябва да започва с буква или символа "\_" и е case-sensitive (различават се главните от малките букви).  
**Значението** на една променлива е символен низ.  
**Декларирането** на променлива не е задължително. То може да става при присвояване на значение. По default, променливата има значение "" (празен низ), което по-късно може да бъде зададено с командата **declare**.  
Променливата е част от обкръжението на процеса shell (environment), докато процесът завърши или преди да се подаде командата **unset**, която изключва променливата от обкръжението на процеса.

## *Присвояване на значение на променлива*

То става по два начина:

### С ****оператор за присвояване****, който има следния вид:

**име\_на\_променлива=[значение]**  
Пример:  
x=123  
y=  
На променливата с име x ще се присвои значението "123", а на променливата с име y ще се присвои празен низ за значение - "".  
Значението е низ, който може да е ограден с единични или двойни кавички, но те може и да не присъстват. Ако в низа няма интервали или метасимволи, тогава значението може да е без кавички. Ако присъстват интервали, то е необходимо значението да бъде заградено в кавички, а ако има (и) метасимволи, те се екранират по следното правило:

* ако низът е заграден в двойни кавички ("низ"), то се екранират всички метасимволи, без **$**, **\**, **'**, **"**;
* ако низът е заграден в единични кавички ('низ'), то се екранират всички метасимволи, без **'**.

### С команда ****read****

**read име\_на\_променлива**  
Командата може да е вътрешна или външна за shell-а. Чете се един ред от стандартния вход и се раделя на думи (разделителен символ е интервал), като всяка дума се съпоставя на съответната променлива (аргумент на променливата). При тази команда може да възникнат несъответствия. Например, при подадени команда **read** и променливи **x**, **y** и **z**, ето какво ще се случи в следните два случая на четене на низове от стандартния вход:

* 123 aa bb cc

На x ще се присвои значението "123", на y - значението "аа", а на z - "bb cc".

* 123 аа

На x ще се присвои значението "123", на y - "aa", a нa z - "" (празен низ).

Други примери за присвояване на значения на променливи:

* L=Linux, L="Linux" и L='Linux' са един и същ вариант, понеже в значението на променливата няма интервали;
* D=/home/student/proc/dir1 - тук символът "/" не е метасимвол;
* OS="Red Hat Linux" и OS='Red Hat Linux' са едно и също, понеже имаме само интервали, но нямаме метасимволи;
* A="a<b" и A='a<b' са едно и също, т.к. няма нито метасимволи, нито интервали.

## *Заместване на променлива*

Замествания се правят от shell-a, а не от команди.

### Конструкциите

**$име\_на\_променлива** и **${име\_на\_променлива}**  
се заместват със значението на променливата. Втората конструкция се използва, когато {} ограничават името на променливата, например когато след променливата следват символи, които не са отделени от нея с интервали:

* Пример 1:

L=Linux  
OS="Red Hat $L"  
Тук се извършва заместване на $L със значението "Linux" на променливата L. В резултат, на променливата с име OS ще се присвои стойността "Red Hat Linux".  
OS="Red Hat ${L}7.2"  
На променливата с име OS ще се присвои стойността "Red Hat Linux7.2".

* Пример 2:

D=/home/student/proc/dir1  
cd $D  
ls -l $D/\*.c  
Тук $D се замества със значението на променливата D, което в случая е "/home/student/proc/dir1". Пример 2 е еквивалентен на следното:  
cd /home/student/proc/dir1  
ls -l /home/student/proc/dir1/\*.c

### Други замествания на променливи:

**${име\_на\_променлива:=дума}**  
**${име\_на\_променлива:-дума}**  
**${име\_на\_променлива:+дума}**  
**${име\_на\_променлива:?дума}**  
При тези конструкции се проверява дали променливата е определена. Ако променливата е без значение ("", празен низ), то се дава новото значение, което е определено от дума, и се извършва заместването; в противен случай заместване се извършва със старото значение на променливата. Ако липсва ":", то проверката за празен низ не се извършва, а директно се прави заместване на значението на променливата с новото, определено от дума.  
трябва да се опише какво правят конструкциите с -, + и ?.//

**${име\_на\_променлива:отместване[:брой]}**  
С тази конструкция се извършва заместване на подниз, с подадено начало в отместване и дължина, равна на брой. Ако броят на символите, които са дължината на подниза, не е зададен, то се счита, че подниза е до края на значението на променливата.

**${#име\_на\_променлива}**  
Конструкцията се замества с броя символи в значението на променливата.

## *Системни променливи*

В login shell (все още нямаме никакви задавания), shell-a прави инициализация на обкръжението. След това, с подаване на set или declare, получаваме няколко екрана с променливи. Някои от тях са доста важни.  
Кои са вътрешните системни променливи, зависи и то това, дали имаме shell, csh, bash или др. Ето някои променливи:

* **HOME**

Тази променлива означава пълно име на началния каталог на потребителя. Тя може да се променя.  
Командата **cd** е еквивалентна на командата **cd $HOME**.

* **PATH**

Променливата съдържа списък от имена на каталози, в които shell-a търси външни команди.  
**:** се използва за разделител между имената на каталозите.  
**PATH** може да изглежда така:  
PATH=/bin:/usr/bin:/usr/local/bin  
Примери:  
PATH=$PATH:$HOME/bin # това е традиционно; bin може да е с друго име  
export PATH # хубаво е да се прави, независимо дали PATH е export или не  
# export е добре да се използва за системните променливи

* **PS1**

Първична покана (prompt) на shell.  
Стандартното означение при обикновен shell e: **$** за обикнове потребител и **#** за администратора.  
Означението при bash е: **bash$** за обикновен потребител и **bash#** за администратора.  
Но ВЕЧЕ стандартните означения може да са променени в различните шелове.  
Ето как може да изглежда променливата **PS1**:  
**PS1='\u @ \h \w \$/# '**  
Тук "\u" означава "user", "\h" - "host", "\w" - "текуща директория (work dir)", "\$/#" - "$" или "#".

* **PS2**

Вторична покана на shell. Стандартното означение е **>**.  
Съществуват и **PS3** и **PS4**.

* **UID**

**User ID** - вътрешен идентификатор на потребителя. Има атрибут **readonly** и **intern** (ползва се в ядрото).

* **IFS**

**Internal Field Seperator** - това са символите, които се разглеждат като разделители на думи при **word splitting** и командата **read**. Стандартно са: **"<space><tab><newline>"**.

* **PWD**

Име на текущ каталог.

* **OLDPWD**

Име на предишния текущ каталог.

* **USER**

Име на потребителя.

* **SHELL**

Име на потребителския login shell.

* **BASH**

Име на файла с изпълнимия код на bash. Стандартно е /bin/bash, но може да е и /usr/bash или др.

* **BASH\_VERSION**

Версията на bash.

Следва списък с такива (срещу името на съответната променлива е значението й):

* **$#** - брой аргументи, подадени на shell-a, без нулевия (аргументът с името на файла с процедурата);
* **$@** - извежда всички аргументи, предадени на shell-a, без нулевия, разделени с интервали; # "$1" "$2" …
* **$**\* - извежда всичко аргументи, предадени на shell-a, без нулевия, разделени с IFS (input field seperator); не се прави word splitting (word splitting може да прехамва символи, например символа за нов ред); # "$1c$2c…"
* **$?** - код на завършване на последната изпълнена команда в привилегирован режим;
* **$$** - идентификатор (pid) на процеса shell;
* **$!** - идентификатор (pid) на последния изпълнен фонов процес.

## Setting values to environment variables

In order to set a value to an existing environment variable, we use an assignment expression. For instance, to set the value of the "LANG" variable to "he\_IL.UTF-8", we use the following command:

LANG=he\_IL.UTF-8

If we use an assignment expression for a variable that doesn't exist, the shell will create a shell variable, which is similar to an environment variable but does not influence the behaviour of other applications.

A shell variable can be exported to become an environment variable with the exportcommand. To create the "EDITOR" environment variable and assign the value "nano" to it, you can do:

EDITOR=nano export EDITOR

The bash shell (the default command-line shell in Ubuntu) provides a shortcut for creating environment variables. The previous example could be performed with the following single command:

export EDITOR=nano

## Examining values of environment variables

The printenv command prints the names and values of all currently defined environment variables:

printenv

To examine the value of a particular variable, we can specify its name to the printenv command:

printenv TERM

Another way to achieve that is to use the dollar sign ($), as used in the following example:

echo $TERM

There is a command for doing temporary, short-term changes to the environment. It can also be used to display the current environment. This command is env.

env

The dollar sign can actually be used to combine the values of environment variables in many shell commands. For example, the following command can be used to list the contents of the "Desktop" directory within the current user's home directory.

ls $HOME/Desktop

For the sake of completeness: If you want to print the names and values also of the non-exported shell variables, i.e. not only the environment variables, this is one way:

( set -o posix ; set ) | less

### Desktop environment specifics

If you use a terminal window to examine environment variables, you actually study the environment variables of the shell that is running in the terminal. Those variables are not necessarily available in the graphical environment of the desktop. To examine the environment variables that are effective when you for instance start an application from [the launcher](https://help.ubuntu.com/stable/ubuntu-help/unity-launcher-intro.html), you can follow the guidance in [this Ask Ubuntu answer](http://askubuntu.com/questions/275965/how-to-list-all-variables-names-and-their-current-values/356973#356973).

## Erasing environment variables

While simply setting an empty value to an environment variable, as shown in the example below, may nullify its effect in most cases, there are a few variables such as "POSIXLY\_CORRECT" whose mere existence, even with an empty value, influences the behavior of programs.

export LC\_ALL=

The unset command can be used in order to completely erase the existence of an environment variable:

unset LC\_ALL

It is also possible to use the "-n" switch to the export command in order to un-export an environment variable and therefore demote it to become a shell variable while preserving its value.

export -n LC\_ALL

The earlier discussion of quoting mentioned that you could use either single or double quotes. There is an important difference between them. The shell expands shell variables that are between double quotes ($quot;), but expansion is not done when single quotes (') are used. In the previous example, we started another shell within our shell and we got a new process id. Using the -c option, you can pass a command to the other shell, which will execute the command and return. If you pass a quoted string as a command, your outer shell will strip the quotes and pass the string. If double quotes are used, variable expansion occurs **before** the string is passed, so the results may not be as you expect. The shell and command will run in another process so they will have another PID. Listing 9 illustrates these concepts. The PID of the top-level bash shell is highlighted.

##### **Listing 9. Quoting and shell variables**

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | [ian@atticf20 ~]$ echo "$SHELL" '$SHELL' "$$" '$$'  /bin/bash $SHELL 3175 $$  [ian@atticf20 ~]$ bash -c "echo Expand in parent $$ $PPID"  Expand in parent 3175 2457  [ian@atticf20 ~]$ bash -c 'echo Expand in child $$ $PPID'  Expand in child 4541 3175 |

### Env

The env command without any options or parameters displays the current environment variables. You can also use it to execute a command in a custom environment. The -i (or just -) option clears the current environment before running the command, while the -u option unsets environment variables that you do not wish to pass.

Listing 11 shows partial output of the env command without any parameters and then three examples of invoking different shells without the parent environment. Look carefully at these before we discuss them.

**Note:** If your system does not have the ksh (Korn) or tcsh shells installed, you will need to install them to do these exercises yourself.

##### **Listing 11. The env command**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31 | [ian@atticf20 ~]$ env  XDG\_VTNR=2  XDG\_SESSION\_ID=1  HOSTNAME=atticf20  GPG\_AGENT\_INFO=/run/user/1000/keyring/gpg:0:1  SHELL=/bin/bash  TERM=xterm-256color  XDG\_MENU\_PREFIX=gnome-  VTE\_VERSION=4002  HISTSIZE=1000  GJS\_DEBUG\_OUTPUT=stderr  WINDOWID=35651982  GJS\_DEBUG\_TOPICS=JS ERROR;JS LOG  QT\_GRAPHICSSYSTEM\_CHECKED=1  USER=ian  ...\_=/usr/bin/env  OLDPWD=/home/ian/Documents  [ian@atticf20 ~]$ env -i bash -c 'echo $SHELL; env'  /bin/bash  PWD=/home/ian  SHLVL=1  \_=/usr/bin/env  [ian@atticf20 ~]$ env -i ksh -c 'echo $SHELL; env'  /bin/sh  \_=\*3175\*/usr/bin/env  PWD=/home/ian  SHLVL=1  \_AST\_FEATURES=UNIVERSE - ucb  A\_\_z="\*SHLVL  [ian@atticf20 ~]$ env -i tcsh -c 'echo $SHELL; env'  SHELL: Undefined variable. |

### Exec

One final command to cover is exec. You can use the exec command to run another program that**replaces** the current shell. Listing 13 starts a child bash shell and then uses exec to replace it with a Korn shell. Upon exit from the Korn shell, you are back at the original bash shell (PID 2852, in this example).

##### **Listing 13. Using exec**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | [ian@atticf20 ~]$ echo $$  3175  [ian@atticf20 ~]$ bash  [ian@atticf20 ~]$ echo $$  4994  [ian@atticf20 ~]$ exec ksh  $ echo $$  4994  $ exit  [ian@atticf20 ~]$ echo $$  3175 |

## System information with uname

The uname command prints information about your system and its kernel. Listing 14 shows the various options for uname and the resulting information; each option is defined in Table 3.

##### **Listing 14. The uname command**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | [ian@atticf20 ~]$ uname  Linux  [ian@atticf20 ~]$ uname -s  Linux  [ian@atticf20 ~]$ uname -n  atticf20  [ian@atticf20 ~]$ uname -r  4.0.4-303.fc22.x86\_64  [ian@atticf20 ~]$ uname -v  #1 SMP Thu May 28 12:37:06 UTC 2015  [ian@atticf20 ~]$ uname -m  x86\_64  [ian@atticf20 ~]$ uname -o  GNU/Linux  [ian@atticf20 ~]$ uname -a  Linux atticf20 4.0.4-303.fc22.x86\_64 #1 SMP Thu May 28 12:37:06 UTC 2015 x86\_64 x86\_64 x86\_64 GNU/Linux |

##### **Table 3. Options for uname**

| **Option** | **Description** |
| --- | --- |
| -s | Print the kernel name. This is the default if no option is specified. |
| -n | Print the nodename or hostname. |
| -r | Print the release of the kernel. This option is often used with module-handling commands. |
| -v | Print the version of the kernel. |
| -m | Print the machine's hardware (CPU) name. |
| -o | Print the operating system name. |
| -a | Print all of the above information. |

## Command history

If you are typing in commands as you read, you may notice that you often use a command many times, either exactly the same or with slight changes. The good news is that the bash shell can maintain a history of your commands. By default, history is on. You can turn it off using the command set +o history and turn it back on using set -o history. An environment variable called HISTSIZE tells bash how many history lines to keep. A number of other settings control how history works and is managed. See the bash man pages for full details.

Some of the commands that you can use with the history facility are:

**history**

Displays the entire history.

**history N**

Displays the last N lines of your history.

**history -d N**

Deletes line N form your history; you might do this if the line contains a password, for example.

**!!**

Your most recent command.

**!N**

The Nth history command.

**!-N**

The command that is N commands back in the history (!-1 is equivalent to !!).

**!#**

The current command you are typing.

**!string**

The most recent command that starts with string.

**!?string?**

The most recent command that contains string.

### Where does the shell find commands?

External commands are just files in your file system. Basic file management is covered in another tutorial in this series. (See [Related topics](https://www.ibm.com/developerworks/library/l-lpic1-103-1/#artrelatedtopics) for the series roadmap). On Linux and UNIX systems, all files are accessed as part of a single large tree that is rooted at /. In our examples so far, our current directory has been the user's home directory. Non-root users usually have a home directory within the /home directory, such as /home/ian, in my case. Root's home is usually /root. If you type a command name, then bash looks for that command on your path, which is a colon-separated list of directories in the PATH environment variable.

If you want to know what command will be executed if you type a particular string, use the which or type command. Listing 17 shows my default path along with the locations of several commands.

The which command reported that the ls command is an *alias* and that the setcommand could not be found. In this case, we interpret that to mean that it does not exist or that it is a builtin. The type command reports that the ls command is an *alias*, but it identifies the set command as a shell builtin. It also reports that there is a builtin echo command as well as the one in /bin that was found by which. The commands also produce output in different orders.

In addition to manual pages accessible from a command line, the Free Software Foundation has created a number of *info* files that are processed with the *info* program. These provide extensive navigation facilities including the ability to jump to other sections. Try man info or info info for more information. Not all commands are documented with info, so you will find yourself using both man and info if you become an info user.

he man page for bash is very long and it can take some time to search through it, even using the lesspager. Fortunately, if you just want some quick help on a bash builtin you can use the help builtin as shown in Listing 22. Use help with no arguments to get a list of available help and use help help to find out how to use help.

# 103.2 Text streams and filters

**Cat, od, and split**

Now that you have created the text1 file, you might want to check what is in it. Use the cat (short for *concatenate)*command to display the contents of a file on stdout. Listing 3 verifies the contents of the file created in Listing 2.

Listing 3. Displaying file contents with cat

|  |  |
| --- | --- |
| 1  2  3  4 | ian@Z61t-u14:~/lpi103-2$ cat text1  1 apple  2 pear  3 banana |

The cat command takes input from stdin if you do not specify a filename (or if you specify - as the filename). Let's use this along with output redirection to create another text file as shown in Listing 4.

Listing 4. Creating a text file with cat

|  |  |
| --- | --- |
| 1  2  3  4 | ian@Z61t-u14:~/lpi103-2$ cat >text2  9       plum  3       banana  10      apple |

In Listing 4, cat keeps reading from stdin until the end of the file. Use the **Ctrl-d** key (hold **Ctrl** and press **d**) combination to signal end of file. This is the same key combination to exit from the bash shell. Use the tab key to line up the fruit names in a column.

Remember that *cat* was short for *concatenate*? You can use cat to concatenate several files together for display. Listing 5 shows the two files that we have just created.

Listing 5. Concatenating two files with cat

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | ian@Z61t-u14:~/lpi103-2$ cat text\*  1 apple  2 pear  3 banana  9   plum  3   banana  10  apple |

When you display these two text files using cat, you notice alignment differences. To learn what causes this, you need to look at the control characters that are in the file. These are acted on in text display output rather than having some representation of the control character itself displayed, so we need to *dump* the file in a format that allows you to find and interpret these special characters. The GNU text utilities include an od (or *O*ctal *D*ump) command for this purpose.

There are several options for od, such as the -A option to control the radix of the file offsets and -t to control the form of the displayed file contents. The radix can be specified as o, (octal, the default), d (decimal), x (hexadecimal), or n (no offsets displayed). You can display output as octal, hex, decimal, floating point, ASCII with backslash escapes, or named characters (for example, nl for newline or ht for horizontal tab). Listing 6 shows some of the formats available for dumping the text2 example file.

Listing 6. Dumping files with od

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | ian@Z61t-u14:~/lpi103-2$ od text2  0000000 004471 066160 066565 031412 061011 067141 067141 005141  0000020 030061 060411 070160 062554 000012  0000031  ian@Z61t-u14:~/lpi103-2$ od -A d -t c text2  0000000   9  \t   p   l   u   m  \n   3  \t   b   a   n   a   n   a  \n  0000016   1   0  \t   a   p   p   l   e  \n  0000025  ian@Z61t-u14:~/lpi103-2$ od -A n -t a text2     9  ht   p   l   u   m  nl   3  ht   b   a   n   a   n   a  nl     1   0  ht   a   p   p   l   e  nl |

**Notes:**

* The -A option of cat provides an alternate way of seeing where your tabs and line endings are. See the man page for more information.
* If you see spaces instead of tabs in your own text2 file, refer to [Expand, unexpand, and tr](https://www.ibm.com/developerworks/library/l-lpic1-103-2/#3-expand), later in this tutorial to see how to switch between tabs and spaces in a file.
* If you have a mainframe background, you might be interested in the hexdump utility, which is part of a different utility set. It's not covered here, so check the man pages.

Our sample files are very small, but sometimes you have large files that you need to split into smaller pieces. For example, you might want to break a large file into CD-sized chunks so you can write it to a CD for sending through the mail to someone who could create a DVD for you. The split command does this in such a way that the cat command can be used to re-create the file easily. By default, the files resulting from the split command have a prefix in their name of 'x' followed by a suffix of 'aa', 'ab', 'ac', ..., 'ba', 'bb', and so on. Options permit you to change these defaults. You can also control the size of the output files and whether the resulting files contain whole lines or just byte counts.

Listing 7 illustrates splitting our two text files with different prefixes for the output files. We split text1 into files containing at most two lines, and text2 into files containing at most 18 bytes. We then use cat to display some of the pieces individually as well as to display a complete file using *globbing*, which is covered in the tutorial on [*basic file and directory management*](http://www.ibm.com/developerworks/library/l-lpic1-v3-103-3/).

Listing 7. Splitting and recombining with split and cat

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | ian@Z61t-u14:~/lpi103-2$ split -l 2 text1  ian@Z61t-u14:~/lpi103-2$ split -b 17 text2 y  ian@Z61t-u14:~/lpi103-2$ cat yaa  9   plum  3   banana  1ian@Z61t-u14:~/lpi103-2$ cat yab  0   apple  ian@Z61t-u14:~/lpi103-2$ cat y\* x\*  9   plum  3   banana  10  apple  1 apple  2 pear  3 banana |

Note that the split file named yaa did not finish with a newline character, so our prompt was offset after we used cat to display it.

**Wc, head, and tail**

Cat displays the whole file. That's fine for small files like our examples, but suppose you have a large file. Well, first you might want to use the wc (*W*ord *C*ount) command to see how big the file is. The wc command displays the number of lines, words, and bytes in a file. You can also find the number of bytes by using ls -l. Listing 8 shows the long format directory listing for our two text files, as well as the output from wc.

Listing 8. Using wc with text files

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | ian@Z61t-u14:~/lpi103-2$ ls -l text\*  -rw-rw-r-- 1 ian ian 24 Jun  8 13:26 text1  -rw-rw-r-- 1 ian ian 25 Jun  8 13:36 text2  ian@Z61t-u14:~/lpi103-2$ wc text\*   3  6 24 text1   3  6 25 text2   6 12 49 total |

Options allow you to control the output from wc or to display other information such as maximum line length. See the man page for details.

Two commands allow you to display either the first part (*head*) or last part (*tail*) of a file. These commands are the head and tail commands. They can be used as filters, or they can take a filename as an argument. By default, they display the first (or last) 10 lines of the file or stream. Listing 9 uses the dmesg command to display bootup messages, in conjunction with wc, tail, and head to discover that there are 791 messages, then to display the last 10 of these, and finally to display the six messages starting 15 from the end.

Listing 9. Using wc, head, and tail to display boot messages

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | ian@Z61t-u14:~/lpi103-2$ dmesg|wc     1100    9541   74856  ian@Z61t-u14:~/lpi103-2$ dmesg | tail  [ 8334.796068] wlan0: direct probe to 00:02:6f:ff:39:0c (try 2/3)  [ 8335.000081] wlan0: direct probe to 00:02:6f:ff:39:0c (try 3/3)  [ 8335.204065] wlan0: authentication with 00:02:6f:ff:39:0c timed out  [ 8351.649200] wlan0: authenticate with 6c:b0:ce:fb:43:02  [ 8351.658626] wlan0: send auth to 6c:b0:ce:fb:43:02 (try 1/3)  [ 8351.661222] wlan0: authenticated  [ 8351.664069] wlan0: associate with 6c:b0:ce:fb:43:02 (try 1/3)  [ 8351.678348] wlan0: RX AssocResp from 6c:b0:ce:fb:43:02 (capab=0x431 status=0 aid=3)  [ 8351.678479] wlan0: associated  [12050.026812] perf interrupt took too long (2502 > 2500), lowering kernel.perf\_event\_max\_sample\_ra  te to 50000  ian@Z61t-u14:~/lpi103-2$ dmesg | tail -n15 | head -n 6  [ 8324.132080] wlan0: direct probe to 6c:b0:ce:fb:43:02 (try 2/3)  [ 8324.336072] wlan0: direct probe to 6c:b0:ce:fb:43:02 (try 3/3)  [ 8324.540069] wlan0: authentication with 6c:b0:ce:fb:43:02 timed out  [ 8334.589068] wlan0: authenticate with 00:02:6f:ff:39:0c  [ 8334.594323] wlan0: direct probe to 00:02:6f:ff:39:0c (try 1/3)  [ 8334.796068] wlan0: direct probe to 00:02:6f:ff:39:0c (try 2/3) |

Another common use of tail is to *follow* a file that uses the -f option, usually with a line count of 1. You might use this when you have a background process that is generating output in a file and you want to check in and see how it is doing. In this mode, tail runs until you cancel it (using **Ctrl-c**), displaying lines as they are written to the file.

**Expand, unexpand, and tr**

When we created our text1 and text2 files, we created text2 with tab characters. Sometimes, you might want to swap tabs for spaces or vice versa. The expand and unexpand commands do this. With the -t option for both commands, you can set the tab stops. A single value sets repeated tabs at that interval. Listing 10expand and unexpand that unaligns the text in text2.

Listing 10. Using expand and unexpand

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | ian@Z61t-u14:~/lpi103-2$ expand -t 1 text2  9 plum  3 banana  10 apple  ian@Z61t-u14:~/lpi103-2$ expand -t8 text2|unexpand -a -t2|expand -t3  9           plum  3           banana  10       apple  ian@Z61t-u14:~/lpi103-2$ cat text1 |tr ' ' '\t'|cat - text2  1   apple  2   pear  3   banana  9   plum  3   banana  10  apple |

Unfortunately, you cannot use unexpand to replace the spaces in text1 with tabs, as unexpand requires at least two spaces to convert to tabs. However, you can use the tr command, which translates characters in one set (*set1*) to corresponding characters in another set (*set2*). Listing 11 shows how to use tr to translate spaces to tabs. Because tr is purely a filter, you generate input for it using the cat command. This example also illustrates the use of - to signify standard input to cat, so we can concatenate the output of tr and the text2 file.

The tr utility copies the given input to produced the output with substitution or deletion of selected characters. tr abbreviated as translate or transliterate. It takes as parameters two sets of characters, and replaces occurrences of the characters in the first set with the corresponding elements from the other set i.e. it is used to translate characters.

1. Convert lower case to upper case

The following tr command is used to convert the lower case to upper case

$ tr abcdefghijklmnopqrstuvwxyz ABCDEFGHIJKLMNOPQRSTUVWXYZ

thegeekstuff

THEGEEKSTUFF

The following command will also convert lower case to upper case

$ tr [:lower:] [:upper:]

thegeekstuff

THEGEEKSTUFF

You can also use ranges in tr. The following command uses ranges to convert lower to upper case.

$ tr a-z A-Z

thegeekstuff

THEGEEKSTUFF

2. Translate braces into parenthesis

You can also translate from and to a file. In this example we will translate braces in a file with parenthesis.

$ tr '{}' '()' < inputfile > outputfile

The above command will read each character from “inputfile”, translate if it is a brace, and write the output in “outputfile”.

3. Translate white-space to tabs

The following command will translate all the white-space to tabs

$ echo "This is for testing" | tr [:space:] '\t'

This is for testing

4. Squeeze repetition of characters using -s

In Example 3, we see how to translate space with tabs. But if there are two are more spaces present continuously, then the previous command will translate each spaces to a tab as follows.

$ echo "This is for testing" | tr [:space:] '\t'

This is for testing

We can use -s option to squeeze the repetition of characters.

$ echo "This is for testing" | tr -s [:space:] '\t'

This is for testing

Similarly you can convert multiple continuous spaces with a single space

$ echo "This is for testing" | tr -s [:space:] ' '

This is for testing

5. Delete specified characters using -d option

tr can also be used to remove particular characters using -d option.

$ echo "the geek stuff" | tr -d 't'

he geek suff

To remove all the digits from the string, use

$ echo "my username is 432234" | tr -d [:digit:]

my username is

Also, if you like to delete lines from file, you can use [sed d command](http://www.thegeekstuff.com/2009/09/unix-sed-tutorial-delete-file-lines-using-address-and-patterns/).

6. Complement the sets using -c option

You can complement the SET1 using -c option. For example, to remove all characters except digits, you can use the following.

$ echo "my username is 432234" | tr -cd [:digit:]

432234

7. Remove all non-printable character from a file

The following command can be used to remove all non-printable characters from a file.

$ tr -cd [:print:] < file.txt

**Pr, nl, and fmt**

The pr command is used to format files for printing. The default header includes the filename and file creation date and time, along with a page number and two lines of blank footer. When output is created from multiple files or the standard input stream, the current date and time are used instead of the filename and creation date. You can print files side-by-side in columns and control many aspects of formatting through options. As usual, refer to the man page for details.

The nl command numbers lines, which can be convenient when printing files. You can also number lines with the -n option of the cat command. Listing 12 shows how to print our text file, and then how to number text2 and print it side-by-side with text1.

##### **Listing 12. Numbering and formatting for print**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | ian@Z61t-u14:~/lpi103-2$ pr text1 | head      2015-06-08 13:26                      text1                       Page 1      1 apple  2 pear  3 banana      ian@Z61t-u14:~/lpi103-2$ nl text2 | pr -m - text1 | head      2015-06-08 16:10                                                  Page 1           1  9   plum                1 apple       2  3   banana              2 pear       3  10  apple               3 banana |

Another useful command for formatting text is the fmt command, which formats text so it fits within margins. You can join several short lines as well as split long ones. In Listing 13, we create text3 with a single long line of text using variants of the !#:\* history feature to save typing our sentence four times. We also create text4 with one word per line. Then we use cat to display them unformatted including a displayed '$' character to show line endings. Finally, we use fmt to format them to a maximum width of 60 characters. Again, consult the man page for details on additional options.

##### **Listing 13. Formatting to a maximum line length**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | ian@Z61t-u14:~/lpi103-2$ echo "This is a sentence. " !#:\* !#:1->text3  echo "This is a sentence. " "This is a sentence. " "This is a sentence. ">text3  ian@Z61t-u14:~/lpi103-2$ echo -e "This\nis\nanother\nsentence.">text4  ian@Z61t-u14:~/lpi103-2$ cat -et text3 text4  This is a sentence.  This is a sentence.  This is a sentence. $  This$  is$  another$  sentence.$  ian@Z61t-u14:~/lpi103-2$ fmt -w 60 text3 text4  This is a sentence.  This is a sentence.  This is a  sentence.  This is another sentence. |

## Sort and uniq

## Cut, paste, and join

CUT

When invoking **cut**, use the **-b**, **-c**, or **-f** option, but only one of them.

If no *FILE* is specified, **cut** reads from the [standard input](https://www.computerhope.com/jargon/s/stdin.htm).

For example, let's say you have a file named **data.txt** which contains the following text:

one two three four five alpha beta gamma delta epsilon

In this example, each of these words is separated by a tab character, not spaces. The tab character is the default delimiter of **cut**, so it will by default consider a field to be anything delimited by a tab.

To "cut" only the third field of each line, use the command:

cut -f 3 data.txt

...which will output the following:

three gamma

If instead you want to "cut" only the second-through-fourth field of each line, use the command:

cut -f 2-4 data.txt

...which will output the following:

two three four beta gamma delta

If you want to "cut" only the first-through-second and fourth-through-fifth field of each line (omitting the third field), use the command:

cut -f 1-2,4-5 data.txt

...which will output the following:

one two four five alpha beta delta epsilon

Or, let's say you want the third field and every field after it, omitting the first two fields. In this case, you could use the command:

cut -f 3- data.txt

...which will output the following:

three four five gamma delta epsilon

Specifying a range with *LIST* also applies to **cut**ting characters (**-c**) or bytes (**-b**) from a line. For example, to output only the third-through-twelfth character of every line of **data.txt**, use the command:

cut -c 3-12 data.txt

...which will output the following:

e two thre pha beta g

Remember that the "space" in between each word is actually a single tab character, so both lines of output are displaying ten characters: eight [alphanumeric](https://www.computerhope.com/jargon/a/alphanum.htm) characters and two tab characters. In other words, **cut** is omitting the first two characters of each line, counting tabs as one character each; outputting characters three through twelve, counting tabs as one character each; and omitting any characters after the twelfth.

Counting bytes instead of characters will result in the same output in this case, because in an [ASCII](https://www.computerhope.com/jargon/a/ascii.htm)-[encoded](https://www.computerhope.com/jargon/c/charcode.htm) text file, each character is represented by a single byte (eight bits) of data. So the command:

cut -b 3-12 data.txt

...will, for our file **data.txt**, produce exactly the same output:

e two thre pha beta g

## Specifying A Delimiter Other Than Tab

The tab character is the default delimiter that **cut** uses to determine what constitutes a field. So, if your file's fields are already delimited by tabs, you don't need to specify a different delimiter character.

You can specify any character as the delimiter, however. For instance, the file **/etc/passwd** contains information about each user on the system, one user per line, and each information field is delimited by a colon ("**:**"). For example, the line of **/etc/passwd** for the **root** user may look like this:

root:x:0:0:root:/root:/bin/bash

These fields contain the following information, in the following order, separated by a colon character:

1. Username
2. Password (shown as **x** if [encrypted](https://www.computerhope.com/jargon/e/encrypt.htm))
3. User ID number (UID)
4. Group ID number (GID)
5. Comment field (used by the [finger](https://www.computerhope.com/unix/ufinger.htm) command)
6. [Home Directory](https://www.computerhope.com/jargon/h/homedir.htm)
7. [Shell](https://www.computerhope.com/jargon/s/shell.htm)

The username is the first field on the line, so to display each username on the system, use the command:

cut -f 1 -d ':' /etc/passwd

...which will output, for example:

root daemon bin sys chope

(There are many more user accounts on a typical system, including many accounts specific to system services, but for this example we will pretend there are only five users.)

The third field of each line in the **/etc/passwd** file is the UID (user ID number), so to display each username and user ID number, use the command:

cut -f 1,3 -d ':' /etc/passwd

...which will output the following, for example:

root:0 daemon:1 bin:2 sys:3 chope:1000

As you can see, the output will be delimited, by default, using the same delimiter character specified for the input. In this case, that's the colon character ("**:**"). You can specify a different delimiter for the input and output, however. So, if you wanted to run the previous command, but have the output delimited by a space, you could use the command:

cut -f 1,3 -d ':' --output-delimiter=' ' /etc/passwd

root 0 daemon 1 bin 2 sys 3 chope 1000

But what if you want the output to be delimited by a tab? Specifying a tab character on the command line is a bit more complicated, because it is an unprintable character. To specify it on the command line, you must "protect" it from the shell. This is done differently depending on which shell you're using, but in the Linux default shell ([bash](https://www.computerhope.com/unix/ubash.htm)), you can specify the tab character with **$'\t'**. So the command:

cut -f 1,3 -d ':' --output-delimiter=$'\t' /etc/passwd

...will output the following, for example:

root 0 daemon 1 bin 2 sys 3 chope 1000

**PASTE**

The **paste** command displays the corresponding lines of multiple files side-by-side.

paste file1.txt file2.txt

**JOIN**

Joins the lines of two files which share a common field of data.

If we have a file, **myfile1.txt**, whose contents are:

1 India 2 US 3 Ireland 4 UK 5 Canada

...and another file, **myfile2.txt**, whose contents are:

1 NewDelhi 2 Washington 3 Dublin 4 London 5 Toronto

The common fields are the fields which begin with the same number. We can join the contents using the following command:

join myfile1.txt myfile2.txt

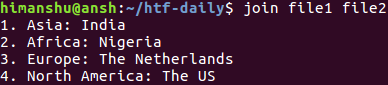
...which outputs the following to standard output:

1 India NewDelhi 2 US Washington 3 Ireland Dublin 4 UK London 5 Canada Toronto

By default, the join command only prints pairable lines. For example, even if file1 contains an extra field (line number 5):

1. Asia:  
2. Africa:  
3. Europe:  
4. North America:  
5. South America:

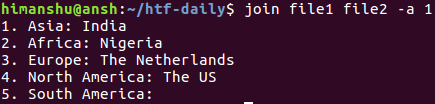
joining file1 and file2 won't produce any different output:

[](https://www.howtoforge.com/images/linux_join_command/big/join-in-action1.png)

That's because unpairable lines are left out in the output. However, if you want, you can still have them in the output using the ***-a*** command line option. This option requires you to pass a file number so that the tool knows which file you are talking about.

For example, in our case, the command would be:

*join file1 file2 -a 1*

[](https://www.howtoforge.com/images/linux_join_command/big/join-display-unpaired.png)

So you can see that the unpaired line from file number 1 (file1 in our case) was also displayed in the output.

Note that in case you just want to print unpaired lines (meaning, suppress the paired lines in output), you can do this using the ***-v*** command line option. This options works exactly the way -a works.

Here's an example of the -v option:

[https://www.howtoforge.com/images/linux_join_command/join-suppress-paired.png](https://www.howtoforge.com/images/linux_join_command/big/join-suppress-paired.png)

## 3. How to provide custom join fields?

As we already know, join combines lines of files on a common field, which is the first field by default. However, if you want, you can specify a different field for each file. For example, consider the following contents in file1 and file2, respectively.

\* 1. Asia:  
& 2. Africa:  
@ 3. Europe:  
# 4. North America:

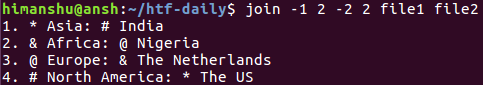
# 1. India  
@ 2. Nigeria  
& 3. The Netherlands  
\* 4. The US

Now, if you want the second field of each line to be the common field for join, you can tell this to the tool by using the ***-1*** and ***-2*** command line options. While the former represents the first file, the latter refers to the second file. These options requires a numeric argument that refers to the joining field for the corresponding file.

For example, in our case, the command will be:

*join -1 2 -2 2 file1 file2*

And here's the output of this command:

[](https://www.howtoforge.com/images/linux_join_command/big/join-custom-fields.png)

**SED – Stream Editor**

Listing 20 shows three simple sed scripts. In the first one, we use the s (substitute) command to substitute an uppercase for a lowercase 'a' on each line. This example replaces only the first 'a', so in the second example, we add the 'g' (for global) flag to cause sed to change all occurrences. In the third script, we introduce the d (delete) command to delete a line. In our example, we use an address of 2 to indicate that only line 2 should be deleted. We separate commands using a semi-colon (;) and use the same global substitution that we used in the second script to replace 'a' with 'A'.

##### **Listing 20. Beginning sed scripts**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | ian@Z61t-u14:~/lpi103-2$ sed 's/a/A/' text1  1 Apple  2 peAr  3 bAnana  ian@Z61t-u14:~/lpi103-2$ sed 's/a/A/g' text1  1 Apple  2 peAr  3 bAnAnA  ian@Z61t-u14:~/lpi103-2$ sed '2d;$s/a/A/g' text1  1 apple  3 bAnAnA |

In addition to operating on individual lines, sed can operate on a range of lines. The beginning and end of the range is separated by a comma (,) and can be specified as a line number, a regular expression, or a dollar sign ($) for the end of file. Given an address or a range of addresses, you can group several commands between curly braces, { and } to have these commands operate only on lines selected by the range. Listing 21 illustrates two ways of having our global substitution applied to only the last two lines of our file. It also illustrates the use of the -e option to add multiple commands to the script.

##### **Listing 21. Sed addresses**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | ian@Z61t-u14:~/lpi103-2$ sed -e '2,${' -e 's/a/A/g' -e '}' text1  1 apple  2 peAr  3 bAnAnA  ian@Z61t-u14:~/lpi103-2$ sed -e '/pear/,/bana/{' -e 's/a/A/g' -e '}' text1  1 apple  2 peAr  3 bAnAnA |

### Filtering With the Sed Command

Sed is also frequently used to filter lines in a file or stream. For example, if you only want to see the lines containing "John," you use:

sed -n '/John/p' songs.txt > johns.txt

which writes the following lines to file johns.txt:

5, Johnny Trash, Title 482, Price $6.50

7, John Lennon, Title 271, Price $7.90

The character after the *s* is the delimiter. It is conventionally a slash, because this is what *ed*, *more*, and *vi* use. It can be anything you want, however. If you want to change a pathname that contains a slash - say /usr/local/bin to /common/bin - you could use the backslash to quote the slash:

sed 's/\/usr\/local\/bin/\/common\/bin/' <old >new

Gulp. Some call this a 'Picket Fence' and it's ugly. It is easier to read if you use an underline instead of a slash as a delimiter:

sed 's\_/usr/local/bin\_/common/bin\_' <old >new

Some people use colons:

sed 's:/usr/local/bin:/common/bin:' <old >new

Others use the "|" character.

sed 's|/usr/local/bin|/common/bin|' <old >new

Pick one you like. As long as it's not in the string you are looking for, anything goes. And remember that you need three delimiters. If you get a "Unterminated `s' command" it's because you are missing one of them.

# 103.3 File and directory management

Повече информация в файла линукс курс.

### Using gzip and gunzip

Compression generally works well on text files. Many image formats already compress the data, so compression might not work well on these or other binary files. To illustrate compression on a reasonably large text file, let's copy /etc/services to the directory you have been using and compress it using gzip as shown in Listing 27. You use the -p option of cp to preserve the time stamp of /etc/services. Note that the compressed file has the same time stamp and has a .gz suffix.

##### **Listing 27. Compressing with gzip**

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | ian@Z61t-u14:~/lpi103-2$ cp -p /etc/services .  ian@Z61t-u14:~/lpi103-2$ ls -l serv\*  -rw-r--r-- 1 ian ian 19558 Dec 30  2013 services  ian@Z61t-u14:~/lpi103-2$ gzip services  ian@Z61t-u14:~/lpi103-2$ ls -l serv\*  -rw-r--r-- 1 ian ian 7538 Dec 30  2013 services.gz |

You decompress a gzipped file using the -d option of gzip or, more commonly, using the gunzipcommand. Listing 28 shows the first of these choices. Note that the uncompressed file now has the original file name and time stamp.

##### **Listing 28. Decompressing with gzip**

|  |  |
| --- | --- |
| 1  2  3 | ian@Z61t-u14:~/lpi103-2$ gzip -d services.gz  ian@Z61t-u14:~/lpi103-2$ ls -l serv\*  -rw-r--r-- 1 ian ian 19558 Dec 30  2013 services |

### Using bzip2 and bunzip2

The bzip2 command operates in a similar manner to gzip as shown in Listing 29.

##### **Listing 29. Compressing with bzip2**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | ian@Z61t-u14:~/lpi103-2$ ls -l serv\*  -rw-r--r-- 1 ian ian 19558 Dec 30  2013 services  ian@Z61t-u14:~/lpi103-2$ bzip2 services  ian@Z61t-u14:~/lpi103-2$ ls -l serv\*  -rw-r--r-- 1 ian ian 7208 Dec 30  2013 services.bz2  ian@Z61t-u14:~/lpi103-2$ bunzip2 services.bz2  ian@Z61t-u14:~/lpi103-2$ ls -l serv\*  -rw-r--r-- 1 ian ian 19558 Dec 30  2013 services |

### Using xz and unxz

The xz command is a newer compression command that uses a Lempel–Ziv–Markov chain algorithm (LZMA), first used in the 7-Zip archiver. LZMA2 is a container format that can hold data compressed with possibly different LZMA parameters and plain uncompressed data. The xz command's native format (.xz) is a container for a single compressed stream, making it similar to both gzip and bzip2 in this regard. See Resources for more information.

Not surprisingly, the xz command operates in a similar manner to gzip and bzip2 as shown in Listing 30.

##### **Listing 30. Compressing with xz**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | ian@Z61t-u14:~/lpi103-2$ ls -l serv\*  -rw-r--r-- 1 ian ian 19558 Dec 30  2013 services  ian@Z61t-u14:~/lpi103-2$ xz services  ian@Z61t-u14:~/lpi103-2$ ls -l serv\*  -rw-r--r-- 1 ian ian 7144 Dec 30  2013 services.xz  ian@Z61t-u14:~/lpi103-2$ unxz services.xz  ian@Z61t-u14:~/lpi103-2$ ls -l serv\*  -rw-r--r-- 1 ian ian 19558 Dec 30  2013 services |

# 103.4 Streams, pipes, and redirects

Повече информация в линукс курс файла.

### Using the xargs command

### Xargs Basic Example

The xargs command (by default) expects the input from stdin, and executes /bin/echo command over the input. The following is what happens when you execute xargs without any argument, or when you execute it without combining with any other commands.

## xargs examples

find /tmp -name core -type f -print | xargs /bin/rm -f

Find files named **core** in or below the directory **/tmp** and delete them. Note that this will work incorrectly if there are any filenames containing newlines or spaces.

find /tmp -name core -type f -print0 | xargs -0 /bin/rm -f

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 spaces or newlines are correctly handled.

cut -d: -f1 < /etc/passwd | sort | xargs echo

Uses [**cut**](https://www.computerhope.com/unix/ucut.htm) to generate a compact listing of all the users on the system.

**Using TEE Command**

Reads from [standard input](https://www.computerhope.com/jargon/s/stdin.htm), and writes to standard output and to [files](https://www.computerhope.com/jargon/f/file.htm).

## tee examples

ls -1 \*.txt | wc -l | tee count.txt

In the above example, the [**ls**](https://www.computerhope.com/jargon/l/ls.htm) command lists all files in the current directory that have the file name extension **.txt**, one file per line; this output is [piped](https://www.computerhope.com/jargon/p/pipe.htm) to [**wc**](https://www.computerhope.com/jargon/w/wc.htm), which counts the lines and outputs the number; this output is piped to **tee**, which writes the output to the [terminal](https://www.computerhope.com/jargon/t/terminal.htm), and writes the same information to the file **count.txt**. If **count.txt** already exists, it is overwritten.

You can also use tee command to store the output of a command to a file and redirect the same output as an input to another command.

The following command will take a backup of the crontab entries, and pass the crontab entries as an input to sed command which will do the substituion. After the substitution, it will be added as a new cron job.

$ crontab -l | tee crontab-backup.txt | sed 's/old/new/' | crontab –

By default tee command overwrites the file. You can instruct tee command to append to the file using the option –a as shown below.

$ ls | tee –a file

You can also write the output to multiple files as shown below.

$ ls | tee file1 file2 file3

# 103.5 Create, monitor, and kill processes

Run a process after log out

In practice, you probably want to have standard IO streams for background processes redirected to or from a file. There is another related question: what happens to the process if the controlling terminal closes or the user logs off? The answer depends on the shell in use. If the shell sends a SIGHUP (or hangup) signal, then the application is likely to close. I cover signals shortly, but for now, we'll consider another way around this problem.

**nohup**

The nohup command is used to start a command that ignores hangup signals and appends stdout and stderr to a file. The default file is either nohup.out or $HOME/nohup.out. If the file cannot be written, then the command does not run. If you want output to go somewhere else, redirect stdout, or stderr as discussed in the tutorial "[Learn Linux 101: Streams, pipes and redirects](http://www.ibm.com/developerworks/library/l-lpic1-103-4/index.html)."

The nohup command will not execute a pipeline or a command list. You can save a pipeline or list in a file and then run it using the sh (default shell) or the bash command. Another tutorial in this series shows you how to make the script file executable, but for now we'll stick to running scripts by using the sh or the bash command. Listing 8 shows how we might do this for our poor man's digital clock. Needless to say, having the time written to a file isn't particularly useful, and the file will keep growing, so we'll set the clock to update every 30 seconds instead of every second.

Listing 8. Using nohup with a command list in a script

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | [ian@atticf20 ~]$ echo "while sleep 30; do date;done">pmc.sh  [ian@atticf20 ~]$ nohup sh pmc.sh&  [3] 6955  [ian@atticf20 ~]$ nohup: ignoring input and appending output to ‘nohup.out’    [ian@atticf20 ~]$ nohup bash pmc.sh&  [4] 6970  nohup: ignoring input and appending output to ‘nohup.out’ |

**Monitor processes**

PS Command

There is another command, the ps command, which we use to display various pieces of process status information. Remember "ps" as an acronym for "process status." The ps command accepts zero or more PIDs as arguments and displays the associated process status. If we use the jobs command with the -p option, the output is simply the PID of the *process group leader* for each job.

Several options, including -f (full), -j (jobs), and -l (long) give control of how much information is displayed. If you do not specify any PIDs, then another useful option is the --forest option, which displays the commands in a tree hierarchy, showing which process has which other process as a parent. In particular, you see that the sleep commands of the previous listing are children of the scripts you have running in background. If you happened to run the command at a different instant, you might see the date command listed in the process status instead, but the odds are very small with this script.

**FREE - Command**

The free command displays the amount of free and used memory in your system. By default the display is in kilobytes, but you can override this using -b for bytes, -k for kilobytes, -m for megabytes, or -g for gigabytes. The -t option displays a total line, and the -s option along with a value refreshes the info with the frequency specified. The number is in seconds but might be a floating point value. Listing 15 shows two examples.

Listing 15. Using the free command

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | [ian@atticf20 ~]$ free                total        used        free      shared  buff/cache   available  Mem:        4046816     1845624      334164       21212     1867028     1926056  Swap:             0           0           0  [ian@atticf20 ~]$ free -mt                total        used        free      shared  buff/cache   available  Mem:           3951        1801         327          20        1823        1882  Swap:             0           0           0  Total:         3951        1801         327 |

**Uptime**

The uptime command shows you a one-line display that includes the current time, how long the system has been running, how many users are currently logged on, and the system load averages for the past 1, 5, and 15 minutes. Listing 16 shows an example.

Listing 16. Showing uptime information

|  |  |
| --- | --- |
| 1  2 | [ian@atticf20 ~]$ uptime   17:22:39 up  8:53,  4 users,  load average: 0.41, 0.62, 0.66 |

**Select and sort processes for display**

**Using ps**

The ps commands discussed so far only list processes that were started from your terminal session (note the SID, or session id, column in the second example of [Listing 14](https://www.ibm.com/developerworks/library/l-lpic1-103-5/index.html#listing14)). To see all the processes with controlling terminals, use the -aoption. The -x option displays processes without a controlling terminal, and the -e option displays information for **every**process. You'll find if you run this on a system where you or anyone else is logged in on a graphical desktop that you will have a lot of output. Listing 17 shows the full (-f) format for all the processes with a controlling terminal on two different systems (Ubuntu 15 and Fedora 22). For the second system, we've truncated the output but then showed how many lines of output there really were. I suspect there is a bug in Fedora 22 in that a daemon process should normally detach from its controlling terminal, so we should not be seeing daemon processes in this output.

Listing 17. Displaying other processes

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26 | jenni@yoga-u15:~$ ps -af # Ubuntu 15.04  jenni@yoga-u15:~$ ps -af  UID        PID  PPID  C STIME TTY          TIME CMD  root      3168  1934  0 12:23 pts/1    00:00:00 sudo -s  root      3169  3168  0 12:23 pts/1    00:00:00 /bin/bash  root      4938  4769  0 22:03 pts/5    00:00:00 su - jenni  jenni     4939  4938  0 22:03 pts/5    00:00:00 -su  jenni     5013  4994  0 22:06 pts/19   00:00:00 ps -af    [ian@atticf20 ~]$ ps -af #Fedora 22  UID        PID  PPID  C STIME TTY          TIME CMD  gdm       1050  1045  0 08:29 tty1     00:00:00 dbus-daemon --print-address 3 --  gdm       1089  1045  0 08:29 tty1     00:00:00 /usr/bin/gnome-session --autosta  ...  ian       4788     1  0 14:09 tty2     00:00:00 /usr/libexec/gvfsd-http --spawne  ian       6250  4111  4 16:20 pts/1    00:14:24 cairo-clock --seconds  ian       6268  4111  6 16:21 pts/1    00:21:55 cairo-clock --seconds --theme an  ian       6955  4111  0 17:01 pts/1    00:00:00 sh pmc.sh  ian       6970  4111  0 17:02 pts/1    00:00:00 bash pmc.sh  ian      10758 10715  0 21:24 pts/3    00:00:00 ssh ian@192.168.1.42  ian      10962  6955  0 21:36 pts/1    00:00:00 sleep 30  ian      10963 10922  0 21:36 pts/4    00:00:00 ssh ian@192.168.1.24  ian      11138  6970  0 21:36 pts/1    00:00:00 sleep 30  ian      11139 10974  0 21:36 pts/5    00:00:00 ps -af  [ian@atticf20 ~]$ ps -af | wc -l  83 |

Note the controlling terminal listed in the TTY column. For the Fedora 22 part of this listing, I switched to an ssh login (pts/5), so the ps -af command is running under pts/5, while most of the commands created for this tutorial are running under pts/1.

There are many more options for ps, including a number that provide significant control over what fields are displayed and how they are displayed. Others provide control over the selection of processes for display, for example, by selecting those processes for a particular user (-u) or a particular command (-C). In Listing 18, user jenni finds all processes running the bash command; we use the -o option to specify the columns that are displayed. We've added the user option to the normal list that you get with just plain ps, so you can see which user runs bash.

Listing 18. Who is running the bash command?

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | [jenni@atticf20 ~]$ ps -C bash -o user,pid,tty,time,comm  jenni@yoga-u15:~$ ps -af # Ubuntu 15.04  UID        PID  PPID  C STIME TTY          TIME CMD  root      3168  1934  0 12:23 pts/1    00:00:00 sudo -s  root      3169  3168  0 12:23 pts/1    00:00:00 /bin/bash  root      4938  4769  0 22:03 pts/5    00:00:00 su - jenni  jenni     4939  4938  0 22:03 pts/5    00:00:00 -su  jenni     5014  4994  0 22:08 pts/19   00:00:00 ps –af |

Sometimes you might want to sort the output by particular fields, and you can do that too using the --sort option to specify the sort fields. The default is to sort in ascending order (+), but you can also specify descending order (-). Listing 19 shows the final ps example where all processes are listed using the jobs format, and the output is sorted by session id and command name. For the first, we use the default sort order, and for the second, we specify both sort orders explicitly.

Listing 19. Sorting the output from ps

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | jenni@yoga-u15:~$ ps -aj --sort -sid,+comm # Ubuntu 15.04    PID  PGID   SID TTY          TIME CMD   5083  5083  4994 pts/19   00:00:00 ps   4939  4939  4769 pts/5    00:00:00 bash   4938  4938  4769 pts/5    00:00:00 su   3169  3169  1934 pts/1    00:00:00 bash   3168  3168  1934 pts/1    00:00:00 sudo  jenni@yoga-u15:~$ ps -aj --sort sid,comm # Ubuntu 15.04    PID  PGID   SID TTY          TIME CMD   3169  3169  1934 pts/1    00:00:00 bash   3168  3168  1934 pts/1    00:00:00 sudo   4939  4939  4769 pts/5    00:00:00 bash   4938  4938  4769 pts/5    00:00:00 su   5085  5085  4994 pts/19   00:00:00 ps |

**Pgrep**

Another useful command, albeit with fewer options, is the pgrep command, which allows you to use a regular expression to search for processes by name. The -a option also prints the command line and the -f option matches against the full command line. An example is shown in Listing 20 searching for our clock processes.

Listing 20. Using pgrep to find clock processes

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | [ian@atticf20 ~]$ pgrep iro # Fedora 22  6250  6268  [ian@atticf20 ~]$ pgrep -af "i.\*seconds"  6250 cairo-clock –seconds  6268 cairo-clock --seconds --theme antique |

**Using top**

If you run ps several times in a row to see what is changing, you probably need the top command instead. It displays a continuously updated process list, along with useful summary information. Listing 21 shows the first few lines of a topdisplay. Use the **q** subcommand to quit **top**.

Listing 21. Displaying processes using top

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | jenni@yoga-u15:~$ top # Ubuntu 15.04  top - 22:20:01 up 10:11,  5 users,  load average: 0.00, 0.01, 0.05  Tasks: 206 total,   1 running, 205 sleeping,   0 stopped,   0 zombie  %Cpu(s):  0.4 us,  0.1 sy,  0.0 ni, 99.5 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st  KiB Mem:   8090136 total,  2355204 used,  5734932 free,    50760 buffers  KiB Swap:  4095996 total,        0 used,  4095996 free.  1713188 cached Mem      PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND   1257 ian       20   0 1225160 176140  85496 S   1.7  2.2   3:31.96 compiz   1252 ian       20   0  628292  37536  25508 S   0.3  0.5   0:05.15 unity-pane+   4026 root      20   0       0      0      0 S   0.3  0.0   0:00.54 kworker/2:0   5086 jenni     20   0   29184   3084   2560 R   0.3  0.0   0:00.05 top      1 root      20   0  182724   5400   3676 S   0.0  0.1   0:02.08 systemd      2 root      20   0       0      0      0 S   0.0  0.0   0:00.00 kthreadd |

The top command has a number of subcommands, of which the most useful to start with are:

* **h**: Gets you help
* **q**: Quits the top command
* **f**: Lets you add or remove fields from the display
* F: Selects fields to sort on

**Send signals to processes**

Let's now look at Linux *signals*, which are an asynchronous way to communicate with processes. We have already mentioned the SIGHUP signal, and we have used both Ctrl-c and Ctrl-z, which are other ways of sending a signal to processes. The general way to send a signal is with the kill command. You can also use the pkill or killall commands to signal multiple processes with one command.

**Sending signals using kill**

The kill command sends a signal to a specified job or process. Listing 23 shows the use of the SIGTSTP and SIGCONT signals to stop and resume a background job. Using the SIGTSTP signal is equivalent to using the fg command to bring the job to the foreground and then Ctrl-z to suspend it. Using SIGCONT is like using the bg command.

Listing 23. Stopping and restarting background jobs

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | [ian@atticf20 ~]$ kill -s SIGTSTP %1    [1]+  Stopped                 cairo-clock --seconds 2> /dev/null  [ian@atticf20 ~]$ jobs -l  [1]+  6250 Stopped                 cairo-clock --seconds 2> /dev/null  [2]   6268 Running                 cairo-clock --seconds --theme antique 2> /dev/null &  [3]   6955 Running                 nohup sh pmc.sh &  [4]-  6970 Running                 nohup bash pmc.sh &  [ian@atticf20 ~]$ kill -s SIGCONT 6250  [ian@atticf20 ~]$ jobs -l  [1]   6250 Running                 cairo-clock --seconds 2> /dev/null &  [2]   6268 Running                 cairo-clock --seconds --theme antique 2> /dev/null &  [3]-  6955 Running                 nohup sh pmc.sh &  [4]+  6970 Running                 nohup bash pmc.sh & |

There are a number of other possible signals that you can display on your system using kill -l. Some are used to report errors such as illegal operation codes, floating point exceptions, or attempts to access memory that a process does not have access to. Notice that signals have both a number, such as 20, and a name, such as SIGTSTP. You can use either the number prefixed by a - sign, or the -s option and the signal name. On my system, I could have used kill -20 instead ofkill -s SIGTSTP. You should always check the signal numbers on your system before assuming which number belongs to which signal.

**Signal handlers and process termination**

You have seen that Ctrl-c terminates a process. In fact, it sends a SIGINT (or interrupt) signal to the process. If you use kill without any signal name, it sends a SIGTERM signal. For most purposes, these two signals are equivalent.

You also saw that the nohup command makes a process immune to the SIGHUP signal. In general, a process can implement a *signal handler* to *catch* signals. So a process could implement a signal handler to catch either SIGINT or SIGTERM. Because the signal handler knows what signal was sent, it might choose to ignore SIGINT and only terminate when it receives SIGTERM, for example. Listing 24 shows how to send the SIGTERM signal to job %1 using the kill and the send the SIGTERM (default) signal to the other cairo-clock process using part of the command name as a pattern. Notice that the process status shows as "Terminated" right after we send the signal. This would show as "Interrupt" if we used SIGINT instead. After a few moments, the process cleanup has occurred and the job no longer shows in the job list. Finally, we send the SIGHUP signal to all of our processes that are running the sleep command. Because this is what pmc.sh does most of the time, this is what we find if we use pgrep to look for processes running sleep. Not surprisingly, the processes keep right on running, because we started them with the intention that they would ignore the SIGHUP signal.

Listing 24. Terminating a process with SIGTERM

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | [ian@atticf20 ~]$ kill -s SIGTERM %1  [1]   Terminated              cairo-clock --seconds 2> /dev/null  [ian@atticf20 ~]$ pkill cairo  [2]   Terminated              cairo-clock --seconds --theme antique 2> /dev/null  [ian@atticf20 ~]$ pgrep -a sleep; pkill --signal SIGHUP sleep;jobs -l  13628 sleep 30  13631 sleep 30  [3]-  6955 Running                 nohup sh pmc.sh &  [4]+  6970 Running                 nohup bash pmc.sh & |

**Logout and nohup**

Remember you saw that using nohup would allow your processes to keep running after you log out. Well, let's do that and then log back in again. After you log back in, check your remaining poor man's clock process using jobs as we have done above. The output is shown in Listing 25.

Listing 25. Logging back in

|  |  |
| --- | --- |
| 1 | [ian@atticf20 ~]$ jobs -l |

There is no sign of our jobs! Not perhaps what we were expecting. However, all is not lost. Suppose you can't remember whether you terminated the nohup job that you started with bash or the one you started with sh. You saw previously how to find the processes that were running the bash command, so you can use the same trick to display just the SID, PID, PPID, and command string. Then you can use the -js option to display all the processes in the session using the SID (Session ID) value. Listing 26 shows the result. Think about other ways you might have found these processes, such as searching by username and then filtering using grep.

Listing 26. Finding our lost commands

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | [ian@atticf20 ~]$ ps -C bash -C sh -o pid,sid,tname,cmd    PID   SID TTY      CMD   2180  2180 pts/0    bash   3779  1474 tty2     /bin/sh /home/ian/Oxygen XML Author 17/oxygenAuthor17.0   4158  4158 pts/2    bash   6955  4111 ?        sh pmc.sh   6970  4111 ?        bash pmc.sh  10715 10715 pts/3    bash  10922 10922 pts/4    bash  10974 10974 pts/5    -bash  11262 11262 pts/6    bash  13702 13702 pts/1    bash  [ian@atticf20 ~]$ ps -js 4111    PID  PGID   SID TTY          TIME CMD   6955  6955  4111 ?        00:00:00 sh   6970  6970  4111 ?        00:00:00 bash  14394  6955  4111 ?        00:00:00 sleep  14402  6970  4111 ?        00:00:00 sleep |

Note that the pmc.sh is still running but now it has a question mark (?) for the controlling TTY.

Given what you have now learned about killing processes, you should be able to kill the remaining poor man's clock processes using their PIDs and the kill command. There is one more command we'll introduce for killing processes, the killall command, which has many parameters in common with pgrep and pkill. Unfortunately it does not identify our processes by the pmc.sh name, so we'll just use it to kill all the sleep commands. This is used here solely for illustration; selecting one command from a running shell script is NOT generally a good way to find a particular set of processes. You might accidentally kill something totally unrelated. Using killall would have been more appropriate for killing the cairo-clock processes, but we already killed those.

Listing 27. Using the killall command

|  |  |
| --- | --- |
| 1  2  3 | [ian@atticf20 ~]$ killall sleep  [ian@atticf20 ~]$ ps -js 4111    PID  PGID   SID TTY          TIME CMD |

# 103.6 Process execution priorities

Knowing your priorities

If you run the top command, its default is to display processes in decreasing order according to their CPU usage, as shown in Listing 1. In the previous tutorial, "[Learn Linux 101: Create, monitor, and kill processes](http://www.ibm.com/developerworks/library/l-lpic1-103-5/)," we showed a Poor Man's Clock script, which prints the time on the console every 30 seconds and does nothing for the rest of the time. If we ran that process, it probably wouldn't make it onto the output list from top because the process spends most of its time not using the CPU.

Listing 1. Typical output from top on a Linux workstation

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24 | top - 22:47:44 up 1 day, 12:44,  3 users,  load average: 0.00, 0.01, 0.05  Tasks: 188 total,   1 running, 187 sleeping,   0 stopped,   0 zombie  %Cpu(s):  0.2 us,  0.0 sy,  0.0 ni, 99.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st  KiB Mem:   8090144 total,  2145616 used,  5944528 free,    81880 buffers  KiB Swap:  4095996 total,   100660 used,  3995336 free.  1464920 cached Mem      PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND   2215 ian       20   0 1549252 162644  79992 S   0.7  2.0  16:11.61 compiz      9 root      20   0       0      0      0 S   0.3  0.0   0:08.04 rcuos/0   4918 ian       20   0   29184   3120   2612 R   0.3  0.0   0:00.47 top      1 root      20   0  182732   5392   3648 S   0.0  0.1   0:03.74 systemd      2 root      20   0       0      0      0 S   0.0  0.0   0:00.02 kthreadd      3 root      20   0       0      0      0 S   0.0  0.0   0:00.10 ksoftirqd/0      5 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/0:0H      7 root      20   0       0      0      0 S   0.0  0.0   0:08.42 rcu\_sched      8 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcu\_bh     10 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcuob/0     11 root      rt   0       0      0      0 S   0.0  0.0   0:00.00 migration/0     12 root      rt   0       0      0      0 S   0.0  0.0   0:00.77 watchdog/0     13 root      rt   0       0      0      0 S   0.0  0.0   0:00.78 watchdog/1     14 root      rt   0       0      0      0 S   0.0  0.0   0:00.18 migration/1     15 root      20   0       0      0      0 S   0.0  0.0   0:00.11 ksoftirqd/1     17 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/1:0H     18 root      20   0       0      0      0 S   0.0  0.0   0:00.46 rcuos/1 |

When you only have one or a limited number of CPUs, you need to decide how to share those limited CPU resources among several competing processes. This is generally done by selecting one process for execution and letting it run for a short period (called a timeslice), or until it needs to wait for some event, such as IO to complete. To ensure that important processes don't get starved out by CPU hogs, the selection is done based on a scheduling priority. The **NI** column in Listing 1 above, shows the scheduling priority or niceness of each process. Niceness generally ranges from -20 to 19, with -20 being the most favorable or highest priority for scheduling and 19 being the least favorable or lowest priority.

**Using ps to find niceness**

In addition to the top command, you can also display niceness values using the ps command. You can either customize the output as you saw in the tutorial "[Learn Linux 101: Create, monitor, and kill processes](http://www.ibm.com/developerworks/library/l-lpic1-103-5/)," or you can just use the -l option to get a long listing. The output of ps -l is shown in Listing 2. As with top, look for the niceness value in the **NI** column.

Listing 2. Using ps to find niceness

|  |  |
| --- | --- |
| 1  2  3  4 | ian@yoga-u15:~$  ps -l  F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD  0 S  1000  3850  3849  0  80   0 -  6726 wait   pts/5    00:00:00 bash  0 R  1000  4924  3850  0  80   0 -  3561 -      pts/5    00:00:00 ps |

**Default niceness**

You may have guessed from Listing 1 or Listing 2 that the default niceness, at least for processes started by regular users, is 0. This is usually the case on current Linux systems. You can verify the value for your shell and system by running the nicecommand with no parameters as shown in Listing 3.

Listing 3. Checking default niceness

|  |  |
| --- | --- |
| 1  2 | ian@yoga-u15:~$ nice  0 |

**Using nice to set priorities**

Now that we can keep a CPU busy for a while, we'll see how to set a priority for a process. To summarize what we've learned so far:

* Linux and UNIX® systems use a priority system with 40 priorities, ranging from -20 (highest priority) to 19 (lowest priority.
* Processes started by regular users usually have priority 0.
* The ps command can display the priority (nice, or NI, level, for example) using the -l option.
* The nice command displays our default priority.

The nice command can also be used to start a process with a different priority. You use the -n or (--adjustment) option with a positive value to increase the priority value and a negative value to decrease it. Remember that processes with the lowest priority value run at highest scheduling priority, so think of increasing the priority value as being *nice* to other processes. Note that you usually need to be the superuser (root) to specify negative priority adjustments. In other words, regular users can usually only make their processes nicer.

**Changing priorities**

**Renice**

If you happen to start a process and realize that it should run at a different priority, there is a way to change it after it has started, using the renice command. You specify an absolute priority (and not an adjustment) for the process or processes to be changed as shown in Listing 10.

Listing 10. Using renice to change priorities

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | ian@yoga-u15:~$ sh count1.sh 100000000 A&  [1] 5724  ian@yoga-u15:~$ A Fri Jul 17 17:30:20 EDT 2015  renice 1 5724;ps -l 5724  5724 (process ID) old priority 0, new priority 1  F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY        TIME CMD  0 R  1000  5724  3850 99  81   1 -  1118 -      pts/5      0:35 sh count1.sh 10000  ian@yoga-u15:~$ renice +3 5724;ps -l 5724  5724 (process ID) old priority 1, new priority 3  F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY        TIME CMD  0 R  1000  5724  3850 99  83   3 -  1118 -      pts/5      0:50 sh count1.sh 10000  ian@yoga-u15:~$ sudo renice -8 5724;ps -l 5724  5724 (process ID) old priority 3, new priority -8  F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY        TIME CMD  0 R  1000  5724  3850 99  72  -8 -  1118 -      pts/5      1:01 sh count1.sh 10000 |

Remember that you have to be the superuser to give your processes higher scheduling priority and make them less nice.

# 103.7 Search text files using regular expressions

**Regular expression building blocks**

Two forms of regular expression syntax are used with the GNU grep program found on most Linux systems: *basic* and *extended*. With GNU grep, there is no difference in functionality. Basic syntax is described here, along with the differences between it and extended syntax.

Regular expressions are built from *characters* and *operators*, augmented by *metacharacters*. Most characters match themselves, and most metacharacters must be escaped using a backslash (\). The fundamental operations are:

**Concatenation**

Concatenating two regular expressions creates a longer expression. For example, the regular expression **a** will match the string **abcdcba** twice (the first and last **a**) and so will the regular expression **b**. However, **ab** will match only **ab**cdcba, while **ba** will match only abcdc**ba**.

**Repetition**

The Kleene \* or repetition operator will match zero or more occurrences of the preceding regular expression. Thus, an expression like **a\*b** will match any string of **a**s terminated by a **b**, including just **b** itself (that is a string of 0 **a**s followed by**b**). The Kleene \* does not have to be escaped, so an expression in which you want to match a literal asterisk (\*) must have the asterisk escaped. The use of \* here is different from the use in globbing, where it matches any string.

**Alternation**

The alternation operator (|) matches either the preceding or following expression. It must be escaped in basic syntax. For example, the expression **a\*\|b\*c** will match a string consisting of any number of **a**s or a string of any number of **b**s (but not both) terminated by a single **c**. Again, the single character **c** matches.

You often need to quote your regular expressions to avoid shell expansion.

**Search files and filesystems**

We will use the text files that we created earlier as examples (see "[Setting up the examples](https://www.ibm.com/developerworks/library/l-lpic1-103-7/index.html#setup-ex)"). Study the simple examples of Listing 3. Note that grep takes a regular expression as a required parameter and a list of zero or more files to search. If no files are given, grep searches stdin, which makes it a filter that can be used in pipelines. If no lines match, there is no output from grep, although its exit code can be tested.

Listing 3. Simple regular expressions

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | ian@yoga-u15:~/lpi103-7$ grep p text1  1 apple  2 pear  ian@yoga-u15:~/lpi103-7$ grep pea text1  2 pear  ian@yoga-u15:~/lpi103-7$ grep "p\*" text1  1 apple  2 pear  3 banana  ian@yoga-u15:~/lpi103-7$ grep "pp\*" text1  1 apple  2 pear  ian@yoga-u15:~/lpi103-7$ grep "x" text1; echo $?  1  ian@yoga-u15:~/lpi103-7$ grep "x\*" text1; echo $?  1 apple  2 pear  3 banana  0  ian@yoga-u15:~/lpi103-7$ cat text1 | grep "l\|n"  1 apple  3 banana  ian@yoga-u15:~/lpi103-7$ echo -e "find an \ns\* here" | grep "s\\*" |

**First shortcuts**

Now that you can use the basic building blocks of regular expressions with grep, here are some convenient shortcuts.

**+**

The + operator is like the \* operator, except that it matches **one** or more occurrences of the preceding regular expression. It must be escaped for basic expressions.

**?**

The ? indicates that the preceding expression is optional, so it represents zero or one occurrence of it. This is not the same as the ? used in globbing.

**.**

The . (dot) is a metacharacter that stands for any character. One of the most commonly used patterns is **.\***, which matches an arbitrary length string containing any characters (or no characters at all). Needless to say, you will find this used as part of a longer expression. Compare a single dot with the ? used in globbing and .\* with the \* used in globbing.

Listing 4. More regular expressions

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | ian@yoga-u15:~/lpi103-7$ grep "pp\+" text1 # at least two p's  1 apple  ian@yoga-u15:~/lpi103-7$ grep "pl\?e" text1  1 apple  2 pear  ian@yoga-u15:~/lpi103-7$ grep "pl\?e" text1 # pe with optional l between  1 apple  2 pear  ian@yoga-u15:~/lpi103-7$ grep "p.\*r" text1 # p, some string then r  2 pear  ian@yoga-u15:~/lpi103-7$ grep "a.." text1 # a followed by two other letters  1 apple  3 banana |

Matching beginning or end of line

The ^ (caret) matches the beginning of a line while the $ (dollar sign) matches the end of a line. So **^..b** matches any two characters at the beginning of a line followed by a **b**, while **ar$** matches any line ending in **ar**. The regular expression **^$**matches an empty line.

**More complex expressions**

# 103.8 File editing with vi

**Moving around**

These commands help you move around in a file:

**h**

Move left one character on the current line

**j**

Move down to the next line

**k**

Move up to the previous line

**l**

Move right one character on the current line

**w**

Move to the next word on the current line

**e**

Move to the next end of word on the current line

**b**

Move to the previous beginning of the word on the current line

**Ctrl-f**

Scroll forward one page

**Ctrl-b**

Scroll backward one page

If you type a number before any of these commands, then the command will be executed that many times. This number is called a *repetition count* or simply *count*. For example, 5h will move left five characters. You can use repetition counts with many vi commands.

**Moving to lines**

The following commands help you move to specific lines in your file:

**G**

Moves to a specific line in your file. For example, 3G moves to line 3. With no parameter, G moves to the last line of the file.

**H**

Moves relative to the top line on the screen. For example, 3H moves to the line currently 3rd from the top of your screen.

**L**

Is like H, except that movement is relative to the last line on screen. Thus, 2L moves to the second-to-last line on your screen.

Practice these commands until you are comfortable moving around the file. If you get stuck and things aren't working as expected, read on and learn how to get out of the file.

**Getting out of vi**

One of the most useful things to know about a new editor is how to get out of it before you do anything you shouldn't do, such as destroying an important configuration file. You can get out of vi by saving or abandoning your changes, or by restarting from the beginning. If these commands don't seem to work for you, you may be in insert mode, which you will learn about in a moment. If in doubt, pressing **Esc** will leave insert mode and return you to command mode where these commands should work.

**:q!**

Quit editing the file and abandon all changes. This is a very common idiom for getting out of trouble.

**:w!**

Write the file (whether modified or not). Attempt to overwrite existing files or read-only or other unwritable files. You may give a filename as a parameter, and that file will be written instead of the one your started with. It's generally safer to omit the ! unless you know what you're doing here.

**ZZ**

Write the file if it has been modified. Then exit. This is a very common idiom for normal vi exit.

**:e!**

Edit the current disk copy of the file. This will reload the file, abandoning changes you have made. You may also use this if the disk copy has changed for some other reason and you want the latest version.

**:!**

Run a shell command. Type the command and press **Enter**. When the command completes, you will see the output and a prompt to return to vi editing.

Notes:

1. When you type the colon (:), your cursor will move to the bottom line of your screen where you can type in the command and any parameters.
2. If you omit the exclamation point from the above commands, you may receive an error message such as one saying changes have not been saved, or the output file cannot be written (for example, you are editing a read-only file).
3. The : commands have longer forms (:quit, :write, :edit), but the longer forms are seldom used.

**vi modes**

The vi editor has two modes of operation:

**Command mode**

In command mode, you move around the file and perform editing operations such as searching for text, deleting text, changing text, and so on. You usually start in command mode.

**Insert mode**

In insert mode, you type new text into the file at the insertion point. To return to command mode, press the **Esc** (Escape) key.

These two modes determine the way the editor behaves. Anything you type in insert mode is considered text to be inserted into the file. If you are trying to type a command and nothing happens, or the character appears under the cursor, then you probably forgot to press **Esc** to escape from insert mode.

**Editing text**

Now that you can open a file in vi, move around it and get out, it's time to learn how to edit the text in the file.

Modifying text

Use the following commands when you need to insert, delete, or modify text. Note that some of these commands have an uppercase form that is similar to the lowercase form; see the descriptions below.

**i**

Enter insert mode before the character at the current position. Type your text and press **Esc** to return to command mode. Use I to insert at the beginning of the current line.

**a**

Enter insert mode after the character at the current position. Type your text and press **Esc** to return to command mode. Use A to insert at the end of the current line.

**c**

Use c to change the current character and enter insert mode to type replacement characters.

**o**

Open a new line for text insertion below the current line. Use O to open a line above the current line.

**cw**

Delete the remainder of the current word and enter insert mode to replace it. Use a repetition count to replace multiple words. Use c$ to replace to end of line.

**dw**

Same as for cw (and c$) above, except that insert mode is not entered.

**dd**

Delete the current line. Use a repetition count to delete multiple lines.

**x**

Delete the character at the cursor position. Use a repetition count to delete multiple characters.

**p**

Put the last deleted text after the current character. Use P to put it before the current character.

**xp**

This combination of x and p is a useful idiom. This swaps the character at the cursor position with the one on its right.

**Searching text**

You can search for text in your file using regular expressions:

**/**

Use / followed by a regular expression to search forward in your file.

**?**

Use ? followed by a regular expression to search backward in your file.

**n**

Use n to repeat the last search in either direction.

You may precede any of the above search commands with a number indicating a repetition count. So 3/x will find the third occurrence of x from the current point, as will /x followed by 2n. Similarly, 2/^e will find the second line from the current position that starts with e.

Note that search will wrap around to the top once the bottom of file is reached.

**Getting help**

Another useful command in vi is the help command, which you invoke by typing :help. Help will open inside vi; use the :qcommand to leave help and go back to your work. If you want help on some particular topic, say wrapping of lines, try adding a word after the :help command, for example: :help wrap.

ПОВЕЧЕ ИНФО ЗА ВИАЙ В ФАЙЛА ЗА ЛИНУКС

# 104.1 Create partitions and filesystems

**Displaying MBR partition information using fdisk**

The fdisk command with the -l option is used to list partitions. Add a device name, such as /dev/sda, if you want to look at the partitions on a particular drive. Note that partitioning tools require root access. Listing 2 shows the partitions on the primary hard drives of two of my systems.

Listing 2. Listing MBR partitions using fdisk

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37 | ian@Z61t-u14:~$ sudo fdisk -l /dev/sda # Ubuntu 14    Disk /dev/sda: 500.1 GB, 500107862016 bytes  255 heads, 63 sectors/track, 60801 cylinders, total 976773168 sectors  Units = sectors of 1 \* 512 = 512 bytes  Sector size (logical/physical): 512 bytes / 512 bytes  I/O size (minimum/optimal): 512 bytes / 512 bytes  Disk identifier: 0xed1f86f7       Device Boot      Start         End      Blocks   Id  System  /dev/sda1   \*          63   185915519    92957728+   7  HPFS/NTFS/exFAT  /dev/sda2       185915520   195365519     4725000   12  Compaq diagnostics  /dev/sda3       195366910   976773119   390703105    5  Extended  /dev/sda5       195366912   293021695    48827392   83  Linux  /dev/sda6       293023744   312553471     9764864   82  Linux swap / Solaris  /dev/sda7       312555520   333035519    10240000   83  Linux    [root@atticf20 ~]# fdisk -l /dev/sda # Fedora 22  Disk /dev/sda: 596.2 GiB, 640135028736 bytes, 1250263728 sectors  Units: sectors of 1 \* 512 = 512 bytes  Sector size (logical/physical): 512 bytes / 512 bytes  I/O size (minimum/optimal): 512 bytes / 512 bytes  Disklabel type: dos  Disk identifier: 0x00064a1a    Device     Boot      Start        End    Sectors   Size Id Type  /dev/sda1               63    2040254    2040192 996.2M 83 Linux  /dev/sda2          2040255   22523129   20482875   9.8G 82 Linux swap / Solaris  /dev/sda4         22523191 1250258624 1227735434 585.4G  5 Extended  /dev/sda5         22523193  167397299  144874107  69.1G 83 Linux  /dev/sda6        167397363  310761359  143363997  68.4G 83 Linux  /dev/sda7  \*     310761423  455442749  144681327    69G 83 Linux  /dev/sda8        455442813  600092009  144649197    69G 83 Linux  /dev/sda9        600092073  744436034  144343962  68.8G 83 Linux  /dev/sda10       744436098  918439935  174003838    83G 83 Linux  /dev/sda11       918441984 1079044095  160602112  76.6G 83 Linux  /dev/sda12      1079053983 1250258624  171204642  81.7G 83 Linux |

**Notes:**

1. The header information shows the disk size and geometry. Most large disks using LBA have 255 heads per cylinder and 63 sectors per track, making a total of 16065 sectors, or 8225280 bytes per cylinder.
2. In the second example, a logical partition (/dev/sda7) is marked *bootable* (or *active*). This flag normally enables the standard DOS PC master boot record to boot the partition. However, that loader can only boot primary partitions. This flag has no significance for the LILO or GRUB boot loaders. Both example now use GRUB2 as the boot loader, and the fact that /dev/sda7 is marked bootable is probably an accident of the history of my use of this drive.
3. The Start and End columns show the starting and ending cylinder for each partition. These must not overlap and will generally be contiguous, with no intervening space.
4. The Blocks column shows the number of 1K (1024 byte) blocks in the partition. For most disks in use at the time of writing, the sector size is 512 bytes, so the maximum number of blocks in a partition is therefore half of the product of the number of cylinders (End + 1 - Start) and the number of sectors per cylinder. A trailing + sign indicates that not all sectors in the partition are used. Some disks can now be formatted with larger sectors
5. The Id field indicates the intended use of the partition. Type 82 is a Linux swap partition, and type 83 is a Linux data partition. There are approximately 100 different partition types defined. The second disk is shared between several operating systems, including Windows/XP, hence the presence of Windows NTFS (and possibly FAT32) partitions.

**Displaying GPT partition information using gdisk**

Like the fdisk command for MBR formatted disks, the gdisk with the -l option is used to list partitions for GPT formatted disks. Add a device name, such as /dev/sda, if you want to look at the partitions on a particular drive. As with fdisk, gdiskrequire root access. Listing 3 shows the partition a GPT formatted hard drive of tone of my systems where the partition takes up almost half the available disk.

Listing 3. Listing GPT partitions using gdisk

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | [root@attic-f21 ~]# # Fedora 21  [root@attic-f21 ~]# gdisk -l /dev/sdc  GPT fdisk (gdisk) version 1.0.0    Partition table scan:    MBR: protective    BSD: not present    APM: not present    GPT: present    Found valid GPT with protective MBR; using GPT.  Disk /dev/sdc: 1953525168 sectors, 931.5 GiB  Logical sector size: 512 bytes  Disk identifier (GUID): E5D8C34A-33BF-49EA-8800-AE195E292F1D  Partition table holds up to 128 entries  First usable sector is 34, last usable sector is 1953525134  Partitions will be aligned on 2048-sector boundaries  Total free space is 1004111213 sectors (478.8 GiB)    Number  Start (sector)    End (sector)  Size       Code  Name     1            2048       949415935   452.7 GiB   0700 |

Note the presence of a protective MBR. This enables programs that are unaware of GPT formatting to see the disk as if the whole disk were occupied by a single partition. Current versions of fdisk are aware of GPT formatting and will also display the partitions on a GPT formatted disk.

**Displaying partition information using parted**

The parted command is a partition editor that will work with both MBR and GPT formatted disks. Listing 4 shows the second disk from [Listing 2](https://www.ibm.com/developerworks/library/l-lpic1-104-1/index.html#listing2) and the disk from [Listing 3](https://www.ibm.com/developerworks/library/l-lpic1-104-1/index.html#gdisk-l).

Listing 4. Listing MBR and GPT partitions using parted

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30 | [root@attic-f21 ~]# # Fedora 21  [root@attic-f21 ~]# parted /dev/sda p  Model: ATA WDC WD6401AALS-0 (scsi)  Disk /dev/sda: 640GB  Sector size (logical/physical): 512B/512B  Partition Table   : msdos  Disk Flags:    Number  Start   End     Size    Type      File system     Flags   1      32.3kB  1045MB  1045MB  primary   ext3   2      1045MB  11.5GB  10.5GB  primary   linux-swap(v1)   4      11.5GB  640GB   629GB   extended   5      11.5GB  85.7GB  74.2GB  logical   ext4   6      85.7GB  159GB   73.4GB  logical   ext4   7      159GB   233GB   74.1GB  logical   ext4            boot   8      233GB   307GB   74.1GB  logical   ext3   9      307GB   381GB   73.9GB  logical   ext3  10      381GB   470GB   89.1GB  logical   ext4  11      470GB   552GB   82.2GB  logical   ext3  12      552GB   640GB   87.7GB  logical   ext4    [root@attic-f21 ~]# parted /dev/sdc p  Model: ATA WDC WD1003FZEX-0 (scsi)  Disk /dev/sdc: 1000GB  Sector size (logical/physical): 512B/4096B  Partition Table   : gpt  Disk Flags:    Number  Start   End    Size   File system  Name  Flags   1      1049kB  486GB  486GB  ext4               msftdata |

Note the different physical sector sizes on the two disks.

**Partitioning an GPT disk using gdisk**

We will now look at adding partitions to GPT formatted disks using the gdisk command. We will add a 4GB swap partition and a 40GB data partition. The examples in this section use a CentOS 6 system.

**Warning:** If you use gdisk on an MBR formatted disk, it will offer to convert the disk to GPT format. This might destroy data on your disk, so make sure you have good backups before you try this.

Listing 12 shows gdisk starting using the MBR disk we used above (/dev/sda), and then using a GPT formatted disk (/dev/sdc). We will use /dev/sdc for our two new partitions.

Listing 14 shows how to add our two partitions. Initially, they will both default to Linux data partitions (type 8300). Note the use of +4G and +40G to specify the last sectors of the partitions.

Listing 14. Adding partitions using gdisk

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | Command (? for help): n  Partition number (2-128, default 2):  First sector (34-1953525134, default = 949415936) or {+-}size{KMGTP}:  Last sector (949415936-1953525134, default = 1953525134) or {+-}size{KMGTP}: +4G  Current type is 'Linux filesystem'  Hex code or GUID (L to show codes, Enter = 8300):  Changed type of partition to 'Linux filesystem'    Command (? for help): n  Partition number (3-128, default 3):  First sector (34-1953525134, default = 957804544) or {+-}size{KMGTP}:  Last sector (957804544-1953525134, default = 1953525134) or {+-}size{KMGTP}: +40G  Current type is 'Linux filesystem'  Hex code or GUID (L to show codes, Enter = 8300):  Changed type of partition to 'Linux filesystem'    Command (? for help): |

Now we will change the partition type for /dev/sdc2 to Linux swap. Listing 15 shows the available types and how to change /dev/sdc2. We also show the resulting partition table, including the first partition that was already on the disk.

Listing 15. Changing a partition type using gdisk

Command (? for help): t

Partition number (1-3): 2

Current type is 'Linux filesystem'

Hex code or GUID (L to show codes, Enter = 8300): 8200

Changed type of partition to 'Linux swap'

Command (? for help): p

Disk /dev/sdc: 1953525168 sectors, 931.5 GiB

Logical sector size: 512 bytes

Disk identifier (GUID): E5D8C34A-33BF-49EA-8800-AE195E292F1D

Partition table holds up to 128 entries

First usable sector is 34, last usable sector is 1953525134

Partitions will be aligned on 2048-sector boundaries

Total free space is 911836525 sectors (434.8 GiB)

Number  Start (sector)    End (sector)  Size       Code  Name

   1            2048       949415935   452.7 GiB   0700

   2       949415936       957804543   4.0 GiB     8200  Linux swap

   3       957804544      1041690623   40.0 GiB    8300  Linux filesystem

Command (? for help): i

Notice that /dev/sdc1 has a type of 0700 which is Microsoft basic data. The partition is actually formatted as a Linux ext4 filesystem. Some partition types, such as Microsoft basic data, matter to some operating systems. Others, as in this case, are merely hints to an operating system.

Our final task is to verify the partition table and write it. We show this in Listing 16.

Listing 16. Verifying and writing a partition table using gdisk

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | Command (? for help): v    No problems found. 911836525 free sectors (434.8 GiB) available in 2  segments, the largest of which is 911834511 (434.8 GiB) in size.    Command (? for help): w    Final checks complete. About to write GPT data. THIS WILL OVERWRITE EXISTING  PARTITIONS!!    Do you want to proceed? (Y/N): y  OK; writing new GUID partition table (GPT) to /dev/sdc.  Warning: The kernel is still using the old partition table.  The new table will be used at the next reboot.  The operation has completed successfully. |

The final warning about needing to reboot is because we had /dev/sdc1 mounted. As we said earlier, it's generally better to repartition a disk when it has no mounted filesystems.

**Partitioning using parted**

By now, you will not be surprised to learn that you can also use parted in interactive mode. You can use it for either MBR or GPT formatted disks. One big difference between parted and either fdisk or gdisk, is that parted executes its subcommands immediately and updates the partition table as you go.

Listing 17 shows how to start parted to update /dev/sdc. You can get a list of commands and help on individual commands. We illustrate individual help for the mklabel subcommand. Commands can be abbreviated to the minimum unique abbreviation, so we could have used h mklabel instead of help mklabel. We also print the partition table and you can confirm that /dev/sdc1 is the only partition that is formatted and indeed it is formatted as ext4.

Listing 17. Starting parted to update /dev/sdc

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53 | [root@attic4-cent ~]# parted /dev/sdc  GNU Parted 2.1  Using /dev/sdc  Welcome to GNU Parted! Type 'help' to view a list of commands.  (parted) help    align-check TYPE N                        check partition N for TYPE(min|opt)          alignment    check NUMBER                             do a simple check on the file system    cp [FROM-DEVICE] FROM-NUMBER TO-NUMBER   copy file system to another partition    help [COMMAND]                           print general help, or help on          COMMAND    mklabel,mktable LABEL-TYPE               create a new disklabel (partition          table)    mkfs NUMBER FS-TYPE                      make a FS-TYPE file system on          partition NUMBER    mkpart PART-TYPE [FS-TYPE] START END     make a partition    mkpartfs PART-TYPE FS-TYPE START END     make a partition with a file system    move NUMBER START END                    move partition NUMBER    name NUMBER NAME                         name partition NUMBER as NAME    print [devices|free|list,all|NUMBER]     display the partition table,          available devices, free space, all found partitions, or a particular          partition    quit                                     exit program    rescue START END                         rescue a lost partition near START          and END    resize NUMBER START END                  resize partition NUMBER and its file          system    rm NUMBER                                delete partition NUMBER    select DEVICE                            choose the device to edit    set NUMBER FLAG STATE                    change the FLAG on partition NUMBER    toggle [NUMBER [FLAG]]                   toggle the state of FLAG on partition          NUMBER    unit UNIT                                set the default unit to UNIT    version                                  display the version number and          copyright information of GNU Parted  (parted) help mklabel    mklabel,mktable LABEL-TYPE               create a new disklabel (partition          table)        LABEL-TYPE is one of: aix, amiga, bsd, dvh, gpt, mac, msdos, pc98, sun,          loop  (parted) p  Model: ATA WDC WD1003FZEX-0 (scsi)  Disk /dev/sdc: 1000GB  Sector size (logical/physical): 512B/4096B  Partition Table : gpt    Number  Start   End    Size    File system  Name              Flags   1      1049kB  486GB  486GB   ext4   2      486GB   490GB  4295MB               Linux swap   3      490GB   533GB  42.9GB               Linux filesystem    (parted) |

Note that parted uses decimal powers rather than binary powers for MB, GB, etc, so the 40GB partition that was added bygdisk now appears as 42.9GB using parted.

We will now add two more Linux filesystem partitions using the mkpart subcommand. You could use the mkpartfscommand to both make a partition and format it, but parted recommends using the more robust partition formatting tools that we will meet shortly, so we'll just use mkpart as shown in Listing 18.

Listing 18. Using mkpart to add two Linux filesystem partitions

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | (parted) mkpart  Partition name?  []? "Linux filesystem"  File system type?  [ext2]? xfs  Start? 533GB  End? 573GB  Partition name?  []? "Linux filesystem"  File system type?  [ext2]?  Start? 573GB  End? 613GB  (parted) p  Model: ATA WDC WD1003FZEX-0 (scsi)  Disk /dev/sdc: 1000GB  Sector size (logical/physical): 512B/4096B  Partition Table : gpt    Number  Start   End    Size    File system  Name              Flags   1      1049kB  486GB  486GB   ext4   2      486GB   490GB  4295MB               Linux swap   3      490GB   533GB  42.9GB               Linux filesystem   4      533GB   573GB  39.7GB               Linux filesystem   5      573GB   613GB  40.0GB               Linux filesystem    (parted) |

You may have noticed that parted asked us about the filesystem type but not the partition type code. We specified xfs for one and defaulted to ext2 for the second. In theory, parted uses the intended filesystem type to determine the correct partition type. Listing 19 uses gdisk to show the types that parted assigned to our new partitions.

Listing 19. Using gdisk to show the partition types created by parted

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24 | [root@attic4-cent ~]# gdisk -l /dev/sdc  GPT fdisk (gdisk) version 0.8.10    Partition table scan:    MBR: protective    BSD: not present    APM: not present    GPT: present    Found valid GPT with protective MBR; using GPT.  Disk /dev/sdc: 1953525168 sectors, 931.5 GiB  Logical sector size: 512 bytes  Disk identifier (GUID): E5D8C34A-33BF-49EA-8800-AE195E292F1D  Partition table holds up to 128 entries  First usable sector is 34, last usable sector is 1953525134  Partitions will be aligned on 2048-sector boundaries  Total free space is 756262253 sectors (360.6 GiB)    Number  Start (sector)    End (sector)  Size       Code  Name     1            2048       949415935   452.7 GiB   0700     2       949415936       957804543   4.0 GiB     8200  Linux swap     3       957804544      1041690623   40.0 GiB    8300  Linux filesystem     4      1041690624      1119139839   36.9 GiB    0700  Linux filesystem     5      1119139840      1197264895   37.3 GiB    0700  Linux filesystem |

Probably /dev/sdc1 was created using parted or perhaps gparted.

You have now seen several ways to add partitions to a Linux workstation. We will show you a few other tools to add to your toolbox later in this tutorial. Filesystem considerations that might influence the choices you might make are covered in the tutorial, "[Learn Linux, 101: Find and place system files](http://www.ibm.com/developerworks/library/l-lpic1-104-7/index.html)."

**Creating filesystems**

Linux uses the mkfs command to create filesystems and mkswapcommand to make swap space. The mkfs command is actually a front end to several filesystem-specific commands such as mkfs.ext3 for ext3, mkfs.ext4 for ext4 and mkfs.btrfs for btrfs.

What filesystem support is already installed on your system? Use the ls /sbin/mk\* command to find out. An example is shown in Listing 20.

Listing 20. Filesystem creation commands

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | [[ian@attic4-cent ~]$ ls /sbin/mk\*  /sbin/mkdosfs     /sbin/mkfs.cramfs   /sbin/mkfs.msdos        /sbin/mksquashfs  /sbin/mkdumprd    /sbin/mkfs.ext2     /sbin/mkfs.vfat         /sbin/mkswap  /sbin/mke2fs      /sbin/mkfs.ext3     /sbin/mkfs.xfs  /sbin/mkfs        /sbin/mkfs.ext4     /sbin/mkhomedir\_helper  /sbin/mkfs.btrfs  /sbin/mkfs.ext4dev  /sbin/mkinitrd |

You will notice various forms of some commands. For example, you will usually find that the files mke2fs, mkfs.ext2, and mkfs.ext3 are all the same file, while mkfs.msdos and mkfs.vfat are usually symbolic links to mkdosfs. Filesystems that may be needed to boot the system will usually use hard links to provide the different names for the same file. Filesystems that cannot be used for the / filesystem in Linux, such as vfat or msdos, may use symbolic links instead. The tutorial "[Learn Linux, 101: Create and change hard and symbolic links](http://www.ibm.com/developerworks/library/l-lpic1-v3-104-6/index.html)" will help you learn about these different kinds of links.

There are a few common options for all mkfs commands. Options that are specific to the type of filesystem being created are passed to the appropriate creation command, based on the type of filesystem specified in the -type option. Our examples use mkfs -type, but you may use the other forms directly with equal effect. For example, you may use mkfs -type ext2, mk2fs, or mkfs.ext2. For the manual pages for a specific filesystem, use the appropriate mkfs command as the name, for example, man mkfs.ext3. Many of the values displayed in the output examples below can be controlled by options to mkfs.

Now that we have created all our partitions, we will reboot the CentOS 6 system and format the filesystems using that rather than the somewhat slower live Knoppix DVD. Of course, you could continue to use a live system if you wished. Remember that you need root authority to create filesystems.

Note that a journal is created with ext3. If you wish to add a journal to an existing ext2 system, use the tune2fs command with the -j option.

A useful option for ext2, ext3, and ext4 filesystems is the -L option with a name, which assigns a label to the partition. This can be used instead of the device name when mounting filesystems; it provides some level of insulation against changes that may need to be reflected in various control files. To display or set a label for an existing ext2, ext3, or ext3 filesystem, use the e2label command. Labels are limited to a maximum size of 16 characters.

A more recent development is to use a universally unique identifier, or UUID, rather than a label. A UUID is a 128-bit identifier usually displayed as 32 hexadecimal digits in groups of 8, separated by four hyphens. Most Linux filesystems generate a UUID automatically when the filesystem is formatted. Use the blkid command (which does not need root authority) as shown in Listing 22 to see the UUID for the partition we just formatted. UUIDs are more likely to be unique than labels and are especially useful for hot-plugged devices such as USB drives.

Listing 22. Displaying a UUID using blkid

|  |  |
| --- | --- |
| 1  2 | [ian@attic4-cent ~]$ blkid /dev/sdc3  /dev/sdc3: UUID="05efd400-c689-4205-a53a-e8b009eb5a55" SEC\_TYPE="ext2" TYPE="ext3" |

# 104.2 Maintain the integrity of filesystems

**Filesystem checking tools**

The main tool for checking filesystems is fsck, which, like mkfs, is really a front end to filesystem-checking routines for the various filesystem types. Some of the underlying check routines are shown in Listing 1.

Listing 1. Some of the fsck programs

|  |  |
| --- | --- |
| 1  2  3  4 | [ian@attic4-cent ~]$ ls /sbin/\*fsck\*  /sbin/btrfsck  /sbin/fsck         /sbin/fsck.ext3     /sbin/fsck.msdos  /sbin/dosfsck  /sbin/fsck.cramfs  /sbin/fsck.ext4     /sbin/fsck.vfat  /sbin/e2fsck   /sbin/fsck.ext2    /sbin/fsck.ext4dev  /sbin/fsck.xfs |

You may be surprised to learn that several of these files are hard links to just one file as shown in Listing 2. Remember that these programs may be used so early in the boot process that the filesystem may not be mounted and symbolic link support may not yet be available. See our tutorial [Learn Linux, 101: Create and change hard and symbolic links](http://www.ibm.com/developerworks/library/l-lpic1-104-6/) for more information about hard and symbolic links.

Listing 2. One fsck program with many faces

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | [ian@attic4-cent ~]$ find /sbin -samefile /sbin/e2fsck  /sbin/fsck.ext3  /sbin/fsck.ext4  /sbin/e2fsck  /sbin/fsck.ext4dev  /sbin/fsck.ext2 |

The system boot process uses fsck with the -A option to check the root filesystem and any other filesystems that are specified for checking in the /etc/fstab control file. If the filesystem was not cleanly unmounted, a consistency check is performed and repairs are made, if they can be done safely. This is controlled by the *pass* (or *passno*) field (the sixth field) of the /etc/fstab entry. Filesystems with pass set to 0 are not checked at boot time. The root filesystem has a pass value of 1 and is checked first. Other filesystems will usually have a pass value of 2 (or higher), indicating the order in which they should be checked.

Multiple fsck operations can run in parallel if the system determines it is advantageous, so different filesystems are allowed to have the same pass value, as is the case for the /grubfile and /home/ian/data filesystems shown in Listing 3. Note that fsck will avoid running multiple filesystem checks on the same physical disk. To learn more about the layout of /etc/fstab, check the man pages for fstab.

You can check filesystems after the system is booted. You will need root authority, and the filesystem you want to check should be unmounted first. Listing 4 shows how to check two of our filesystems, using the device name, label, or UUID. You can use the blkid command to find the device given a label or UUID, and the label and UUID, given the device.

Listing 4. Using fsck to check filesystems

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | [root@attic4-cent ~]# # Find label and UUID for /dev/sdc4  [root@attic4-cent ~]# blkid /dev/sdc4  /dev/sdc4: LABEL="IAN-GPT-EXT4" UUID="f69e0b28-beda-4255-ad5a-4d73672ac9e4" TYPE="ext4"  [root@attic4-cent ~]# # Check /dev/sdc4  [root@attic4-cent ~]# fsck /dev/sdc4  fsck from util-linux-ng 2.17.2  e2fsck 1.41.12 (17-May-2010)  IAN-GPT-EXT4: clean, 11/2424832 files, 197218/9681152 blocks  [root@attic4-cent ~]# # Check it by label using fsck.ext4  [root@attic4-cent ~]# fsck.ext4 LABEL=IAN-GPT-EXT4  e2fsck 1.41.12 (17-May-2010)  IAN-GPT-EXT4: clean, 11/2424832 files, 197218/9681152 blocks  [root@attic4-cent ~]# # Check it by UUID using e2fsck  [root@attic4-cent ~]# e2fsck UUID=f69e0b28-beda-4255-ad5a-4d73672ac9e4  e2fsck 1.41.12 (17-May-2010)  IAN-GPT-EXT4: clean, 11/2424832 files, 197218/9681152 blocks  [root@attic4-cent ~]# # Finally check the small vfat partition /dev/sda3  [root@attic4-cent ~]# fsck /dev/sda3  fsck from util-linux-ng 2.17.2  dosfsck 3.0.9, 31 Jan 2010, FAT32, LFN  /dev/sda3: 0 files, 0/1265 clusters |

If you attempt to check a mounted filesystem, you may see a warning. More recent versions of fsck will abort the check as Listing 5 shows where we try to check our root filesystem. If the check does not abort, heed the warning and do not do it!

Listing 5. Do not attempt to check a mounted filesystem

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | [root@attic4-cent ~]# df /  Filesystem     1K-blocks    Used Available Use% Mounted on  /dev/sda11      79040416 7444796  67580568  10% /  [root@attic4-cent ~]# fsck /dev/sda11  fsck from util-linux-ng 2.17.2  e2fsck 1.41.12 (17-May-2010)  /dev/sda11 is mounted.  e2fsck: Cannot continue, aborting. |

It is also a good idea to let fsck figure out which check to run on a filesystem. Running the wrong check can corrupt the filesystem. If you want to see what fsck would do for a given filesystem or set of filesystems, use the -N option as

shown in Listing 6.

Listing 6. Finding what fsck would do to check /dev/sda7, /dev/sda8, and /dev/sda9

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | [root@attic4-cent ~]# fsck -N /dev/sda11 /dev/sdb\*  fsck from util-linux-ng 2.17.2  [/sbin/fsck.ext3 (1) -- /] fsck.ext3 /dev/sda11  [/sbin/fsck.ext2 (2) -- /dev/sdb] fsck.ext2 /dev/sdb  [/sbin/fsck.ext3 (3) -- /dev/sdb1] fsck.ext3 /dev/sdb1  [/sbin/fsck.ext4 (4) -- /home/ian/data] fsck.ext4 /dev/sdb2  [/sbin/fsck.ext3 (5) -- /home/ian/research] fsck.ext3 /dev/sdb3  [/sbin/fsck.ext4 (6) -- /dev/sdb4] fsck.ext4 /dev/sdb4 |

So far, we have checked ext and vfat filesystems. Let's now check the XFS filesystem on /dev/sdb3. As you can see in Listing 7, the fsck command simply tells us that we should use the xfs\_check command. If there are no errors, then xfs\_checkdoes not display any output. There is a -v option for verbose output, but it is much too verbose for a simple check.

Listing 7. Using fsck with XFS

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | [root@attic4-cent ~]# fsck -N /dev/sdc3  fsck from util-linux-ng 2.17.2  [/sbin/fsck.xfs (1) -- /dev/sdc3] fsck.xfs /dev/sdc3  [root@attic4-cent ~]# fsck /dev/sdc3  fsck from util-linux-ng 2.17.2  If you wish to check the consistency of an XFS filesystem or  repair a damaged filesystem, see xfs\_check(8) and xfs\_repair(8).  [root@attic4-cent ~]# xfs\_check /dev/sdc3 |

**Monitoring free space**

On a storage device, a file or directory is contained in a collection of *blocks*. Information about a file is contained in an *inode*, which records information such as who the owner is, when the file was last accessed, how large it is, whether it is a directory, and who can read from or write to it. The inode number is also known as the file serial number and is unique within a particular filesystem. See our tutorial [Learn Linux, 101: File and directory management](http://www.ibm.com/developerworks/library/l-lpic1-v3-103-3/) for more information on files and directories.

Data blocks and inodes each take space on a filesystem, so you need to monitor the space usage to ensure that your filesystems have space for growth.

**The df command**

The df command displays information about mounted filesystems. If you add the -T option, the filesystem type is included in the display; otherwise, it is not. The output from df for the CentOS 6 system that we used above is shown in Listing 8. For additional interest, we have created a mount point at /mnt/btrfs-test and mounted our btrfs filesystem (/dev/sdc5) over it. We have done the same for our small vfat partition on /dev/sda3.

Listing 8. Displaying filesystem usage

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | [root@attic4-cent ~]# mkdir /mnt/btrfs-test  [root@attic4-cent ~]# mount /dev/sdc5 /mnt/btrfs-test  [[root@attic4-cent ~]# mkdir /mnt/vfat-test  [root@attic4-cent ~]# mount /dev/sda3 /mnt/vfat-test  root@attic4-cent ~]# df -T  Filesystem     Type  1K-blocks      Used Available Use% Mounted on  /dev/sda11     ext3   79040416   7444800  67580564  10% /  tmpfs          tmpfs   1961548       232   1961316   1% /dev/shm  /dev/sda1      ext3     988420    116067    821349  13% /grubfile  /dev/sdb2      ext4  124077136  49155856  68611636  42% /home/ian/data  /dev/sdb3      ext3   60458064  30808664  26578276  54% /home/ian/research  /dev/sdc1      ext4  467126880 134497524 308894012  31% /home/ian/pictures  /dev/sda5      ext4   71168700  31178752  36368096  47% /mnt/sda5  /dev/sdc5      btrfs  39062528        56  36936704   1% /mnt/btrfs-test  /dev/sda3      vfat       2530         0      2530   0% /mnt/vfat-test |

Notice that the output includes the total number of blocks as well as the number used and free (available). Also notice the filesystem, such as ext3 for our root filesystem on /dev/sda11, and its mount point: /. The tmpfs entry is for a virtual memory filesystem. These exist only in RAM or swap space and are created when mounted without need for a mkfscommand.

For specific information on inode usage, use the -i option on the df command. You can exclude certain filesystem types using the -x option, or restrict information to just certain filesystem types using the -t option. Use these multiple times if necessary. See the examples in Listing 9.

Listing 9. Displaying inode usage

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | [root@attic4-cent ~]# df -i -x tmpfs  Filesystem       Inodes  IUsed    IFree IUse% Mounted on  /dev/sda11      5021696 236498  4785198    5% /  /dev/sda1        251000    537   250463    1% /grubfile  /dev/sdb2       7904304 125115  7779189    2% /home/ian/data  /dev/sdb3       3842048  40341  3801707    2% /home/ian/research  /dev/sdc1      29671424  48944 29622480    1% /home/ian/pictures  /dev/sda5       4530176 245603  4284573    6% /mnt/sda5  /dev/sdc5             0      0        0     - /mnt/btrfs-test  /dev/sda3             0      0        0     - /mnt/vfat-test  [root@attic4-cent ~]# df -iT -t ext4 -t vfat  Filesystem     Type   Inodes  IUsed    IFree IUse% Mounted on  /dev/sdb2      ext4  7904304 125115  7779189    2% /home/ian/data  /dev/sdc1      ext4 29671424  48944 29622480    1% /home/ian/pictures  /dev/sda5      ext4  4530176 245603  4284573    6% /mnt/sda5  /dev/sda3      vfat        0      0        0     - /mnt/vfat-test |

You may not be surprised to see that the FAT32 filesystem does not have inodes. You may be more surprised to see no inode information for the btrfs filesystem. Btrfs (and also ReiserFS) keep inode information in dynamically allocated structures so there are no special inode blocks as there are for ext2, ext3 or ext4.

There are several other options you may use with df to limit the display to local filesystems or control the format of output. For example, use the -h option to display human readable sizes, such as 1K for 1024, or use the -H (or --si) option to get sizes in powers of 10 (1K=1000).

**The tune2fs command**

The ext family of filesystems also has a utility called tune2fs, which can be used to inspect information about the block count as well as information about whether the filesystem is journaled (ext3 or ext4) or not (ext2). The command can also be used to set many parameters or convert an ext2 filesystem to ext3 by adding a journal. Listing 11 shows the output for a newly created ext4 filesystem using the -l option to simply display the existing information.

**The du command**

The df command gives information about a whole filesystem. Sometimes you might want to know how much space is used by your home directory, or how big a partition to use if you wanted to move /usr to its own filesystem. To answer this kind of question, use the du command.

The du command displays information about the filename (or filenames) given as parameters. If a directory name is given, then du recurses and calculates sizes for every file and subdirectory of the given directory. The result can be a lot of output. Fortunately, you can use the -s option to request just a summary for a directory. If you use du to get information for multiple directories, then you can add the -c option to get a grand total. You can also control output format using the same set of size options (-h, -H, --si, and so on) that are used for df. Listing 13 shows two views of the home directory of a newly created user who has logged in once and created an index.html file.

Listing 13. Using du

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | [testuser@attic4-cent ~]$ du -hc \*  4.0K    Desktop  4.0K    Documents  4.0K    Downloads  16K index.html  4.0K    Music  4.0K    Pictures  4.0K    Public  4.0K    Templates  4.0K    Videos  48K total  [testuser@attic4-cent ~]$ du -hs .  980K    . |

The reason for the difference between the 48K total from du -c \* and the 980K summary from du -s is that the latter includes the entries starting with a dot, such as .bashrc, while the former does not.

One other thing to note about du is that you must be able to read the directories that you are running it against.

So now, let's use du to display the total space used by the /usr tree and each of its first-level subdirectories. The result is shown in Listing 14. Use root authority to make sure you have appropriate access permissions.

Listing 14. Using du on /usr

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | [root@attic4-cent ~]# du -shc /usr/\*  257M    /usr/bin  4.0K    /usr/etc  4.0K    /usr/games  132M    /usr/include  383M    /usr/lib  1.5G    /usr/lib64  62M     /usr/libexec  136K    /usr/local  67M     /usr/sbin  2.4G    /usr/share  98M     /usr/src  0       /usr/tmp  4.8G    total |

**Repairing filesystems**

Occasionally, very occasionally we hope, the worst will happen and you will need to repair a filesystem because of a crash or other failure to unmount cleanly. The fsck command that you saw above can repair filesystems as well as check them. Usually the automatic boot-time check will fix the problems and you can proceed.

If the automatic boot-time check of filesystems is unable to restore consistency, you are usually dumped into a single user shell with some instructions to run fsck manually. For an ext2 filesystem, which is not journaled, you may be presented with a series of requests asking you to confirm proposed actions to fix particular blocks on the filesystem. You should generally allow fsck to attempt to fix problems, by responding y (for yes). When the system reboots, check for any missing data or files.

If you suspect corruption, or want to run a check manually, most of the checking programs require the filesystem to be unmounted, or at least mounted read-only. Because you can't unmount the root filesystem on a running system, the best you can do is drop to single user mode (using telinit 1) and then remount the root filesystem read-only, at which time you should be able to perform a consistency check. A better way to check a filesystem is to boot a recovery system, such as a live CD or a USB memory key, and perform the check of your unmounted filesystems from that.

If fsck cannot fix the problem, you do have some other tools available, although you will generally need advanced knowledge of the filesystem layout to successfully fix it.

**Why journal?**

An fsck scan of an ext2 disk can take quite a while to complete, because the internal data structure (or *metadata*) of the filesystem must be scanned completely. As filesystems get larger and larger, this takes longer and longer, even though disks also keep getting faster, so a full check may take one or more hours.

This problem was the impetus for *journaled*, or *journaling*, filesystems. Journaled filesystems keep a log of recent changes to the filesystem metadata. After a crash, the filesystem driver inspects the log in order to determine which recently changed parts of the filesystem may possibly have errors. With this design change, checking a journaled filesystem for consistency typically takes just a matter of seconds, regardless of filesystem size. Furthermore, the filesystem driver will usually check the filesystem on mounting, so an external fsck check is generally not required. In fact, for the xfs filesystem, fsck does nothing!

If you do run a manual check of a filesystem, check the man pages for the appropriate fsck command (fsck.ext3, e2fsck , xfs\_check, and so on) to determine the appropriate parameters. The -p option, when used with ext2, ext3, or ext4 filesystems will cause fsck to automatically fix all problems that can be safely fixed. This is, in fact, what happens at boot time.

We'll illustrate the use of e2fsck and xfs\_check by first running e2fsck on an empty XFS filesystem and then using xfs\_check to fix it. Remember we suggested that you use the fsck front end to be sure you are using the right checker, and we warned you that failure to do so may result in filesystem corruption.

In Listing 15, we start running e2fsck against /dev/sda8, which contains an XFS filesystem. After a few interactions we use ctrl-Break to break out, but it is too late. **Warning**: Do **NOT** do this unless you are willing to destroy your filesystem.

Listing 15. Deliberately running e2fsck manually on an XFS filesystem

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | [root@attic4-cent ~]# xfs\_check -s /dev/sdc3  [root@attic4-cent ~]# e2fsck /dev/sdc3  e2fsck 1.41.12 (17-May-2010)  e2fsck: Group descriptors look bad... trying backup blocks...  /dev/sdc3 was not cleanly unmounted, check forced.  Resize inode not valid.  Recreate<y>? yes    Pass 1: Checking inodes, blocks, and sizes  Deleted inode 163841 has zero dtime.  Fix<y>? ctrl-Break    /dev/sdc3: e2fsck canceled.    /dev/sdc3: \*\*\*\*\* FILE SYSTEM WAS MODIFIED \*\*\*\*\* |

**Superblocks**

You may be wondering how all these checking and repairing tools know where to start. Linux and UNIX filesystems usually have a *superblock*, which describes the filesystem *metadata*, or data describing the filesystem itself. This is usually stored at a known location, frequently at or near the beginning of the filesystem, and replicated at other well-known locations. You can use the -n option of mke2fs to display the superblock locations for an existing filesystem. If you specified parameters such as the bytes per inode ratio, you should invoke mke2fs with the same parameters when you use the -n option. Listing 18 shows the location of the superblocks on /dev/sda5. Note that the filesystem must not be mounted when you do this.

Listing 18. Finding superblock locations

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | [root@attic4-cent ~]# mke2fs -n /dev/sda5  mke2fs 1.41.12 (17-May-2010)  Filesystem label=  OS type: Linux  Block size=4096 (log=2)  Fragment size=4096 (log=2)  Stride=0 blocks, Stripe width=0 blocks  4530176 inodes, 18109263 blocks  905463 blocks (5.00%) reserved for the super user  First data block=0  Maximum filesystem blocks=4294967296  553 block groups  32768 blocks per group, 32768 fragments per group  8192 inodes per group  Superblock backups stored on blocks:      32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,      4096000, 7962624, 11239424 |

**Tools for ext2 and ext3 filesystems**

**tune2fs**

Adjusts parameters on ext2 and ext3 filesystems. Use this to add a journal to an ext2 system, making it an ext3 system, as well as display or set the maximum number of mounts before a check is forced. You can also assign a label and set or disable various optional features.

**dumpe2fs**

Prints the super block and block group descriptor information for an ext2 or ext3 filesystem.

**debugfs**

Is an interactive filesystem debugger. Use it to examine or change the state of an ext2 or ext3 filesystem.

# 104.3 Control mounting and unmounting of filesystems

When you mount a filesystem over an existing directory, the files on the filesystem you are mounting become the files and subdirectories of the mount point. If the mount point directory already contained files or subdirectories, they are not lost, but are no longer visible until the mounted filesystem is unmounted, at which point they become visible again. It is a good idea to avoid this problem by using only empty directories as mount points.

After mounting a filesystem, any files or directories created or copied to the mount point or any directory below it will be created on the mounted filesystem. So a file such as /dos/sampdir/file.txt will be created on the FAT32 filesystem that we mounted at /dos in our example.

Usually, the mount command will automatically detect the type of filesystem being mounted. Occasionally you may need to specify the filesystem type explicitly using the -t option as shown in Listing 2.

Listing 2. Mounting with explicit filesystem type

|  |  |
| --- | --- |
| 1 | [root@attic4-cent ~]# mount -t vfat /dev/sda3 /dos |

To see what filesystems are mounted, use the mount command with no parameters. Listing 3 shows our example system. Note that you do not need root authority to simply list mounted filesystems.

Listing 3. Displaying mounted filesystems

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | [ian@attic4-cent ~]$ mount  /dev/sda11 on / type ext3 (rw)  proc on /proc type proc (rw)  sysfs on /sys type sysfs (rw)  devpts on /dev/pts type devpts (rw,gid=5,mode=620)  tmpfs on /dev/shm type tmpfs (rw,rootcontext="system\_u:object\_r:tmpfs\_t:s0")  /dev/sda1 on /grubfile type ext3 (rw)  /dev/sdb2 on /home/ian/data type ext4 (rw)  /dev/sdb3 on /home/ian/research type ext3 (rw)  /dev/sdc1 on /home/ian/pictures type ext4 (rw)  none on /proc/sys/fs/binfmt\_misc type binfmt\_misc (rw)  /dev/sda3 on /dos type vfat (ro)  /dev/sr0 on /media/KNOPPIX type iso9660 (ro,nosuid,nodev,uhelper=udisks,uid=1000,  gid=1000,iocharset=utf8,mode=0400,dmode=0500) |

You can also view similar information by displaying /proc/mounts or /etc/mtab, both of which contain information about mounted filesystems.

**Mount options**

The mount command has several options that override the default behavior. For example, you can mount a filesystem read-only by specifying -o ro. If the filesystem is already mounted, add remount as shown in Listing 4.

Listing 4. Remounting read-only

|  |  |
| --- | --- |
| 1 | [root@attic4-cent ~]# mount -o remount,ro /dos |

**Notes:**

* Use commas to separate multiple options, such as remount and ro.
* When remounting an already mounted filesystem, it suffices to specify either the mount point or the device name. It is not necessary to specify both.
* You cannot mount a read-only filesystem as read-write. Media that cannot be modified, such as CD-ROM discs, will automatically be mounted read-only.
* To remount a writable device read-write, specify -o remount,rw

Remount commands will not complete successfully if any process has open files or directories in the filesystem you are remounting. See [Unmounting filesystems](https://www.ibm.com/developerworks/library/l-lpic1-104-3/index.html#umount) below for additional information.

**Labels, UUIDs, and links**

In UNIX and early Linux systems, the /dev directory usually contained entries for all the devices that might ever be attached to a system. Any device that was used was always located in the same place in the /dev tree, so using names such as /dev/sda6 was natural. With the advent of hot-plugging of devices such as USB or Firewire (IEEE 1394) attached devices, a given device might appear in one USB port today, and that same device might be plugged into a different USB port tomorrow. In this environment, you might want to always mount your USB stick at /media/myusbstick, regardless of which USB port you plug it in to. In the tutorial for topic 102, "[Learn Linux, 101: Boot managers](http://www.ibm.com/developerworks/library/l-lpic1-102-2/)," you learned about using labels and UUIDs (Universally Unique IDs) instead of device names to identify partitions. If the filesystem on the partition supports either, you can use these with the mount command too. Use the blkid command to find out the UUID and label (if present) associated with a device.

Listing 5 shows how to use blkid to find the label and UUID for our root partition and then how to create two additional mount points and mount the root partition at these two additional points. This example is for illustration. You would not normally do this in a production environment.

Listing 5. Mount using labels or UUIDs

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | [root@attic4-cent ~]# blkid /dev/sda11  /dev/sda11: UUID="2f60a3b4-ef6c-4d4c-9ef4-50d7f75124a2" TYPE="ext3" LABEL="CentOS 6"  [root@attic4-cent ~]# mkdir /mnt/sda11label  [root@attic4-cent ~]# mkdir /mnt/sda11uuid  [root@attic4-cent ~]# mount LABEL="CentOS 6" /mnt/sda11label  [root@attic4-cent ~]# mount UUID="2f60a3b4-ef6c-4d4c-9ef4-50d7f75124a2" /mnt/sda11uuid |

With the advent of udev, you will find additional symbolic links in the /dev directory for devices such as hard drives. Listing 6 shows the links to /dev/sda6 on my CentOS 6 system.

Listing 6. Symbolic links to /dev/sda6

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | [root@attic4-cent ~]# find /dev -lname "\*sda11"/dev/root  /dev/disk/by-label/CentOS\x206  /dev/disk/by-uuid/2f60a3b4-ef6c-4d4c-9ef4-50d7f75124a2  /dev/disk/by-id/wwn-0x50014ee056628af6-part11  /dev/disk/by-id/scsi-SATA\_WDC\_WD6401AALS-\_WD-WMASY6347052-part11  /dev/disk/by-id/ata-WDC\_WD6401AALS-00L3B2\_WD-WMASY6347052-part11  /dev/disk/by-path/pci-0000:00:11.0-scsi-0:0:0:0-part11  /dev/block/8:11  /dev/.udev/watch/113  /dev/.udev/links/disk\x2fby-label\x2fCentOS\x5cx206/b8:11  /dev/.udev/links/root/b8:11  /dev/.udev/links/disk\x2fby-id\x2fwwn-0x50014ee056628af6-part11/b8:11  /dev/.udev/links/disk\x2fby-uuid\x2f2f60a3b4-ef6c-4d4c-9ef4-50d7f75124a2/b8:11  /dev/.udev/links/disk\x2fby-path\x2fpci-0000:00:11.0-scsi-0:0:0:0-part11/b8:11  /dev/.udev/links/disk\x2fby-id\x2fscsi-SATA\_WDC\_WD6401AALS-\_WD-WMASY6347052-part11/b8:11  /dev/.udev/links/disk\x2fby-id\x2fata-WDC\_WD6401AALS-00L3B2\_WD-WMASY6347052-part11/b8:11 |

**Boot time and fstab**

In the tutorial for topic 102, "[Learn Linux, 101: Boot managers](http://www.ibm.com/developerworks/library/l-lpic1-102-2/)," you learned how to use the root= parameter in both GRUB and LILO to tell the boot loader what filesystem should be mounted as root. For GRUB2, this is the set root statement. Once the root filesystem is mounted, the initialization process runs mount with the -a option to automatically mount a set of filesystems. The set is specified in the file /etc/fstab.

Listing 7 shows /etc/fstab for a sample CentOS 6 system. In this example, most hard drive partitions are identified by UUID. I have added examples of how to mount /dev/sda3 at /dos as we did earlier, and also how to use a label to mount a labeled partition at /mnt/fedora22.

Listing 7. Example CentOS 6 fstab

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | [root@attic4-cent ~]# cat /etc/fstab    #  # /etc/fstab  # Created by anaconda on Wed Jul  8 09:34:46 2015  #  # Accessible filesystems, by reference, are maintained under '/dev/disk'  # See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info  #  UUID=2f60a3b4-ef6c-4d4c-9ef4-50d7f75124a2 /                   ext3    defaults        1 1  UUID=3c3de27e-779a-44d5-ad7a-61c5fd03d9e7 /grubfile           ext3    defaults        1 2  UUID=158d605e-2591-4749-bf59-5e92e1b1c01d swap                swap    defaults        0 0  tmpfs                                     /dev/shm            tmpfs   defaults        0 0  devpts                                    /dev/pts            devpts  gid=5,mode=620  0 0  sysfs                                     /sys                sysfs   defaults        0 0  proc                                      /proc               proc    defaults        0 0  UUID=4c962b67-c646-467f-96fb-cbbd6de40140 /home/ian/data      ext4    defaults        1 2  UUID=0998d33c-3398-463d-b0e3-7c13ca0c675f /home/ian/research  ext3    defaults        1 2  UUID=e3be4658-b79b-470d-82fe-bb434bcdcc2f /home/ian/pictures  ext4    defaults        1 2  LABEL="FEDORA22"                          /mnt/fedora22       ext4    defaults        1 2  /dev/sda3                                 /dos                vfat    defaults        0 0 |

Lines starting with a # character are comments. Remaining lines contain six fields. Because the fields are positional, they must all be specified.

**file system**

This may be a device name such as /dev/sda1, or a label (LABEL=) or UUID (UUID=). For the root filesystem of our CentOS 6 example, it could be /dev/sda11, LABEL="CentOS 6", or UUID=2f60a3b4-ef6c-4d4c-9ef4-50d7f75124a2. Using a label or UUID makes your system more robust when devices are added or removed.

**mount point**

This is the mount point we discussed in [Mounting filesystems](https://www.ibm.com/developerworks/library/l-lpic1-104-3/index.html#4-mount) above. For swap space, this should be the value 'none' or 'swap'. On older systems you will usually find the value 'none'.

**type**

Specifies the type of filesystem. CD/DVD drives will often support either ISO9660 or UDF filesystems, so you may specify multiple possibilities in a comma-separated list if you specify such a drive in /etc/fstab. If you want mount to automatically determine the type, specify auto. For example, you might see lines like the following on some older systems for a CD or DVD and a floppy disk.

|  |  |
| --- | --- |
| 1  2 | /dev/scd0       /media/cdrom0   udf,iso9660 user,noauto,exec,utf8 0       0  /dev/fd0        /media/floppy0  auto    rw,user,noauto,exec,utf8  0       0 |

**option**

Specifies the mount options. Specify defaults if you want default mount options. Some options you will want to know about are:

* rw and ro specify whether the filesystem should be mounted read-write or read-only.
* noauto specifies that this filesystem should not be automatically mounted at boot time or whenever mount -a is issued. In our example, this is done for the removable drives.
* user specifies that a non-root user is permitted to mount and unmount the filesystem. This is especially useful for removable media. In older systems, this option is specified in /etc/fstab rather than on the mount command. With newer systems, it may be specified in udev rules that are located in rules files within /lib/udev/rules.d or /etc/udev/rules.d. The options for the DVD drive on my CentOS 6 system come from udev rules, and that is why there is no entry in /etc/fstab for an optical drive.
* exec and noexec specify whether or not to allow execution of files from the mounted filesystem. User-mounted filesystems default to noexec unless exec is specified afteruser.
* noatime will disable recording of access times. Not using access times may improve performance.

**dump**

Specifies whether the dump command should consider this ext2 or ext3 filesystem for backups. A value of 0 tells dump to ignore this filesystem.

**pass**

Non-zero values of pass specify the order of checking filesystems at boot time, as discussed in our tutorial "[Learn Linux, 101: Maintain the integrity of filesystems](http://www.ibm.com/developerworks/library/l-lpic1-104-2/)."

When you mount a filesystems that is listed in /etc/fstab, you can give either the device name or the mount point when mounting the filesystem. You do not need to give both.

On some systems, for example SUSE 11.2, you may find that the fstab generated at install time uses symbolic links to the device. So, you may see /dev/disk/by-id/ata-WDC\_WD1001FALS-00J7B1\_WD-WMATV3772868-part6, rather than /dev/sda6 for the file system value. Refer back to [Listing 6](https://www.ibm.com/developerworks/library/l-lpic1-104-3/index.html#listing7) for additional possibilities

Consult the man pages for fstab, mount, and udev for additional information, including options not covered here.

**Unmounting filesystems**

All mounted filesystems are usually unmounted automatically by the system when it is rebooted or shut down. When a filesystem is unmounted, any cached filesystem data in memory is flushed to the device.

You may also unmount filesystems manually. Indeed, you **should** do this when removing writable media such as diskettes or USB drives or memory keys.

Use the unmount command to unmount the filesystem, specifying *either* the device name or mount point as an argument. This listing shows how to unmount /dos, then remount it and unmount again using the device name.

Listing 8. Unmounting filesystems

|  |  |
| --- | --- |
| 1  2  3 | [root@attic4-cent ~]# umount /dos  [root@attic4-cent ~]# mount /dev/sda3 /dos  [root@attic4-cent ~]# umount /dev/sda3 |

After a filesystem is unmounted, any files in the directory used for the mount point are visible again.

If you attempt to unmount a filesystem while a process has open files on that filesystem, you will see an error message. Before unmounting a filesystem, you should check that there are no processes running that have open files on the filesystem. Use the lsof or fusercommand to determine what files are open or what process has open files. You may need the -w option on lsof to avoid warning messages related to the Gnome Virtual File system (gvfs). Check the man pages to learn about additional mount options and lsof. If you are checking a whole device, you can specify the device name or the mount point. You may also check whether an individual file is in use or not.

To illustrate these commands, I created a copy of /etc/fstab on /dos and a small script to read lines from stdin and print them to stdout with a 10 second pause between each line. Listing 9 shows the error message from umount when files are in use and the use of lsof and fuser to check for open files on /dos, or the underlying device /dev/sda9.

Listing 9. Checking for open files

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | [root@attic4-cent ~]# umount /dos  umount: /dos: device is busy.          (In some cases useful info about processes that use           the device is found by lsof(8) or fuser(1))  [root@attic4-cent ~]# lsof -w /dos  COMMAND     PID USER   FD   TYPE DEVICE SIZE/OFF NODE NAME  slowread. 28960  ian    0r   REG    8,3     1207    2 /dos/fstab  sleep     28972  ian    0r   REG    8,3     1207    2 /dos/fstab  [root@attic4-cent ~]# lsof -w /dev/sda3  COMMAND     PID USER   FD   TYPE DEVICE SIZE/OFF NODE NAME  slowread. 28960  ian    0r   REG    8,3     1207    2 /dos/fstab  sleep     28978  ian    0r   REG    8,3     1207    2 /dos/fstab  [root@attic4-cent ~]# lsof -w /dos/fstab  COMMAND     PID USER   FD   TYPE DEVICE SIZE/OFF NODE NAME  slowread. 28960  ian    0r   REG    8,3     1207    2 /dos/fstab  sleep     28989  ian    0r   REG    8,3     1207    2 /dos/fstab  [root@attic4-cent ~]# fuser -m /dos  /dos:                28960 29001  [root@attic4-cent ~]# fuser -m /dev/sda3  /dev/sda3:           28960 29001 |

At this point you can either wait until the filesystem is no longer busy, or you can do a *lazy unmount*, by specifying the -loption. A lazy unmount detaches the filesystem from the filesystem tree immediately, and cleans the references to the filesystem when it is no longer busy.

**Removable filesystems**

We mentioned some issues with removable devices such as USB or Firewire (IEEE 1394) attached devices. It is inconvenient to switch to root access every time you need to mount or unmount such a device. The same goes for CD, DVD, and floppy drives, where you need to unmount the device to change media. In the discussion of fstab above, we mentioned the user option, which allows ordinary users to mount and unmount devices.

Note that the filesystem types for the optical drive are specified as udf,iso9660, while the filesystem type for the floppy is specified as auto. For the optical drive, the mount process will check first for a udf filesystem (common on DVD) and then for an iso9660 filesystem (common on CD). For the floppy drive, the mount process will probe for a filesystem type. You can create or edit /etc/filesystems to change the order in which the filesystems will be probed.

**Note:** You should **always** unmount removable drives or media before disconnecting the drive or attempting to remove the media. Failure to do so may result in loss of data that has not yet been written to the device.

If you run a graphical desktop such as Nautilus, you will usually find options that allow removable devices and media to be automatically mounted. For example, if I insert a Knoppix DVD into the DVD drive of my system, I might see a mount entry such as shown in Listing 10. The presence of 'uid=1000' shows that the user with id 1000 can unmount this disc. The idcommand shows the uid for user ian is 1000, so ian can unmount this disc.

Listing 10. Desktop mounting of DVD

|  |  |
| --- | --- |
| 1  2  3  4  5 | [ian@attic4-cent ~]$ mount | grep sr0  /dev/sr0 on /media/KNOPPIX type iso9660 (ro,nosuid,nodev,uhelper=udisks,uid=1000,  gid=1000,iocharset=utf8,mode=0400,dmode=0500)  [ian@attic4-cent ~]$ id ian  uid=1000(ian) gid=1000(ian) groups=1000(ian) |

You may also use the eject command to eject removable media when the drive supports the operation as most CD and DVD drives do. If you have not unmounted the device first, then eject will both unmount and eject the disc.

# 104.4 Manage disk quotas

**Enabling quotas**

There are three different types of quota support:

1. Version 1 quota format used up to kernel 2.2
2. Version 2 quota format used on 2.4 and later kernels.
3. xfs, the quota on XFS filesystems

Quotas require kernel support that was introduced in later 2.4 kernels. 2.6 kernels have the support you need.

XFS quotas are always journaled. Journaled version 2 quotas are supported on kernel 2.6.11 and above.

Version 1 quota support is sometimes referred to as *vfsold*, while Version 2 quota support is sometimes referred to as *vfsv0*(32-bit UIDs / GIDs, 64-bit space usage, 32-bit inode usage and limits) or *vfsv1* (64-bit quota limits and usage).

Adding quota support to /etc/fstab

The next step to enable quotas is to add the appropriate options to the filesystem definitions in /etc/fstab, according to whether you want to implement user quotas, group quotas, or both. XFS filesystems also support project quotas.

At the time of writing, the options for enabling quotas are not centrally documented in an obvious man page. Table 1 shows the options that are available and the type of quota system they are used for.

Table 1. Quota options in /etc/fstab

| **Option** | | | **Applies to** | **Use** | |
| --- | --- | --- | --- | --- | --- |
| usrquota | | | All types | Enable user quotas | |
| usrjquota=*filename* | | | vfsv0, vfsv1 (Version 2) | Enable journaled user quotas; requires a quota database file name (usually aquota.user) and specification of jqfmt option | |
| uquota | | | xfs | Equivalent to usrquota | |
| grpquota | | | All types | Enable group quotas | |
| grpjquota=*filename* | | | vfsv0, vfsv1 (Version 2) | Enable journaled group quotas; requires a quota database file name (usually aquota.group) and specification of jqfmt option | |
| quota | vfsold (Version 1), vfsv0, vfsv1 (Version 2) | | | Equivalent to usrquota | |
| noquota | vfsold (Version 1), vfsv0, vfsv1 (Version 2) | | | Do not enable quotas | |

We use an ext4 partition and an XFS partition for demonstration purposes. We'll add user and group quota to these filesystems so you can see how quotas work on two different filesystems. Our /etc/fstab entries are shown in Listing 5.

Listing 5. Enabling quota support in /etc/fstab

|  |  |
| --- | --- |
| 1  2 | /dev/sdc6  /quotatest/ext4  ext4    defaults,usrquota,grpquota          1 2  /dev/sdc3  /quotatest/xfs   xfs     defaults,usrquota,grpquota          1 2 |

**Remount the filesystem**

After you edit /etc/fstab and add quotas, you need to remount the filesystems. For XFS filesystems, quota data is considered part of the filesystem metadata. For other filesystems, user quota information is stored in the aquota.user file in the root of the filesystem, and group quota is similarly stored in aquota.group. Version 1 quotas used quota.user and quota.group. For these filesystems, after you remount the filesystem, you must create the quota files and enable quota checking. The quotacheck command checks the quotas on all filesystems and creates the required aquota.user and aquota.group files if they do not exist. It can also repair damaged quota files. See the man pages for more information.

The following are some of the common options used with the quotacheck command:

**-a or --all**

Check all mounted filesystems in /etc/mtab (except NFS filesystems)

**-c or --create-files**

Ignore existing quota files. Run a new scan and write the results to disk

**-u or --user**

Check user quotas (this is the default)

**-g or --group**

Check group quotas

**-v or --verbose**

Verbose output

Listing 6 shows the result of unmounting the filesystems and then remounting them using the definitions from /etc/fstab, then running the quotacheck command on our freshly remounted filesystems. If you do not use the -a option, you must specify the filesystem that you want to check.

Listing 6. Creating quota database files for vfsold (Version 1) and vfsv0 or vfsv1 (Version 2)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22 | [root@attic4-cent ~]# umount /quotatest/\*  [root@attic4-cent ~]# mount /quotatest/ext4  [root@attic4-cent ~]# mount /quotatest/xfs  [root@attic4-cent ~]# quotacheck -augvc  quotacheck: Your kernel probably supports journaled quota but you are not using it.  Consider switching to journaled quota to avoid running quotacheck after an unclean shutdown.  quotacheck: Scanning /dev/sdc6 [/quotatest/ext4] done  quotacheck: Cannot stat old group quota file /quotatest/ext4/aquota.group: No such file  or directory. Usage will not be subtracted.  quotacheck: Cannot stat old group quota file /quotatest/ext4/aquota.group: No such file  or directory. Usage will not be subtracted.  quotacheck: Checked 2 directories and 1 files  quotacheck: Old file not found.  quotacheck: Skipping /dev/sdc3 [/quotatest/xfs]  [root@attic4-cent ~]# # find the created quota ext4 files  [root@attic4-cent ~]# find /quotatest  /quotatest  /quotatest/ext4  /quotatest/ext4/aquota.group  /quotatest/ext4/lost+found  /quotatest/ext4/aquota.user  /quotatest/xfs |

If your output includes messages like:

|  |  |
| --- | --- |
| 1 | quotacheck: Cannot create new quotafile /quotatest/ext4/aquota.user.new: Permission denied |

then you probably did not update your SELinux contexts. Go back and check [Listing 3](https://www.ibm.com/developerworks/library/l-lpic1-104-4/index.html#fixfiles-selinux).

Notice the warning from the quotacheck command suggesting that we switch to using journaled quotas. Do this by changing the options in /etc/fstab for the ext4 filesystem on /dev/sdc6 from  
defaults,usrquota,grpquota  
to  
defaults,usrjquota=aquota.user,grpjquota=aquota.group,jqfmt=vfsv0

After you update /etc/fstab, you will need to remount your filesystems again.

Notice also that the XFS filesystem was skipped by the quotacheck command because the XFS quota structures are contained in the meta-data and journaled.

**Start or stop quota checking**

To enforce vfsold and vfsv0 quota checking, you must turn it on using the quotaon command. The common options -a, -g, -u, and -v have the same meaning as for the quotacheck command. Similarly, if you do not specify the -a option, you must specify a filesystem. Use the -p option if you just want to display whether quotas are on or off. Use the quotaoff command to turn off quota checking. Listing 7 shows examples of both these commands.

Listing 7. Turning on quota checking for vfsold and vfsv0 quotas

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | [root@attic4-cent ~]# quotaon -p /quotatest/ext4/  group quota on /quotatest/ext4 (/dev/sdc6) is off  user quota on /quotatest/ext4 (/dev/sdc6) is off  [root@attic4-cent ~]# quotaon -uagv  /dev/sdc6 [/quotatest/ext4]: group quotas turned on  /dev/sdc6 [/quotatest/ext4]: user quotas turned on  [root@attic4-cent ~]# quotaoff -ugv /quotatest/ext4/  /dev/sdc6 [/quotatest/ext4]: group quotas turned off  /dev/sdc6 [/quotatest/ext4]: user quotas turned off  [root@attic4-cent ~]# quotaon -ugv /quotatest/ext4/  /dev/sdc6 [/quotatest/ext4]: group quotas turned on  /dev/sdc6 [/quotatest/ext4]: user quotas turned on |

**Setting quota limits**

As you have seen, quotas are controlled either through binary files in the root of the filesystem or through filesystem metadata. To set a quota for a particular user, use the edquota command. This command extracts the quota information for the user from the various filesystems with quotas enabled, creates a temporary file, and opens an editor for you to adjust the quotas. By default, edquota uses the vi editor. Set the EDITOR or the VISUAL environment variables if you want to use a different editor. To edit user quotas, use the -u option (default) with one or more user names. To edit group quotas, use the-g with one or more group names.

You must be root to edit quotas. Using vi as the editor, the information displayed will look something like Listing 9.

Listing 9. Running edquota -u ian

|  |  |
| --- | --- |
| 1  2  3  4 | Disk quotas for user ian (uid 1000):    Filesystem                   blocks       soft       hard     inodes     soft     hard    /dev/sdc6                         0          0          0          0        0        0    /dev/sdc3                         0          0          0          0        0        0 |

As you can see in Listing 9, edquota displays my current usage of both 1K blocks and inodes on each of the filesystems that have quota turned on. There are also soft and hard limits for both block and inode usage. In this example, these are 0, meaning no quota limit is enforced.

The soft limit is the value at which a user will receive email warnings about being over quota. The hard limit is the value that a user may not exceed. You can think of block limits as an approximate limit on the amount of data that a user may store, and inode limits as a limit on the number of files and directories.

**Changing quota limits**

You change the quota limits by changing the values in the temporary file and then saving the file. If you do not want to make changes, quit the file without saving. We will use some very small limits in our examples, so you can easily see the effects. Suppose you want to set my quota to 10MB of data and 50 files on each of our test filesystems. Allowing 10% additional for hard limits, you would set values as shown in Listing 10.

##### **Listing 10. Setting limits**

|  |  |
| --- | --- |
| 1  2  3  4 | Disk quotas for user ian (uid 1000):    Filesystem                   blocks       soft       hard     inodes     soft     hard    /dev/sdc6                         0      11264          0          0       50       55    /dev/sdc3                         0      11264          0          0       50       55 |

Save the file, and the new quotas will take effect. Note that any changes you make to the used blocks or inodes values will be ignored.

### Copying quotas

Now suppose you are creating IDs for additional developers. Assume you have users mary and jenni, and you'd like them both to have the same quota as ian. You do this using the -p option of edquota, which uses the quota values of ian as a prototype for those of the other users as shown in Listing 11.

##### **Listing 11. Setting quotas from a prototype**

|  |  |
| --- | --- |
| 1 | [root@attic4-cent ~]# edquota -p ian mary jenni |

### Group limits

You can also use edquota to restrict the allocation of disk space based on the group ownership of files. For example, the three developers above are members of the development group. To limit the total amounts used by the all members of the group to 250MB and 100 files, use the command edquota -g development and set the values as shown in Listing 12.

##### **Listing 12. Setting quotas for a group**

|  |  |
| --- | --- |
| 1  2  3  4 | Disk quotas for group development (gid 1002):    Filesystem                   blocks       soft       hard     inodes     soft     hard    /dev/sdc6                         4      25600      28160          1      100      110    /dev/sdc3                         0      25600      28160          1      100      110 |

You may wonder why there are already some blocks and inodes used on /dev/sdc6, our ext4 filesystem. Remember back in [Listing 2](https://www.ibm.com/developerworks/library/l-lpic1-104-4/index.html#listing2) we used chown and chmod to set the default group for all new files to be the development group? That also applies to the two quota database files. You can change these back to being in root's group as shown in Listing 13. Be sure to turn off quota checking to make the change and then turn it back on again.

##### **Listing 13. Resetting group owner for quota database files**

|  |  |
| --- | --- |
| 1  2  3 | [root@attic4-cent ~]# quotaoff /quotatest/ext4/  [root@attic4-cent ~]# chown :root /quotatest/ext4/aquota.\*  [root@attic4-cent ~]# quotaon /quotatest/ext4 |

### The grace period

Users or groups may exceed their soft limit for a grace period, which defaults to seven days. After the grace period, the soft limit is enforced as a hard limit. Once the hard limit is reached, some files must be deleted before new files can be created. Set grace periods using edquota -t. Again, you will be placed in an editor with data similar to that of Listing 14. As before, save changes to update the values. Be sure to leave your users enough time to receive their warning email and delete some files accordingly.

##### **Listing 14. Setting grace periods**

|  |  |
| --- | --- |
| 1  2  3  4  5 | Grace period before enforcing soft limits for users:  Time units may be: days, hours, minutes, or seconds    Filesystem             Block grace period     Inode grace period    /dev/sdc6                     7days                  7days    /dev/sdc3                     7days                  7days |

## Checking quotas

The quota command with no options displays the quotas for the invoking user on any filesystems for which the user has quotas set if the user has files on that system. The -v option displays the information for all filesystems that have quota enabled. The root user may also add a user name to the command to view quotas for a particular user. In Listing 15, we create a 1MB file on our ext4 filesystem and show the use of the quota command with and without the -v option.

##### **Listing 15. Displaying quotas**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | [ian@attic4-cent ~]$ dd if=/dev/zero of=/quotatest/ext4/ianfile1 bs=1024 count=1024  1024+0 records in  1024+0 records out  1048576 bytes (1.0 MB) copied, 0.00736743 s, 142 MB/s  [ian@attic4-cent ~]$ quota  Disk quotas for user ian (uid 1000):       Filesystem  blocks   quota   limit   grace   files   quota   limit   grace        /dev/sdc6    1024   11264       0               1      50      55  [ian@attic4-cent ~]$ quota -v  Disk quotas for user ian (uid 1000):       Filesystem  blocks   quota   limit   grace   files   quota   limit   grace        /dev/sdc6    1024   11264       0               1      50      55        /dev/sdc3       0   11264       0               0      50      55 |

Along with the statistics on current usage, you see the soft and hard quota limits displayed. If you run the quota command immediately after creating the file, you may see a slightly larger block count displayed for a short time.

Listing 16 shows what happens if you exceed the soft limit and then what happens if you attempt to exceed the hard limit. In this example, we add a 9.5MB file to the 1MB we already created, which is sufficient to exceed the soft limit. Notice how the soft limit has an asterisk beside it indicating that the user is over quota. Note also that the grace period columns now indicate how long the user has to correct the problem. Finally, when we try to make a copy of the 1MB file, the operation fails because this would exceed the hard limit for user ian.

##### **Listing 16. Exceeding block quotas**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | [ian@attic4-cent ~]$ dd if=/dev/zero of=/quotatest/ext4/ianfile2 bs=1024 count=9500  9500+0 records in  9500+0 records out  9728000 bytes (9.7 MB) copied, 0.0503342 s, 193 MB/s  [ian@attic4-cent ~]$ quota  Disk quotas for user ian (uid 1000):       Filesystem  blocks   quota   limit   grace   files   quota   limit   grace        /dev/sdc6   10524   11264       0               2      50      55  [ian@attic4-cent ~]$ cp /quotatest/ext4/ianfile1 /quotatest/ext4/ianfile3  sdc6: warning, user block quota exceeded.  [ian@attic4-cent ~]$ quota  Disk quotas for user ian (uid 1000):       Filesystem  blocks   quota   limit   grace   files   quota   limit   grace        /dev/sdc6   11548\*  11264       0   6days       3      50      55 |

To illustrate what happens if you exceed the inode (file) quote, we'll switch to user jenni, and create 52 empty files in /quotatest/ext4. Listing 17 shows the results.

##### **Listing 17. Exceeding Jenni's file quota**

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | [jenni@attic4-cent ~]$ for n in $(seq 1 52); do touch /quotatest/ext4/jenni$n;done  sdc6: warning, user file quota exceeded.  [jenni@attic4-cent ~]$ quota  Disk quotas for user jenni (uid 1004):       Filesystem  blocks   quota   limit   grace   files   quota   limit   grace        /dev/sdc6       0   11264       0              52\*     50      55   7days |

## Quota reports

Checking user quotas one user at a time is not very useful, so you will want to use the repquotacommand to generate quota reports. Listing 18 shows how to see the quotas for all users and groups on /quotatest/ext4/.

##### **Listing 18. Generating quota reports**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | [root@attic4-cent ~]# repquota -ug /quotatest/ext4  \*\*\* Report for user quotas on device /dev/sdc6  Block grace time: 7days; Inode grace time: 7days                          Block limits                File limits  User            used    soft    hard  grace    used  soft  hard  grace  ----------------------------------------------------------------------  root      --      16       0       0              1     0     0  ian       +-   11548   11264       0  6days      22    50    55  development --       4       0       0              1     0     0  jenni     -+       0   11264       0             52    50    55  6days  mary      --    1024   11264       0             35    50    55      \*\*\* Report for group quotas on device /dev/sdc6  Block grace time: 7days; Inode grace time: 7days                          Block limits                File limits  Group           used    soft    hard  grace    used  soft  hard  grace  ----------------------------------------------------------------------  root      --      16       0       0              1     0     0  development -+   12576   25600   28160            110   100   110  7days |

Note the plus sign in Listing 18 for users ian and jenni and for group development indicating that each is now over quota. User ian has too much data. User jenni has too many files. And group development also has too many files.

As with other quota commands, the -a option produces a report for all mounted filesystems that have quota enabled. The -v option produces more verbose output. And the -n option produces the listing by numeric user number rather than resolving the user number to a name. This may provide a performance boost for large reports, but is generally less useful to human readers.

## Project quotas on XFS filesystems

The xfs quota system also supports project quotas, which are an alternative to group quotas. You cannot use both group and project quotas on one XFS filesystem. Project quotas must be enabled by adding the prjquota (or pquota) to /etc/fstab.

Project quotas use the notion of a numeric id and a more descriptive project name. The file /etc/projects is used to map the numeric project identifiers to directory trees, while the file /etc/projid maps the numeric project identifiers to project names. It is possible, but less convenient, to operate without these two files in /etc. See the man pages for details.

To set project quotas, you must use the xfs\_quota command in expert mode (-x option). First you must use the project subcommand with the -s option to set up the project quotas by marking all the affected inodes as part of the project. You then use the limit subcommand to set hard or soft limits. Listing 19 shows how to set up a project tree under /quotatest/xfs/proj-dir1, with a 15MB project hard limit. We will also set up a parallel /quotatest/xfs/proj-dir2 with no limits for illustration.

##### **Listing 19. Setting up an xfs project quota.**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | [root@attic4-cent ~]# mkdir -m ag+w /quotatest/xfs/proj-dir{1,2}  [root@attic4-cent ~]# chown development:development /quotatest/xfs/proj-dir\*  [root@attic4-cent ~]# echo "50:/quotatest/xfs/proj-dir1" >> /etc/projects  [root@attic4-cent ~]# echo "dev-projects:50" >> /etc/projid  [root@attic4-cent ~]# xfs\_quota -x  xfs\_quota> path        Filesystem          Pathname   000  /quotatest/xfs      /dev/sdc3 (uquota, gquota)  [001] /quotatest/xfs/proj-dir1 /dev/sdc3 (project 50, dev-projects)  xfs\_quota> project -s dev-projects  Setting up project dev-projects (path /quotatest/xfs/proj-dir1)...  Processed 1 (/etc/projects and cmdline) paths for project dev-projects with  recursion depth infinite (-1).  xfs\_quota> limit -p bhard=15m dev-projects  xfs\_quota> q |

We'll illustrate the project quota in action by having user chris create some files. User chris is not a member of the development group and does not have any quotas set for either his id or group. Listing 20 shows that chris cannot create more than 15MB of data in /quotatest/xfs/proj-dir1, but is not limited in /quotatest/xfs/proj-dir2.

##### **Listing 20. Project quotas in action**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29 | [chris@attic4-cent ~]$ id chris  uid=1006(chris) gid=1006(chris) groups=1006(chris)  [chris@attic4-cent ~]$ cd /quotatest/xfs/  [chris@attic4-cent xfs]$ dir  proj-dir1  proj-dir2  [chris@attic4-cent xfs]$ dd if=/dev/zero of=proj-dir1/chris1 bs=1024 count=10000  10000+0 records in  10000+0 records out  10240000 bytes (10 MB) copied, 0.0386164 s, 265 MB/s  [chris@attic4-cent xfs]$ cp proj-dir1/chris1 proj-dir1/chris2  cp: writing `proj-dir1/chris2': No space left on device  [chris@attic4-cent xfs]$ ls -l proj-dir1  total 15356  -rw-rw-r--. 1 chris chris 10240000 Aug  8 18:01 chris1  -rw-rw-r--. 1 chris chris  5484544 Aug  8 18:01 chris2  [chris@attic4-cent xfs]$ # Note: second copy was truncated at hard limit  [chris@attic4-cent xfs]$ cp proj-dir1/chris1 proj-dir2/chris1  [chris@attic4-cent xfs]$ cp proj-dir1/chris1 proj-dir2/chris2  [chris@attic4-cent xfs]$ cp proj-dir1/chris1 proj-dir2/chris3  [chris@attic4-cent xfs]$ du -sh \*  15M proj-dir1  30M proj-dir2  [chris@attic4-cent xfs]$ du -sh $(find /quotatest/xfs/proj-dir1 -user chris)  9.8M    /quotatest/xfs/proj-dir1/chris1  5.3M    /quotatest/xfs/proj-dir1/chris2  [chris@attic4-cent xfs]$ du -sh $(find /quotatest/xfs/proj-dir2 -user chris)  9.8M    /quotatest/xfs/proj-dir2/chris1  9.8M    /quotatest/xfs/proj-dir2/chris2  9.8M    /quotatest/xfs/proj-dir2/chris3 |

# 104.5 Manage file permissions and ownership

### Who am I?

If you have not become another user, your ID is still the one you used to log in. If you have become another user, your prompt may include your user ID, as most of the examples in this tutorial do. If your prompt does not include your user ID, then you can use the whoami command to check your current effective id. shows some examples where the prompt strings (from the PS1 environment variable) are different from the other examples in this tutorial. Having your ID in the prompt string can be a useful feature.

##### **Determining effective user id**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | [ian@attic4-cent ~]$ <**strong**>whoami</**strong**>  ian  [ian@attic4-cent ~]$ <**strong**>ksh -l</**strong**>  $ <**strong**>whoami</**strong**>  ian  $ <**strong**>su jenni</**strong**>  Password:  [jenni@attic4-cent ian]$ <**strong**>whoami</**strong**>  jenni  [jenni@attic4-cent ian]$ <**strong**>echo "$PS1"</**strong**>  [\u@\h \W]\$  [jenni@attic4-cent ian]$ <**strong**>su - mary</**strong**>  Password:  [mary@attic4-cent ~]$ <**strong**>whoami</**strong**>  mary |

### What groups am I in?

Similarly, you can find out what groups you are in by using the groups command. You can find out both user and group information using the id command. Add a user ID parameter to either groups or id to see information for that user ID instead of the current user ID. shows some examples. Note that without a user ID, the id command will also display SELinux context as well as basic ID information.

##### **Determining group membership**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | [ian@attic4-cent ~]$ <**strong**>id</**strong**>  uid=1000(ian) gid=1000(ian) groups=1000(ian),1002(development),8093(editor) context=  unconfined\_u:unconfined\_r:unconfined\_t:s0-s0:c0.c1023  [ian@attic4-cent ~]$ <**strong**>id ian</**strong**>  uid=1000(ian) gid=1000(ian) groups=1000(ian),1002(development),8093(editor)  [ian@attic4-cent ~]$ <**strong**>groups</**strong**>  ian development editor  [ian@attic4-cent ~]$ <**strong**>id jenni</**strong**>  uid=1004(jenni) gid=1004(jenni) groups=1004(jenni),1002(development)  [ian@attic4-cent ~]$ <**strong**>groups jenni</**strong**>  jenni : jenni development  [ian@attic4-cent ~]$ <**strong**>su jenni</**strong**>  Password:  [jenni@attic4-cent ian]$ <**strong**>groups</**strong**>  jenni development  [jenni@attic4-cent ian]$ <**strong**>groups ian</**strong**>  ian : ian development editor |

## File ownership and permissions

### Directories

Directories use the same permissions flags as regular files, but they are interpreted differently.

* Read permission for a directory allows a user with that permission to list the contents of the directory.
* Write permission means a user with that permission can create or delete files in the directory.
* Execute permission allows the user to enter the directory and access any subdirectories.

Without execute permission on a directory, the filesystem objects inside the directory are not accessible. Without read permission on a directory, the filesystem objects inside the directory are not viewable in a directory listing, but these objects can still be accessed as long as you know the full path to the object on disk. is an artificial example to illustrate these points.

### Immutable files

The access modes and permissions provide extensive control over who can do what with files and directories. However, they do not prevent things such as inadvertent deletion of files by the root user. Although beyond the scope of LPI Topic 104.5, there are some additional attributes available on various filesystems that provide additional capabilities. One of these is the immutable attribute. If this is set, even root cannot delete the file until the attribute is unset.

Use the lsattr command to see whether the immutable flag (or any other attribute) is set for a file or directory. To make a file immutable, use the chattr command with the -i flag.

shows that user root can create an immutable file but cannot delete it until the immutable flag is removed.

##### **Immutable files**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | [root@attic4-cent ~]# <**strong**>touch keep.me</**strong**>  [root@attic4-cent ~]# <**strong**>chattr +i keep.me</**strong**>  [root@attic4-cent ~]# <**strong**>lsattr keep.me</**strong**>  ----i---------- keep.me  [root@attic4-cent ~]# r<**strong**>m -f keep.me</**strong**>  rm: cannot remove `keep.me': Operation not permitted  [root@attic4-cent ~]# <**strong**>chattr -i keep.me</**strong**>  [root@attic4-cent ~]# <**strong**>rm -f keep.me</**strong**> |

Changing the immutable flag requires root authority, or at least the CAP\_LINUX\_IMMUTABLE capability. Making files immutable is often done as part of a security or intrusion detection effort. See the capabilities man page (man capabilities) for more information.

## Setting file owner and group

### File group

To change the group of a file, use the chgrp command with a group name and one or more file names. You may also use the group number if you prefer. An ordinary user must own the file and also be a member of the group to which the file's group is being changed. The root user may change files to any group. shows an example.

##### **Changing group ownership**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | [ian@attic4-cent ~]$ <**strong**>touch file{1,2}</**strong**>  [ian@attic4-cent ~]$ <**strong**>ls -l file\*</**strong**>  -rw-rw-r--. 1 ian ian 0 Aug  9 07:16 file1  -rw-rw-r--. 1 ian ian 0 Aug  9 07:16 file2  [ian@attic4-cent ~]$ <**strong**>chgrp development file1</**strong**>  [ian@attic4-cent ~]$ <**strong**>chgrp 1002 file2</**strong**>  [ian@attic4-cent ~]$ <**strong**>ls -l file\*</**strong**>  -rw-rw-r--. 1 ian development 0 Aug  9 07:16 file1  -rw-rw-r--. 1 ian development 0 Aug  9 07:16 file2 |

As with many of the commands covered in this tutorial, chgrp has a -R option to allow changes to be applied recursively to all selected files and subdirectories.

### Default group

When you studied [Access modes](https://www.ibm.com/developerworks/library/l-lpic1-104-5/index.html#modes), you learned how setting the sgid mode on a directory causes new files created in that directory to belong to the group of the directory rather than to the group of the user creating the file.

You may also use the newgrp command to temporarily change your primary group to another group of which you are a member. A new shell will be created, and when you exit the shell, your previous group will be reinstated, as shown in .

##### **Using newgrp to temporarily change default group**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | [ian@attic4-cent ~]$ <**strong**>groups</**strong**>  ian development editor  [ian@attic4-cent ~]$ <**strong**>newgrp development</**strong**>  [ian@attic4-cent ~]$ <**strong**>groups</**strong**>  development ian editor  [ian@attic4-cent ~]$ <**strong**>touch file3</**strong**>  [ian@attic4-cent ~]$ <**strong**>ls -l file3</**strong**>  -rw-r--r--. 1 ian development 0 Aug  9 07:21 file3  [ian@attic4-cent ~]$ <**strong**>exit</**strong**>  exit  [ian@attic4-cent ~]$ <**strong**>groups</**strong**>  ian development editor |

### File owner

The root user can change the ownership of a file using the chown command. In its simplest form, the syntax is like the chgrp command, except that a user name or numeric ID is used instead of a group name or ID. The file's group may be changed at the same time by adding a colon and a group name or ID right after the user name or ID. If only a colon is given, then the user's default group is used. Naturally, the -R option will apply the change recursively. shows an example.

##### **Using chown to changing file ownership**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | [ian@attic4-cent ~]$ <**strong**>touch file4</**strong**>  [ian@attic4-cent ~]$ <**strong**>su -</**strong**>  Password:  [root@attic4-cent ~]# <**strong**>ls -l ~ian/file4</**strong**>  -rw-rw-r--. 1 ian ian 0 Aug  9 07:27 /home/ian/file4  [root@attic4-cent ~]# <**strong**>chown greg ~ian/file4</**strong**>  [root@attic4-cent ~]# <**strong**>ls -l ~ian/file4</**strong**>  -rw-rw-r--. 1 greg ian 0 Aug  9 07:27 /home/ian/file4  [root@attic4-cent ~]# <**strong**>chown jenni:mary ~ian/file4</**strong**>  [root@attic4-cent ~]# <**strong**>ls -l ~ian/file4</**strong**>  -rw-rw-r--. 1 jenni mary 0 Aug  9 07:27 /home/ian/file4  [root@attic4-cent ~]# <**strong**>chown :jenni ~ian/file4</**strong**>  [root@attic4-cent ~]# <**strong**>ls -l ~ian/file4</**strong**>  -rw-rw-r--. 1 jenni jenni 0 Aug  9 07:27 /home/ian/file4 |

# 104.6 Create and change hard and symbolic links

## Introducing links

On a storage device, a file or directory is contained in a collection of blocks. Information about a file is contained in an inode, which records information such as the owner, when the file was last accessed, how large it is, whether it is a directory or not, and who can read from or write to it. The inode number is also known as the file serial number and is unique within a particular filesystem. A directory entrycontains a name for a file or directory and a pointer to the inode where the information about the file or directory is stored.

A *hard link* is a directory entry that points to an inode, while a *soft link* or *symbolic link* is a directory entry that points to an inode that provides the name of another directory entry. The exact mechanism for storing the second name may depend on both the file system and the length of the name. Symbolic links are also called *symlinks*.

You can create hard links only for files and not for directories. The exception is the special directory entries in a directory for the directory itself and for its parent (. and ..), which are hard links that maintain the count of the number of subdirectories. Because hard links point to an inode, and inodes are only unique within a particular file system, hard links cannot cross file systems. If a file has multiple hard links, the file is deleted only when the last link pointing to the inode is deleted and the link count goes to 0.

Soft links, or symlinks, merely point to another file or directory by name rather than by inode. Soft links can cross filesystem boundaries. Deleting a soft link does not delete the target file or directory, and deleting the target file or directory does not automatically remove any soft links.

### Hard links

Use the ln command to create additional hard links to an existing file (but not to a directory, even though the system sets up . and .. as hard links).

Listing 1 shows how to create a directory containing two files and a subdirectory with two hard links to file1, one in the same directory and one in the subdirectory. We added a word to file1, and then another word to file3 and displayed the contents of the link in the subdirectory to show that all do indeed point to the same data.

##### **Listing 1. Creating hard links**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | [ian@atticf22 ~]$ mkdir -p lpi104-6/subdir  [ian@atticf22 ~]$ touch lpi104-6/file1  [ian@atticf22 ~]$ touch lpi104-6/file2  [ian@atticf22 ~]$ ln lpi104-6/file1 lpi104-6/file3  [ian@atticf22 ~]$ ln lpi104-6/file1 lpi104-6/subdir/file3sub  [ian@atticf22 ~]$ echo "something" > lpi104-6/file1  [ian@atticf22 ~]$ echo "else" >> lpi104-6/file3  [ian@atticf22 ~]$ cat lpi104-6/subdir/file3sub  something  else |

### Soft links

You use the ln command with the -s option to create soft links. Soft links use file or directory names, which may be relative or absolute. If you are using relative names, you will usually want the current working directory to be the directory where you are creating the link. Otherwise, the link you create will be relative to another point in the file system.

Listing 3 shows you two ways to create a soft link for the file1 that we just created, and also how to create soft links instead of the two hard links that failed in Listing 2.

##### **Listing 3. Creating soft links**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | [ian@atticf22 ~]$ # Create symlink using absolute paths  [ian@atticf22 ~]$ ln -s ~/lpi104-6/file1 ~/lpi104-6/file4  [ian@atticf22 ~]$ # Create symlink using relative paths  [ian@atticf22 ~]$ cd lpi104-6/  [ian@atticf22 lpi104-6]$ ln -s file1 file5  [ian@atticf22 lpi104-6]$ cd ..  [ian@atticf22 ~]$ # Create symlink across file systems  [ian@atticf22 ~]$ mkdir -p ~ian/research/lpi104-6  [ian@atticf22 ~]$ ln -s ~/lpi104-6/file1 ~ian/research/lpi104-6/file4  [ian@atticf22 ~]$ # Create symlink for directory  [ian@atticf22 ~]$ ln -s lpi104-6 lpidir104-6 |

As before, you can use any of the links or the target file name to reference the file or directory. Listing 4 shows some examples.

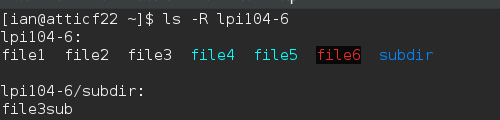
##### **Listing 4. Using soft links**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | [ian@atticf22 ~]$ echo "another line" >> ~ian/research/lpi104-6/file  [ian@atticf22 ~]$ # cat a symlink  [ian@atticf22 ~]$ cat lpi104-6/file5  something  else  [ian@atticf22 ~]$ # cat a hard link  [ian@atticf22 ~]$ cat lpi104-6/file1  something  else  [ian@atticf22 ~]$ # display directory contents using symlink  [ian@atticf22 ~]$ ls lpidir104-6  file1  file2  file3  file4  file5  subdir |

### Finding information

On many systems, the ls command is aliased to ls --color=auto, which prints different types of filesystem objects in different colors. If you use this option, symlinks might show up with cyan text, as illustrated in Figure 1.

##### **Figure 1. Using the --colors option of ls to identify links**

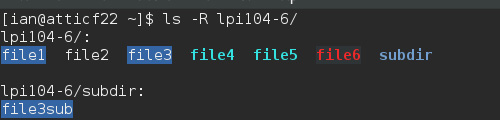


On older configurations hard links might show up with a dark blue background. The colors are configurable using the dircolors program. If you customize your terminal appearance, you will probably want to change colors for some of the output of ls. Listing 6 shows one way of getting a blue background for hard links on our Fedora 22 system, and Figure 2 shows the result. Use the man pages to further understand the example.

##### **Listing 6. Using dircolors to set a blue background for hard links**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | [ian@atticf22 ~]$ # Save a copy of dircolors defaults  [ian@atticf22 ~]$ dircolors -p > dircolors-defaults  [ian@atticf22 ~]$ grep MULTI dircolors-defaults  MULTIHARDLINK 00 # regular file with more than one link  [ian@atticf22 ~]$ # Change MULTIHARDLINK to blue background  [ian@atticf22 ~]$ sed -e'/MULTI/s/00/00;44/' dircolors-defaults > dircolors-new  [ian@atticf22 ~]$ grep MULTI dircolors-new  MULTIHARDLINK 00;44 # regular file with more than one link  [ian@atticf22 ~]$ # Set the new colors for the current terminal session  [ian@atticf22 ~]$ eval $(dircolors dircolors-new ) |

##### **Figure 2. Identifying hard links with a blue background**



While color might be convenient for sighted people who can distinguish them, they are not much use to others, and certainly not much use to shell scripts or programs. Without color, you need more information, such as that provided by a long listing using ls -l. In Listing 7, we explicitly disable color output for the first example, but you could also explicitly call the /bin/ls command as we have done for the other two examples.

##### **Listing 7. Identifying links**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | [ian@atticf22 ~]$ ls --color=none -lR lpi104-6  lpi104-6:  total 12  -rw-rw-r--. 3 ian ian   15 Aug  9 14:19 file1  -rw-rw-r--. 1 ian ian    0 Aug  9 14:19 file2  -rw-rw-r--. 3 ian ian   15 Aug  9 14:19 file3  lrwxrwxrwx. 1 ian ian   24 Aug  9 14:26 file4 -> /home/ian/lpi104-6/file1  lrwxrwxrwx. 1 ian ian    5 Aug  9 14:26 file5 -> file1  lrwxrwxrwx. 1 ian ian   14 Aug  9 14:34 file6 -> lpi104-6/file1  drwxrwxr-x. 2 ian ian 4096 Aug  9 14:19 subdir    lpi104-6/subdir:  total 4  -rw-rw-r--. 3 ian ian 15 Aug  9 14:19 file3sub  [ian@atticf22 ~]$ /bin/ls -l ~ian/research/lpi104-6/file4  lrwxrwxrwx. 1 ian ian 24 Aug  9 14:27 /home/ian/research/lpi104-6/file4 -> /home/ian/lpi104-6/file1  [ian@atticf22 ~]$ /bin/ls -l lpidir104-6  lrwxrwxrwx. 1 ian ian 8 Aug  9 14:27 lpidir104-6 -> lpi104-6 |

### Who links to me?

To find which files are hard links to a particular inode, you can use the find command and the -samefile option with a file name or the -inum option with an inode number, as shown in Listing 12.

##### **Listing 12. Finding hard links to the same file**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | [ian@atticf22 ~]$ find lpi104-6 -samefile lpi104-6/file1  lpi104-6/file1  lpi104-6/file3  lpi104-6/subdir/file3sub  [ian@atticf22 ~]$ ls -i lpi104-6/file1  1988884 lpi104-6/file1  [ian@atticf22 ~]$ find lpi104-6 -inum 1988884  lpi104-6/file1  lpi104-6/file3  lpi104-6/subdir/file3sub |

To find which files link symbolically to a particular file, you can use the find command and the -lnameoption with a file name, as Listing 13 illustrates. Links may use a relative or absolute path, so you probably want a leading asterisk in the name to find all matches.

##### **Listing 13. Finding symbolic links to a file or directory**

|  |  |
| --- | --- |
| 1  2  3  4  5 | [ian@atticf22 ~]$ find lpi104-6 research/lpi104-6 -lname "\*file1"  lpi104-6/file4  lpi104-6/file6  lpi104-6/file5  research/lpi104-6/file4 |

# 104.7 Find and place system files

## Filesystem Hierarchy Standard

By placing files in the same general place across Linux distributions, the FHS simplifies distribution-independent software development. The FHS is also used in the Linux Standard Base (see [Resources)](https://www.ibm.com/developerworks/library/l-lpic1-104-7/index.html#artrelatedtopics). The FHS allows both users and software to predict the location of installed files and directories. An FHS-compliant filesystem assumes that the operating system supports the basic security features found in most UNIX filesystems.

### The two independent FHS categories

At the core of the FHS are two independent characteristics of files:

**Shareable vs. unshareable**

Shareable files can be located on one system and used on another, while unshareable files must reside on the system on which they are used.

**Static vs. variable**

Static files change only through system administrator intervention, such as installing or upgrading a package, and include documentation, libraries, and binaries. Variable files are all other files, such as logs, spool files, databases, and user data, which are subject to change by users and by system processes.

These distinctions allow files with different sets of characteristics to be stored on different filesystems. Table 1 is an example from the FHS document showing a layout that would be FHS-compliant.

##### **Table 1. FHS example**

|  | **Shareable** | **Unshareable** |
| --- | --- | --- |
| **Static** | /usr /opt | /etc /boot |
| **Variable** | /var/mail /var/spool/news | /var/run /var/lock |

Linux systems often contain hundreds of thousands of files. A 64-bit Fedora 22 system that I recently installed has over 170,000 files in the /usr hierarchy alone.

|  |  |
| --- | --- |
| 1  2 | [root@atticf22 ~]# find /usr -type f | wc -l  174182 |

Most of my other installations have over 100,000 files and often 200,000 files or more. The next few sections look at tools to help you find files, particularly programs, in this vast sea of data.

## Your PATH to executable files

If you have used several Linux systems, you may have noticed that if you log in as root, you are able to execute some commands which you cannot execute if you are a user. And some commands can be executed by users but not root. When you run a program at the command line, the bash (or other) shell searches through a list of directories to find the program you requested. The list of directories is specified in your PATH environment variable. On older systems, it was common for root's path to include /sbin, while non-root user paths did not. Listing 1 shows user and root path examples from three different distributions.

##### **Listing 1. Some PATH examples**

[ian@attic4-cent ~]$ # CentOS 6

[ian@attic4-cent ~]$ echo $PATH

/usr/lib64/qt-3.3/bin:/usr/local/bin:/usr/bin:/bin:/usr/local/sbin:/usr/sbin:/sbin:/home/ian/bin

[ian@attic4-cent ~]$ su -

Password:

[root@attic4-cent ~]# echo $PATH

/usr/lib64/qt-3.3/bin:/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin:/root/bin

[ian@atticf22 ~]$ # echo Fedora 22

[ian@atticf22 ~]$ echo $PATH

/usr/local/bin:/usr/local/sbin:/usr/bin:/usr/sbin:/bin:/sbin:/home/ian/.local/bin:/home/ian/bin

[ian@atticf22 ~]$ su -

Password:

[root@atticf22 ~]# echo $PATH

/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin:/root/bin

[root@atticf22 ~]# ls -l /sbin

lrwxrwxrwx. 1 root root 8 Aug 16  2014 /sbin -> usr/sbin

As you can see, the PATH variable is just a list of directory names, separated by colons. Note the differences between the user path and root path order for CentOS and the two extra games directories in the Linux Mint user path. Since the fortune game command is actually located in /usr/games/fortune, the Mint root user will have to stick to work instead of playing games unless he or she provides a fully qualified path name. On systems where the /sbin directory was not included in user paths, it was common for programs such as tune2fs to only be found by the root user.

Usually, your path is set in an initialization file such as .bash\_profile or .bashrc. You can change it for the current bash process by specifying a new path. Remember to export the PATH variable if you want the new value to be available to other processes that you start from this one. Suppose I am developing a "Hello World" application in the my test-hello directory. Listing 2 shows how to add it to my path and run my hello executable without having to give a fully qualified path each time.

##### **Listing 2. Changing your PATH**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | [ian@atticf22 ~]$ hello  bash: hello: command not found...  [ian@atticf22 ~]$ test-hello/hello  Hello World!  [ian@atticf22 ~]$ export PATH=~/test-hello:$PATH  [ian@atticf22 ~]$ hello  Hello World! |

## The which, type, and whereis commands

In the previous section, you saw why the hello command might not be available if you attempted to run it. However, there are several other useful commands that can help you find which command would run if you typed a command name.

### The which command

You can use the which command to search your path and find out which command will be executed (if any) when you type a command. Listing 3 shows an example of finding the hello command before and after we update the path.

##### **Listing 3. Using which**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | [ian@atticf22 ~]$ echo $PATH  /usr/local/bin:/usr/local/sbin:/usr/bin:/usr/sbin:/bin:/sbin:/home/ian/.local/bin:/home/ian/bin  [ian@atticf22 ~]$ which hello  /usr/bin/which: no hello in (/usr/local/bin:/usr/local/sbin:/usr/bin:/usr/sbin:/bin:/sbin:  /home/ian/.local/bin:/home/ian/bin)  [ian@atticf22 ~]$ export PATH=~/test-hello:$PATH  [ian@atticf22 ~]$ echo $PATH  /home/ian/test-hello:/usr/local/bin:/usr/local/sbin:/usr/bin:/usr/sbin:/bin:/sbin:  /home/ian/.local/bin:/home/ian/bin  [ian@atticf22 ~]$ which hello  ~/test-hello/hello |

The which command shows you the first occurrence of a command in your path. If you want to know if there are multiple occurrences, then add the -a option as shown in Listing 4.

##### **Listing 4. Using which to find multiple occurrences**

|  |  |
| --- | --- |
| 1  2  3  4  5 | [ian@atticf22 ~]$ which awk  /usr/bin/awk  [ian@atticf22 ~]$ which -a awk  /usr/bin/awk  /bin/awk |

Here we find the awk command in two places: in /usr/bin (which is the main directory for commands on the system), and in /bin (which contains commands that may be used both by the system administrator and by users, but which are required when no other filesystems are mounted).

Another tutorial in this series, "[Learn Linux 101: Create and change hard and symbolic links](https://www.ibm.com/developerworks/library/l-lpic1-104-6/)," shows you how to check that these three different files all eventually represent the same underlying gawkcommand as shown in Listing 5.

##### **Listing 5. Awk commands lead to gawk**

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | [ian@atticf22 ~]$ ls -l $(which -a awk)  lrwxrwxrwx. 1 root root 4 Jan  2  2015 /bin/awk -> gawk  lrwxrwxrwx. 1 root root 4 Jan  2  2015 /usr/bin/awk -> gawk  [ian@atticf22 ~]$ ls -ld /bin /usr/bin  lrwxrwxrwx. 1 root root     7 Aug 16  2014 /bin -> usr/bin  dr-xr-xr-x. 2 root root 69632 Aug 10 10:40 /usr/bin |

### The type command

There are some commands that the which command will not find, such as shell builtins. The typecommand is a builtin that will tell you how a given command string will be evaluated for execution. Listing 6 uses which and type to show that the type command is not an executable found on your path, but is a shell builtin.

##### **Listing 6. Using type**

|  |  |
| --- | --- |
| 1  2  3  4  5 | [ian@atticf22 ~]$ which type  /usr/bin/which: no type in (/usr/local/bin:/usr/local/sbin:/usr/bin:/usr/sbin:/bin:/sbin:  /home/ian/.local/bin:/home/ian/bin)  [ian@atticf22 ~]$ type type  type is a shell builtin |

### The whereis command

If you want more information than just the location of a program, you can use the whereis command. For example, you can find the man pages or other information, as shown in Listing 7.

##### **Listing 7. Using whereis to find man pages**

|  |  |
| --- | --- |
| 1  2  3 | [ian@atticf22 ~]$ whereis awk  awk: /usr/bin/awk /usr/libexec/awk /usr/share/awk /usr/share/man/man1p/awk.1p.gz  /usr/share/man/man1/awk.1.gz |

Note that the copy of awk in /bin was not found by whereis. The directories used by whereis are fixed, so the command may not always find what you are looking for. The whereis command can also search for source files, specify alternate search paths, and search for unusual entries. Consult the man pages to see how to override this behavior or change the fixed paths used by whereis.

whereis searches through a list of certain directories (listed in man) for source/binary and manuals sections.  
which searches only for binaries in $PATH.

Basically, whereis searches for "possibly useful" files, while which only searches for executables.

## The find command

In an earlier tutorial in this series, "[Learn Linux 101: File and directory management](https://www.ibm.com/developerworks/library/l-lpic1-103-3/)," you learned how to find files based on name (including wildcards), path, size, or timestamp. In another earlier tutorial in this series, "[Learn Linux 101: Create and change hard and symbolic links](https://www.ibm.com/developerworks/library/l-lpic1-104-6/)," you learned how to find the links to a particular file or inode.

The find command is the Swiss Army knife of file-searching tools on Linux systems. Two other capabilities that you may find useful are its ability to find files based on user or group name and its ability to find files based on permissions.

Suppose you want to see what files a user has in the /tmp hierarchy. Listing 8 shows how the root user could find all the files for user ian in /tmp.

##### **Listing 8. Finding files by user and group**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | [root@atticf22 ~]# find /tmp -user ian  /tmp/jna-104022  ...  /tmp/hsperfdata\_ian  /tmp/hsperfdata\_ian/2390  /tmp/tracker-extract-files.1000  /tmp/.esd-1000  /tmp/.esd-1000/socket  /tmp/.X11-unix/X0  /tmp/.ICE-unix/1668 |

You can also find files by group using the -group test. And you can find files that do not belong to any user or group on the system using the -nouser and -nogroup options. As with other tests, you can negate the test using !. I usually set my user number to 1000, as that is the default on many systems. I also create a group called ian with 1000 as the group number if the system does not automatically do that. Other systems still start at 500, or put new users in the group 'users' by default. Some of my older research material that was archived from a Red Hat 6.2 system still has user 500.

Listing 9 shows how to find some directories that are not owned by my current user group. The research/rh62/involution is owned by user 500 and group 4, neither of which exist on my current system. To find files or directories by numeric user id or group id, use the -uid or -gid tests.

##### **Listing 9. Finding directories not owned by ian**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | [ian@atticf22 ~]$ find -L research -maxdepth 2 -type d ! -group ian  research/rh62/involution  research/rh62/programs  research/lost+found  find: ‘research/lost+found’: Permission denied  [ian@atticf22 ~]$ ls -ld research/rh62/involution  drwxr-xr-x. 2 500 adm 4096 Nov 10  1999 research/rh62/involution |

To find files by permission, you can use the -perm test along with symbolic expressions similar to those used with the chmod or umask commands. You can search for exact permissions, but it is often more useful to prefix the permission expression with a hyphen to indicate that you want files with those permissions set, but that you don't care about other permissions. Listing 10 illustrates how to find files that are executable by user, group, and everyone, and two different ways of finding files that are not readable by others.

##### **Listing 10. Finding files by permission**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | [ian@atticf22 ~]$ find . -maxdepth 1 -type f -perm -uga=x  ./hello  [ian@atticf22 ~]$ find . -maxdepth 1 ! -perm  -o=r  .  ./.ssh  ./.ICEauthority  ./.bash\_history  ./.config  ...  ./.kde  ./.cache  [ian@atticf22 ~]$ find . -maxdepth 1 ! -perm  -0004  .  ./.ssh  ./.ICEauthority  ./.bash\_history  ./.config  ...  ./.kde  ./.cache |

We have covered several major types of search that you can do with the find command. To further narrow your output, you can combine multiple expressions, and you can add regular expressions to the mix. To learn more about this versatile command, use the man page, or better, use info find if you have the info system installed.

Listing 11 shows a final example of searching with find. This example does a cd to /usr/include to keep the listing length manageable, then finds all files containing packet in their path name without regard to case. The second example further restricts this output to files that are not directories and that are at least 1500 bytes in size. Actual output on your system may differ depending on which packages you have installed.

##### **Listing 11. A final example of find**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | [ian@atticf22 ~]$ cd /usr/include  [ian@atticf22 include]$ find . -iregex ".\*packet.\*"  ./net/if\_packet.h  ./linux/packet\_diag.h  ./linux/if\_packet.h  ./netpacket  ./netpacket/packet.h  [ian@atticf22 include]$ find . -iregex ".\*packet.\*" ! -type d -size +1500c  ./linux/packet\_diag.h  ./linux/if\_packet.h  ./netpacket/packet.h |

## The locate and updatedb commands

The find command searches all the directories you specify, every time you run it. To speed things up, you can use another command, locate, which uses a database of stored path information rather than searching the filesystem every time.

### The locate command

The locate command searches for matching files in a database that is usually updated daily by a cron job.

The locate command matches against any part of a path name, not just the file name. Put the file name in single quotes and include at least one globbing character to match more precisely. Listing 12 shows how to find paths containing the string bin/ls, and shows two examples of using globbing characters to restrict the output.

##### **Listing 12. Using locate to find paths and restrict output**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | [ian@attic4-cent ~]$ # CentOS 6  [ian@attic4-cent ~]$ locate /bin/ls  /bin/ls  /bin/lsblk  /bin/lscgroup  /bin/lssubsys  /usr/bin/lsattr  /usr/bin/lsb\_release  /usr/bin/lscpu  /usr/bin/lsdiff  /usr/bin/lshal  /usr/bin/lslogins  /usr/bin/lsusb  /usr/bin/lsusb.py  [ian@attic4-cent ~]$ locate '\/bin/ls'  /bin/ls  [ian@attic4-cent ~]$ locate '/bin/ls\*'  /bin/ls  /bin/lsblk  /bin/lscgroup  /bin/lssubsys |

### The updatedb command

The default database used by locate is stored in the /var filesystem, in a location such as /var/lib/locatedb. This may be different on systems that use slocate or mlocate packages to provide additional security or speed. You can find statistics on your locate database using locate -S as shown in Listing 13.

##### **Listing 13. Locatedb statistics**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | [ian@attic4-cent ~]$ # CentOS 6  [ian@attic4-cent ~]$ locate -S  Database /var/lib/mlocate/mlocate.db:      77,143 directories      776,460 files      56,872,710 bytes in file names      21,074,461 bytes used to store database |

The database is created or updated using the updatedb command. This is usually run daily as a cron job. The file /etc/updatedb.conf, or sometimes /etc/sysconfig/locate, is the configuration file forupdatedb. To enable daily updates, the root user needs to edit /etc/updatedb.conf and set DAILY\_UPDATE=yes. To create the database immediately, run the updatedb command as root.

Other considerations for using locate include security considerations and network file I/O considerations for daily builds of the updatedb database. Check the man pages and updatedbconfiguration files for more details.

### /usr and /var hierarchies

The /usr and /var hierarchies are complex enough to have complete sections of the FHS devoted to them. The /usr filesystem is the second major section of the filesystem, containing shareable, read-only data. It can be shared between systems, although present practice does not often do this.

The /var filesystem contains variable data files, including spool directories and files, administrative and logging data, and transient and temporary files. Some portions of /var are not shareable between different systems, but others, such as /var/mail, /var/cache/man, /var/cache/fonts, and /var/spool/news, may be shared.

**Note:** Version 3.0 of the FHS moved the /var/run hierarchy to a separate /run hierarchy. The /run directory contains system information describing the system since it was booted. Files under this directory must be removed or truncated at boot time. Programs may continue to use /var/run to for the purposes of backwards compatibility but should migrate to using /run.