

6th Edition



LINUX

IN A NUTSHELL

A Desktop Quick Reference

O'REILLY®

Ellen Siever, Stephen Figgins,
Robert Love & Arnold Robbins

Linux in a Nutshell



Everything you need to know about Linux is in this book. Written by people with years of active participation in the Linux community, *Linux in a Nutshell*, Sixth Edition, thoroughly covers programming tools, system and network administration tools, the shell, editors, and the GRUB boot loader.

This updated edition offers a tighter focus on Linux system essentials, as well as more coverage of new capabilities. It also highlights the most important options for using the vast number of Linux commands. You'll find many helpful new tips and techniques in this reference, whether you're new to this operating system or have been using it for years.

- Get the Linux commands for system administration and network management
- Learn hundreds of the most important shell commands available on Linux
- Understand the Bash shell command-line interpreter
- Search and process text with regular expressions
- Manage your servers via virtualization with Xen and VMware
- Use the Emacs text editor and development environment, as well as the vi, ex, and vim text-manipulation tools

INTRODUCTORY INTERMEDIATE ADVANCED

Previous Linux experience recommended.

US \$49.99

CAN \$62.99

ISBN: 978-0-596-15448-6

9 780596 154486

5 4 9 9 9

Safari
Books Online

Free online edition
for 45 days with purchase of
this book. Details on last page.

Linux in a Nutshell
*was named the “Best Linux Book of All Time”
by the readers of the
Linux Journal.*

Ellen Siever is a writer and editor specializing in Linux and other open source topics. She also coauthored *Perl in a Nutshell* (O'Reilly).

Stephen Figgins administers Linux servers for Sunflower Broadband in Lawrence, Kansas.

Robert Love is an active contributor to both the Linux kernel and GNOME. He works at Google on Android.

Arnold Robbins, a professional programmer and technical author, has worked with Unix systems since 1980.

O'REILLY®
oreilly.com

Function options

You must use exactly one of these, and it must come before any other options:

-A, --catenate, --concatenate

Concatenate a second tar file to the end of the first.

-c, --create

Create a new archive.

-d, --diff, --compare

Compare the files stored in *tarfile* with *other-files*. Report any differences: missing files, different sizes, different file attributes (such as permissions or modification time).

--delete

Delete from the archive. This option cannot be used with magnetic tape.

-r, --append

Append *other-files* to the end of an existing archive.

-t, --list

Print the names of *other-files* if they are stored in the archive (if *other-files* are not specified, print names of all archived files).

-u, --update

Add files if not in the archive or if modified.

-x, --extract, --get

Extract *other-files* from an archive (if *other-files* are not specified, extract all files).

Options

--anchored

Exclude patterns must match the start of the filename (the default).

--atime-preserve

Preserve original access time on extracted files.

-b *n*, --blocking-factor=*n*

Set block size to $n \times 512$ bytes. By default, $n=20$.

-B, --read-full-records

Reblock while reading; used for reading from 4.2BSD pipes.

--backup[=*type*]

Back up files rather than deleting them. If no backup type is specified, a simple backup is made with \sim as the suffix. (See also **--suffix**.) The possible values of *type* are:

t, numbered

Make numbered backups.

nil, existing

Make numbered backups if there are already numbered backups; otherwise, make simple backups.

never, simple

Always make simple backups.

- C directory, --directory=directory**
cd to *directory* before beginning tar operation.
- checkpoint**
List directory names encountered.
- exclude=pattern**
Remove files matching *pattern* from any list of files.
- f file, --file=file**
Store files in or extract files from archive *file*. Note that *file* may take the form *hostname:filename*.
- F script, --info-script=script, --new-volume-script=script**
Implies -M (multiple archive files). Run *script* at the end of each file.
- force-local**
Interpret filenames in the form *hostname:filename* as local files.
- g file, --listed-incremental=file**
Create, list, or extract new-style incremental backup.
- G, --incremental**
Create, list, or extract old-style incremental backup.
- group=group**
Use *group* as the group for files added to the archive.
- h, --dereference**
Dereference symbolic links, and archive the files they point to rather than the symbolic link.
- help**
Print help message and exit.
- i, --ignore-zeros**
Ignore blocks of zeros (i.e., EOFs).
- ignore-case**
Ignore case when excluding files.
- ignore-failed-read**
Ignore unreadable files to be archived. Default behavior is to exit when encountering these.
- index-file=file**
With -v, send output to *file*.
- j, --bzip2**
Compress files with bzip2 before archiving them, or uncompress them with bunzip2 before extracting them.
- k, --keep-old-files**
When extracting files, do not overwrite files with similar names. Instead, print an error message.
- K file, --starting-file=file**
Begin tar operation at *file* in archive.
- keep-newer-files**
When extracting files, do not overwrite files that are newer than the archive files.

- m, --touch**
Do not restore file modification times; update them to the time of extraction.
- M, --multivolume**
Expect archive to be multivolume. With **-c**, create such an archive.
- mode=permissions**
Use *permissions* when adding files to an archive. The permissions are specified the same way as for the **chmod** command.
- N date, --newer=date, --after-date=date**
Ignore files older than *date*.
- newer-mtime=date**
Add only files whose contents have changed since *date* to the archive.
- no-anchored**
Exclude patterns may match anything following a slash.
- no-ignore-case**
Do not ignore case when excluding files.
- no-same-permissions**
Do not extract permissions information when extracting files from the archive. This is the default for users, and therefore affects only the superuser.
- no-recursion**
Do not move recursively through directories.
- no-wildcards**
Don't use wildcards when excluding files; treat patterns as strings.
- no-wildcards-match-slash**
Wildcards do not match / when excluding files.
- null**
Allow filenames to be null-terminated with **-T**. Override **-C**.
- numeric-owner**
Use the numeric owner and group IDs rather than the names.
- o, --no-same-owner**
When extracting, create files with yourself as owner.
- O, --to-stdout**
Print extracted files to standard output.
- occurrence[=n]**
Process only the *n*th occurrence of each file (default is 1). Use with **--delete**, **--diff**, **--extract**, or **--list**.
- one-file-system**
Do not archive files from other filesystems.
- overwrite**
Overwrite existing files and directory metadata when extracting from archive.
- overwrite-dir**
Overwrite existing directory metadata when extracting from archive.

- owner=*owner***
Set *owner* as the owner of extracted files instead of the original owner. *owner* is first assumed to be a username, then, if there is no match, a numeric user ID.
- p, --same-permissions, --preserve-permissions**
Keep permissions of extracted files the same as the originals.
- P, --absolute-names**
Do not remove initial slashes (/) from input filenames.
- posix**
Create a POSIX-compliant archive.
- preserve**
Equivalent to invoking both the -p and -s options.
- R, --block-number**
Display archive's block number in messages.
- record-size=*size***
Treat each record as having *size* bytes, where *size* is a multiple of 512.
- recursion**
Move recursively through directories.
- recursive-unlink**
Remove existing directory hierarchies before extracting directories with the same name.
- remove-files**
Remove originals after inclusion in archive.
- rsh-command=*command***
Do not connect to remote host with rsh; instead, use *command*.
- s, --same-order, --preserve-order**
When extracting, sort filenames to correspond to the order in the archive.
- S, --sparse**
Treat sparse files more efficiently when adding to archive.
- same-owner**
When extracting, create files with the same ownership as the originals.
- show-defaults**
Display the default tar options.
- show-omitted-dirs**
List directories being omitted when operating on an archive.
- strip-components=*num*, --strip-path=*num***
Strip the specified number of leading components from filenames before extracting. Use --strip-components for tar versions beginning with tar-1.14.90. Earlier versions of tar-1.14 use --strip-path.
- suffix=*suffix***
Use *suffix* instead of the default ~ when creating a backup file.

-T *file*, --files-from=*file*
 Consult *file* for files to extract or create.

--totals
 Print byte totals.

-U, --unlink-first
 Remove each existing file from the filesystem before extracting from the archive.

--use-compress-program=*program*
 Compress archived files with *program*, or uncompress extracted files with *program*.

--utc
 Display file modification dates in UTC format.

-v, --verbose
 Verbose. Print filenames as they are added or extracted.

-V *name*, --label=*name*
 Name this volume *name*.

--version
 Print version information and exit.

--volno-file=*file*
 Use/update the volume number in *file*.

-w, --interactive, --confirmation
 Wait for user confirmation (y) before taking any actions.

-W, --verify
 Check archive for corruption after creation.

--wildcards
 Use wildcards when excluding files.

--wildcards-match-slash
 Wildcards match / when excluding files.

-X *file*, --exclude-from *file*
 Consult *file* for list of files to exclude.

-z, --gzip, --gunzip, --ungzip
 Compress files with **gzip** before archiving them, or uncompress them with **gunzip** before extracting them.

-Z, --compress, --uncompress
 Compress files with **compress** before archiving them, or uncompress them with **uncompress** before extracting them.

Examples

Create an archive of book chapter files (c), show the command working (v), and store the results in *Chapters.tar*:

```
tar cvf Chapters.tar chapter*
```

List the archive's contents in a format like **ls -l**:

```
tar tvf Chapters.tar
```

Extract Chapter 1:

```
tar xvf Chapters.tar chapter1
```

Create an archive of the current directory and store it in a file *backup.tar*:

```
tar cvf - `find . -print` > backup.tar
```

(The **-** tells **tar** to store the archive on standard output, which is then redirected.)

Create an archive and filter it through **bzip2** to compress it:

```
tar cvfj Chapters.tar.bz2 chapter*
```

Filter an existing archive through **gzip**, extracting the contents but leaving the original file compressed:

```
tar xvzf chapters.tar.gz
```

taskset

taskset [options] [mask | list] [pid | command [args]]

taskset is used to retrieve or set the processor affinity mask of either an existing process, given its PID, or to run a new a process, given its command name, with a specified affinity mask. The Linux scheduler will then honor the given affinity mask, ensuring that the process in question runs only on allowed processors.

Options

-c, --cpu-list

The affinity mask is provided in list form, for example, “0,2,5-6,” not as a bitmask.

-p, --pid

Set or retrieve the mask of the given PID. Do not start a new process.

-h, --help

Display usage information and then exit.

-V, --version

Display version information and then exit.

tcpdump

tcpdump [options] [expression]

System administration command. Dump headers and packets of network traffic that match *expression*. The command continues to capture packets until it receives a SIGTERM or SIGINT signal (usually generated by typing the interrupt character **control-C**). When finished, it will generate a report on traffic captured, received, or dropped by the kernel.

Expressions

Create matching expressions using the following primitives followed by an ID or name.

direction

A qualifier indicating whether to match source or destination information. Accepted values are **src**, **dst**, **src or dst**, and **src and dst**. When not specified, the expression will match either source or destination traffic.

protocol

A qualifier restricting matches to a particular kind of packet. Accepted values are: **ether**, **fddi**, **tr**, **wlan**, **ip**, **ip6**, **arp**, **rarp**, **decnet**, **tcp**, and **udp**. If not specified, the match defaults to any appropriate protocol matching type.

type

A qualifier indicating what kind of thing the ID or name references, such as a part of a hostname (**host**), IP address (**net**) or port (**port**). When not specified, the match defaults to **host**.

Options

-A Print packets in ASCII text.

-c n
Exit after receiving *n* packets.

-C n
When saving to a file, do not write files larger than *n* million bytes. Open a new file with the same basename appended by a number. Start with the number 1.

-d, -dd, -ddd

Compile and dump the packet-matching code for the given expression, then exit. Use the second form to dump it as a C programming fragment. Use the third form to dump the code in decimal.

-D Print a list of the available interfaces, then exit.

-e Print the link-level header on each line.

-F file

Read *expression* from the specified *file*.

-i interface

Listen on the specified *interface*. If not specified, **tcpdump** will listen on the lowest-numbered interface available, other than the loopback interface. Use **any** to listen to all available interfaces.

-l Line buffer standard out.

-L Print the data link types for an interface, then exit.

-n, -nn

Print IP addresses instead of converting them to hostnames. Use the second form to leave protocols and port numbers in numeric form, as well.

-N Print hostnames instead of fully qualified domain names.

-p Don't put the interface into promiscuous mode.

-q Abbreviate output, printing less protocol information.

-r file

Read packets from the specified *file*. (You can create such a file with the **-w** option.)

-s n

Read *n* bytes of data from each packet. (The default is 68.)

- S Print absolute TCP sequence numbers.
- T *n*
Read *n* bytes of data from each packet. (The default is 68.)
- t, -tt, -ttt, -tttt
Change display of timestamp. Use the first form to omit the timestamp from each line. Use the second form to print an unformatted timestamp. Use the third form to print the time in seconds between the current and the previous dump line. The final form prints the date before the timestamp on each dump line.
- u Print undecoded NFS handles.
- v, -vv, -vvv
Increase the verbosity of the printout. Each additional v increases the detail of the information printed.
- w *file*
Write the raw packet information to *file* without parsing or printing it. Specify - to write to standard output.
- W *n*
Wrap files creating a rotating buffer. Used with -C, this will set a limit on the number of files created. Upon reaching the limit, **tcpdump** will begin overwriting earlier files.
- x, -xx
Print packets in hex. Use the second form to print the packet's link level header in hex as well.
- X, -XX
Print packets in hex and ASCII text. Use the second form to print the packet's link level header in hex and ASCII as well.
- Z *user*
Drop root privileges and change to the specified user. Use the primary group of the specified user.

Examples

Place full packets into a file named *tcpdump.cap* for later analysis:

```
tcpdump -v -w tcpdump.cap -x -s 0
```

Read all packet headers received on the eth0 interface, except for arp and SSH packets:

```
tcpdump -i eth0 not arp and not port ssh
```

tcpslice

tcpslice [options] [*start* [*end*]] *files*

System administration command. Reads and manipulates packet capture files created by **tcpdump** -w. Based on timestamps, extract portions of or merge together *files*. Display all packets between the given *start* and *end* times. **tcpslice** understands most time and date formats. **tcpslice** also understands a relative time format specified as a unit of time—e.g., +1h10m to specify the first hour and ten minutes of packets in the specified *files*. This format is named

ymdhmsu after the letters it uses to denote units of time: years, months, days, hours, minutes, seconds, and microseconds. If no constraining dates are specified, the command will print out all packets contained in *files*.

Options

- d Print the start and end time of the specified range, then exit.
- D When merging files, don't discard duplicate packets.
- l Merge packets based on the time relative to the start of the file. The default is to merge based on the absolute timestamp.
- r Print the time and date of the first and last packet in each file, then exit.
- R Print the raw timestamp of the first and last packet in each file, then exit.
- t Print times associated with the first and last packet in each file in *ymdhmsu* format.
- w *file* Write output to *file* instead of standard output.

tee

`tee [options] files`

Accept output from another command and send it both to standard output and to *files* (like a T or fork in the road).

Options

- a, --append Append to *files*; do not overwrite.
- i, --ignore-interrupts Ignore interrupt signals.
- help Print a help message and then exit.
- version Print version information and then exit.

Example

`ls -l | tee savefile` View listing and save for later

telinit

`telinit [option] [runlevel]`

System administration command. Signal **init** to change the system's runlevel. **telinit** is actually just a link to **init**, the ancestor of all processes.

Option

- t *seconds* Send SIGKILL *seconds* after SIGTERM. Default is 20.

Runlevels

The default runlevels vary from distribution to distribution, but these are standard:

0 Halt the system.

1, s, S

Single user.

6 Reboot the system.

a, b, c

Process only entries in */etc/inittab* that are marked with runlevel **a**, **b**, or **c**.

q, Q

Reread */etc/inittab*.

Check the */etc/inittab* file for runlevels on your system.

telnet

telnet [options] [host [port]]

Access remote systems. **telnet** is the user interface that communicates with another host using the Telnet protocol. If **telnet** is invoked without *host*, it enters command mode, indicated by its prompt, **telnet>**, and accepts and executes commands. Type **?** at the command prompt to see the available commands. If invoked with arguments, **telnet** performs an **open** command (shown in the following list) with those arguments. *host* indicates the host's official name, alias, or Internet address. *port* indicates a port number (default is the Telnet port).

The Telnet protocol is often criticized because it uses no encryption and makes it easy for snoopers to pick up user passwords. Most sites now use **ssh** instead.

Options

-7 Request 7-bit operation.

-8 Request 8-bit operation.

-a Automatic login to the remote system.

-b hostalias

Use **bind** to bind the local socket to an aliased address or the address of an interface other than the one that would be chosen by **connect**.

-c Disable reading of the user's *.telnetrc* file.

-d Turn on socket-level debugging.

-e [escape_char]

Set initial **telnet** escape character to *escape_char*. If *escape_char* is omitted, no escape character is predefined.

-E Disable the escape character functionality.

-f With Kerberos V5 authentication, allow forwarding of the local credentials to the remote system.

telnetd

- F With Kerberos V5 authentication, allow local credentials to be forwarded to the remote system, including any that were already forwarded to the local environment.
- k *realm* With Kerberos authentication, obtain tickets for the remote host in *realm*, instead of in the remote host's realm.
- K Do not allow automatic login to the remote system.
- l *user* When connecting to remote system and if remote system understands ENVIRON, send *user* to the remote system as the value for variable USER. Implies the -a option.
- L Specify an 8-bit data path on output.
- n *tracefile* Open *tracefile* for recording the trace information.
- r Emulate rlogin. The default escape character for this mode is a tilde (~); an escape character followed by a dot causes telnet to disconnect from the remote host; a ^Z instead of a dot suspends telnet; and a ^] (the default telnet escape character) generates a normal telnet prompt. These codes are accepted only at the beginning of a line.
- x Turn on data-stream encryption if possible.
- X *atype* Disable the *atype* type of authentication.

telnetd

telnetd [*options*]

TCP/IP command. Telnet protocol server. **telnetd** is invoked by the Internet server for requests to connect to the Telnet port (port 23 by default). **telnetd** allocates a pseudo-terminal device for a client, thereby creating a login process that has the slave side of the pseudo-terminal serving as stdin, stdout, and stderr. **telnetd** manipulates the master side of the pseudo-terminal by implementing the Telnet protocol and by passing characters between the remote client and the login process.

The Telnet protocol is often criticized because it uses no encryption and makes it easy for snoopers to pick up user passwords. Most sites now use **ssh** instead.

Options

-a *type*

When compiled with authentication support, this option sets the authentication type. Accepted values are:

debug

Debug authentication code.

none

No authentication required, but accept it if offered. Use **login** for any further verification needed to access an account.

off
Disable authentication.

user
Allow only authenticated remote users with permission to access their accounts without giving a password.

valid
Allow only authenticated remote users. Use **login** for any additional verification needed to access an account.

-debug [port]
Start **telnetd** manually instead of through **inetd**. *port* may be specified as an alternate TCP port number on which to run **telnetd**.

-D modifier(s)
Debugging mode. This allows **telnet** to print out debugging information to the connection, enabling the user to see what telnet is doing. Several modifiers are available for the debugging mode:

- netdata**
Display data stream received by **telnetd**.
- options**
Print information about the negotiation of the Telnet options.
- ptydata**
Display data written to the pseudo-terminal device.
- report**
Print **options** information, as well as some additional information about what processing is going on.

-edebug
When compiled with support for encryption, enable encryption debugging code.

-h Don't print host-specific information until after login is complete.

-L path
Specify path to alternative login program. By default **telnetd** uses */bin/login*.

-n Disable checking for lost connections with TCP keep-alives.

-U Refuse connections from IP addresses with no reverse DNS information.

-X type
Disable authentication *type*.

test *test expression*
[*expression*]
Evaluate an *expression* and, if its value is true, return a zero exit status; otherwise, return a nonzero exit status. In shell scripts, you can use the alternate form [*expression*]. This command is generally used with conditional constructs in shell programs. Also exists as a built-in in most shells.

File testers

The syntax for all of these options is **test option file**. If the specified file does not exist, they return false. Otherwise, they test the file as specified in the option description.

- b Is the file block special?
- c Is the file character special?
- d Is the file a directory?
- e Does the file exist?
- f Is the file a regular file?
- g Does the file have the set-group-ID bit set?
- G Is the file owned by the process's effective group ID?
- k Does the file have the sticky bit set?
- L, -h Is the file a symbolic link?
- O Is the file owned by the process's effective user ID?
- p Is the file a named pipe?
- r Is the file readable by the current user?
- s Is the file nonempty?
- S Is the file a socket?
- t [*file-descriptor*] Is the file associated with *file-descriptor* (or 1, standard output, by default) connected to a terminal?
- u Does the file have the set-user-ID bit set?
- w Is the file writable by the current user?
- x Is the file executable?

File comparisons

The syntax for file comparisons is **test file1 option file2**. A string by itself, without options, returns true if it's at least one character long.

- ef Do the files have identical device and inode numbers?
- nt Is *file1* newer than *file2*? Check modification date, not creation date.
- ot Is *file1* older than *file2*? Check modification date, not creation date.

String tests

The syntax for string tests is **test option string** or **test string1 [!=] string2**.

- n Is the string at least 1 character long?
- z Is the string 0 characters long?

string1 = string2
Are the two strings equal?

string1 != string2
Are the strings unequal?

Expression tests

Note that an expression can consist of any of the previous tests.

(expression)

Is the expression true?

! expression

Is the expression false?

expression -a expression

Are the expressions both true?

expression -o expression

Is either expression true?

Integer tests

The syntax for integer tests is **test integer1 option integer2**. You may substitute **-l string** for an integer; this evaluates to *string*'s length.

-eq Are the two integers equal?

-ge Is *integer1* greater than or equal to *integer2*?

-gt Is *integer1* greater than *integer2*?

-le Is *integer1* less than or equal to *integer2*?

-lt Is *integer1* less than *integer2*?

-ne Are the two integers unequal?

time

time [options] command [arguments]

Run the specified command, passing it any *arguments*, and time the execution. Note that there is also a shell **time** command, so you might need to specify the full path, usually */usr/bin/time*, to run this version of **time**. **time** displays its results on standard error. The output includes elapsed time, user CPU time, system CPU time, and other information such as memory used and number of I/O operations. The output can be formatted using **printf** format strings specified with the **-f** option or the **TIME** environment variable.

Options

-- The end of the options. Anything after the **--** is treated as the command or one of its arguments.

-a, --append

Used with **-o** to append the output to *file* instead of overwriting it.

-f format, --format=format

Specify the output format. Overrides any format specified in the **TIME** environment variable.

--help

Print help message and exit.

-o file, --output=file

Send the output from **time** to the specified file instead of to standard error. If *file* exists, it is overwritten.

-p, --portability

Use portable output format (POSIX).

-v, --verbose

Give verbose output, providing all available information.

-V, --version

Print version information and exit.

Resources

The following resources can be specified in format strings:

- c** Number of involuntary context switches because of time slice expiring.
- C** Name and arguments of command being timed.
- D** Average size of unshared data area, in kilobytes.
- e** Elapsed real time, in seconds.
- E** Elapsed real time as *hours:minutes:seconds*.
- F** Number of major (I/O-requiring) page faults.
- I** Number of filesystem inputs.
- k** Number of signals delivered to the process.
- K** Average total (data+stack+text) memory use, in kilobytes.
- M** Maximum resident set size, in kilobytes.
- O** Number of filesystem outputs.
- p** Average unshared stack size, in kilobytes.
- P** Percent of CPU used.
- r** Number of socket messages received.
- R** Number of minor (recoverable) page faults.
- s** Number of socket messages sent.
- S** Total CPU seconds used by the system on behalf of the process.
- t** Average resident set size, in kilobytes.
- U** Total CPU seconds used directly by the process.
- w** Number of voluntary context switches.
- W** Number of times the process was swapped out of main memory.
- x** Exit status of the command.
- X** Average shared text size, in kilobytes.
- Z** System page size, in bytes.

Example

Time the execution of the command `ls -l` and display the user time, system time, and exit status of the command:

```
/usr/bin/time -f "\t%U user,\t%S system,\t%x status" ls -Fs
```

tload	<code>tload [options] [tty]</code>
	Display system load average in graph format. If <i>tty</i> is specified, print it to that terminal.
Options	
-d delay	Specify the delay, in seconds, between updates.
-s scale	Specify scale (number of characters between each graph tick). A smaller number results in a larger scale.
-V	Print version information and exit.

tmpwatch	<code>tmpwatch [options] hours directory</code>
	System administration command. Recursively remove regular files and directories in <i>directory</i> with access times older than <i>hours</i> . Specify the directory as an absolute path. This command is usually invoked by cron to remove old files in the <i>/tmp</i> directory.
Options	
-a, --all	Remove all file types.
-c, --ctime	Make decision on last inode change time for files and modification time for directories instead of access time.
-d, --nodirs	Do not remove directories.
-f, --force	Force removal of read-only files (similar to rm -f).
-l, --nosymlinks	Don't remove symbolic links.
-m, --mtime	Make decision on last modification time instead of access time.
-M, --dirmtime	Make directory deletion decisions based on modification time.
-q, --quiet	Report only fatal errors.
-s, --fuser	Before deleting, attempt to use fuser to see if a file is in use.
-t, --test	Verbosely test command, but don't actually remove files.
-u, --atime	Make decision on access time. (This is the default.)
-U user, --exclude-user user	Don't delete files owned by <i>user</i> , specified by name or user ID.

-v, --verbose

Print more details. Use two times to further increase the detail of the output.

-x, --exclude=path

Skip the specified *path*, the absolute path of a directory or file.

top**top [options]**

Provide information (frequently refreshed) about the most CPU-intensive processes currently running. You do not need to include a - before options. See **ps** for explanations of the field descriptors.

Options

-b Run in batch mode; don't accept command-line input. Useful for sending output to another command or to a file.

-c Show command line or program name in display. **-c** is a toggle; **top** starts with the last remembered setting.

-d delay

Specify delay between refreshes. Specify as *ss.tt* (seconds and tenths).

-f Add or remove fields or columns.

-h Print a help message and exit.

-H Display either all individual threads or a summary of all threads in process. **-H** is a toggle; **top** starts with the last remembered setting.

-i Suppress display of idle and zombie processes. **-i** is a toggle; **top** starts with the last remembered setting.

-n num

Update display *num* times, then exit.

-p pids

Monitor only processes with the specified process IDs.

-s Secure mode. Disable some (dangerous) interactive commands.

-S Cumulative mode toggle. Print total CPU time of each process, including dead child processes when on. **top** starts with the last remembered setting.

-u user

Monitor only processes with the specified effective UID or username.

-U user

Monitor only processes with the specified UID or username, matching real, effective, saved, and filesystem ids.

-v Print version information and exit.

Interactive commands

= Remove restrictions on which tasks are shown. Reverses the effect of an active **i** or **n** command.

space, Enter

Update display immediately.

<, >

Move the sort field. Use **<** to move one column left and **>** to move one column to the right.

- A** Toggle alternate display mode between a single window or multiple windows. See the following section, “Alternate display mode commands,” for the commands that work with **A**.
- b** Toggle between bold and reverse display. Only works with **x** and/or **y**.
- B** Globally toggle bold display.
- c** Toggle display of program name or full command line.

d, s

Change delay between refreshes. Prompt for new delay time, which should be in seconds. Suppressed in secure mode.

f Prompt to add fields to or remove fields from the display.**F, O**

Select sort field.

G Select another field group and make it current, or change by selecting a number from the following list:

- 1** Def
- 2** Job
- 3** Mem
- 4** Usr

h, ?

Display help about commands and the status of secure and cumulative modes.

H Toggle between displaying all individual threads and a summary of all threads in process.

I, 1

Toggle SMP view. Use **I** to toggle IRIX/Solaris mode (divide CPU usage by number of CPUs), **1** to toggle single/separate states.

k Prompt for process ID to kill, and signal to send (default is SIGTERM) to kill it.

i Toggle suppression of idle and zombie processes.

l Toggle display of load-average and uptime information.

m Toggle display of memory information.

n, #

Prompt for maximum number of processes to show. If **0** is entered, show as many as will fit on the screen (default).

o Prompt to change order of displayed fields.

q Exit.

- r** Apply `renice` to a process. Prompt for PID and `renice` value. Suppressed in secure mode.
- R** Toggle normal or reverse sort.
- S** Toggle cumulative mode. (See the `-S` option.)
- t** Toggle display of **processes** and **CPU states** lines.
- u** Prompt for user to show; matches on effective UID.
- U** Prompt for user to show; matches on real, effective, saved, and filesystem UID.
- W** Write current setup to `~/.toprc`. This is the recommended way to write a `top` configuration file.
- x** Toggle highlighting for sort field.
- y** Toggle highlights for running tasks.
- z** Toggle between color and mono display.
- Z** Globally change color mappings.

Alternate display mode commands

- = Rebalance tasks in the current window.
- + Rebalance tasks in every window.
- Show or hide the current window.
- _ Show all invisible windows or hide all visible windows.
- a Cycle forward through all four windows.
- g Change the name of the current window or group.
- w Cycle backward through all four windows.

Field descriptions

The first five entries in the following list describe the lines that appear at the top of the `top` display. The rest are the fields that can be displayed for each task (sizes are in kilobytes). Use the interactive `f` command to add or remove fields.

`top`

Display the time the system has been up, the number of users, and three load averages consisting of the average number of processes ready to run in the last 1, 5, and 15 minutes.

Tasks

The total number of processes running when the last update was taken, shown as the number of running, sleeping, stopped, or undead tasks.

`Cpu(s)`

The percentage of CPU time spent in user mode, in system mode, on tasks with a negative nice value, and idle.

`Mem`

Memory statistics, including total available memory, free memory, memory used, shared memory, and memory used for buffers.

Swap

Swapspace statistics, including total, available, used, and cached.

PID

Process ID.

PPID

Parent process ID.

UID

Effective user ID of task's owner.

USER

Effective username of task's owner.

RUSER

Real username of task's owner.

GROUP

The effective group name of task's owner.

PR

Priority.

NI

Nice value.

nFLT

Page fault count.

CODE

Code size.

DATA

Data plus stack size.

RES

Resident task size.

SWAP

Size of swapped-out portion of task.

VIRT

The total amount of virtual memory used by the task.

nDRT

Number of pages marked dirty.

#C

Last-used processor, for multiprocessor systems.

SHR

Amount of shared memory used.

S State of the task. Values are **S** (sleeping), **D** (uninterruptible sleep), **R** (running), **Z** (zombies), or **T** (stopped or traced).

WCHAN

Address or name of the kernel function in which the task is currently sleeping.

TIME

Total CPU time used by task and any children.

touch

TIME+

Like **TIME**, but shows the time down to hundredths of a second.

%CPU

Share of CPU time since last update, as percentage of total CPU time.

%MEM

Share of physical memory.

TTY

Controlling tty.

COMMAND

Command line (truncated if too long) or name of program depending on the state of the C toggle. Processes with no command line are shown in parentheses.

FLAGS

Task flags.

touch

touch [options] files

For one or more *files*, update the access time and modification time (and dates) to the current time and date. **touch** is useful in forcing other commands to handle files a certain way; for example, the operation of **make**, and sometimes **find**, relies on a file's access and modification time. If a file doesn't exist, **touch** creates it with a file size of 0.

Options

-a, --time=atime, --time=access, --time=use

Update only the access time.

-c, --no-create

Do not create any file that doesn't already exist.

-d time, --date=time

Change the time value to the specified *time* instead of the current time. *time* can use several formats and may contain month names, time zones, a.m. and p.m. strings, etc.

-m, --time=mtime, --time=modify

Update only the modification time.

-r file, --reference=file

Change times to be the same as those of the specified *file*, instead of the current time.

-t time

Use the time specified in *time* instead of the current time. This argument must be of the format [[cc]yy]mmddhhmm[.ss], indicating optional century and year, month, date, hours, minutes, and optional seconds.

--help

Print help message and then exit.

--version

Print the version number and then exit.

tr

tr [options] [string1 [string2]]

Translate characters. Copy standard input to standard output, substituting characters from *string1* to *string2*, or deleting characters in *string1*.

Options

-c, -C, --complement

Complement characters in *string1* with respect to ASCII 001-377.

-d, --delete

Delete characters in *string1* from output.

-s, --squeeze-repeats

Squeeze out repeated output characters in *string2*.

-t, --truncate-set1

Truncate *string1* to the length of *string2* before translating.

--help

Print help message and then exit.

--version

Print the version number and then exit.

Special characters

Include brackets ([]) where shown.

\a Ctrl-G (bell)

\b Ctrl-H (backspace)

\f Ctrl-L (form feed)

\n Ctrl-J (newline)

\r Ctrl-M (carriage return)

\t Ctrl-I (tab)

\v Ctrl-K (vertical tab)

\nnn

Character with octal value *nnn*

**** Literal backslash

char1-char2

All characters in the range *char1* through *char2*. If *char1* does not sort before *char2*, produce an error.

[char*]

In *string2*, expand *char* to the length of *string1*.

[char*number]

Expand *char* to *number* occurrences. [*x*4*] expands to **xxxx**, for instance.

[:class:]

Expand to all characters in *class*, where *class* can be:

alnum

Letters and digits

alpha

Letters

tracepath

blank	Whitespace
cntrl	Control characters
digit	Digits
graph	Printable characters except space
lower	Lowercase letters
print	Printable characters
punct	Punctuation
space	Whitespace (horizontal or vertical)
upper	Uppercase letters
xdigit	Hexadecimal digits
[=char=]	The class of characters to which <i>char</i> belongs.

Examples

Change uppercase to lowercase in a file:

```
cat file | tr 'A-Z' 'a-z'
```

Turn spaces into newlines (ASCII code 012):

```
tr ' ' '\012' < file
```

Strip blank lines from *file* and save in *new.file*:

```
cat file | tr -s "" "\012" > new.file
```

Delete colons from *file* and save result in *new.file*:

```
tr -d : < file > new.file
```

tracepath

tracepath [options] host [port]

TCP/IP command. Trace path to *host* and report the Maximum Transmission Unit (MTU). A simplified version of **traceroute** without options meant for use by unprivileged users. If specified, it will use *port* to send UDP probe packets. *host* is the destination hostname or the IP number of the host to reach.

Options

- n** Use alternative packet length of *n*. The default is 65536 for IPv4 and 128000 for IPv6.
- n** Don't look up host names, just print IP addresses.

traceroute	<code>traceroute [options] host [packetsize]</code>
	TCP/IP command. Trace route taken by packets to reach network host. traceroute attempts tracing by launching UDP probe packets with a small TTL (time-to-live), then listening for an ICMP “time exceeded” reply from a gateway. <i>host</i> is the destination hostname or the IP number of the host to reach. <i>packetsize</i> is the packet size in bytes of the probe datagram. Default is 40 bytes.
	Options
-4, -6	Force IPv4 or IPv6 tracerouting.
-A	Perform AS path lookups.
-d	Turn on socket-level debugging.
-e	Show ICMP extensions.
-f n	Set the initial time-to-live to <i>n</i> hops.
-F	Set the “don’t fragment” bit.
-g addr	Enable the IP LSRR (Loose Source Record Route) option in addition to the TTL tests, to ask how someone at IP address <i>addr</i> can reach a particular target.
-i interface	Specify the network interface for getting the source IP address for outgoing probe packets. Useful with a multihomed host. Also see the -s option.
-I	Use ICMP ECHO requests instead of UDP datagrams.
-m max_ttl	Set maximum time-to-live used in outgoing probe packets to <i>max_ttl</i> hops. Default is 30.
-n	Show numerical addresses; do not look up hostnames. (Useful if DNS is not functioning properly.)
-N n	Send <i>n</i> probe packets simultaneously. The default is 16.
-p port	Set base UDP port number used for probe packets to <i>port</i> . Default is (decimal) 33434.
-q n	Set number of probe packets per hop to the value <i>n</i> . Default is 3.
-r	Bypass normal routing tables and send directly to a host on an attached network.
-s src_addr	Use <i>src_addr</i> as the IP address that will serve as the source address in outgoing probe packets.

troff

- t *tos*
Set the type-of-service in probe packets to *tos* (default is 0).
The value must be a decimal integer in the range 0 to 255.
- T Use TCP SYN packets instead. This may help bypass some firewall rules.
- v Verbose; received ICMP packets (other than TIME_EXCEEDED and PORT_UNREACHABLE) will be listed.
- w *wait*
Set time to wait for a response to an outgoing probe packet to *wait* seconds (default is 5).
- x Toggle IP checksums, usually to turn them off. IP checksums are always calculated if -I is specified.
- z *msecs*
Set the delay between probes, in milliseconds. The default is 0.

troff

troff

See **groff**.

true

true

A null command that returns a successful (0) exit status. See also **false**.

tset

tset [*options*] [*terminal*]
reset [*options*] [*terminal*]

Initialize a terminal. The terminal to be initialized is whichever is found first from the value of *terminal*, the value of the TERM environment variable, or the default terminal type. See also the **reset** command.

Options

- c Set control characters.
- echar Set the erase character to *char*.
- ichar Set the interrupt character to *char*.
- I Do not send terminal or tab initialization strings to the terminal.
- kchar Set line-kill character to *char*.
- m *arg*
Specify a mapping from a port type to a terminal, where *arg* looks like this:
$$[\text{port_type}][\text{operator}][\text{baud_rate}][:]\text{terminal_type}$$
operator can be any combination of < (less than), > (greater than), @ (equal), and ! (not). The terminal type is a string (e.g., **vt100** or **xterm**).

- q Print the terminal type on standard output but do not initialize the terminal.
- Q Don't display values for the erase, interrupt, and line kill characters.
- r Print the terminal type to standard error.
- s Print the shell commands that initialize the TERM environment variable on standard output.
- V Print the version of **ncurses** used for this program and exit.

tsort**tsort** [*option*] [*file*]

Perform a topological sort on partially ordered strings in the specified file. If no file is specified or is -, read standard input. Multiple strings on a line are separated by spaces, where each line indicates a partial ordering. The fully ordered results are written to standard output. See the **tsort** info page for an example of the use of **tsort** for sorting lists of functions into the order they are called.

Options**--help**

Print help information and exit.

--version

Print version information and exit.

tty**tty** [*options*]

Print the filename of the terminal connected to standard input.

Options**--help**

Print help message and exit.

-s, --silent, --quiet

Print nothing to standard output, but return an exit status.

--version

Display version information and exit.

tune2fs**tune2fs** [*options*] *device*

System administration command. Tune the parameters of a Linux Second Extended Filesystem by adjusting various parameters. You must specify the *device* on which the filesystem resides; it must not be mounted read/write when you change its parameters.

Options**-c *max-mount-counts***

Specify the maximum number of mount counts between two checks on the filesystem.

-e behavior

Specify the kernel's behavior when encountering errors.
behavior must be one of:

continue

Continue as usual.

remount-ro

Remount the offending filesystem in read-only mode.

panic

Cause a kernel panic.

-f Force completion even if there are errors.**-g group**

Allow *group* (a group ID or name) to use reserved blocks.

-j Add an ext3 journal to the filesystem. If specified without **-J**, use the default journal parameters.**-J jrn1-options**

Specify ext3 journal parameters as a comma-separated list of *option=value* pairs. The specified options override the default values. Only one size or device option can be specified for a filesystem. Possible options are:

device=ext-jrn1

Attach to the journal block device on *ext-jrn1*, which must exist and must have the same block size as the filesystem to be journaled. *ext-jrn1* can be specified by its device name, by the volume label (**LABEL=label**), or by the Universal Unique Identifier (UUID) stored in the journal's ext2 superblock (**UUID=uuid**; see **uuidgen**). Create the external journal with:

mke2fs -0 jrn1-devext-jrn1

size=jrn1-size

The size of the journal in megabytes. The size must be at least equivalent to 1024 blocks and not more than 102,400 blocks.

-l Display a list of the superblock's contents.**-L label**

Specify the volume label of filesystem. The label must be no more than 16 characters.

-m percentage

Specify the percentage of blocks that will be reserved for use by privileged users.

-M dir

Specify the filesystem's last-mounted directory.

-o mount-options

Set or clear the specified default *mount-options*. Mount options specified in */etc/fstab* or on the command line for **mount** will override these defaults. Specify multiple options as

a comma-separated list. Prefixing an option with a caret (^) clears the option. No prefix or a plus sign (+) causes the option to be set. The following options can be cleared or set:

acl

Enable Posix Acess Control Lists.

bsdgroups

Assign new files the group-id of the directory in which they are created instead of the group-id of the process creating them.

debug

Enable debugging code.

journal_data

When journaling, commit all data to journal before writing to the filesystem.

journal_data_ordered

When journaling, force data to the filesystem before committing metadata to the journal.

journal_data_writeback

When journaling, force data to the filesystem after committing metadata to the journal.

-O option

Set or clear the specified filesystem options in the filesystem's superblock. Specify multiple options as a comma-separated list. Prefixing an option with a caret (^) clears the option. No prefix or a plus sign (+) causes the option to be set. Run **e2fsck** after changing **filetype** or **sparse_super**. The following options can be cleared or set:

dir_index

Use B-trees to speed up lookups on large directories.

filetype

Save file type information in directory entries.

has_journal

Create an ext3 journal. Same as the -j option.

sparse_super

Save space on large filesystems by limiting the number of backup superblocks. Same as -s.

-r num

Specify the number of blocks that will be reserved for use by privileged users.

-s [0|1]

Turn the sparse superblock feature on or off. Run **e2fsck** after changing this feature.

-u user

Allow *user* (a user ID or name) to use reserved blocks.

tunelp

-U *uuid*

Set the UUID of the filesystem to a UUID generated by **uuidgen** or to one of the following:

clear

Clear the existing UUID.

random

Randomly generate a new UUID.

time

Generate a new time-based UUID.

tunelp

tunelp *device* [*options*]

System administration command. Control a line printer's device parameters. Without options, print information about device(s).

Options

-a [on|off]

Specify whether or not to abort if the printer encounters an error. By default, do not abort.

-c *n*

Retry device *n* times if it refuses a character. (Default is 250.) After exhausting *n*, sleep before retrying.

-i *irq*

Use *irq* for specified parallel port. Ignore **-t** and **-c**. If 0, restore noninterrupt-driven (polling) action.

-o [on|off]

Specify whether to abort if device is not online or is out of paper.

-q [on|off]

Specify whether to print current IRQ setting.

-r

Reset port.

-s

Display printer's current status.

-t *time*

Specify a delay of *time* in jiffies to sleep before resending a refused character to the device. A jiffy is defined as either one tick of the system clock or one AC cycle time; it should be approximately 1/100 of a second.

-w *time*

Specify a delay of *time* in jiffies to sleep before resending a strobe signal.

ul

ul [*options*] [*filenames*]

Translate underscores to underlining. The process will vary by terminal type. Some terminals are unable to handle underlining.

Options

- i When on a separate line, translate - to underline instead of translating underscores.
- t *terminal-type*
Specify terminal type. By default, TERM is consulted.

umount

umount [options] [directory]

System administration command. Unmount filesystem specified by directory. You may also specify the filesystem by device name. **umount** announces to the system that the removable file structure previously mounted on the specified directory is to be removed. Any pending I/O for the filesystem is completed, and the file structure is flagged as clean. A busy filesystem cannot be unmounted.

Options

- a Unmount all filesystems listed in */etc/mtab* other than */proc*.
- d If the unmounted device was a loop device, free the loop device too. See also the **losetup** command.
- f Force the umount. This option requires kernel 2.1.116 or later.
- h Print help message and exit.
- i Don't execute */sbin/umount.<filesystem>* helper programs.
- l Lazy umount. Detach the filesystem from the hierarchy immediately, but don't clean up references until it is no longer busy. Requires kernel 2.4.11 or later.
- n Unmount, but do not record changes in */etc/mtab*.
- O *options*
Unmount only filesystems with the specified options in */etc/fstab*. Specify multiple options as a comma-separated list. Add **no** as a prefix to an option to indicate filesystems that should not be unmounted.
- r If unmounting fails, try to remount read-only.
- t *type*
Unmount only filesystems of type *type*. Multiple types can be specified as a comma-separated list, and any type can be prefixed with **no** to specify that filesystems of that type should not be unmounted.
- v Verbose mode.
- V Print version information and exit.

uname

uname [options]

Print information about the machine and operating system. Without options, print the name of the kernel (Linux).

unexpand

Options

-a, --all

Combine all the system information from the other options.

-i, --hardware-platform

Print the system's hardware platform.

-m, --machine

Print the name of the hardware that the system is running on.

-n, --nodename

Print the machine's hostname.

-o, --operating-system

Print the operating system name.

-p, --processor

Print the type of processor.

-r, --kernel-release

Print the release number of the kernel.

-s, --kernel-name

Print the name of the kernel (Linux). This is the default action.

-v, --kernel-version

Print build information about the kernel.

--help

Display a help message and then exit.

--version

Print version information and then exit.

unexpand

unexpand [options] [files]

Convert strings of initial whitespace, consisting of at least two spaces and/or tabs, to tabs. Read from standard input if given no file or a file named -.

Options

-a, --all

Convert all, not just leading, strings of spaces and tabs.

--first-only

Convert only leading spaces and tabs. Overrides -a.

-t *nums*, --tabs *nums*

nums is a comma-separated list of integers that specify the placement of tab stops. If a single integer is provided, the tab stops are set to every *integer* spaces. By default, tab stops are eight spaces apart. This option implies -a.

--help

Print help message and then exit.

--version

Print the version number and then exit.

unicode_start `unicode_start [font [umap]]`
 Put keyboard and console in Unicode mode, setting the font to *font* and the Unicode map to *umap* if the font doesn't have its own map. If no font is specified, use the default.

unicode_stop `unicode_stop`
 Take keyboard and console out of Unicode mode.

uniq `uniq [options] [file1 [file2]]`
 Remove duplicate adjacent lines from sorted *file1* or from standard input, sending one copy of each line to *file2* (or to standard output). Often used as a filter. Specify only one of **-d** or **-u**. See also **comm** and **sort**.

Options

- c, --count**
 Print each line once, prefixing number of instances.
- d, --repeated**
 Print duplicate lines once but no unique lines.
- D, --all-repeated[=method]**
 Print all duplicate lines. **-D** takes no delimiter method. The delimiter method *method* takes one of the following values: **none** (default), **prepend**, or **separate**. Blank lines are used as the delimiter.
- f n, --skip-fields=n**
 Ignore first *n* fields of a line. Fields are separated by spaces or by tabs.
- i, --ignore-case**
 Ignore case differences when checking for duplicates.
- s n, --skip-chars=n**
 Ignore first *n* characters of a field.
- u, --unique**
 Print only unique lines (no copy of duplicate entries is kept).
- w n, --check-chars=n**
 Compare only first *n* characters per line (beginning after skipped fields and characters).
- help**
 Print a help message and then exit.
- version**
 Print version information and then exit.

Examples

Send one copy of each line from **list** to output file **list.new**:

```
uniq list list.new
```

Show which names appear more than once:

```
sort names | uniq -d
```

unlink

unlink	<i>unlink filename</i> <i>unlink option</i>
Remove the specified file using the system unlink function.	

Options

--help

Print help information and exit.

--version

Print version information and exit.

uptime	<i>uptime [option]</i>
---------------	------------------------

Print the current time, how long the system has been running, the number of users currently logged in (which may include the same user multiple times), and system load averages. This output is also produced by the first line of **w** command output.

Option

-V Print version information and exit.

useradd	<i>useradd [options] [user]</i>
----------------	---------------------------------

System administration command. Create new user accounts or update default account information. Unless invoked with the **-D** option, *user* must be given. **useradd** will create new entries in system files. Home directories and initial files may also be created as needed.

Options

-b dir, --base-dir dir

Specify base *dir* for home directories. The default is */home*.

-c comment, --comment comment

Comment field.

-d dir, --home dir

Home directory. The default is to use *user* as the directory name under the *home* directory specified with the **-D** option.

-D [options]

Set or display defaults. If *options* are specified, set them. If no options are specified, display current defaults. The options are:

-b dir, --base-dir dir

Home directory prefix to be used in creating home directories. If the **-d** option is not used when creating an account, the *user* name is appended to *dir*.

-e date, --expiredate date

Expire *date*. Requires the use of shadow passwords.

-f days, --inactive days

Number of *days* after a password expires to disable an account. Requires the use of shadow passwords.

- g group, --gid group**
Initial *group* name or ID number.
- s shell, --shell shell**
Default login *shell*.
- e date, --expiredate date**
Account expiration *date*. Use the format *MM/DD/YYYY*. Two-digit year fields are also accepted. The value is stored as the number of days since January 1, 1970. This option requires the use of shadow passwords.
- f days, --inactive days**
Permanently disable account this many *days* after the password has expired. A value of **-1** disables this feature. This option requires the use of shadow passwords.
- g group, --gid group**
Initial *group* name or ID number. If a different default group has not been specified using the **-D** option, the default group is 1.
- G groups, --groups groups**
Supplementary *groups* given by name or number in a comma-separated list with no whitespace.
- k [dir], --skel [dir]**
Copy default files to the user's home directory. Meaningful only when used with the **-m** option. Default files are copied from */etc/skel/* unless an alternate *dir* is specified.
- K key=value, --key key=value**
Override */etc/login.defs* defaults. This option can be given multiple times.
- l** Keep old entries for *user* in lastlog and faillog databases. By default old user data in these are reset.
- m, --create-home**
Make user's home directory if it does not exist. The default is not to make the home directory.
- M** Do not create a home directory for the user, even if the system default in */etc/login.defs* is to create one.
- o, --non-unique**
Override. Accept a nonunique *uid* with the **-u** option. (Probably a bad idea.)
- p passwd, --password passwd**
The encrypted password, as returned by **crypt(3)**.
- r, --system**
Red Hat-specific option. Create a system account with a non-expiring password and a UID lower than the minimum defined in */etc/login.defs*. Do not create a home directory for the account unless **-m** is also specified.
- s shell, --shell shell**
Login *shell*.

userdel**-u uid, --uid uid**

Numerical user ID. The value must be unique unless the **-o** option is used. The default value is the smallest ID value greater than 99 and greater than every other *uid*.

userdel**userdel [option] user**

System administration command. Delete all entries for *user* in system account files.

Option**-f, --force**

Remove the *user* even if they are currently logged in. Remove home directory and mail spool even if they are used by another user. Remove group too, if USERGROUPS_ENAB is set to yes in */etc/login.defs*.

-r, --remove

Remove the home directory of *user* and any files contained in it.

usermod**usermod [options] user**

System administration command. Modify *user* account information.

Options**-a, --append**

Used with the **-G** option. Add *user* to the specified *groups*, but don't remove *user* from groups not in the current list.

-c comment, --comment comment

Comment field.

-d dir, --home dir

Home directory.

-e date, --expiredate date

Account expiration *date*. *date* is in the format MM/DD/YYYY; two-digit year fields are also accepted. The value is stored as the number of days since January 1, 1970. This option requires the use of shadow passwords.

-f days, --inactive days

Permanently disable account this many *days* after the password has expired. A value of **-1** disables this feature. This option requires the use of shadow passwords.

-g group, --gid group

Initial *group* name or number.

-G groups, --groups groups

Supplementary *groups* given by name or number in a comma-separated list with no whitespace. *user* will be removed from any groups to which it currently belongs that are not included in *groups*.

-l name, --login name

Login *name*. This cannot be changed while the user is logged in.

-L, --lock

Lock user's password by putting a ! in front of it. This option cannot be used with **-p** or **-U**.

-o, --non-unique

Override. Accept a nonunique *uid* with the **-u** option.

-p pw, --password pw

Encrypted password, as returned from `crypt(3)`.

-s shell, --shell shell

Login *shell*.

-u uid, --uid uid

Numerical user ID. The value must be unique unless the **-o** option is used. Any files owned by *user* in the user's home directory will have their user ID changed automatically. Files outside of the home directory will not be changed. *user* should not be executing any processes while this is changed.

-U, --unlock

Unlock the user's password by removing the ! that **-L** put in front of it. This option cannot be used with **-p** or **-L**.

users

`users [file]`

`users option`

Print a space-separated list of each login session on the host. Note that this may include the same user multiple times. Consult *file* or, by default, `/var/log/utmp` or `/var/log/wtmp`.

Options**--help**

Print usage information and exit.

--version

Print version information and exit.

usleep

`usleep [microseconds]`

`usleep [option]`

Sleep some number of microseconds (default is 1).

Options**-?, --help**

Print help information and then exit.

--usage

Print brief usage message and then exit.

-v, --version

Print version information.

uuidgen	<code>uuidgen [option]</code> Create a new Universal Unique Identifier (UUID) and print it to standard output. The generated UUID consists of five hyphen-separated groups of hex digits (e.g., 3cdfc61d-87d3-41b5-ba50-32870b33dc67). The default is to generate a random-based UUID, but this requires that a high-quality random-number generator be available on the system.
	Options -r Generate a random-based UUID. -t Generate a time-based UUID.
vdir	<code>vdir [options] [files]</code> Verbosely list directory contents. Equivalent to ls -lb . By default, list the current directory. Directory entries are sorted alphabetically unless overridden by an option. vdir takes the same options as ls .
vi	<code>vi [options] [files]</code> A screen-oriented text editor based on ex . vi is bi-modal, with a command mode and an insert mode. For more information on vi , see Chapter 9.
vidmode	<code>vidmode [option] image [mode [offset]]</code> System administration command. Set the video mode for a kernel <i>image</i> . If no arguments are specified, print current <i>mode</i> value. <i>mode</i> is a 1-byte value located at offset 506 in a kernel image. You may change the <i>mode</i> by specifying the kernel <i>image</i> to change, the new <i>mode</i> , and the byte offset at which to place the new information (the default is 506). Note that rdev -v is a synonym for vidmode . If LILO is used, vidmode is not needed. The video mode can be set from the LILO prompt during a boot.
	Modes -3 Prompt -2 Extended VGA -1 Normal VGA 0 Same as entering 0 at the prompt 1 Same as entering 1 at the prompt 2 Same as entering 2 at the prompt 3 Same as entering 3 at the prompt n Same as entering n at the prompt
	Option -o offset Same as specifying an <i>offset</i> as an argument.

vim**vim**

An enhanced version of the **vi** screen editor. Both **vi** and **vim** are covered in Chapter 9.

vmstat**vmstat [options] [interval [count]]**

System administration command. Print report on virtual memory statistics, including information on processes, memory, paging block I/O, traps, system and CPU usage. **vmstat** initially reports average values since the last system reboot. If given a sampling period *interval* in seconds, it prints additional statistics for each interval. If specified, **vmstat** exits when it has completed *count* reports. Otherwise, it continues until it receives a Ctrl-C, printing a new header line each time it fills the screen.

Options

- a** Display active and inactive memory.
- d** Display disk statistics.
- f** Display the number of forks since the system was booted.
- m** Display the names and sizes of various kernel objects stored in a cache known as the slab layer. Also see the **slabtop** command.
- n** Don't print new header lines when the screen is full.
- p partition** Display detailed statistics for the specified partition.
- s** Display various event counters and memory statistics.
- S units** Switch the output units. Possible values are **k**, **K**, **m**, or **M**.
- t** Add a timestamp to output.
- V** Print version number, then exit.

VM mode fields**procs**

- r** Processes waiting for runtime.
- b** Uninterruptible sleeping processes.

memory**swpd**

Virtual memory used, in kilobytes.

free

Idle memory, in kilobytes.

buff

Memory used as buffers, in kilobytes.

cache

Cache memory, in kilobytes.

inactive

Inactive memory, in kilobytes, displayed with **-a**.

active

Active memory, in kilobytes; displayed with **-a**.

swap

si Memory swapped in from disk each second, in kilobytes.

so Memory swapped out to disk each second, in kilobytes.

io

bi Blocks received from block devices each second.

bo Blocks sent to block devices each second.

system

in Interrupts per second, including clock interrupts.

cs Context switches per second.

cpu

us Percentage of CPU time consumed by user processes.

sy Percentage of CPU time consumed by system processes.

id Percentage of CPU time spent idle.

wa Percentage of CPU time spent waiting for I/O.

Disk mode fields**Reads and Writes****total**

Total reads or writes completed successfully.

merged

Reads or writes grouped into one I/O.

sectors

Sectors read or written successfully.

ms

Milliseconds spent reading or writing.

IO

cur I/O in progress

s Seconds spent doing I/O.

Disk partition mode fields**reads**

Total reads issued to this partition.

read sectors

Total sectors read for this partition.

writes

Total writes issued to this partition.

requested writes

Total write requests for this partition.

Slab mode fields

cache

Cache name.

num

Number of currently active objects.

total

Total number of available objects.

size

Size of each object.

pages

Number of pages with at least one active object.

totpages

Total number of allocated pages.

pslab

Number of pages per slab.

volname

`volname [devfile]`

Return the volume name for a device such as a CD-ROM that was formatted with an ISO-9660 filesystem. The default device file *devfile* is `/dev/cdrom`.

w

`w [options] [user]`

Print summaries of system usage, currently logged-in users, and what those users are doing. **w** is essentially a combination of **uptime**, **who**, and **ps -a**. Display output for one user by specifying *user*.

Options

- f Toggle printing the from (remote hostname) field.
- h Suppress heading and **uptime** information.
- s Use the short format.
- u Ignore the username while figuring out the current process and CPU times.
- V Display version information.

File

`/var/run/utmp`

List of users currently logged in.

wall

`wall [file]`

`wall [-n] [message]`

Write to all users. Depending on your Linux distribution, **wall** uses one of the two syntaxes shown. In both versions, the default is for **wall** to read a message from standard input and send the message to all users currently logged in, preceded by “Broadcast Message from....” With the first syntax, which comes with Debian-based systems, for example, if *file* is specified, **wall** reads

warnquota

input from that file rather than from standard input, and only the superuser can write to a terminal if the user has disallowed messages. With the second syntax, distributed with Red Hat-based systems, for example, the text of the message can be included on the command line, and the message is limited to 20 lines. In this form, if **-n** is specified, the default banner message is replaced with “Remote broadcast message.” **-n** can only be specified by the superuser, and only if **wall** was installed set-group-id.

Example

Send the message contained in the file *message.txt* to all users:

```
$ wall < message.txt
```

warnquota

warnquota [options] [filesystem]

System administration command. Mail warning messages to users that have exceeded their soft limit.

Options

-a file, --admins-file=file

Read group administrator information from *file* instead of */etc/quotagrpadmins*.

-c file, --config=file

Read configuration information from *file* instead of */etc/warnquota.conf*.

-d, --no-details

Send messages without attaching quota reports.

-F format, --format=format

Read quota information of the specified format. (See **quota** for valid formats.)

-g, --group

Send messages for group quotas. Send the message to the user specified in */etc/quotagrpadmins*.

-i, --no-autofs

Ignore automount mount points.

-q file, --quota-tab=file

Read device description strings from *file* instead of */etc/quotagrpadmins*.

-s, --human-readable

Report sizes in more human-readable units.

-u, --user

Send messages for user quotas. (This is the default.)

watch

watch [options] command [cmd_options]

Run the specified command repeatedly (by default, every two seconds) and display the output so you can watch it change over time. The command and any options are passed to **sh -c**, so you may need to use quotes to get correct results.

Options

-d, --differences[=cumulative]

Highlight changes between iterations. If **cumulative** is specified, the highlighting remains on the screen throughout, giving a cumulative picture of the changes.

-h, --help

Display help message and exit.

-n secs, --interval=secs

Run the command every *secs* seconds.

-t, --no-title

Do not display the header or the blank line following the header.

-v, --version

Print version information and exit.

wc

`wc [options] [files]`

Print byte, word, and line counts for each file. Print a total line for multiple *files*. If *files* is omitted or is `-`, read standard input. See other examples under **ls** and **sort**.

Options

-c, --bytes

Print byte count only.

-l, --lines

Print line count only.

-L, --max-line-length

Print length of longest line.

-m, --chars

Print character count only.

-w, --words

Print word count only.

--help

Print help message and then exit.

--version

Print the version number and then exit.

Examples

Count the number of users logged in:

```
who | wc -l
```

Count the words in three essay files:

```
wc -w essay.[123]
```

Count lines in the file named by variable \$file (don't display filename):

```
wc -l < $file
```

wget **wget [options] [urls]**

Perform noninteractive file downloads from the Web. **wget** works in the background and can be used to set up and run a download without the user having to remain logged on. **wget** supports HTTP, HTTPS, and FTP, as well as downloads through HTTP proxies. **wget** uses a global startup file that you may find at */etc/wgetrc* or */usr/local/etc/wgetrc*. In addition, users can define their own *\$HOME/.wgetrc* files.

Options**-4, --inet4-only**

Force connection to IPv4 hosts only.

-6, --inet6-only

Force connection to IPv6 hosts only.

-a *logfile*, --append-output=*logfile*

Append output messages to *logfile*, instead of overwriting the contents as **-o** does. If *logfile* doesn't exist, create it.

-A *acclist*, --accept=*acclist*

Specify a comma-separated list of filename suffixes or patterns to accept.

-b, --background

Go into the background immediately after startup, writing output to the file specified with **-o** or to **wget-log**.

-B *url*, --base=*url*

Used with **-F** to prepend the specified URL to relative links in the input file specified with **-i**.

--bind-address=*address*

When making client TCP/IP connections, **bind()** to the specified local address, which can be specified as a hostname or IP address. Useful if your system is bound to multiple IP addresses.

-c, --continue

Continue getting a partially downloaded file. Affects the restarting of downloads from an earlier invocation of **wget**. Works only with FTP servers and HTTP servers that support the Range header.

--connect-timeout=*seconds*

Set the timeout for a connection to be established in seconds. The default is never to time out, unless a timeout is implemented by system libraries.

--cut-dirs=*num*

Ignore the specified number of directory components when creating the local directory structure.

-d, --debug

Turn on debugging. **wget** must have been compiled with debug support.

- D *domainlist*, --domains=*domainlist***
Specify a comma-separated list of domains to be followed.
Does not turn on **-H**.
- delete-after**
Delete each retrieved file from the local machine after downloading it. Useful for prefetching pages through a proxy. **-k** is ignored if specified with **--delete-after**.
- dns-timeout=*seconds***
Set the DNS lookup timeout to *seconds*. The default is to never time out.
- e *command*, --execute=*command***
Execute the specified command after the commands in *.wgetrc*, overriding any *.wgetrc* commands. Can be included multiple times, once for each command to execute.
- E, --html-extension**
Append the suffix *.html* to the filenames of downloaded files where the URL does not include it (for example, an *.asp* file).
- exclude-domains=*domainlist***
Specify a comma-separated list of names that are never to be followed.
- F, --force-html**
When reading input from a file, force the file to be treated as an HTML file.
- follow-ftp**
Follow FTP links from HTML documents. The default is to ignore FTP links.
- follow-tags=*list***
Specify a comma-separated list of tags to be considered, overriding the internal table that **wget** normally uses during a recursive retrieval.
- ftp-user, --ftp-password**
Specify the user name and password on an FTP server.
- h, --help**
Display usage information and exit.
- H, --span-hosts**
Enable spanning across hosts when doing recursive retrieval.
- header=*header***
Add an additional header to be passed to the HTTP server. The header must include a colon (:) preceded by at least one nonblank character, and with no newline characters. Can be specified multiple times. If *header* is an empty string, all user-defined headers are cleared.
- http-user=*user*, --http-password=*password***
Specify the username and password on an HTTP server.
- i *file*, --input-file=*file***
Read URLs from the specified file. URLs specified on the command line are accessed before URLs in the file. If *file* is given as **-**, read from standard input.

`wget`

-I list, --include-directories=list

Specify a comma-separated list of directories to follow when downloading. The list elements may contain wildcards.

--ignore-case

Ignore case when matching files and directories. Affects the behavior of **-A**, **-I**, **-R**, **-X**, and FTP globbing.

--ignore-length

Ignore the “Content-Length” header on the HTTP server.

--ignore-tags=list

Specify a comma-separated list of tags to be ignored for recursive retrievals.

-k, --convert-links

Convert document links after the download is complete so they work locally.

-K, --backup-converted

When converting a file, back up the original and add an **.orig** suffix. Affects the behavior of **-N**.

--keep-session-cookies

Causes **--save-cookies** to also save session cookies.

-l depth, --level=depth

For recursive retrievals, specify the maximum recursion depth. The default depth is 5.

-L, --relative

Follow relative links only.

--limit-rate=rate

Set the maximum download speed. The default is to specify the rate in bytes, or add a **k** suffix for kilobytes or **m** for megabytes.

--load-cookies=file

Load cookies from the specified file before the first HTTP retrieval.

-m, --mirror

Turn on options suitable for mirroring a remote site. Equivalent to **-r -N -l inf --no-remove-listing**.

--max-redirect=num

Set the maximum number of redirections to follow (default is 20).

-N, --timestamping

Turn on timestamping.

-nc, --no-clobber

Do not download a file if there is already a copy on the disk. The default is to preserve the original copy and rename successive downloads, adding **.1**, **.2**, etc. to their name. May not be specified with **-N**.

-nd, --no-directories

Do not create a directory hierarchy when doing recursive retrievals.

-nH, --no-host-directories

Disable creation of directories prefixed by the name of the host. The default is to include the hostname.

--no-cache

Disable server-side cache for an HTTP retrieval. The default is for caching to be on.

--no-cookies

Disable the use of cookies.

--no-dns-cache

Disable caching of DNS lookups; look up hostname again for each new connection.

--no-glob

Turn off FTP globbing to prevent the use of wildcards for multiple file retrievals.

--no-http-keep-alive

Turn off the keep-alive feature for HTTP retrievals.

--np, --no-parent

In recursive retrievals, do not ever go up to the parent directory.

--no-passive-ftp

Do not allow use of passive FTP transfer mode.

--no-proxy

Do not use proxies.

--no-remove-listing

Do not remove the temporary *.listing* files generated by FTP retrievals.

-nv, --non-verbose

Turn off verbose mode, but don't run completely quietly.
Displays error messages and basic information.

-o *logfile*, --output-file=*logfile*

Log output messages to *logfile*, instead of the default standard error.

-O *file*, --output-document=*file*

Concatenate all documents into the specified file. If the file exists, it is overwritten. Specify the file as - to write to standard output.

-p, --page-requisites

Download all files necessary to display an HTML page.

-P *prefix*, --directory-prefix=*prefix*

Set the directory prefix to the specified value.

--post-data=*string*, --post-file=*file*

Use POST as the method for HTTP requests and send the specified data in the request body. Use **--post-data** to send *string* as data and **--post-file** to send the *file* contents.

--prefer-family=*family*

Connect to the addresses of the specified family first when there is a choice. The possible values for *family* are **IPv4** (the default), **IPv6**, and **none**.

```
wget
```

--progress=type[:style]

Set the progress indicator to *type*. Valid types are **dot** and **bar**; the default is **bar**. With **--progress=dot**, you can also set a style. The default style is for each dot to represent 1 KB, with 10 dots in a cluster and 50 dots per line. Alternatives are **binary**, with each dot representing 8 KB, 16-dot clusters, and 48 dots per line; **mega**, for downloading very large files, with each dot representing 64 KB, 8 dots per cluster, and 48 dots per line; and **giga**, with each dot representing 1 MB, 8 dots per cluster, and 4 clusters per line.

--protocol-directories

Use the protocol name as part of the local filename.

--proxy-user=user, --proxy-passwd=password

Specify the username and password for authentication on a proxy server.

-q, --quiet

Run quietly; don't produce output.

-Q quota, --quota=quota

Specify download quota for automatic retrievals. The default value is in bytes; add **k** suffix for kilobytes, or **m** for megabytes.

-r, --recursive

Turn on recursive retrieving.

-R rejlist, --reject=rejlist

Specify a comma-separated list of filename suffixes or patterns to reject.

--random-wait

Set a random wait time to prevent being identified by websites that look for patterns in time between requests so they can block access.

--read-timeout=seconds

Set the read (and write) timeout to the specified number of seconds. The default is 900 seconds.

--referer=url

Include a “Referer: url” header in an HTTP request.

--restrict-file-names=mode[,nocontrol]

Restrict the characters found in remote URLs from appearing in local filenames. The value of *mode* is the operating system—e.g., **unix** or **windows** (use **unix** for Linux). Such characters are escaped with a percent sign (%). The default is to escape characters not valid on your operating system. Appending **,nocontrol** turns off escaping of control characters.

--retr-symlinks

When retrieving FTP directories recursively, follow symbolic links and retrieve the linked-to files.

--retry-connrefused

Retry after getting a “connection refused” error. Useful for mirroring unreliable sites whose servers are likely to go down briefly.

- S, --server-response**
Print HTTP server headers and FTP server responses.
- save-cookies=*file***
Save cookies in the specified file before exiting. Does not save expired cookies, and only saves session cookies if **--keep-session-cookies** is also specified.
- save-headers**
Save the headers sent by an HTTP server to the file, preceding the contents and separated by a blank line.
- spider**
Behave like a web spider, checking that pages exist but not downloading them.
- strict-comments**
Turn on strict parsing of HTML comments, instead of terminating comments at the first occurrence of -->.
- t *num*, --tries=*num***
Set the number of retries to the specified value of *num*. Set *num* to 0 or **inf** to keep trying forever (infinitely) (default is 20 retries), unless there is a fatal error such as “connection refused.”
- T *seconds*, --timeout=*seconds***
Set network timeout to the specified number of seconds. Equivalent to specifying all of **--dns-timeout**, **--connect-timeout**, and **--read-timeout**.
- user=*user*, --password=*password***
Specify the user name and password for both FTP and HTTP file retrieval.
- U *agent*, --user-agent=*agent***
Specify an agent string to the HTTP server to replace the default identification of Wget/*version*, where *version* is the current wget version. This string is used in the User-Agent header field.
- v, --verbose**
Turn on verbose output, printing all available data. This is the default.
- V, --version**
Display version information and exit.
- w *seconds*, --wait=*seconds***
Specify the wait in seconds between retrievals. Used to lighten server load. Use the suffix **m** to specify the wait in minutes, **h** for hours, or **d** for days.
- waitretry=*seconds***
Specify the number of seconds to wait between retries if the download fails. The default in the global configuration file is not to wait.
- x, --force-directories**
Create a hierarchy of directories even if one wouldn’t otherwise be created.

whatis

-X list, --exclude-directories=list

Specify a comma-separated list of directories to exclude from download. List elements may contain wildcards.

whatis

whatis keywords

Search the short manual page descriptions in the **whatis** database for each *keyword* and print a one-line description to standard output for each match. Like **apropos**, except that it searches only for complete words. Equivalent to **man -f**.

whereis

whereis [options] files

Locate the binary, source, and manual page files for specified commands/files. The supplied filenames are first stripped of leading pathname components and any (single) trailing extension of the form *.ext* (for example, *.c*). Prefixes of *s*, resulting from use of source code control are also dealt with. **whereis** then attempts to locate the desired program in a list of standard Linux directories (*/bin*, */etc*, */usr/bin*, */usr/local/bin*, etc.).

Options

-b Search only for binaries.

-B *directories*

Change or otherwise limit the directories to search for binaries.

-f Terminate the last directory list and signal the start of filenames. Required when the **-B**, **-M**, or **-S** option is used.

-m Search only for manual sections.

-M *directory*

Change or otherwise limit the directories to search for manual sections.

-s Search only for sources.

-S *directory*

Change or otherwise limit the directories to search for sources.

-u Search for unusual entries—that is, files that do not have one entry of each requested type. Thus, the command **whereis -m -u *** asks for those files in the current directory that have no documentation.

Example

Find all files in */usr/bin* that are not documented in */usr/share/man/man1* but that have source in */usr/src*:

```
$ cd /usr/bin  
$ whereis -u -M /usr/share/man/man1 -S /usr/src -f *
```

which**which** [options] [--] [commands]

List the full pathnames of the files that would be executed if the named *commands* had been run. **which** searches the user's \$PATH environment variable.

Options**-a, --all**

Print all matches, not just the first.

-i, --read-alias

Read aliases from standard input and write matches to standard output. Useful for using an alias for **which**.

--read-functions

Read shell functions from standard input and report matches to standard output. Useful for also using a shell function for **which** itself.

--skip-alias

Ignore **--read-alias** if present. Useful for finding normal binaries while using **--read-alias** in an alias for **which**.

--show-dot

If a matching command is found in a directory that starts with a dot, print *./cmdname* instead of the full pathname.

--show-tilde

Print a tilde (~) to indicate the user's home directory. Ignored if the user is root.

--skip-dot

Skip directories that start with a dot.

--skip-functions

Ignore **--read-functions** if present. Useful when searching for normal binaries while using **--read-functions** in an alias or function for **which**.

--skip-tilde

Skip directories that start with a tilde (~) and executables in \$HOME.

--tty-only

Stop processing options on the right if not on a terminal.

-v, -V, --version

Print version information and then exit.

--help

Print help information and then exit.

Example

```
$ which cc ls
/usr/bin/cc
ls:      aliased to ls -sFC
```

who	<code>who [options] [file]</code>
	<code>who am i</code>
	Show who is logged into the system. With no options, list the names of users currently logged in, their terminal, the time they have been logged in, and the name of the host from which they have logged in. An optional system <i>file</i> (default is <i>/etc/utmp</i>) can be supplied to give additional information.
	Options
-a, --all	Equivalent to -b -d --login -p -r -t -T -u .
am i	Print information for the invoking user.
-b, --boot	Print time of last system boot.
-d, --dead	Print a list of dead processes.
-H, --heading	Print column headings.
--help	Print a help message and then exit.
-l, --login	Print list of system login processes.
--lookup	Attempt to include canonical hostnames via DNS.
-m	Same as who am i .
-p, --process	Print active processes spawned by init .
-q, --count	“Quick.” Display all usernames and the total number of users.
-r, --runlevel	Print the current runlevel.
-s, --short	Print only name, line, and time. This is the default behavior.
-t, --time	Print the last system clock change.
-u, --users	Print a list of the users who are logged in.
--version	Print version information and then exit.
-w, -T, --mesg, --message, --writable	Include user’s message status in the output: <ul style="list-style-type: none">+ mesg y (write messages allowed)- mesg n (write messages refused)? Cannot find terminal device

Example

This sample output was produced at 8 a.m. on April 17:

```
$ who -uH
  NAME    LINE    TIME      IDLE   PID  COMMENTS
  Earvin  ttyp3  Apr 16 08:14 16:25  2240
  Larry   ttyp0  Apr 17 07:33 . 15182
```

Since Earvin has been idle since yesterday afternoon (16 hours), it appears that he isn't at work yet. He simply left himself logged in. Larry's terminal is currently in use.

whoami

`whoami`

Print current user ID. Equivalent to `id -un`.

whois

`whois[options] query[@server[:port]]`

`jwhois [options] query[@server[:port]]`

Search a **whois** database for a domain name, IP address, or NIC name. The information returned varies, but usually contains administrative and technical contacts so that you can find a person to handle problems at that domain. By default, the command returns information on `.com`, `.net`, and `.edu` domains, but other hosts can be queried for other domains using `host` or the `-h` option.

Options

- Indicate the end of **whois** options. A subsequent string that begins with a hyphen on the command line is taken as a query string.

`-a, --raw`

Do not rewrite query according to configuration before sending to server.

`-c file, --config=file`

Specify a configuration file to use instead of the default `/etc/jwhois.conf`.

`-d, --disable-cache`

Disable reading and writing to the cache.

`-f, --force-lookup`

Force the lookup query to go to the host, even if it is available from the cache.

`-h host, --host=host`

Query the **whois** server on the specified host. Same as `host` on the command line. By default, queries the server in the environment variable `NICNAMESEVER` or `WHOISSERVER` if either is set; otherwise queries `whois.internic.net`.

`--help`

Print help message and exit.

`-i, --display-redirections`

Display every step in a redirection. The default is to display only the last step.

- n, --no-redirect**
Disable redirection from one server to the next.
- p port, --port=port**
Connect to the specified port. Same as *port* on the command line. Default is 43.
- r, --rwhois**
Force use of the **rwhois** protocol, instead of HTTP or **whois**.
- rwhois-display=display**
Request receiving **rwhois** servers to display the results in the specified display instead of the default.
- rwhois-limit=limit**
Request receiving **rwhois** servers to limit the number of matches to the specified limit.
- s, --no-whois-servers**
Disable built-in support for *whois-servers.net*.
- v** Verbose. Print debugging information while running. Specify **-vv** to increase verbosity.
- version**
Print version information and exit.

wodim

wodim [options] track1,track2...

Record data or audio CD or DVD media. This program normally requires root access to the device file; run as root or install suid-root. It has a large number of options and settings; see the manpage for a complete list as well as a number of useful examples.

General options

General option flags go directly after the **wodim** command and before any track options or arguments. The general options are:

- atip**
Display the ATIP (Absolute Time In Pregroove) information for a disc. Not all drives allow you to read this information.
- blank=type**
Erase data from a CD-RW in one of the following ways:
 - all**
Erase all information on the disc. May take a long time.
 - fast**
Perform a quick erase of the disc, erasing only the PMA, TOC, and pregap.
 - help**
Display a possible list of blanking methods.
 - session**
Blank the last session.
 - track**
Blank a track.
 - trtail**
Blank the tail of a track only.

unclose
 Unclose the last session.

unreserve
 Unreserve a track previously marked as reserved.

-checkdrive
 Check to see if there are valid drivers for the current drive.
 Returns 0 if the drive is valid.

-d, debug=n
 Set the debug level to an integer (greater numbers are more verbose), or use multiple **-d** flags as with the **-v** and **-V** flags.

-dao, -sao
 Disk-at-once (session-at-once) mode. Works only with MMC drives that support nonraw session-at-once modes.

dev=target
 Set the SCSI target for the CD/DVD recorder. May be specified as a device name or as three comma-separated integers representing bus, target, and logical unit. To check the options that are available, use the **-scanbus** option. By default **wodim** looks in the CDR_DEVICE environment variable.

driver=name
 Lets you specify a driver for your system. Suggested for experts only. The special drivers **cdr_simul** and **dvd_simul** are used for simulation and profiling tests if your drive does not support the **-dummy** option.

driveropts=optlist
 Specify a comma-separated list of driver options. To get a list of valid options, use **driveropts=help** together with **-checkdrive**.

-dummy
 Perform a dry run, doing all the steps of recording with the laser turned off. This will let you know whether the process is going to work.

-eject
 Eject disc after recording. Some hardware may need to eject a disc after a dummy recording and before the actual recording.

-fix
 Close (“fixate”) the session, preventing future multisession recordings and allowing the disc to be played in standard audio CD players (some can also play a disc that has not been closed).

-force
 Override errors if possible. May allow you to blank an otherwise broken CD-RW.

fs=n
 Set the FIFO buffer size to *n*, in bytes. You may use **k**, **m**, **s**, or **f** to specify kilobytes, megabytes, or units of 2048 and 2352 bytes, respectively. The default is 4 MB.

gracetime=*n*

Set the number of seconds of grace time before writing. The value of *n* should be at least 2 (default is 4 seconds).

kdebug=*n*, kd=*n*

Set the kernel's debug notification value to *n* during SCSI command execution. Works through the usal-driver.

mcn=*n*

Set the Media Catalog Number to *n*.

msifile=*file*

Like **-msinfo**, but also saves the information in the specified file.

-msinfo

Get multisession information from the CD. Used only with multisession discs onto which you can still record more sessions.

-multi

Set to record in multisession mode. Must be present on all sessions but the last one for a multisession disc.

-nofix

Do not close the disc after writing.

-reset

Attempt to reset the SCSI bus. Does not work on all systems.

-s, -silent

Silent mode. Do not print any SCSI command errors.

speed=*n*

Set the speed to *n*, a multiple of the audio speed. Normally, **wodim** gets the speed from */etc/wodim.conf* or the CDR_SPEED environment variable. If your drive has trouble with higher numbers, try 0 as a value.

timeout=*n*

Set the SCSI command timeout to *n* seconds. Defaults to 40.

-tao

Track-at-once (TAO) write mode. Most drives require TAO mode for multisession recording.

-toc

Display the table of contents for the CD currently in the drive. Works for CD-ROM, as well as CD-R and CD-RW drives.

ts=*n*

Set the maximum transfer size for a single SCSI command to *n*. Defaults to 63 KB.

-scanbus

Scan SCSI devices. Useful for finding the SCSI address of the drive.

-useinfo

Use *.inf* files to override audio options set elsewhere.

-v Verbose mode. Use one **v** for each level of verbosity. **-vv** would be very verbose, and **-vvv** would be even more so.

-V A verbose mode counter that applies only to SCSI transport messages. This slows the application but can be useful for debugging.

-version

Print version information and exit.

-waiti

Wait for input on standard input before opening the SCSI driver. Useful for letting **wodim** read input from a pipe while writing additional sessions to a multisession disk.

Track options and arguments

Track options may be mixed with track arguments, and normally apply to the track immediately after them or to all tracks after them. The track arguments themselves should be the files that you will be writing to the CD. Options are:

-audio

Write all tracks after this track in digital audio format (playable by standard CD players). If you do not use this flag or the **-data** flag, **wodim** assumes that **.au** and **.wav** files are to be recorded as raw audio and that all other files are data.

-cdi

Write subsequent tracks in CDI format. Use with XA disks only.

-data

Record subsequent tracks as CD-ROM data. If you do not use this flag or the **-audio** flag, all files except for those that end in **.wav** or **.au** are assumed to be data.

-index=a,b,c

Set the index list for the next track. The values should be increasing comma-separated integers, starting with index 1 and counting in sectors (75ths of a second). For example, you could set three indices in a track with **index=0,750,7500** and they would occur at the beginning of the track, after 10 seconds, and after 100 seconds.

-isosize

The size of the next track should match the size of the ISO-9660 filesystem. This is used when duplicating CDs or copying from raw-data filesystems.

isrc=n

Set the International Standard Recording Number for the track argument following this option.

-mode2

Write all subsequent tracks in CD-ROM mode 2 format.

-nopad

Do not insert blank data between data tracks following this flag. This is the default behavior.

write

-pad

Insert 15 sectors of blank data padding between data tracks.
Applies to all subsequent tracks or until you use the **-nopad** argument, and is overridden by the **padszie=n** argument.

padszie=n

Insert *n* sectors of blank data padding after the next track.
Applies only to the track immediately after it.

-swab

Declare that your data is in byte-swapped (little-endian) byte order. This is not normally necessary.

tsize=n

Set the size of the next track. Useful only if you are recording from a raw disk for which **wodim** cannot determine the file size. If you are recording from an ISO 9660 filesystem, use the **-isosize** flag instead.

-xa, -xa1, -xa2

Write subsequent tracks in CD-ROM XA mode 2 format. **-xa1** writes in mode 2 form 1 format, and subheaders must be supplied by the application providing the data. **-xa** and **-xa2** write in mode 2 form 1 and mode 2 form 2, respectively, and subheaders are created by the drive.

write

write user [tty]

message

Initiate or respond to an interactive conversation with *user*. A **write** session is terminated with EOF. If the user is logged into more than one terminal, specify a *tty* number. See also **talk**; use **mesg** to keep other users from writing to your terminal.

xargs

xargs [options] [command]

Execute *command* (with any initial arguments), but read remaining arguments from standard input instead of specifying them directly. **xargs** passes these arguments in several bundles to the command, allowing it to process more arguments than it could normally handle at once. The arguments are typically a long list of filenames (generated by **ls** or **find**, for example) that get passed to **xargs** via a pipe. The default command is **/bin/echo**.

Options

-0, --null

Expect filenames to be terminated by NULL instead of whitespace. Do not treat quotes or backslashes specially.

-a file, --arg-file=file

Read arguments from *file*, not standard input.

-E string

Set EOF to *string*. Default is no EOF string.

-help

Print usage information and then exit.

-I string

Replace all occurrences of *string* in the initial arguments with names read from standard input. Unquoted blanks are not considered argument terminators; newline character is used. Implies **-x** and **-L 1**.

-L lines

Allow no more than *lines* nonblank input lines on the command line (default is 1). Implies **-x**.

-n args, --max-args=args

Allow no more than *args* arguments on the command line. Overridden by the maximum number of characters set with **-s**.

-p, --interactive

Prompt for confirmation (y or Y) before running each command line. Implies **-t**.

-P max, --max-procs=max

Allow no more than *max* processes to run at once. The default is 1. A maximum of 0 allows as many as possible to run at once.

-r, --no-run-if-empty

Do not run command if standard input contains only blanks.

-s max, --max-chars=max

Allow no more than *max* characters per command line.

--show-limits

Display upper and lower limits on command-line length (based on system limits, **xarg** buffer size, and **-s** if specified) before running the command.

-t, --verbose

Verbose mode. Print command line on standard error before executing.

-x, --exit

If the maximum size (as specified by **-s**) is exceeded, exit.

--version

Print the version number of **xargs** and then exit.

Examples

grep for *pattern* in all files on the system:

```
find / | xargs grep pattern> out &
```

Run **diff** on file pairs (e.g., **f1.a** and **f1.b**, **f2.a** and **f2.b**, etc.):

```
echo $* | xargs -n2 diff
```

The previous line would be invoked as a shell script, specifying filenames as arguments. Display *file*, one word per line (same as **deroff -w**):

```
cat file | xargs -n1
```

Move files in **olddir** to **newdir**, showing each command:

```
ls olddir | xargs -i -t mv olddir/{ } newdir/{ }
```

xinetd	xinetd [options]
	TCP/IP command. The extended Internet services daemon. xinetd saves system resources by listening to multiple sockets on the behalf of other server programs, invoking necessary programs as requests are made for their services. Beyond this, xinetd provides better logging facilities, including remote user ID, access times, and server-specific information. It also provides access-control facilities. Not limited to system administration use, it can launch services that are not listed in <i>/etc/services</i> . Unprivileged users can use this tool to start their own servers.
	Options
-cc num	Perform an internal-state consistency check every <i>num</i> seconds.
-d	Turn on debugging support.
-dontfork	Execute in the foreground. This option automatically sets the -stayalive option.
-f file	Read configuration from the specified <i>file</i> instead of <i>/etc/xinetd.conf</i> .
-filelog file	Write log messages to the specified <i>file</i> . Cannot be combined with -syslog or -d .
-inetd_compat	Read the <i>/etc/inetd.conf</i> file after reading <i>/etc/xinetd.conf</i> .
-limit num	Start no more than <i>num</i> concurrent processes.
-logprocs num	Limit processes used to look up remote user IDs to <i>num</i> .
-pidfile file	Write xinetd 's process ID to <i>file</i> .
-stayalive	Keep running even when no services have been specified.
-syslog facility	Log messages to the specified syslogd facility. Accepted values are daemon , auth , user , and localn , where <i>n</i> can range from 0 to 7. Cannot be combined with -syslog or -d . The default behavior is to write messages to syslogd using the daemon facility.
-version	Print version information, then exit.

Configuration files

By default **xinetd** reads its configuration information from file */etc/xinetd.conf*. Lines in this file beginning with # are treated as

comments. The entries for each service differ completely from */etc/inetd* entries. **xinetd** configuration entries for services follow the pattern:

```
service servicename
{
    attribute1 = valueset1
    attribute2 = valueset2
}
```

Some attributes allow assignment operators other than `=`. Other operators are `+=`, to add to a value set, and `-=`, to remove a value from a value set. There are many attributes available to control services. The following are the most common:

cps

Limit incoming connection rate. Accepts two numeric arguments: the number of connections per second to allow and the number of seconds to wait to accept a new connection when the rate is exceeded. The default is 50 incoming connections and a 10-second wait.

disable

Accept a Boolean **yes** or **no**. When disabled, **xinetd** will ignore the entry.

flags

Accept a set of the following values defining **xinetd**'s behavior:

IDONLY

Accept only connections when the remote user's ID can be verified by an identification server. Cannot be used with USERID logging.

INTERCEPT

Intercept packets to ensure they are coming from allowed locations. Cannot be used with internal or multithreaded services.

IPv4

Service is an IPv4 service.

IPv6

Service is an IPv6 service.

KEEPALIVE

Set flag on socket, enabling periodic checks to determine if the line is still receiving data.

NAMEINARGS

Expect the first argument for the **server_args** attribute to be the command to run. This flag is necessary to wrap services with **tcpd**.

NODELAY

Set socket's **NODELAY** flag.

NOLIBWRAP

Don't use **xinetd**'s internal TCP wrapping facilities.

NORETRY

If service fails to fork, don't try to fork again.

SENSOR

Instead of launching a service, add IP addresses that attempt to access this service to a list of denied addresses for a time specified by the **deny_time** attribute.

group

Specify a group ID for the server process. This may be used only when **xinetd** runs as root.

nice

Set service priority. This attribute accepts the same values as the **renice** command.

id

Specify a unique identifier for the service. Useful when creating multiple entries with the *servicename*. For example, two versions of the echo service, one supporting UDP and the other TCP, might be given the identifiers **echo-stream** and **echo-dgram**.

log_on_failure

Specify values to log when a server cannot be started. Accepted values are HOST, USERID, or just ATTEMPT.

log_on_success

Specify values to log when a server is started. Accepted values are PID, HOST, USERID, EXIT, and DURATION.

no_access

Specify hosts that should not be allowed access to a service. May be given as an IP address, a netmask, a hostname, a network name from */etc/networks*, or a group of IP addresses like so: 192.168.1.{10,11,12,15,32}.

only_from

Restrict access to the service to the specified hosts. This attribute accepts the same values as **no_access**.

per_source

Specify the maximum number of instances allowed to a single source IP address. The default is “UNLIMITED”.

port

Specify the service port to listen to. This attribute is required for non-RPC services not listed in */etc/services*. If the service is listed, the value of **port** cannot differ from what is listed.

protocol

Specify protocol to use, usually **tcp** or **udp**. The protocol must be listed in */etc/protocols*. This attribute is required for RPC services, as well as services not found in */etc/services*.

rpc_version

The RPC version used by the service. This can be a single number or a range of numbers from *x-y*. This attribute is required for RPC services.

rpc_number

Specify RPC ID number. This is required only for services not listed in */etc/rpc*; otherwise it's ignored.

server

The program to execute for the service. When using **tcpd** to wrap a service, also set the **NAMEINARGS** flag and use the server's program name as the first argument for **server_args**. This attribute is required for all noninternal services.

server_args

Arguments to pass to the server program.

socket_type

Specify the socket type to create. Accepted values are **stream**, **dgram**, **raw**, and **seqpacket**.

type

Describe the type of service. Accepted values are **RPC**, **INTERNAL**, and **UNLISTED**.

user

Specify a user ID for the server process. This may be used only when **xinetd** runs as root.

wait

Determine whether services should be treated as single-threaded (**yes**) and **xinetd** should wait until the server exits to resume listening for new connections, or multithreaded (**no**) and **xinetd** should not wait to resume listening. This attribute is required for all serices.

Files

/etc/xinetd.conf

Default configuration file.

/etc/xinetd.d

Common directory containing configuration files included from */etc/xinetd.conf*.

yacc

yacc [options] file

Given a *file* containing context-free grammar, convert *file* into tables for subsequent parsing, and send output to *y.tab.c*. This command name stands for yet another compiler-compiler. See also **flex**, **bison**, and the book *lex & yacc* (O'Reilly).

Options

-b *prefix*

Prepend *prefix*, instead of *y*, to the output file.

-d Generate *y.tab.h*, producing **#define** statements that relate *yacc*'s token codes to the token names declared by the user.

-g Generate a VCG description.

-l Exclude **#line** constructs from code produced in *y.tab.c*. (Use after debugging is complete.)

yes

-o *outfile*

Write generated code to *outfile* instead of the default *y.tab.c*.

-p *prefix*

Change the symbol **yacc** uses for symbols it generates from the default *yy* to *prefix*.

-t Compile runtime debugging code.

-v Generate *y.output*, a file containing diagnostics and notes about the parsing tables.

yes

yes [*strings*]

yes [*option*]

Print the command-line arguments, separated by spaces and followed by a newline, until killed. If no arguments are given, print **y** followed by a newline until killed. Useful in scripts and in the background; its output can be piped to a program that issues prompts.

Options

--help

Print a help message and then exit.

--version

Print version information and then exit.

ypbind

ypbind [*options*]

NFS/NIS command. NIS binder process. **ypbind** is a daemon process typically activated at system startup time. Its function is to remember information that lets client processes on a single node communicate with some **ypserv** process. The information **ypbind** remembers is called a *binding*—the association of a domain name with the Internet address of the NIS server and the port on that host at which the **ypserv** process is listening for service requests. This information is cached in the file */var/yp/binding/domainname.version*.

Options

-broadcast

Ignore configuration information in */etc/yp.conf* and directly request configuration information from a remote system using **ypset**.

-broken-server

Allow connections to servers using normally illegal port numbers. Sometimes needed for compatibility with other versions of **ypserv**.

-c Check configuration file for syntax errors, then exit.

-debug

Run in the foreground process instead of detaching and running as a daemon.

-f file

Read configuration information from *file* instead of */etc/yp.conf*.

-no-ping

Don't ping remote servers to make sure they are alive.

--version

Print version information, then exit.

-ypset

Allow remote machine to change the local server's bindings.
This option is very dangerous and should be used only for debugging the network from a remote machine.

-ypsetme

ypset requests may be issued from this machine only. Security is based on IP address checking, which can be defeated on networks on which untrusted individuals may inject packets.
This option is not recommended.

ypcat**ypcat [options] map**

NFS/NIS command. Print values in an NIS database specified by *map* name or nickname.

Options**-d domain**

Specify *domain* other than the default domain.

-h host

Specify a *ypbind host* other than the default.

-k Display keys for maps in which values are null or key is not part of value.

-t Do not translate *mname* to map name.

-x Display map nickname table listing the nicknames (*mnames*) known and map name associated with each nickname. Do not require an *mname* argument.

ypinit**ypinit [options]**

NFS/NIS command. Build and install an NIS database on an NIS server. **ypinit** can be used to set up a master server, slave server, or slave copier. Only a privileged user can run **ypinit**.

Options

-m Indicate that the local host is to be the NIS master server.

-s master_name

Set up a slave server database. *master_name* should be the hostname of an NIS server, either the master server for all the maps, or a server on which the database is up to date and stable.

ypmatch	<code>ypmatch [options] key... mname</code> NFS/NIS command. Print value of one or more keys from an NIS map specified by <i>mname</i> . <i>mname</i> may be either a map name or a map nickname.
----------------	--

Options

- d *domain*
Specify *domain* other than default domain.
- k Before printing value of a key, print the key itself, followed by a colon (:).
- t Do not translate nickname to map name.
- x Display map nickname table listing the nicknames (*mnames*) known, and the map name associated with each nickname. Do not require an *mname* argument.

yppasswd	<code>yppasswd [options] [name]</code> NFS/NIS command. Change login password in Network Information Service. Create or change your password, and distribute the new password over NIS. The superuser can change the password for any <i>user</i> . This command may also be invoked as ypchfn and ypchsh .
-----------------	--

Options

- f Update the password information field (the **GECOS** field). Using this option is the same as **ypchfn**.
- l Update the login shell. Using this option is the same as **ypchsh**.
- p Update the password. This is the default behavior for **yppasswd**.

ypasswdd	<code>rpc.yppasswdd [options]</code> NFS/NIS command. Server for modifying the NIS password file. yppasswdd handles password-change requests from yppasswd . It changes a password entry only if the password represented by yppasswd matches the encrypted password of that entry and if the user ID and group ID match those in the server's <i>/etc/passwd</i> file. Then it updates <i>/etc/passwd</i> and the password maps on the local server. If the server was compiled with the CHECKROOT=1 option, the password is also checked against the root password.
-----------------	--

Options

- D *dir*
Specify a directory that contains the *passwd* and *shadow* files for **rpc.yppasswdd** to use instead of */etc/passwd* and */etc/shadow*. Useful to prevent all users in the NIS database from automatically gaining access to the NIS server.
- e [**chsh|chfn**]
Permit users to change the shell or user information in the **GECOS** field of their *passwd* entry. By default, **rpc.yppasswdd** does not permit users to change these fields.

-E program

Specify a program to edit the *passwd* and *shadow* files instead of **rpc.yppasswdd**. The program should return 0 for successful completion; 1 for successful completion, but the **pwupdate** program should not be run to update the NIS server's maps; and anything else if the change failed.

-p pwfile

Specify an alternative *passwd* file to */etc/passwd*, to prevent all users in the NIS database from automatically gaining access to the NIS server.

--port num

Specify a port that **rpc.yppasswdd** will try to register itself, allowing a router to filter packets to the NIS ports.

-s shadowfile

Use *shadowfile* instead of */etc/passwd* for shadow password support.

-v Print version information and whether the package was compiled with **CHECKROOT**.

-x program

Modify files using the specified *program* instead of using internal default functions. **rpc.yppasswdd** passes information to *program* in the following format:

```
username o:oldpassword p:password s:shell g:gcos
```

Any of the fields **p**, **s**, or **g** may be missing.

yppoll**yppoll [options] map**

NFS/NIS command. Determine version of NIS map at NIS server. **yppoll** asks a **ypserv** process for the order number and the hostname of the master NIS server for the *map*.

Options**-h host**

Ask the **ypserv** process at *host* about the map parameters. If *host* is not specified, the hostname of the NIS server for the local host (the one returned by **ypwhich**) is used.

-d domain

Use *domain* instead of the default domain.

yppush**yppush [options] mapnames**

NFS/NIS command. Force propagation of changed NIS map. **yppush** copies a new version of an NIS map, *mapname*, from the master NIS server to the slave NIS servers. It first constructs a list of NIS server hosts by reading the NIS map **ypservers** with the **-d** option's *domain* argument. Keys within this map are the ASCII names of the machines on which the NIS servers run. A map transfer request is sent to the NIS server at each host, along with the information needed by the transfer agent to call back to

yppush. When the attempt has been completed and the transfer agent has sent **yppush** a status message, the results may be printed to standard error. Normally invoked by */var/yp/Makefile* after commenting out the **NOPUSH=true** line.

Options

-d domain

Specify a *domain*.

-h host

Specify one or a group of systems to which a map should be transferred instead of using the list of servers in the **ypservers** map. Multiple **-h** options can be specified to create a list of hosts.

-p count, --parallel count

Send maps to *count* NIS slaves simultaneously (in parallel). By default, **yppush** sends maps to one server at a time (serially).

--port num

Specify a port to listen on for callbacks. This will not work when sending maps in parallel. By default, the command chooses a random port.

-t secs

Specify a timeout value in seconds. The timeout determines how long **yppush** will wait for a response from a slave server before sending a map transfer request to the next server. The default timeout is 90 seconds, but for big maps a longer timeout may be needed.

-v Verbose; print message when each server is called and for each response. Specify twice to make **yppush** even more verbose.

NFS/NIS command. NIS server process. **ypserv** is a daemon process typically activated at system startup time. It runs only on NIS server machines with a complete NIS database. Its primary function is to look up information in its local database of NIS maps. The operations performed by **ypserv** are defined for the implementor by the NIS protocol specification, and for the programmer by the header file *<rpcv/yp_prot.h>*. Communication to and from **ypserv** is by means of RPC calls. On startup or when receiving the signal SIGHUP, **ypserv** parses the file */etc/ypserv.conf*. **ypserv** supports **securenets**, which can be used to restrict access to a given set of hosts.

Options

-d [path], --debug [path]

Run in debugging mode without going into background mode, and print extra status messages to standard error for each request. If *path* is specified, use it instead of */var/yp*.

-i interface, --iface interface

Only provide service on the specified network *interface*.

-p port, --port port

Bind to the specified port. For use with a router to filter packets so that access from outside hosts can be restricted.

-v, --version

Print version information and exit.

Files and directories*/etc/yp.conf*

Configuration file.

/var/yp/domainname

Location of NIS databases for *domainname*.

/var/yp/Makefile

Makefile that is responsible for creating NIS databases.

/var/yp/securenets

securenets information containing netmask/network pairs separated by whitespace.

yptest**yptest [options] server**

NFS/NIS command. Point **ypbind** at a particular server. **yptest** tells **ypbind** to get NIS services for the specified domain from the **ypserv** process running on *server*. *server* indicates the NIS server to bind to and can be specified as a name or an IP address.

Options**-d domain**

Use *domain* instead of the default domain.

-h host

Set **ypbind**'s binding on *host* instead of the local host. *host* can be specified as a name or an IP address.

yptest**yptest [options]**

NFS/NIS command. Check configuration of NIS services by calling various NIS functions. Without arguments, **yptest** queries the NIS server for the local machine.

Options**-d domainname**

Use *domainname* instead of the current host's default domain. This option may cause some tests to fail.

-h host

Test **ypserv** on the specified *host* instead of the current host. This option may cause some tests to fail.

-m map

Use the specified *map* instead of the default map.

-q

Quiet mode. Print no messages.

-u user

Run tests as *user* instead of as nobody.

ypwhich	<code>ypwhich [options] [host]</code> NFS/NIS command. Return hostname of NIS server or map master. Without arguments, ypwhich cites the NIS server for the local machine. If <i>host</i> is specified, that machine is queried to find out which NIS master it is using.
----------------	---

Options

-d domain

Use *domain* instead of the default domain.

-m [map]

Find master NIS server for a map. No host can be specified with **-m**. *map* may be a map name or a nickname for a map. If no map is specified, display a list of available maps.

-t mapname

Inhibit nickname translation.

-Vn

Version of **ypbind** (default is v2).

-x Display map nickname table. Do not allow any other options.

ypxfr	<code>ypxfr [options] mapname</code> NFS/NIS command. Transfer an NIS map from the server to the local host by making use of normal NIS services. ypxfr creates a temporary map in the directory <code>/var/yp/domain</code> (where <i>domain</i> is the default domain for the local host), fills it by enumerating the map's entries, and fetches the map parameters and loads them. If run interactively, ypxfr writes its output to the terminal. However, if it is invoked without a controlling terminal, its output is sent to syslogd .
--------------	---

Options

-c Do not send a “Clear current map” request to the local **ypserv** process.

-C tid prog ipadd port

This option is for use only by **ypserv**. When **ypserv** invokes **ypxfr**, it specifies that **ypxfr** should call back a **yppush** process at the host with IP address *ipadd*, registered as program number *prog*, listening on port *port*, and waiting for a response to transaction *tid*.

-d domain

Specify a domain other than the default domain.

-f Force the transfer to occur even if the version on the master server is older than the local version.

-h host

Get the map from *host* instead of querying NIS for the map's master server. *host* may be specified by name or IP address.

-p dirUse *dir* as the path to the NIS map directory instead of */var/yp*.**-s domain**Specify a source *domain* from which to transfer a map that should be the same across domains (such as the *servicesbyname* map).**ypxfrd****rpc.ypxfrd [options]**NFS/NIS command. This server is used to copy a master's NIS map using an RPC based file transfer program instead of building a local map the way **ypxfr** would. This will speed up the transfer of large maps.**Options****--debug**

Debug mode. Do not fork.

-d dirUse *dir* instead of */var/yp*.**-p port**Bind to the specified *port*.**-v, --version**

Print version information and exit.

zcat**zcat [options] [files]**Read one or more *files* that have been compressed with **gzip** or **compress** and write them to standard output. Read standard input if no *files* are specified or if - is specified as one of the files; end input with EOF. **zcat** is identical to **gunzip -c** and takes the options described for **gzip/gunzip**.**zcmp****zcmp [options] files**Read compressed files and pass them uncompressed to the **cmp** command, along with any command-line options. If a second file is not specified for comparison, look for a file called *file.gz*.**zdiff****zdiff [options] files**Read compressed files and pass them, uncompressed, to the **diff** command, along with any command-line options. If a second file is not specified for comparison, look for a file called *file.gz*.**zforce****zforce [names]**Rename all **gzipped** files to *filename.gz*, unless file already has a .gz extension.

zgrep	zgrep [options] [files] Uncompress files and pass to grep , along with any command-line arguments. If no files are provided, read from (and attempt to uncompress) standard input. May be invoked as zegrep or zfgrep and will in those cases invoke egrep or fgrep .
zless	zless files Uncompress files and allow paging through them. Equivalent to running zmore with the environment variable PAGER set to less . See zmore for the available commands.
zmore	zmore [files] Similar to more . Uncompress files and print them one screenful at a time. Works on files compressed with compress , gzip , or pack , and with uncompressed files. The argument <i>i</i> in the following zmore commands is an optional integer argument.

Commands

space

Print next screenful.

ispace

Print next *i* lines.

Return

Print one more line.

id, i^D

Print next *i*, or 11, lines.

iz Print next *i* lines or a screenful. If *i* is specified, treat it as the new window size for the rest of the current file, then revert to the default.

is Skip *i* lines, then print the next screenful.

if Skip *i* screens, then print the next screenful.

q, Q, :q, :Q

Go to next file or, if current file is the last, exit **zmore**.

e, q

Exit **zmore** when the prompt “--More-- (Next file: *file*)” is displayed.

s Skip next file and continue when the prompt “--More-- (Next file: *file*)” is displayed.

= Print line number.

i/expr

Search forward for *i*th occurrence (in all files) of *expr*, which should be a regular expression. Display occurrence, including the two previous lines of context.

in Search forward for the *i*th occurrence of the last regular expression searched for.

!command

Execute *command* in shell. If *command* is not specified, execute last shell command. To invoke a shell without passing it a command, enter `\!`.

- Repeat the previous command.

znew

znew [options] [files]

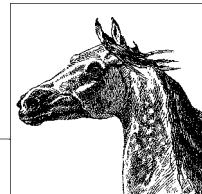
Uncompress .Z files and recompress them in .gz format.

Options

- 9** Optimal (and slowest) compression method.
 - f** Recompress even if *filename.gz* already exists.
 - K** If the original .Z file is smaller than the .gz file, keep it.
 - P** Pipe data to conversion program. This saves disk space.
 - t** Test new .gz files before removing .Z files.
 - v** Verbose mode.
-

4

Boot Methods



This chapter describes techniques for booting your Linux system. Depending on your hardware and whether you want to run any other operating systems, you can configure the system to automatically boot Linux or to provide a choice between several operating systems. Choosing between operating systems is generally referred to as *dual booting*, although you can select between more than two. We talk more about dual booting in the section “Dual-Booting Linux and Windows 2000/XP/Vista” on page 536.

An alternative to dual booting is virtualization, where you run one or more virtual operating systems inside a real operating system. The real system is known as the host, and the virtual systems are known as guests. Virtualization makes it easy to switch between systems without having to reboot. Two ways to run virtual systems are to make Linux the host system with another operating system running in a virtual machine. See Chapter 15 for an overview of virtualization concepts and for information on how to run guest systems under Linux. You can also run Linux as a guest with another operating system such as Windows as the host. Two ways to do this are with Microsoft’s Virtual PC and VMware server. Both are free downloads and are available at www.microsoft.com and www.vmware.com, respectively.

Once your Linux system is installed, rebooting the system is generally straightforward. There are several possibilities for configuring your boot process. The most common choices are:

- Boot Linux from a bootable disk, most likely a CD or an installation CD/DVD, leaving another operating system to boot from the hard drive.
- Use the Linux Loader, LILO. This used to be the traditional method of booting and lets you boot both Linux and other operating systems.
- Use GRUB (GRand Unified Bootloader), the GNU graphical boot loader and command shell. Like LILO, GRUB lets you boot both Linux and other operating systems. GRUB, which has additional functionality not found in LILO, is now the *de facto* Linux boot loader.

Whatever method you choose for booting, be sure to have a working boot disk available for emergency use. In particular, don't experiment with the files and options in this chapter unless you have a boot disk, because any error could leave you unable to boot from the hard disk. Note, though, that one of the advantages of using GRUB is that if there is a problem booting from the menu, it drops you down to the command-line interface so you can enter commands directly and try to recover. In addition, your distribution CD or DVD undoubtedly has a recovery option on it. Or you can boot from a live Linux CD such as Knoppix.

The Boot Process

On an x86-based PC, the first sector of every hard disk is known as the *boot sector* and contains the partition table for that disk and possibly also code for booting an operating system. The boot sector of the first hard disk is known as the *master boot record* (MBR), because when you boot the system, the BIOS transfers control to a program that lives on that sector along with the partition table. That code is the *boot loader*, the code that initiates an operating system. When you add Linux to the system, you need to modify the boot loader, replace it, or boot from a floppy or CD to start Linux.

In Linux, each disk and each partition on the disk is treated as a device. For example, the entire first hard disk is known as */dev/hda*, and the entire second hard disk is */dev/hdb*. The first partition of the first hard drive is */dev/hda1*, and the second partition is */dev/hda2*. The first partition of the second hard drive is */dev/hdb1*, and so on. If your drives are SCSI or SATA instead of IDE, the naming works the same way, except that the devices are */dev/sda*, */dev/sda1*, and so on. Thus, if you want to specify that the Linux partition is the second partition of the first hard drive (as in the examples in this chapter), you refer to it as */dev/hda2*. Note that GRUB has its own disk naming convention, described later in this chapter in “GRUB: The Grand Unified Bootloader” on page 516.

Once you've made the decision to install LILO or GRUB, you still need to decide how it should be configured. Most Linux distributions will automatically set up the booting environment for you, whether you are installing Linux as the primary operating system or into a dual-booting environment (or in a virtual guest system where the real MBR is not modified). If for some reason your distribution doesn't do it for you, or you want to do it manually, the rest of this chapter will help you. If your distribution does set up the boot environment for you, you might still want to read the sections on LILO or GRUB to find out how to customize your boot loader.

If you want your system to dual-boot Linux and Windows, you need to know that Windows has its own loader installed on the MBR, and it expects that loader to be in charge. The standard solution described in this chapter is to add Linux as an option in the Windows loader and install LILO or GRUB in the Linux partition as a secondary boot loader. The result is that the Windows loader transfers control to the secondary loader, which then boots Linux. See “Dual-Booting Linux and Windows 2000/XP/Vista” on page 536 for more information. You can also install one of the Linux boot loaders in the MBR and use it to boot Windows. (See the “Linux+WinNT” and the “Multiboot with GRUB” mini-HOWTOs at the Linux Documentation Project [www.tldp.org] if you're interested in doing that.)

When you install the boot loader (either LILO or GRUB) on the MBR, it replaces the Windows boot loader. If you have problems with your installation or you simply want to restore the original boot loader, you can do one of the following:

- If you’re running LILO, you can boot Linux from a boot disk (CD or floppy) and restore the boot sector, which LILO automatically backs up:
`$ /sbin/lilo -u`
- For Windows 2000, XP, and Vista, boot your computer from the Windows CD. When you see “Welcome to Setup,” press R (for repair) and, in Windows 2000, then press C. Select your Windows installation from the numbered list that is displayed (there may be only one entry) and enter the administrator password at the prompt. Enter the command `fixmbr` at the command-line prompt and confirm it with `y`. After the MBR has been restored, type `exit` to reboot.

The common element in both methods is that they replace the boot loader on the MBR with the original Microsoft boot loader.

Whatever boot loader is on the MBR is the one that will be used to boot the system. This means that if you want to switch from LILO to GRUB, say, or from GRUB to LILO, you don’t need to uninstall the old loader; simply install the new one.

The rest of this chapter describes the various techniques for booting Linux and the options that you can specify to configure both the boot loader and the Linux kernel. Whether you use GRUB or LILO, you can pass options to the loader and specify options for the kernel.

LILO: The Linux Loader

In addition to booting Linux, LILO can boot other operating systems, such as Windows or any of the BSD systems. During installation, some Linux distributions provide the opportunity to install LILO (most now install GRUB by default). LILO can also be installed later if necessary. LILO can be installed on the MBR of your hard drive or as a secondary boot loader on the Linux partition. LILO consists of several pieces, including the boot loader itself, a configuration file (`/etc/lilo.conf`), a map file (`/boot/map`) containing the location of the kernel, and the `lilo` command (`/sbin/lilo`), which reads the configuration file and uses the information to create or update the map file and to install the files LILO needs.

One thing to remember about LILO is that it has two aspects: the boot loader and the `lilo` command. The `lilo` command configures and installs the boot loader and updates it as necessary. The boot loader is the code that executes at system boot time and boots Linux or another operating system.

You can make a rescue CD for LILO with the LILO command `mkrescue --iso` to make an image that can be burned to CD. Use `mkrescue` by itself or with other options to make a rescue floppy disk. See the `mkrescue` manpage for more information.

The LILO Configuration File

The **lilo** command reads the LILO configuration file, */etc/lilo.conf*, to get the information it needs to install LILO. Among other things, it builds a map file containing the locations of all disk sectors needed for booting.

Note that any time you change */etc/lilo.conf* or rebuild or move a kernel image, you need to rerun **lilo** to rebuild the map file and update LILO.

The configuration file starts with a section of global options, described in the next section. Global options are those that apply to every system boot, regardless of the operating system you are booting. Here is an example of a global section (a hash sign, **#**, begins a comment):

```
boot=/dev/hda           # The boot device is /dev/hda
map=/boot/map           # Save the map file as /boot/map
install=/boot/boot.b    # The file to install as the new boot sector
prompt                 # Always display the boot prompt
timeout=30              # Set a 3-second (30 tenths of a second) timeout
```

Following the global section, there is one section of options for each Linux kernel and for each non-Linux operating system that you want LILO to be able to boot. Each of these sections is referred to as an *image* section because each boots a different kernel image (shorthand for a binary file containing a kernel) or another operating system. Each Linux image section begins with an **image=** line.

```
image=/boot/vmlinuz      # Linux image file
label=linux               # Label that appears at the boot prompt
root=/dev/hda2            # Location of the root filesystem
vga=ask                  # Always prompt the user for VGA mode
read-only                # Mount read-only to run fsck for a filesystem check
```

The equivalent section for a non-Linux operating system begins with **other=** instead of **image=**. For example:

```
other=/dev/hda1          # Location of the partition
label=winxp               # Label that appears at the boot prompt
table=/dev/hda             # Location of the partition table
```

Put LILO configuration options that apply to all images into the global section of */etc/lilo.conf*, and options that apply to a particular image into the section for that image. If an option is specified in both the global section and an image section, the setting in the image section overrides the global setting for that image.

Here is an example of a complete */etc/lilo.conf* file for a system that has the Linux partition on */dev/hda2*:

```
## Global section
boot=/dev/hda2
map=/boot/map
delay=30
timeout=50
prompt
vga=ask
```

```

## Image section: For regular Linux
image=/boot/vmlinuz
label=linux
root=/dev/hda2
install=/boot/boot.b
map=/boot/map
read-only

## Image section: For testing a new Linux kernel
image=/testvmlinuz
label=testlinux
root=/dev/hda2
install=/boot/boot.b
map=/boot/map
read-only
optional           # Omit image if not available when map is built

## Image section: For booting Windows XP
other=/dev/hda1
label=winxp
loader=/boot/chain.b
table=/dev/hda      # The current partition table

```

Global options

In addition to the options listed here, the kernel options **append**, **read-only**, **read-write**, **root**, and **vga** (described later in “Kernel options” on page 513) can also be set as global options.

backup=*backup-file*

Copy the original boot sector to *backup-file* instead of to */boot/boot.nnnn*, where *nnnn* is a number that depends on the disk device type.

boot=*boot-device*

Set the name of the device that contains the boot sector. **boot** defaults to the device currently mounted as root, such as */dev/hda2*. Specifying a device such as */dev/hda* (without a number) indicates that LILO should be installed in the master boot record; the alternative is to set it up on a particular partition, such as */dev/hda2*.

change-rules

Begin a section that redefines partition types at boot time for hiding and unhiding partitions. See the LILO User’s Guide, which comes with the LILO distribution, for detailed information on using this option and creating a new rule set.

compact

Merge read requests for adjacent disk sectors to speed up booting. Use of **compact** is particularly recommended when booting from a floppy disk. Use of **compact** may conflict with **linear**.

default=*name*

Use the image *name* as the default boot image. If **default** is omitted, the first image specified in the configuration file is used.

delay=tsecs

Specify, in tenths of a second, how long the boot loader should wait before booting the default image. If **serial** is set, **delay** is set to a minimum of 20. The default is not to wait. See “Boot-Time Kernel Options” on page 539 for ways to get the boot prompt if no delay is set.

disk=device-name

Define parameters for the disk specified by *device-name* if LILO can’t figure them out. Normally, LILO can determine the disk parameters itself, and this option isn’t needed. When **disk** is specified, it is followed by one or more parameter lines, such as:

```
disk=/dev/sda
bios=0x80      # First disk is usually 0x80, second is usually 0x81
sectors=...
heads=...
```

Note that this option is not the same as the disk geometry parameters you can specify with the **hd** boot command-line option. With **disk**, the information is given to LILO; with **hd**, it is passed to the kernel. Note also that if either **heads** or **sectors** is specified, they must both be specified. The parameters that can be specified with **disk** are listed briefly here; they are described in detail in the LILO User’s Guide.

bios=bios-device-code

The number the BIOS uses to refer to the device. See the previous example.

cylinders=cylinders

The number of cylinders on the disk.

heads=heads

The number of heads on the disk.

inaccessible

Tell LILO that the BIOS can’t read the disk; used to prevent the system from becoming unbootable if LILO thinks the BIOS can read it. If this parameter is specified, it must be the only parameter.

partition=partition-device

Start a new section for a partition. The section contains one variable, **start=partition-offset**, which specifies the zero-based number of the first sector of the partition:

```
partition=/dev/sda1
start=2048
```

sectors=sectors

The number of sectors per track.

disktab=disktab-file

This option has been superseded by the **disk=** option.

fix-table

If set, allow **lilo** to adjust 3-D addresses (addresses specified as sector/head/cylinder) in partition tables. This is sometimes necessary if a partition isn’t track-aligned and another operating system is on the same disk. See the *lilo.conf* manpage for details.

force-backup=*backup-file*

Like **backup**, but overwrite an old backup copy if one exists.

ignore-table

Tell **lilo** to ignore corrupt partition tables.

install=*boot-sector*

Install the specified file as the new boot sector. If **install** is omitted, the boot sector defaults to */boot/boot.b*.

lba32

Generate 32-bit Logical Block Addresses instead of sector/head/cylinder addresses, allowing booting from any partition on hard disks greater than 8.4 GB (i.e., remove the 1024-cylinder limit). Requires BIOS support for the EDD packet call interface* and at least LILO version 21–4.

linear

Generate linear sector addresses, which do not depend on disk geometry, instead of 3-D (sector/head/cylinder) addresses. If LILO can't determine your disk's geometry itself, you can try using **linear**; if that doesn't work, then you need to specify the geometry with **disk=**. Note, however, that **linear** sometimes doesn't work with floppy disks, and it may conflict with **compact**.

lock

Tell LILO to record the boot command line and use it as the default for future boots until it is overridden by a new boot command line. **lock** is useful if there are kernel options that you need to enter on the boot command line every time you boot the system.

map=*map-file*

Specify the location of the map file. Defaults to */boot/map*. The map file records the location of the kernel(s) used on the system.

message=*message-file*

Specify a file containing a message to be displayed before the boot prompt. The message can include a formfeed character (**Ctrl-L**) to clear the screen. The map file must be rebuilt by rerunning the **lilo** command if the message file is changed or moved. The maximum length of the file is 65,535 bytes.

nowarn

Disable warning messages.

optional

Specify that any image that is not available when the map is created should be omitted and not offered as an option at the boot prompt. Like the per-image option **optional**, but applies to all images.

password=*password*

Specify a password that the user is prompted to enter when trying to load an image. The password is not encrypted in the configuration file, so if passwords are used, permissions should be set so that only the superuser is able to read the file. This option is like the per-image version, except that all images are password-protected and they all have the same password.

* As long as your BIOS is dated after 1998, it should include EDD packet call interface support.

prompt

Automatically display the boot prompt without waiting for the user to press the Shift, Alt, or Scroll Lock key. Note that setting **prompt** without also setting **timeout** prevents unattended reboots.

restricted

Can be used with **password** to indicate that a password needs to be entered only if the user specifies parameters on the command line. Like the per-image **restricted** option, but applies to all images.

serial=parameters

Allow the boot loader to accept input from a serial line as well as from the keyboard. Sending a break on the serial line corresponds to pressing a Shift key on the console to get the boot loader's attention. All boot images should be password-protected if serial access is insecure (e.g., if the line is connected to a modem). Setting **serial** automatically raises the value of **delay** to 20 (i.e., two seconds) if it is less than that. The parameter string *parameters* has the following syntax:

port[,bps[parity[bits]]]

For example, to initialize COM1 with the default parameters:

`serial=0,2400n8`

The parameters are:

port

The port number of the serial port. The default is 0, which corresponds to COM1 (*/dev/ttys0*). The value can be one of 0 through 3, for the four possible COM ports.

bps

The baud rate of the serial port. Possible values of *bps* are 110, 300, 1200, 2400, 4800, 9600, 19200, 38400, 57600, and 115200. The default is 2400 bps.

parity

The parity used on the serial line. Parity is specified as **n** or **N** for no parity, **e** or **E** for even parity, and **o** or **O** for odd parity. However, the boot loader ignores input parity and strips the 8th bit.

bits

Specify whether a character contains 7 or 8 bits. Default is 8 with no parity and 7 otherwise.

timeout=tsecs

Set a timeout (specified in tenths of a second) for keyboard input. If no key has been pressed after the specified time, the default image is booted automatically. **timeout** is also used to determine when to stop waiting for password input. The default timeout is infinite.

verbose=level

Turn on verbose output, where higher values of *level* produce more output. If **-v** is also specified on the **lilo** command line, the level is incremented by 1 for each occurrence of **-v**. The maximum verbosity level is 5.

Image options

The following options are specified in the image section for a particular boot image. The image can be a Linux kernel or a non-Linux operating system.

alias=name

Provide an alternate name for the image that can be used instead of the name specified with the **label** option.

image=pathname

Specify the file or device containing the boot image of a bootable Linux kernel. Each per-image section that specifies a bootable Linux kernel starts with an **image** option. See also the **range** option.

label=name

Specify the name that is used for the image at the boot prompt. Defaults to the filename of the image file (without the path).

loader=chainloader

For a non-Linux operating system, specify the chain loader to which LILO should pass control for booting that operating system. The default is */boot/chain.b*. If the system will be booted from a drive that is neither the first hard disk nor a floppy or CD, the chainloader must be specified.

lock

Like **lock**, as described in the previous global options section; it can also be specified in an image section.

optional

Specify that the image should be omitted if it is not available when the map is created by the **lilo** command. Useful for specifying test kernels that are not always present.

other=pathname

Specify the path to a file that boots a non-Linux system. Each per-image section that specifies a bootable non-Linux system starts with an **other** option.

password=password

Specify that the image is password-protected and provide the password that the user is prompted for when booting. The password is not encrypted in the configuration file, so if passwords are used, permissions should be set so only the superuser can read the file.

range=sectors

Used with the **image** option, when the image is specified as a device (e.g., **image=/dev/fd0**), to indicate the range of sectors to be mapped into the map file. *sectors* can be given as the range *start-end* or as *start+number*, where *start* and *end* are zero-based sector numbers and *number* is the increment beyond *start* to include. If only *start* is specified, only that one sector is mapped. For example:

```
image=/dev/fd0
range=1+512 # take 512 sectors, starting with sector 1
```

restricted

Specify that a password is required for booting the image only if boot parameters are specified on the command line.

table=device

Specify, for a non-Linux operating system, the device that contains the partition table. If **table** is omitted, the boot loader does not pass partition information to the operating system being booted. Note that */sbin/lilo* must be rerun if the partition table is modified. This option cannot be used with **unsafe**.

unsafe

Can be used in the per-image section for a non-Linux operating system to indicate that the boot sector should not be accessed when the map is created. If **unsafe** is specified, then some checking isn't done, but the option can be useful for running the **lilo** command without having to insert a floppy disk when the boot sector is on a fixed-format floppy disk device. This option cannot be used with **table**.

Kernel options

The following kernel options can be specified in */etc/lilo.conf* as well as on the boot command line:

append=string

Append the options specified in *string* to the parameter line passed to the kernel. This typically is used to specify certain hardware parameters. For example, while BIOSes on modern systems can recognize more than 64 MB of memory, BIOSes on older systems were limited to 64 MB. If you are running Linux on such a system, you can use **append**:

```
append="mem=128M"
```

initrd=filename

Specify the file to load into */dev/initrd* when booting with a RAM disk. See also the options **load_ramdisk** (in “Boot-Time Kernel Options” on page 539) and **prompt_ramdisk**, **ramdisk_size**, and **ramdisk_start** in this section.

literal=string

Like **append**, but replace all other kernel boot options.

noinitrd

Preserve the contents of */dev/initrd* so they can be read after the kernel is booted.

prompt_ramdisk=n

Specify whether the kernel should prompt you to insert the floppy disk that contains the RAM disk image, for use during Linux installation. Values of *n* are:

- 0** Don't prompt. Usually used for an installation in which the kernel and the RAM disk image both fit on one floppy.
- 1** Prompt. This is the default.

ramdisk_size=n

Specify the amount of memory, in kilobytes, to be allocated for the RAM disk. The default is 4096, which allocates 4 MB.

ramdisk_start=offset

Used for a Linux installation in which both the kernel and the RAM disk image are on the same floppy. *offset* indicates the offset on the floppy where the RAM disk image begins; it is specified in kilobytes.

read-only

Specify that the root filesystem should be mounted read-only for filesystem checking (`fsck`), after which it is typically remounted read/write.

read-write

Specify that the root filesystem should be mounted read/write.

root=root-device

Specify the device that should be mounted as root. If the special name **current** is used as the value, the root device is set to the device on which the root filesystem currently is mounted. Defaults to the root-device setting contained in the kernel image.

vga=mode

Specify the VGA text mode that should be selected when booting. The mode defaults to the VGA mode setting in the kernel image. The values are case-insensitive. They are:

ask

Prompt the user for the text mode. Pressing Enter in response to the prompt displays a list of the available modes.

extended (or ext)

Select 80×50 text mode.

normal

Select normal 80×25 text mode.

number

Use the text mode that corresponds to *number*. A list of available modes for your video card can be obtained by booting with `vga=ask` and pressing Enter.

The lilo Command

You need to run the **lilo** command to install the LILO boot loader and to update it whenever the kernel changes or to reflect changes to `/etc/lilo.conf`. Note that if you replace your kernel image without rerunning **lilo**, your system may be unable to boot.

The path to the **lilo** command is usually `/sbin/lilo`. The syntax of the command is:

```
lilo [options]
```

Some of the options correspond to `/etc/lilo.conf` keywords:

Configuration keyword	Command option
<code>boot=bootdev</code>	<code>-b bootdev</code>
<code>compact</code>	<code>-c</code>
<code>delay=tsecs</code>	<code>-d tsecs</code>
<code>default=label</code>	<code>-D label</code>
<code>disktab=file</code>	<code>-f file</code>
<code>install=bootsector</code>	<code>-i bootsector</code>
<code>lba32</code>	<code>-L</code>

Configuration keyword	Command option
<code>linear</code>	<code>-l</code>
<code>map=mapfile</code>	<code>-m mapfile</code>
<code>fix-table</code>	<code>-P fix</code>
<code>ignore-table</code>	<code>-P ignore</code>
<code>backup=file</code>	<code>-s file</code>
<code>force-backup=file</code>	<code>-S file</code>
<code>verbose=level</code>	<code>-v</code>

These options should be put in the configuration file whenever possible; putting them on the `lilo` command line instead of in `/etc/lilo.conf` is deprecated. The next section describes those options that can be given only on the `lilo` command line; the others were described earlier.

lilo Command Options

The following list describes `lilo` command options that are available only on the command line. Multiple options are given separately; for example:

- `$ lilo -q -v`
- C config-file**
Specify an alternative to the default configuration file (`/etc/lilo.conf`). `lilo` uses the configuration file to determine which files to map when it installs LILO.
- I label**
Print the path to the kernel specified by *label* to standard output, or an error message if no matching label is found. For example:
`$ lilo -I linux`
`/boot/vmlinuz-2.0.34-0.6`
- q** List the currently mapped files. `lilo` maintains a file (`/boot/map` by default) containing the name and location of the kernel(s) to boot. Running `lilo` with this option prints the names of the files in the map file to standard output, as in this example (the asterisk indicates that `linux` is the default):
`$ lilo -q`
`linux *`
`test`
- r root-directory**
Specify that before doing anything else, `lilo` should `chroot` to the indicated directory. Used for repairing a setup from a boot CD or floppy; you can boot from that disk but have `lilo` use the boot files from the hard drive. For example, if you issue the following commands, `lilo` will get the files it needs from the hard drive:
`$ mount /dev/hda2 /mnt`
`$ lilo -r /mnt`
- R command-line**
Set the default command for the boot loader the next time it executes. The command executes once and then is removed by the boot loader. This option typically is used in reboot scripts, just before calling `shutdown -r`.

- t Indicate that this is a test—do not really write a new boot sector or map file. Can be used with -v to find out what **lilo** would do during a normal run.
- u *device-name* Uninstall **lilo** by restoring the saved boot sector from */boot/boot.nnnn*, after validating it against a timestamp. *device-name* is the name of the device on which LILO is installed, such as */dev/hda2*.
- U *device-name* Like -u, but do not check the timestamp.
- V Print the **lilo** version number.

LIFO Boot Errors

As LILO loads itself, it displays the letters of the word LILO, one at a time as it proceeds. Once LILO is correctly loaded, you'll see the full word printed on the screen. If nothing prints, then LILO has not been loaded at all; most likely LILO isn't installed or it is installed, but on a partition that is not active. If LILO started loading, but there was a problem, you can see how far it got by how many letters printed:

- L The first stage boot loader is loaded and running, but it can't load the second stage. There should be an error code indicating the type of problem; usually the problem is a media failure or bad disk parameters. See the LILO User's Guide for the meaning of the error codes.
- LI The first stage boot loader loaded the second stage but was not able to run it. The problem is most likely bad disk parameters or the file */boot/boot.b* (the boot sector) was moved but the **lilo** command wasn't run.

LIL

The second stage boot loader was run, but it couldn't load the descriptor table from the map file. This is usually caused by a media failure or bad disk parameters.

LIL?

The second stage boot loader was loaded at an incorrect address, probably because of bad disk parameters or by moving */boot/boot.b* without running **lilo**.

LIL-

The descriptor table is corrupt. The problem is probably bad disk parameters or moving */boot/map* without running **lilo**.

LIGO

LILO was successfully loaded.

GRUB: The Grand Unified Bootloader

Like LILO, the GRUB boot loader can load other operating systems in addition to Linux. GRUB has become the default bootloader for most Linux variants. It was written by Erich Boleyn to boot operating systems on PC-based hardware and is now developed and maintained by the GNU project. GRUB was intended to boot operating systems that conform to the Multiboot Specification, which was

designed to create one booting method that would work on any conforming PC-based operating system. In addition to multiboot-conforming systems, GRUB can boot directly into Linux, FreeBSD, OpenBSD, and NetBSD. It can also boot other operating systems such as Microsoft Windows indirectly, through the use of a *chainloader*. The chainloader loads an intermediate file, and that file loads the operating system's boot loader.

GRUB provides a graphical menu interface. It also provides a command interface that is accessible both while the system is booting (the native command environment) and from the command line once Linux is running.

While LILO works perfectly well, especially if you usually boot the default image, GRUB has some advantages. The graphical menu interface shows you exactly what your choices are for booting, so you don't have to remember them. It also lets you easily edit an entry on the fly, or drop down into the command interface. In addition, if you are using the menu interface and something goes wrong, GRUB automatically puts you into the command interface so you can attempt to recover and boot manually. Another advantage of GRUB is that if you install a new kernel or update the configuration file, that's all you have to do; with LILO, you also have to remember to rerun the `lilo` command to reinstall the boot loader. On the other hand, if you are used to LILO, don't need to see the prompts often, and have a stable system, LILO is quick and convenient.

A GRUB installation consists of at least two and sometimes three executables, known as stages. The stages are:

Stage 1

Stage 1 is the piece of GRUB that resides in the MBR or the boot sector of another partition or drive. Since the main portion of GRUB is too large to fit into the 512 bytes of a boot sector, Stage 1 is used to transfer control to the next stage, either Stage 1.5 or Stage 2.

Stage 1.5

Stage 1.5 is loaded by Stage 1 only if the hardware requires it. Stage 1.5 is filesystem-specific; that is, there is a different version for each filesystem that GRUB can load. The name of the filesystem is part of the filename (`e2fs_stage1_5`, `fat_stage1_5`, etc.). Stage 1.5 loads Stage 2.

Stage 2

Stage 2 runs the main body of the GRUB code. It displays the menu, lets you select the operating system to be run, and starts the system you've chosen.

If it was compiled with netboot support, GRUB can also be used to boot over a network. We don't describe that process here; see the file `netboot/README.netboot` in the GRUB source directory for detailed information.

One of the first things to understand about GRUB is that it uses its own naming conventions. Drives are numbered starting from 0; thus, the first hard drive is `hd0`, the second hard drive is `hd1`, the first floppy drive is `fd0`, and so on. Partitions are also numbered from 0, and the entire name is put in parentheses. For example, the first partition of the first drive, `/dev/hda1`, is known as `(hd0,0)` to GRUB, and the third partition of the second drive is `(hd1,2)`. GRUB makes no distinction between drive types; thus the first drive is `hd0` regardless of whether it is IDE, SCSI, or SATA.

Files are specified either by the filename or by *blocklist*, which is used to specify files such as chainloaders that aren't part of a filesystem. A filename looks like a standard Unix path specification with the GRUB device name prepended; for example:

```
(hdo,0)/grub/grub.conf
```

If the device name is omitted, the GRUB root device is assumed. The GRUB root device is the disk or partition where the kernel image is stored, set with the **root** command. See “GRUB Commands” on page 525 for the command descriptions.

When you use blocklist notation, you tell GRUB which blocks on the disk contain the file you want. Each section of a file is specified as the offset on the partition where the block begins plus the number of blocks in the section. The offset starts at 0 for the first block on the partition. The syntax for blocklist notation is:

```
[device][offset]+length[,offset]+length...
```

In this case, too, the device name is optional for a file on the root device. With blocklist notation, you can also omit the offset if it is 0. A typical use of blocklist notation is when using a chainloader to boot Windows. If GRUB is installed in the MBR, you can chainload Windows by setting the root device to the partition that has the Windows boot loader, making it the active partition, and then using the **chainloader** command to read the Windows boot sector:

```
rootnoverify (hdo,0)
makeactive
chainloader +1
```

In this example, the blocklist notation (+1) does not include either the device name or the offset because we set the root device to the Windows partition, and the Windows loader begins at offset 0 of that partition.

GRUB also includes a *device map*. The device map is an ASCII file, usually */boot/grub/device.map*. Since the operating system isn't loaded yet when you use GRUB to boot Linux (or any other operating system), GRUB knows only the BIOS drive names. The purpose of the device map is to map the BIOS drives to Linux devices. For example:

```
(fd0)    /dev/fd0
(hdo)    /dev/hda
```

Installing GRUB

Installing GRUB involves two stages. First, you install the GRUB files on your system, either by compiling and installing the source tarball or from a package. That puts the GRUB files in the correct locations on your system. The second step is to install the GRUB software as your boot manager. This is the step we describe in this section.

If you installed GRUB as part of your Linux installation, the distribution's installation program took care of both stages of installing GRUB, and you'll see the GRUB menu when you boot Linux. If you didn't install GRUB as part of your Linux installation, you have two choices. The easiest way to install GRUB is with the **grub-install** shell script that comes with GRUB. If **grub-install** doesn't work,

or if you want to do the installation manually, you can run the **grub** command and issue the installation commands yourself.

The following sections describe how to create a GRUB boot CD, a GRUB boot floppy, and how to install GRUB. You can create a GRUB boot disk for everyday use or to have for an emergency.

Creating a GRUB boot CD

The following instructions make a CD that boots to GRUB:

1. Make a directory that will hold the GRUB iso image to be written to CD:
`$ mkdir -p grubiso/boot/grub # make parent dirs if needed`
2. Copy the file *stage2_eltorito** to the new directory from the directory where GRUB was installed (in this example */usr/lib/grub/x86_64-pc*):
`$ cp /usr/lib/grub/x86_64-pc grubiso/boot/grub`

You can move other files to the directory as well, such as *menu.lst* to display the menu when you boot.

3. Run **genisoimage** to make an ISO9660 image file, *grub.iso*:
`$ genisoimage -R -b boot/grub/stage2_eltorito -no-emul-boot \
-boot-load-size 4 -boot-info-table -o grub.iso grubiso`

This command takes the contents of *grubiso/boot/grub* (only the top of the directory tree needs to be specified) and makes the image file *grub.iso*. See the **genisoimage** command in Chapter 3 for information on the options.

4. The image file can now be burned onto a CD (or DVD) with the burning software of your choice.

Creating a GRUB boot floppy

The following instructions make a floppy that boots to the GRUB command line:

1. From the directory where GRUB was installed (e.g., */usr/share/grub/i386-pc*), use the **dd** command to write the file *stage1* to the floppy:
`$ dd if=stage1 of=/dev/fd0 bs=512 count=1`

This command writes one block, with a block size of 512, from the input file *stage1* to the floppy device */dev/fd0*.

2. Now write the file *stage2* to the floppy, skipping over the first block (**seek=1**) so you don't overwrite *stage1*:
`$ dd if=stage2 of=/dev/fd0 bs=512 seek=1`

Put together, the process looks like this:

```
$ dd if=stage1 of=/dev/fd0 bs=512 count=1
1+0 records in
1+0 records out
$ dd if=stage2 of=/dev/fd0 bs=512 seek=1
254+1 records in
254+1 records out
```

* El Torito is a specification that lets you create a bootable CD.

The boot floppy is now ready to boot to the GRUB command line.

You can also make a boot floppy that boots to the GRUB menu:

1. Create a GRUB configuration file (*/boot/grub/menu.lst*) if you don't already have one. The configuration file is described later in "The GRUB Configuration File" on page 521.
2. Create a filesystem on your floppy disk. For example:

```
$ mke2fs /dev/fdo
```
3. Mount the floppy drive and create the directory */boot/grub*:

```
$ mount /mnt  
$ mkdir /mnt/boot  
$ mkdir /mnt/boot/grub
```

4. Copy the *stage1*, *stage2*, and *grub.conf* GRUB images from */boot/grub* on your Linux partition to */mnt/boot/grub*.
5. Run the **grub** command. This example assumes the command is in */sbin/grub*, but it might be in */usr/sbin/grub* on your system:

```
$ /sbin/grub --batch <<EOT  
root (fd0)  
setup (fd0)  
quit  
EOT
```

You should now be able to boot to the GRUB menu from the floppy disk you just created.

Using grub-install

GRUB comes with a shell script, **grub-install**, which uses the GRUB shell to automate the installation. The command syntax is:

```
grub-install options install-device
```

where *install-device* is the name of the device on which you want to install GRUB, specified as either the GRUB device name (e.g., *(hd0)*) or the system device (e.g., */dev/hda*). For example, you might issue the following command (as root):

```
$ grub-install /dev/hda
```

This command installs GRUB into the MBR of the first hard drive. The **grub-install** options are:

--force-lba

Force GRUB to use LBA mode, to allow booting from partitions beyond cylinder 1024.

--grub-shell=*file*

Specify that *file* is to be used as the GRUB shell. You might want to use this option to append options to **grub**. For example:

```
$ grub-install --grub-shell="grub --read-only" /dev/fdo
```

-h, --help

Print a help message on standard output and exit.

--recheck

Force probing of a device map. You should run **grub-install** with this option if you add or remove a disk from your system. The device map is found at */boot/grub/device.map*.

--root-directory=dir

Install GRUB images in the directory *dir* instead of the GRUB root directory.

-v, --version

Print the GRUB version number to standard output and exit.

Installing from the GRUB command line

To install GRUB from the native command environment, make a GRUB boot disk as described previously. You will use that disk to boot to the GRUB command line to do the installation. If you know which partition holds the GRUB files, you're all set. Otherwise, you can find the partition with the **find** command:

```
grub> find /boot/grub/stage1  
(hd0,0)
```

Here, the files are on (hd0,0). Use that information to set the GRUB root device:

```
grub> root (hd0,0)
```

Run the **setup** command to install GRUB. To install GRUB on the MBR, run **setup** as follows:

```
grub> setup (hd0)
```

If you are going to chainload Linux and want to install GRUB on the boot sector of the Linux partition, run **setup** like this:

```
grub> setup (hd0,0)
```

The GRUB Configuration File

GRUB uses a configuration file that sets up the menu interface. The configuration file is called *menu.lst* and is found with the other GRUB files in the */boot/grub* directory. In some distributions (e.g., Fedora and Red Hat) the configuration file is called *grub.conf*, which is a symbolic link to *menu.lst*.

The configuration file begins with a section containing global commands that apply to all boot entries, followed by an entry for each Linux image or other operating system that you want to be able to boot. Here is an example of a global section (a hash sign, #, begins a comment):

```
default=0                                # default to the first entry  
timeout=20                               # set the timeout to 20 seconds  
splashimage=(hd0,0)/grub/splash.xpm.gz    # the splash image displayed with  
                                              # the menu
```

Certain GRUB commands are available only in the global section of the configuration file, for use with the GRUB menu. These commands are described in the following list. All other commands can be used either in the configuration file or on the command line and are described later in “GRUB Commands” on page 525.

default num

Set the default menu entry to *num*. The default entry is started if the user does not make a selection before the timeout time. Menu entries are numbered from 0. If no default is specified, the first entry (0) is used as the default.

fallback num

Specify the entry to be used if for any reason the default entry has errors. If this command is specified and the default doesn't work, GRUB boots the fallback entry automatically instead of waiting for user input.

hiddenmenu

Specify that the menu is not to be displayed. The user can press Esc before the end of the timeout period to have the menu displayed; otherwise, the default entry is booted at the end of the timeout.

timeout time

Specify the timeout period, in seconds. The timeout is the amount of time GRUB waits for user input before booting the default entry.

title name

Start a new boot entry with specified *name*.

Following the global section, the configuration file includes an entry for each boot image. An entry begins with a **title** command that specifies the text that will appear on the menu for that entry when the system boots. A typical boot entry might look like this:

```
title Linux 2.6.28
root (hd0,1)
kernel /vmlinuz-2.6.28 ro root=LABEL=/
initrd /initrd-2.6.28
```

This entry provides the information GRUB needs to boot to Linux. When the menu is displayed, it will include an entry that says:

```
Linux 2.6.28
```

The GRUB root is on the second partition of the first hard drive (hd0,1). The **kernel** command specifies which Linux kernel to run and passes some parameters to the kernel, and the **initrd** command sets up an initial RAM disk.

The configuration file also provides some security features, such as the ability to set passwords and to lock certain entries so only the root user can boot them. The configuration file can be set up so that a password is required to run interactively (i.e., for editing menu entries or using the command interface) or simply to protect certain menu entries while leaving other entries available to all users. See the explanation of the **password** and **lock** commands in “GRUB Commands” on page 525.

In addition to providing a password feature, GRUB provides the command **md5crypt** to encrypt passwords in MD5 format, and a corresponding Linux command, **grub-md5-crypt**. **grub-md5-crypt** is a shell script that acts as a frontend to the **grub** shell, calling **md5crypt**. Passwords encrypted either directly with **md5crypt** or with **grub-md5-crypt** can be used with the **password** command to set up a GRUB password. **grub-md5-crypt** has three possible options:

- help**
Print help message and exit.
- grub-shell=*file***
Specify that *file* is to be used as the GRUB shell.
- version**
Print version information and exit.

Using the Menu Interface

The most common way to use GRUB is with the menu interface. The Stage 2 loader reads the configuration file *menu.lst* and displays the menu. If a timeout is set in the configuration file, GRUB displays a countdown at the bottom of the window showing how much time is left before it boots to the default entry. Move the cursor to an entry and press Enter to boot; or, press **e** to edit the command line for that entry, **a** to modify the kernel arguments, or **c** to go to the command-line interface to issue commands manually.

If you go to the command line, you can return to the menu at any time by pressing Esc.

Selecting **a** and **e** are similar, except that **a** displays only the **kernel** command line and lets you append options to it, while **e** displays the entire boot entry for you to edit. In either case, the available editing commands are similar to those available on the shell command line. When you are through editing, press Esc to return to the main menu. Your changes take effect for this session only; the configuration file is not permanently changed.

One common use for editing a **kernel** command is to boot to single-user mode. To do that, select **a** from the menu and append the word “single” to the end of the **kernel** command. Then press Esc to return to the menu and select the entry.

The GRUB Shell

In addition to using the command line from within the GRUB menu interface (or booting directly to the command line), you can run a GRUB shell directly from the Linux command line with the **grub** command. For the most part, using the **grub** shell is the same as running in the native command-line environment. The major difference is that the shell uses operating system calls to emulate the BIOS calls that the native environment uses. That can lead to some differences in behavior.

The syntax of the **grub** command is:

```
grub [options]
```

For example:

```
$ grub --no-floppy
```

The **grub** command-line options are:

- batch**
Turn on batch mode for noninteractive use. Equivalent to **grub --no-config-file --no-curses --no-pager**.

- boot-drive=drive**
Use *drive* as the Stage 2 boot drive, specified as a decimal, hexadecimal, or octal integer. The default is hexadecimal 0x0.
 - config-file=file**
Use *file* as the GRUB configuration file. The default is */boot/grub/menu.lst*.
 - device-map=file**
Use *file* for the device map. The value of *file* is usually */boot/grub/device.map*.
 - help**
Display a help message to standard output and exit.
 - hold**
Wait for a debugger to attach before starting **grub**.
 - install-partition=partition**
Use *partition* as the Stage 2 installation partition, specified as a decimal, hexadecimal, or octal number. The default is hexadecimal 0x20000.
 - no-config-file**
Run without reading the configuration file.
 - no-curses**
Don't use the **curses** interface for managing the cursor on the screen.
 - no-floppy**
Don't probe for a floppy drive. This option is ignored if **--device-map** is also specified.
 - no-pager**
Don't use the internal pager.
 - preset-menu**
Use a preset menu, for example if your system has no console and you need to get a serial terminal set up to see messages. To use this option, compile GRUB with the **--enable-preset-menu=file** option and create a menu file. See the GRUB documentation for more information.
 - probe-second-floppy**
Probe the second floppy drive (which is not probed by default). This option is ignored if **--device-map** is also specified.
 - read-only**
Do not write to any disk drives.
 - verbose**
Print verbose messages.
 - version**
Print version information and exit.
- When you run **grub**, you will see something like this:
- ```
GRUB version 0.94 (640K lower / 3072K upper memory)

[Minimal BASH-like line editing is supported. For the first word, TAB lists possible command completions. Anywhere else TAB lists the possible completions of a device/filename.]
```
- grub>

You can now enter commands at the grub> prompt. Press Tab to get a brief help message, listing all the commands:

```
grub>
Possible commands are: blocklist boot cat chainloader cmp color configfile
debug device displayapm displaymem dump embed find ftest geometry halt help
hide impsprobe initrd install ioprobe kernel lock makeactive map md5crypt
module modulenounzip pager partnew parttype password pause quit read reboot
root rootnoverify savedefault serial setkey setup terminal testload testvbe
unhide uppermem vbeprobe
```

Using Tab is a quick way to remind yourself of the commands, but it can be confusing to see them all run together and wrapping across lines. You can also run the **help** command, which lists the most frequently used commands and their syntax:

|                                        |                                        |
|----------------------------------------|----------------------------------------|
| grub> <b>help</b>                      |                                        |
| blocklist FILE                         | boot                                   |
| cat FILE                               | chainloader [--force] FILE             |
| color NORMAL [HIGHLIGHT]               | configfile FILE                        |
| device DRIVE DEVICE                    | displayapm                             |
| displaymem                             | find FILENAME                          |
| geometry DRIVE [CYLINDER HEAD SECTOR [ | halt [--no-apm]                        |
| help [--all] [PATTERN ...]             | hide PARTITION                         |
| initrd FILE [ARG ...]                  | kernel [--no-mem-option] [--type=TYPE] |
| makeactive                             | map TO_DRIVE FROM_DRIVE                |
| md5crypt                               | module FILE [ARG ...]                  |
| modulenounzip FILE [ARG ...]           | pager [FLAG]                           |
| partnew PART TYPE START LEN            | parttype PART TYPE                     |
| quit                                   | reboot                                 |
| root [DEVICE [HDBIAS]]                 | rootnoverify [DEVICE [HDBIAS]]         |
| serial [--unit=UNIT] [--port=PORT] [-- | setkey [TO_KEY FROM_KEY]               |
| setup [--prefix=DIR] [--stage2=STAGE2_ | terminal [--dumb] [--timeout=SECS] [-- |
| testvbe MODE                           | unhide PARTITION                       |
| uppermem KBYTES                        | vbeprobe [MODE]                        |

You can add the **--all** option to see all the commands.

To get help for a specific command, add the command name (e.g., **help read**). **help** treats the text you enter as a pattern; therefore, if you enter **help find**, you'll get help for the **find** command, but if you enter **help module**, you'll get help for both **module** and **modulenounzip**.

## GRUB Commands

The following sections describe two sets of commands. Both can be used at the GRUB command line. In addition, the first set can be used in the global section of the menu, and the second can be used in individual menu entries. A few commands can be used only on the GRUB shell command line; this is noted in the command entry. The commands **default**, **fallback**, **hiddenmenu**, **timeout**, and **title** are available only in the configuration file, for use with the menu interface. They are described in “The GRUB Configuration File” on page 521.

When running commands, if you find that you aren't sure how to complete a pathname, you can use the Tab key to find the possible completions. For example:

```
grub> blocklist (hd0,1)/grub/[Tab]
Possible files are: grub.conf splash.xpm.gz menu.lst device.map stage1
stage2 e2fs_stage1_5 fat_stage1_5 ffs_stage1_5 jfs_stage1_5 minix_stage1_5
reiserfs_stage1_5 vstafs_stage1_5 xfs_stage1_5
grub> blocklist (hd0,1)/grub/stage2
(hd0,1)33306+24,33332+231
```

## Command-Line and Global Menu Commands

The commands available at the command line and in the global section of the configuration file are as follows.

---

**bootp**      `bootp [--with-configfile]`  
Initialize a network device via the Bootstrap Protocol (BOOTP). This command is available only if GRUB was compiled with netboot support. If **--with-configfile** is specified, GRUB automatically loads a configuration file specified by your BOOTP server.

---

**color**      `color normal [highlight]`  
Specify colors for the menu. *normal* represents the color used for normal menu text, while *highlight* represents the color used to highlight the line the cursor is on. Both *normal* and *highlight* are specified as two symbolic color names, for foreground and background color, separated by a slash. For example:

`color light-gray/blue cyan/black`

You can prefix the foreground color with **blink-** (e.g., **blink-cyan/red**) to get a blinking foreground. The colors **black**, **blue**, **green**, **cyan**, **red**, **magenta**, **brown**, and **light-gray** can be specified for foreground or background. Additional colors that can be used only for the foreground are **dark-gray**, **light-blue**, **light-green**, **light-cyan**, **light-red**, **light-magenta**, **yellow**, and **white**.

---

**device**      `device drive file`  
Specify a file to be used as a BIOS drive. This command is useful for creating a disk image and/or for fixing the drives when GRUB fails to determine them correctly. The **device** command is available only from within the **grub** shell, not from the native command line. For example:

```
grub> device (fd0) /floppy-image
grub> device (hd0) /dev/sd0
```

---

| <b>dhcp</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | <code>dhcp [--with-configfile]</code>                                                                                                                                                                                                                                                                                                        |      |        |      |   |                |   |                |   |        |     |                  |     |                  |     |        |      |                |      |            |      |         |      |         |      |        |      |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|--------|------|---|----------------|---|----------------|---|--------|-----|------------------|-----|------------------|-----|--------|------|----------------|------|------------|------|---------|------|---------|------|--------|------|
|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | Initialize a network device via the DHCP protocol. Currently, this command is just an alias for <b>bootp</b> and is available only if GRUB was compiled with netboot support. If specified with <b>--with-configfile</b> , GRUB will fetch and load a configuration file specified by your DHCP server.                                      |      |        |      |   |                |   |                |   |        |     |                  |     |                  |     |        |      |                |      |            |      |         |      |         |      |        |      |
| <b>hide</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | <code>hide <i>partition</i></code>                                                                                                                                                                                                                                                                                                           |      |        |      |   |                |   |                |   |        |     |                  |     |                  |     |        |      |                |      |            |      |         |      |         |      |        |      |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | Hide the specified partition. This is useful when you are booting Windows and there are multiple primary partitions on one disk. Hide all but the one you want to boot. Also see <b>unhide</b> .                                                                                                                                             |      |        |      |   |                |   |                |   |        |     |                  |     |                  |     |        |      |                |      |            |      |         |      |         |      |        |      |
| <b>ifconfig</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | <code>ifconfig [--server=server] [--gateway=gateway] [--mask=mask] [--address=address]</code>                                                                                                                                                                                                                                                |      |        |      |   |                |   |                |   |        |     |                  |     |                  |     |        |      |                |      |            |      |         |      |         |      |        |      |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | Configure a network device manually. If no options are specified, displays the current network configuration. With the server address, gateway, netmask, and IP address specified, <b>ifconfig</b> configures the device. The addresses must be in dotted decimal format (e.g., 192.168.0.4), and the options can be specified in any order. |      |        |      |   |                |   |                |   |        |     |                  |     |                  |     |        |      |                |      |            |      |         |      |         |      |        |      |
| <b>pager</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           | <code>pager [<i>flag</i>]</code>                                                                                                                                                                                                                                                                                                             |      |        |      |   |                |   |                |   |        |     |                  |     |                  |     |        |      |                |      |            |      |         |      |         |      |        |      |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | Enable or disable the internal pager by setting <i>flag</i> to <b>on</b> (enable) or <b>off</b> (disable).                                                                                                                                                                                                                                   |      |        |      |   |                |   |                |   |        |     |                  |     |                  |     |        |      |                |      |            |      |         |      |         |      |        |      |
| <b>partnew</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | <code>partnew <i>part</i> <i>type</i> <i>from</i> <i>to</i></code>                                                                                                                                                                                                                                                                           |      |        |      |   |                |   |                |   |        |     |                  |     |                  |     |        |      |                |      |            |      |         |      |         |      |        |      |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | Make a new primary partition, <i>part</i> , specified in GRUB syntax. <i>type</i> is the partition type, specified as a number in the range 0-0xff. <i>from</i> and <i>to</i> are the starting and ending sectors, specified as absolute numbers. Some of the common partition types are:                                                    |      |        |      |   |                |   |                |   |        |     |                  |     |                  |     |        |      |                |      |            |      |         |      |         |      |        |      |
| <table border="1"> <thead> <tr> <th>Type</th> <th>Number</th> </tr> </thead> <tbody> <tr> <td>None</td> <td>0</td> </tr> <tr> <td>FAT 16, lt 32M</td> <td>4</td> </tr> <tr> <td>FAT 16, gt 32M</td> <td>6</td> </tr> <tr> <td>FAT 32</td> <td>0xb</td> </tr> <tr> <td>FAT 32, with LBA</td> <td>0xc</td> </tr> <tr> <td>WIN 95, extended</td> <td>0xf</td> </tr> <tr> <td>EXT2FS</td> <td>0x83</td> </tr> <tr> <td>Linux extended</td> <td>0x85</td> </tr> <tr> <td>Linux RAID</td> <td>0xfd</td> </tr> <tr> <td>FreeBSD</td> <td>0xa5</td> </tr> <tr> <td>OpenBSD</td> <td>0xa6</td> </tr> <tr> <td>NetBSD</td> <td>0xfd</td> </tr> </tbody> </table> |                                                                                                                                                                                                                                                                                                                                              | Type | Number | None | 0 | FAT 16, lt 32M | 4 | FAT 16, gt 32M | 6 | FAT 32 | 0xb | FAT 32, with LBA | 0xc | WIN 95, extended | 0xf | EXT2FS | 0x83 | Linux extended | 0x85 | Linux RAID | 0xfd | FreeBSD | 0xa5 | OpenBSD | 0xa6 | NetBSD | 0xfd |
| Type                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   | Number                                                                                                                                                                                                                                                                                                                                       |      |        |      |   |                |   |                |   |        |     |                  |     |                  |     |        |      |                |      |            |      |         |      |         |      |        |      |
| None                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   | 0                                                                                                                                                                                                                                                                                                                                            |      |        |      |   |                |   |                |   |        |     |                  |     |                  |     |        |      |                |      |            |      |         |      |         |      |        |      |
| FAT 16, lt 32M                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | 4                                                                                                                                                                                                                                                                                                                                            |      |        |      |   |                |   |                |   |        |     |                  |     |                  |     |        |      |                |      |            |      |         |      |         |      |        |      |
| FAT 16, gt 32M                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | 6                                                                                                                                                                                                                                                                                                                                            |      |        |      |   |                |   |                |   |        |     |                  |     |                  |     |        |      |                |      |            |      |         |      |         |      |        |      |
| FAT 32                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | 0xb                                                                                                                                                                                                                                                                                                                                          |      |        |      |   |                |   |                |   |        |     |                  |     |                  |     |        |      |                |      |            |      |         |      |         |      |        |      |
| FAT 32, with LBA                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | 0xc                                                                                                                                                                                                                                                                                                                                          |      |        |      |   |                |   |                |   |        |     |                  |     |                  |     |        |      |                |      |            |      |         |      |         |      |        |      |
| WIN 95, extended                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | 0xf                                                                                                                                                                                                                                                                                                                                          |      |        |      |   |                |   |                |   |        |     |                  |     |                  |     |        |      |                |      |            |      |         |      |         |      |        |      |
| EXT2FS                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | 0x83                                                                                                                                                                                                                                                                                                                                         |      |        |      |   |                |   |                |   |        |     |                  |     |                  |     |        |      |                |      |            |      |         |      |         |      |        |      |
| Linux extended                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | 0x85                                                                                                                                                                                                                                                                                                                                         |      |        |      |   |                |   |                |   |        |     |                  |     |                  |     |        |      |                |      |            |      |         |      |         |      |        |      |
| Linux RAID                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | 0xfd                                                                                                                                                                                                                                                                                                                                         |      |        |      |   |                |   |                |   |        |     |                  |     |                  |     |        |      |                |      |            |      |         |      |         |      |        |      |
| FreeBSD                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | 0xa5                                                                                                                                                                                                                                                                                                                                         |      |        |      |   |                |   |                |   |        |     |                  |     |                  |     |        |      |                |      |            |      |         |      |         |      |        |      |
| OpenBSD                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | 0xa6                                                                                                                                                                                                                                                                                                                                         |      |        |      |   |                |   |                |   |        |     |                  |     |                  |     |        |      |                |      |            |      |         |      |         |      |        |      |
| NetBSD                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | 0xfd                                                                                                                                                                                                                                                                                                                                         |      |        |      |   |                |   |                |   |        |     |                  |     |                  |     |        |      |                |      |            |      |         |      |         |      |        |      |

---

|                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|-----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>parttype</b> | <code>parttype <i>part type</i></code><br>Change the type of partition <i>part</i> to <i>type</i> . The type must be a number in the range 0-0xff. See <b>partnew</b> for a list of partition types.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| <b>password</b> | <code>password [--md5] <i>passwd</i> [<i>file</i>]</code><br>Set a password for the menu interface. If used in the global section of the configuration file, outside the menu entries, GRUB prompts for a password before processing an <b>a</b> , <b>e</b> , or <b>c</b> entered by the user. Once the password <i>passwd</i> has been entered, if no <i>file</i> was specified, GRUB allows the user to proceed. Otherwise, GRUB loads the file as a new configuration file and restarts Stage 2. If <b>password</b> appears in an individual menu entry, GRUB prompts for the password before continuing. Specify <b>--md5</b> to tell GRUB that the password was encrypted with the <b>md5crypt</b> command.                                                                                                                                                                                                                                                                                                                                 |
| <b>rarp</b>     | <code>rarp</code><br>Initialize a network device via the Reverse Address Resolution Protocol (RARP). This command is available only if GRUB was compiled with netboot support. The use of RARP is deprecated.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| <b>serial</b>   | <code>serial [<i>options</i>]</code><br>Initialize a serial device. The serial port is not used for communication unless <b>terminal</b> is also specified. This command is available only if GRUB was compiled with serial support.<br><br><b>Options</b><br><b>--device=device</b><br>Specify the tty device to be used in the host operating system.<br>This option can be used only in the <b>grub</b> shell.<br><b>--parity=parity</b><br>Specify the parity. The possible values are <b>no</b> , <b>odd</b> , and <b>even</b> ; the default is <b>no</b> .<br><b>--port=port</b><br>Specify the I/O port. The value of <i>port</i> overrides any value specified for <b>--unit</b> .<br><b>--speed=speed</b><br>Specify the transmission speed (default is 9600).<br><b>--stop=num</b><br>Specify the number of stop bits. The value of <i>num</i> is either 1 or 2 (default is 1).<br><b>--unit=num</b><br>Specify the serial port to use. The value of <i>num</i> is a number in the range 0–3; the default is 0, corresponding to COM1. |

**--word=num**

Specify the number of data bits. The value of *num* is a number in the range 5–8 (default is 8).

**setkey****setkey [to-key from-key]**

Configure the keyboard map for GRUB by mapping the key *from-key* to the key *to-key*. With no mappings specified, reset the keyboard map. **setkey** is useful for setting up international keyboards. Possible key values are letters; digits; one of the strings **alt**, **backspace**, **capslock**, **control**, **delete**, **enter**, **escape**, **F<sub>n</sub>** (where *n* is one of the function key numbers), **shift**, **tab**; or one of the strings in the “Key value” columns of the following table.

| Key value          | Character | Key value           | Character |
|--------------------|-----------|---------------------|-----------|
| <b>ampersand</b>   | &         | <b>asterisk</b>     | *         |
| <b>at</b>          | @         | <b>backquote</b>    | `         |
| <b>backslash</b>   | \         | <b>bar</b>          |           |
| <b>braceleft</b>   | {         | <b>braceright</b>   | }         |
| <b>bracketleft</b> | [         | <b>bracketright</b> | ]         |
| <b>caret</b>       | ^         | <b>colon</b>        | :         |
| <b>comma</b>       | ,         | <b>dollar</b>       | \$        |
| <b>doublequote</b> | "         | <b>equal</b>        | =         |
| <b>exclam</b>      | !         | <b>greater</b>      | >         |
| <b>less</b>        | <         | <b>minus</b>        | -         |
| <b>numbersign</b>  | #         | <b>parenleft</b>    | (         |
| <b>parenright</b>  | )         | <b>percent</b>      | %         |
| <b>period</b>      | .         | <b>plus</b>         | +         |
| <b>question</b>    | ?         | <b>quote</b>        | '         |
| <b>semicolon</b>   | ;         | <b>slash</b>        | /         |
| <b>space</b>       |           | <b>tilde</b>        | ~         |
| <b>underscore</b>  | _         |                     |           |

**splashimage****splashimage file**

Use the image in *file* as the background (splash) image. The file should be a gzipped *.xpm* (X pixmap) file, created with a 14-color palette at 640 × 480 resolution and specified with standard GRUB device syntax:

```
splashimage=(hd0,0)/grub/splash.xpm.gz
```

Programs that you can use to create *.xpm* files include the GIMP, xv, and xpaint.

**terminal****terminal [options] [console] [serial]**

Specify a terminal for user interaction. This command is available only if GRUB was compiled with serial support. If both **console**

and **serial** are specified, GRUB uses the first terminal where a key is pressed, or the first after the timeout has expired. If neither is specified, GRUB displays the current setting.

### Options

#### --dumb

The terminal is a dumb terminal; if this option is not specified, the terminal is assumed to be VT100-compatible.

#### --lines=*num*

The terminal has *num* lines. The default is 24.

#### --silent

Suppress the prompt to hit any key (useful if your system does not have a terminal).

#### --timeout=*secs*

Specify the timeout in seconds.

---

|                   |                                                                                                                                                                                                                                                                                        |
|-------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>tftpserver</b> | <code>tftpserver <i>ipaddress</i></code>                                                                                                                                                                                                                                               |
|                   | Specify a TFTP server, overriding the address returned by a BOOTP, DHCP, or RARP server. The IP address must be specified in dotted decimal format. This command is available only if GRUB was compiled with netboot support. This command is deprecated; use <b>ifconfig</b> instead. |

---

|               |                                                                                                                                                                                                                     |
|---------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>unhide</b> | <code>unhide <i>partition</i></code>                                                                                                                                                                                |
|               | Unhide the specified partition. This is useful when booting DOS or Windows when there are multiple primary partitions on one disk. You can <b>unhide</b> the partition you want to boot and <b>hide</b> the others. |

---

## Command-Line and Menu-Entry Commands

The commands available at the command line and in the individual menu entries of the configuration file are as follows.

---

|                  |                                                                                                                        |
|------------------|------------------------------------------------------------------------------------------------------------------------|
| <b>blocklist</b> | <code>blocklist <i>file</i></code>                                                                                     |
|                  | Print the specified file in blocklist notation, where <i>file</i> is an absolute pathname or a blocklist. For example: |

```
grub> blocklist (hd0,1)/grub/grub.conf
(hd0,1)33746+2
```

---

|             |                                                                                                                                                   |
|-------------|---------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>boot</b> | <code>boot</code>                                                                                                                                 |
|             | Boot the operating system or chainloader that has been loaded. You need to run this command only if you are in the interactive command-line mode. |

|                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>cat</b>         | <code>cat <i>file</i></code><br>Display the contents of the specified file.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| <b>chainloader</b> | <code>chainloader [--force] <i>file</i></code><br>Load <i>file</i> as a chainloader. You can use blocklist notation to specify the first sector of the current partition with +1. If <b>--force</b> is specified, the file is loaded forcibly.                                                                                                                                                                                                                                                                                                                                                                                                                      |
| <b>cmp</b>         | <code>cmp <i>file1</i> <i>file2</i></code><br>Compare the two files <i>file1</i> and <i>file2</i> . Report differences by printing nothing if the files are identical, the sizes if they are different, or the bytes at an offset if they differ at that offset.                                                                                                                                                                                                                                                                                                                                                                                                    |
| <b>configfile</b>  | <code>configfile <i>file</i></code><br>Load <i>file</i> as the configuration file.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| <b>debug</b>       | <code>debug</code><br>Toggle debug mode, which prints extra messages to show disk activity. The default debug mode is off.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| <b>displayapm</b>  | <code>displayapm</code><br>Display Advanced Power Management (APM) BIOS information.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| <b>displaymem</b>  | <code>displaymem</code><br>Display the system address space map of the machine, including all regions of physical RAM installed. For example:<br><br><pre>grub&gt; displaymem displaymem EISA Memory BIOS Interface is present Address Map BIOS Interface is present Lower memory: 640K, Upper memory (to first chipset hole): 3072K [Address Range Descriptor entries immediately follow (values are 64-bit)] Usable RAM: Base Address: 0x0 X 4GB + 0x0, Length: 0x0 X 4GB + 0xa0000 bytes Reserved: Base Address: 0x0 X 4GB + 0xa0000, Length: 0x0 X 4GB + 0x60000 bytes Usable RAM: Base Address: 0x0 X 4GB + 0x100000, Length: 0x0 X 4GB + 0x300000 bytes</pre> |
| <b>dump</b>        | <code>dump <i>from</i> <i>to</i></code><br>Dump the contents of one file into another. The file you're dumping <i>from</i> is a GRUB file, and the file you're dumping <i>to</i> is an operating system file.                                                                                                                                                                                                                                                                                                                                                                                                                                                       |

---

|                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>embed</b>     | <code>embed <i>stage1.5 device</i></code><br>Embed the specified Stage 1.5 file in the sectors following the MBR if <i>device</i> is a drive, or in the boot loader area if it is an FFS (Berkeley Fast File System) partition (or, in the future, a ReiserFS partition). If successful, print the number of sectors the Stage 1.5 file occupies. You don't usually need to run this command directly.                                                                                     |
| <b>find</b>      | <code>find <i>file</i></code><br>Search all partitions for the specified file and print the list of devices where it was found. The filename specified should be an absolute filename, such as <i>/boot/grub/stage1</i> , or a blocklist.                                                                                                                                                                                                                                                  |
| <b>fstest</b>    | <code>fstest</code><br>Toggle the filesystem test mode, which prints data for device reads and the values being sent to the low-level routines. The <b>install</b> and <b>testload</b> commands turn off filesystem test mode. The test output is in the following format:<br><br><code>&lt;partition-offset-sector, byte-offset, byte-length&gt;</code><br>for high-level reads in a partition, and:<br><code>[disk-offset-sector]</code><br>for low-level sector requests from the disk. |
| <b>geometry</b>  | <code>geometry <i>drive [cylinder head sector [total_sector]]</i></code><br>Print geometry information for <i>drive</i> . From the GRUB shell, you can specify the number of cylinders, heads, sectors, and total sectors to set the drive's geometry. If <i>total_sector</i> is omitted, it is calculated from the other values.                                                                                                                                                          |
| <b>halt</b>      | <code>halt [--no-apm]</code><br>Shut down the computer. The computer is halted with an APM BIOS call unless the option <b>--no-apm</b> is specified.                                                                                                                                                                                                                                                                                                                                       |
| <b>help</b>      | <code>help [--all] [<i>patterns</i>]</code><br>Provide help for built-in commands. With no options, show the command and any options or parameters for the most common commands. With <b>--all</b> , show the same information for all possible commands. If you specify a pattern (i.e., a partial command name) or a full command name, a more complete description of the command or commands matching the pattern is displayed.                                                        |
| <b>impsprobe</b> | <code>impsprobe</code><br>Probe the Intel Multiprocessor Specification 1.1 or 1.4 configuration table and boot the CPUs that are found into a tight loop. This command can be used only in Stage 2.                                                                                                                                                                                                                                                                                        |

---

---

|                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>initrd</b>  | <code>initrd file [args]</code><br>Load an initial ramdisk <i>file</i> and pass any arguments.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| <b>install</b> | <p><code>install [options] stage1_file [d] dest_dev stage2_file [addr] [p] [config_file] [real_config_file]</code></p> <p>Perform a full GRUB install. See also the <b>setup</b> command, which acts as a frontend to <b>install</b> and is easier to use. The Stage 2 or Stage 1.5 file (both referred to as <i>stage2_file</i> here because they are loaded the same way) must be in its final install location (e.g., in the <i>/boot/grub</i> directory). <b>install</b> loads and validates <i>stage1_file</i>, installs a blocklist in the Stage 1 file for loading <i>stage2_file</i> as Stage 2 or Stage 1.5, and writes the completed Stage 1 file to the first block of the device <i>dest_dev</i>.</p> |

### Options

#### `--force-lba`

If the BIOS has LBA support but might return the incorrect LBA bitmap (which sometimes happens), `--force-lba` forces **install** to ignore the incorrect bitmap.

#### `--stage2=os_stage2_file`

This option is required to specify the operating system name of the Stage 2 file if the filesystem where it is located cannot be unmounted.

### Parameters

#### `addr`

Specify the address at which Stage 1 is to load Stage 2 or Stage 1.5. The possible values are 0x8000 for Stage 2 and 0x2000 for Stage 1.5. If omitted, GRUB determines the address automatically.

#### `config_file`

Specify the location of the configuration file for Stage 2.

- d** Tell Stage 1 to look for the actual disk on which *stage2\_file* was installed if it's not on the boot drive.

#### `dest_dev`

Specify the destination device. The final Stage 1 file is written to this device.

- p** If present, the partition where *stage2\_file* is located is written into the first block of Stage 2.

#### `real_config_file`

If *stage2\_file* is really a Stage 1.5 file, *real\_config\_file* specifies the real configuration filename and is written into the Stage 2 configuration file.

#### `stage1_file`

Specify the Stage 1 file to be written.

#### `stage2_file`

Specify the file that Stage 1 is to load for Stage 2.

---

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>ioprobe</b>       | <code>ioprobe drive</code><br>Probe the I/O ports used for <i>drive</i> and write the results to standard output.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| <b>kernel</b>        | <code>kernel [--non-mem-option] file [...]</code><br>Load the kernel image from <i>file</i> . Any text following <i>file</i> is passed on as the kernel command line. After running this command, you must reload any modules. The option <b>--type</b> specifies the kernel type and is required only for loading a NetBSD ELF kernel; GRUB automatically determines other types. The possible values of type are <b>linux</b> , <b>biglinux</b> , <b>freebsd</b> , <b>multiboot</b> , <b>netbsd</b> , and <b>openbsd</b> . For Linux, <b>--no-mem-option</b> tells GRUB not to pass the <b>mem=</b> option to the kernel. |
| <b>lock</b>          | <code>lock</code><br>Lock the entry until a valid password is entered. This is used in a menu entry immediately after <b>title</b> to prevent nonroot users from executing the entry. This command is most useful in conjunction with the <b>password</b> command.                                                                                                                                                                                                                                                                                                                                                          |
| <b>makeactive</b>    | <code>makeactive</code><br>Set the active partition on the root disk to GRUB's root device. Use only on primary PC hard disk partitions.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| <b>map</b>           | <code>map to from</code><br>Map the <i>from</i> drive to the <i>to</i> drive. You need to do this when chain-loading an operating system such as Windows, if it is not on the first drive. For example, if Windows is on (hd1):<br><pre>grub&gt; map (hd0) (hd1) grub&gt; map (hd1) (hd0)</pre> This swaps the mappings of the first and second hard drives, tricking Windows into thinking it's on the first drive so it can boot.                                                                                                                                                                                         |
| <b>md5crypt</b>      | <code>md5crypt</code><br>Prompt for a password and encrypt it in MD5 format for use with the <b>password</b> command.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| <b>module</b>        | <code>module file [...]</code><br>Load the boot module <i>file</i> for a multiboot format boot image. Anything after the filename is passed as the module command line.                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| <b>modulenounzip</b> | <code>modulenounzip files</code><br>Like <b>module</b> , except that automatic decompression is disabled.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |

---

---

|                                                                                                                                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|-------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>pause</b>                                                                                                                                          | pause <i>messages</i><br>Print the specified message and wait for a key to be pressed before continuing.                                                                                                                                                                                                                                                                                                                                    |
| <b>quit</b>                                                                                                                                           | quit<br>Used only from within the <b>grub</b> shell to exit from the shell. In the native command environment, use <b>reboot</b> instead to reboot the computer.                                                                                                                                                                                                                                                                            |
| <b>read</b>                                                                                                                                           | read <i>addr</i><br>Read a 32-bit value from memory at the specified address and display it in hex.                                                                                                                                                                                                                                                                                                                                         |
| <b>reboot</b>                                                                                                                                         | reboot<br>Reboot the system.                                                                                                                                                                                                                                                                                                                                                                                                                |
| <b>root</b>                                                                                                                                           | root <i>device</i> [ <i>hdbias</i> ]<br>Set the root device to the specified <i>device</i> and attempt to mount it to get the partition size (and some additional information for booting BSD kernels). If you are booting a BSD kernel, you can specify <i>hdbias</i> to tell the kernel how many BIOS drive numbers are before the current one.                                                                                           |
| <b>rootnoverify</b>                                                                                                                                   | rootnoverify <i>device</i> [ <i>hdbias</i> ]<br>Similar to <b>root</b> , but don't attempt to mount the partition. Used when you are booting a non-GRUB-readable partition such as Windows.                                                                                                                                                                                                                                                 |
| <b>savedefault</b>                                                                                                                                    | savedefault<br>Save the current menu entry as the default. GRUB will default to that entry the next time you boot the system.                                                                                                                                                                                                                                                                                                               |
| <b>setup</b>                                                                                                                                          | setup [ <i>options</i> ] <i>install_device</i> [ <i>image_device</i> ]<br>Set up installation of GRUB and run the <b>install</b> command to actually install GRUB onto the device <i>install_device</i> . Find the GRUB images on <i>image_device</i> if it is specified; otherwise use the current root device as set by the <b>root</b> command. If <i>install_device</i> is a hard disk, embed a Stage 1.5 file in the disk if possible. |
| <b>Options</b>                                                                                                                                        |                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| <b>--force-lba</b>                                                                                                                                    |                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| Force <b>install</b> to use LBA mode. Specify this option if your BIOS supports LBA mode but you find that GRUB isn't working in LBA mode without it. |                                                                                                                                                                                                                                                                                                                                                                                                                                             |

**--prefix=dir**  
Specify the directory where the GRUB images are located. If not specified, GRUB searches for them in */boot/grub* and */grub*.

**--stage2=os\_stage2\_file**  
Passed to **install** to tell GRUB the operating system name of the Stage 2 file.

---

|                 |                                                                                                                                                                                                                                                                                                                                            |
|-----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>testload</b> | <code>testload file</code><br>Read the contents of a <i>file</i> in different ways and compare the results to test the filesystem code. If no errors are reported and the final output reports an equal value for the reported variables <i>i</i> and <i>filepos</i> , then the filesystem is consistent and you can try loading a kernel. |
| <b>testvbe</b>  | <code>testvbe mode</code><br>For a VBE (VESA BIOS Extension) BIOS, test the specified VESA BIOS extension mode. You should see an animation loop, which you can cancel by pressing any key.                                                                                                                                                |
| <b>uppermem</b> | <code>uppermem kbytes</code><br>Tell GRUB to assume that only the specified number of kilobytes of upper memory are installed. You should need to use this command only for old systems, where not all the memory may be recognized.                                                                                                       |
| <b>vbeprobe</b> | <code>vbeprobe [mode]</code><br>For a VBE BIOS, probe VESA BIOS extension information. If <i>mode</i> is specified, the output shows only information for that mode; otherwise, all available VBE modes are listed.                                                                                                                        |

---

## Dual-Booting Linux and Windows 2000/XP/Vista

As mentioned earlier, when you run Windows, its boot loader expects to be the one in charge; therefore, the standard way to dual-boot Windows and Linux is to add Linux as an option on the Windows boot menu. This section describes how to do that. The information provided here applies to Windows 2000 and Windows XP, which use the Windows NT loader **ntldr** (so called because it was developed for Windows NT). Windows Vista uses a different boot loader. If you want to set up Vista to dual-boot Linux, you can use the free download EasyBCD by Neosmart Technologies ([neosmart.net](http://neosmart.net)).

Note again that you do not need the information in this section if your Linux installation software set up the dual-booting for you, which it probably did.

To set up dual booting with the NT loader manually, you need to provide the loader with a copy of the Linux boot sector. We'll describe how to do that on a

computer running Windows with an NTFS filesystem (note that Windows should be installed on your system already). See the “Linux+NT-loader” mini-HOWTO for more information and other alternatives.

You should have a Linux boot floppy or CD available so that if necessary you can boot Linux before the Windows boot loader has been modified. You should also have a DOS-formatted floppy to transfer the boot sector to the Windows partition. If you are running LILO and it is already installed, you may need to modify */etc/lilo.conf* as described later. Otherwise, install LILO or GRUB to the boot sector of the Linux partition; once the Linux boot manager is installed and you have a configuration file, you can set up the system for dual booting.

The following instructions assume your Linux partition is on */dev/hda2*. If Linux is on another partition in your system, be sure to replace */dev/hda2* in the following examples with the correct partition. The instructions also assume that you have a floppy drive to make a diskette for transferring the boot sector to your NTFS filesystem. If you don’t have a floppy drive, you will have to use some other means of doing the transfer. If you have a FAT partition, you can mount that on Linux and transfer the file there. Other possibilities include putting it on a CD, transferring it over a network to another system while you reboot to Windows, or even emailing it to yourself and reading it from the Windows side.

1. If you are running LILO, specify the Linux root partition as your boot device in */etc/lilo.conf*. If you are editing */etc/lilo.conf* manually, your entry will look like this:

```
boot=/dev/hda2
```

and will be the same as the **root=** entry.

If you are running GRUB, make sure your configuration file, */boot/grub/menu.lst*, includes a menu entry for booting Linux. The exact values of the entries in the menu depend on the filename of the kernel image that you wish to boot. For example:

```
title Linux 2.6.28
root (hd0,1)
kernel /vmlinuz-2.6.28 ro root=LABEL=/
initrd /initrd-2.6.28
```

You can then skip to Step 3.

2. Run the **lilo** command to install LILO on the Linux root partition.
3. At this point, if you need to reboot Linux, you’ll have to use a boot floppy or CD because the NT loader hasn’t been set up yet to boot Linux.
4. From Linux, run the **dd** command to make a copy of the Linux boot sector:

```
$ dd if=/dev/hda2 of=/bootsect.lnx bs=512 count=1
```

This command copies one block, with a block size of 512 bytes, from the input file */dev/hda2* to the output file */bootsect.lnx*. Note that if you are running GRUB, the boot sector is actually the *stage1* file. (The output filename can be whatever makes sense to you; it doesn’t have to be *bootsect.lnx*.)

5. Copy *bootsect.lnx* to a DOS-formatted floppy disk if that is how you are going to transfer it to Windows:

```
$ mount -t msdos /dev/fdo /mnt
$ cp /bootsect.lnx /mnt
$ umount /mnt
```

6. Reboot the system to Windows and copy the boot sector from the floppy disk to the hard disk. You can drag and drop the file to the hard drive, or use the command line to copy the file, as in the following example:

```
C:> copy a:\bootsect.lnx c:\bootsect.lnx
```

It doesn't matter where on the hard drive you put the file because you'll tell the NT loader where to find it in step 8.

7. Modify the attributes of the file *boot.ini*\* to remove the system and read-only attributes so you can edit it:

```
C:> attrib -s -r c:\boot.ini
```

8. Edit *boot.ini* with a text editor to add the line:

```
C:\bootsect.lnx="Linux"
```

This line adds Linux to the boot menu and tells the NT boot loader where to find the Linux boot sector. You can insert the line anywhere in the [operating systems] section of the file. Its position in the file determines where it will show up on the boot menu when you reboot your computer. Adding it at the end, for example, results in a *boot.ini* file that looks something like this (the second multi(0) entry is wrapped to fit the margins of this page):

```
[boot loader]
timeout=30
default=multi(0)disk(0)rdisk(0)partition(1)\WINNT
[operating systems]
multi(0)disk(0)rdisk(0)partition(1)\WINNT="Windows NT Server Version 4.00"
multi(0)disk(0)rdisk(0)partition(1)\WINNT="Windows NT Server Version
4.00 [VGA mode]" /basevideo /sos
C:\bootsect.lnx="Linux"
```

If you want Linux to be the default operating system, modify the **default=** line:

```
default=C:\bootsect.lnx
```

9. Rerun **attrib** to restore the system and read-only attributes:

```
C:> attrib +s +r c:\boot.ini
```

Now you can shut down Windows and reboot. Windows will prompt you with a menu that looks something like this:

```
OS Loader V4.00
Please select the operating system to start:
Windows NT Workstation Version 4.00
Windows NT Workstation Version 4.00 [VGA mode]
Linux
```

\* *boot.ini* is the Windows counterpart to */etc/lilo.conf*. It defines which operating systems the Windows loader can boot.

Select Linux, and the NT loader will read the Linux boot sector and transfer control to LILO or GRUB on the Linux partition.

If you are using LILO and you later modify */etc/lilo.conf* or rebuild the kernel, you need to rerun the **lilo** command, create a new *bootsect.lnx* file, and replace the version of *bootsect.lnx* on the Windows partition with the new version. In other words, you need to rerun steps 2–6.



If you have any problems or you simply want to remove LILO or GRUB later, you can reverse the installation procedure: boot to Windows, change the system and read-only attributes on *boot.ini*, re-edit *boot.ini* to remove the Linux entry, save the file, restore the system and read-only attributes, and remove the Linux boot sector from the Windows partition.

## Boot-Time Kernel Options

The earlier sections of this chapter described some of the options you can specify when you boot Linux. There are many more options that can be specified. This section touches on the ways to pass options to the kernel and then describes some of the kinds of parameters you might want to use. The parameters in this section affect the kernel and therefore apply regardless of which boot loader you use.

If LILO is your boot loader, you can add to or override the parameters specified in */etc/lilo.conf* during the boot process as follows:

- If **prompt** is set in */etc/lilo.conf*, LILO always presents the boot prompt and waits for input. At the prompt, you can choose the operating system to be booted. If you choose Linux, you can also specify parameters.
- If **prompt** isn't set, press Ctrl, Shift, or Alt when the word "LILO" appears. The boot prompt will then appear. You also can press the Scroll Lock key before LILO is printed and not have to wait poised over the keyboard for the right moment.
- At the boot prompt, specify the system you want to boot, or press Tab to get a list of the available choices. You then can enter the name of the image to boot. For example:

```
LILO boot: <press Tab>
linux test winxp
boot: linux
```

You also can add boot command options:

```
boot: linux single
```

- If you don't provide any input, LILO waits the amount of time specified in the **delay** parameter and then boots the default operating system with the default parameters, as set in */etc/lilo.conf*.

If you are using GRUB, you can pass parameters to the kernel on the **kernel** command line, either in the configuration file or from the command-line interface. If you are booting from the GRUB menu, you can edit or add parameters by entering **e** or **a** when the menu appears.

Some of the boot parameters have been mentioned earlier. Many of the others are hardware-specific and are too numerous to mention here. For a complete list of parameters and a discussion of the booting process, see the “BootPrompt HOWTO.” Some of the parameters not shown earlier that you might find useful are listed next; many more are covered in the HOWTO. Most of the following parameters are used to provide information or instructions to the kernel, rather than to LILO or GRUB:

**acpi=off**

Disable ACPI (Advanced Configuration and Power Interface) if it was to be enabled. This is useful for debugging possible hardware problems.

**debug**

Print all kernel messages to the console.

**hd=cylinders,heads,sectors**

Specify the hard drive geometry to the kernel. Useful if Linux has trouble recognizing the geometry of your drive, especially if it’s an IDE drive with more than 1024 cylinders.

**load\_ramdisk=n**

Tell the kernel whether to load a RAM disk image for use during Linux installation. Values of *n* are:

- 0 Don’t try to load the image. This is the default.
- 1 Load the image from a floppy disk to the RAM disk.

**mem=size**

Specify the amount of system memory installed. Useful if your BIOS reports memory only up to 64 MB and your system has more memory installed. Specify as a number with **M** or **k** (case-insensitive) appended:

`mem=128M`

Because **mem** would have to be included on the command line for every boot, it often is specified on a command line saved with **lock** or with **append** to be added to the parameters passed to the kernel.

**noinitrd**

When set, disable the two-stage boot and preserve the contents of */dev/initrd* so the data is available after the kernel has booted. */dev/initrd* can be read only once, and then its contents are returned to the system.

**number**

Start Linux at the runlevel specified by *number*. A runlevel is an operating state that the system can be booted to, such as a multiuser system or a system configuration running the X Window System. A runlevel is generally one of the numbers from 1 to 6; the default is usually 3. On modern distributions using Upstart, the runlevels and their corresponding states are defined in the *ttyN* files in the directory */etc/event.d*. On older systems using SysVinit, the runlevels are defined in the file */etc/inittab*. See Chapter 2 for a discussion of the init process.

- ro** Mount the root filesystem read-only. Used for doing system maintenance, such as checking the filesystem integrity, when you don't want anything written to the filesystem.
- rw** Mount the root filesystem read/write. If neither **ro** nor **rw** is specified, the default value (usually **rw**) stored in the kernel image is used.

### single

Start Linux in single-user mode. This option is used for system administration and recovery. It gives you a root prompt as soon as the system boots, with minimal initialization. No other logins are allowed.

## initrd: Using a RAM Disk

Modern Linux distributions use a modular kernel, which allows modules to be added without requiring that the kernel be rebuilt. If your root filesystem is on a device whose driver is a module (as is frequently true of SCSI disks), you can use the **initrd** facility, which provides a two-stage boot process, to first set up a temporary root filesystem in a RAM disk containing the modules you need to add (e.g., the SCSI driver) and then load the modules and mount the real root filesystem. The RAM disk containing the temporary filesystem is the special device file */dev/initrd*.

Similarly, you need to use a RAM disk if your root partition uses the ext3 filesystem and ext3 was not compiled into the kernel image. In that case, the ext3 module must be loaded with **initrd**.

Before you can use **initrd**, both RAM disk support (**CONFIG\_BLK\_DEV\_RAM=y**) and initial RAM disk support (**CONFIG\_BLK\_DEV\_INITRD=y**) must be compiled into the Linux kernel. Then you need to prepare the normal root filesystem and create the RAM disk image. Your Linux distribution may have utilities to do some of the setup for you; for example, the Red Hat distribution comes with the **mkinitsrd** command, which builds the **initrd** image. For detailed information, see the **initrd** manpage and the file *initrd.txt* (the path may vary, but it is usually something like */usr/src/linux/Documentation/initrd.txt*).

Once your Linux system has been set up for **initrd**, you can do one of the following, depending on which boot loader you are using:

- If you are using LILO, add the **initrd** option to the appropriate image section:
 

```
image=/vmlinuz
 initrd=/boot/initrd # The file to load as the contents of /dev/initrd
 ...

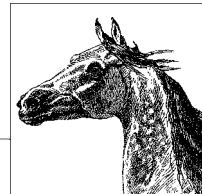
```
- Run the **/sbin/lilo** command, and you can reboot with **initrd**.
- If you are using GRUB, add the **initrd** option to the kernel line of the configuration-file boot entry, or to the **kernel** command if you are booting from the command-line interface:
 

```
kernel /vmlinuz-2.6.28 ro root=LABEL=/
 initrd /initrd-2.6.28

```

# 5

## Package Management



This chapter describes the two major Linux packaging systems: the Red Hat Package Manager (RPM) and the Debian GNU/Linux Package Manager. It also describes the major frontend applications designed to simplify and automate package management: **yum** for RPM-based systems, and **apt**, **aptitude**, and **synaptic** for Debian-based systems (**apt** is now also available for RPM-based systems).

When you install applications on your Linux system, most often you'll find a binary or a source package containing the application you want, instead of (or in addition to) a *.tar.gz* file. A package is a file containing the files necessary to install an application. However, while the package contains the files you need for installation, the application might require the presence of other files or packages that are not included, such as particular libraries (and even specific versions of the libraries), to actually be able to run. Such requirements are known as *dependencies*.

Package-management systems offer many benefits. As a user, you may want to query the package database to find out what packages are installed on the system and their versions. As a system administrator, you need tools to install and manage the packages on your system. And if you are a developer, you need to know how to build a package for distribution.

Among other things, package managers do the following:

- Provide tools for installing, updating, removing, and managing the software on your system.
- Allow you to install new or upgraded software directly across a network.
- Tell you what software package a particular file belongs to or what files a package contains.
- Maintain a database of packages on the system and their status, so you can determine what packages or versions are installed on your system.

- Provide dependency checking, so you don't mess up your system with incompatible software.
- Provide GPG, PGP, MD5, or other signature-verification tools.
- Provide tools for building packages.

Any user can list or query packages. However, installing, upgrading, or removing packages generally requires root privileges. This is because the packages normally are installed in system-wide directories that are writable only by root. Sometimes you can specify an alternate directory to install a package into your home directory or into a project directory where you have write permission, if you aren't running as root.

Signature verification is an important feature of package-management systems that helps maintain the security of your system. An MD5 checksum is used to check the integrity of a package, making sure, for example, that it was downloaded correctly and that it was not tampered with by a malicious user. GPG (and PGP) encrypt a digital signature into the package, which is used to verify the identity of the package creator.

Most often you'll install a binary package, in which the source code has been compiled and the software is ready to run once it is installed. You may also want or need to install source packages, which provide the source code and instructions for compiling and installing it. Source code packages do not contain executable files. Packages follow certain naming conventions, and you can tell from the name whether it is a binary or source package. RPM and Debian package names contain the same information, but they are expressed slightly differently. An RPM package has the form:

*package-version-release.architecture.rpm*

A Debian package has the form:

*package\_version-revision\_architecture.deb*

In both cases, *package* is the name of the package, *version* is the version number of the software, *release* (RPM) and *revision* (Debian) indicate the revision number of the package for that version, and *architecture* shows what system architecture the software was packaged for (e.g., **i386** or **amd64**). The value of *architecture* may also be **noarch** for a package that is not hardware-specific or **src** for an RPM source package (Debian source packages come as **tarred, gzipped** files).

All the package managers check for dependencies when you install a package. In the case of RPM, if there are missing dependencies, it prints an error and terminates without installing the package. To proceed, you need to first install the missing package (or packages). This can become an involved process if the missing package has its own dependencies. A major advantage of the high-level package managers described in this chapter (i.e., **yum**, **apt**, **aptitude**, and **synaptic**) is that they automatically resolve dependencies and install missing packages for you. Another advantage is that they locate and download the package automatically, based on information in configuration files specifying where to look for packages. With RPM, you first have to locate the package, then download it, and only then can you run RPM to do the install. On the other hand, if you already have the package file on your system or on a CD, RPM is quick and easy to run.

Both RPM and the **apt** system back up old files before installing an updated package. Not only does this let you go back if there is a problem, but it also ensures that you don't lose your changes (to configuration files, for example).

The following list shows the package-management programs described in the rest of this chapter. Which program to use is very much a matter of personal preference, and you can use more than one at different times. However, it's best to pick the program you prefer and use it consistently, so all your packages are maintained in a single database that you can query.

### **The Advanced Package Tool (APT)**

APT is a modern, user-friendly package-management tool that consists of a number of commands. The most frequently used of these commands is **apt-get**, which is used to download and install a Debian package. **apt-get** can be run from the command line or selected as a method from **dselect**.

Note that there are versions of the **apt** commands that can be used on an RPM-based system. If you plan to do that, it's best to install the version of **apt** that comes with your Linux distribution.

#### **aptitude**

High-level text-based interface to APT. Runs either from the command line or in a visual mode inside a terminal window such as an xterm.

#### **dpkg**

The original Debian packaging tool. Used to install or uninstall packages, or as a frontend to **dpkg-deb**. Getting and installing packages is usually done with **apt-get**, but **dpkg** is still commonly used to install a package that is already on your system. In fact, **apt-get** calls **dpkg** to do the installation once it's gotten the package.

#### **dpkg-deb**

Lower-level packaging tool. Used to create and manage the Debian package archives. Accepts and executes commands from **dpkg** or can be called directly.

#### **dselect**

An interactive frontend to **dpkg**. With the advent of the newer tools and the increased number of packages, the use of **dselect** is deprecated.

#### **synaptic**

A graphical frontend to APT.

#### **RPM**

The original command-line system for installing and managing RPM packages. RPM has two commands: **rpm** for installing and managing packages, and **rpmbuild** for creating packages.

#### **yum**

A frontend to RPM that runs from the command line.

Another RPM-based package manager, **up2date**, used to be the default for Red Hat Enterprise Linux systems. Red Hat has since switched to **yum** as the default, but **up2date** is still available if you prefer it. **up2date** has both command line and graphical interfaces, and like **yum**, it resolves dependencies as needed.

If you want to update your system regularly, to keep it current and to be sure you have the latest security fixes, you can set up a command that you can reissue at some regular interval (say, every day or once a week), or you can set it up as a **cron** job to run overnight daily or weekly. (See the descriptions of the **cron** and **crontab** commands in Chapter 3 for more information on setting up a **cron** job.)

You can set up your **cron** job to automatically download and install updated packages, but a safer approach is to have your job download the updates and email you a summary, leaving it up to you when and how to do the installation. This is particularly true in a production environment where you want to test changes thoroughly before incorporating them into your system.

## **Yum: Yellowdog Updater Modified**

Yum is a system for managing RPM packages, including installing, updating, removing, and maintaining packages; it automatically handles dependencies between packages. Yum is derived from **yup**, an updating system written for Yellow Dog Linux, an RPM-based PowerPC distribution. Yum downloads the information in the package headers to a directory on your system, which it then uses to make decisions about what it needs to do. Yum obtains both the headers and the RPMs themselves from a collection of packages on a server, known as a *repository*.

A repository consists of a set of RPM packages and the package headers, which are on a server that can be accessed via FTP or HTTP, from an NFS server, or from a local filesystem. A single server can contain one or multiple repositories; repositories are often mirrored on many servers, and you can configure **yum** to use multiple repositories. When they are downloaded to your system, the header and package files are maintained in */var/cache/yum*.

The configuration file, */etc/yum.conf*, is where you customize **yum**. It consists of two section types. The first section, [**main**], sets configuration defaults for **yum** operation. This section is followed by [**server**] sections, where each server is named according to the repository it specifies. For example, for Fedora, you might have [**base**] for the base Fedora repository and [**development**] for the development repository.

The server sections can also be stored, one to a file, in */etc/yum.repos.d*. **yum** comes with a default *yum.conf* file, which you can use as is or as a starting point from which to add additional repositories.

### **The yum Command**

The **yum** command is an automated system for updating **rpm**-based packages, particularly on Fedora and Red Hat Enterprise Linux. Yum can automatically install, upgrade, and remove packages. In addition to individual packages or a list of packages, **yum** can operate on an entire group of packages at a time.

When you run **yum**, it first updates the cache (unless you tell it not to with the **-C** option); then it proceeds to perform the requested operation.

The format of the **yum** command is:

```
yum [options] [command] [package ...]
```

Any general options are specified first, followed by a command telling **yum** what you want it to do, usually followed by a list of one or more packages. The **command** is always required, except with the **--help**, **-h**, and **--version** options.

Package names can be specified in various combinations of name, architecture, version, and release. For example, you could refer to the **bash** package as *bash*, *bash.x86\_64*, *bash-3.2*, *bash-3.2-30*, or *bash-3.2-30.fc10.x86\_64*.

## General options

The following general options can be set on the command line. For those that can also be set in the [**main**] section of the *yum.conf* configuration file, the name of the configuration option is given.

**-c [config-file]**

Specify the location of the **yum** configuration file. The file can be specified as a path to a local file or as an HTTP or FTP URL. The default is */etc/yum.conf*.

**-C** Run entirely from the local cache. Don't download or update headers unless required to complete the requested action.

**-d [num]**

Set the debug level to *num*, which is generally a number between 0 and 10, to specify how much debugging information to print. The configuration option is **debuglevel**.

**--disableexcludes=option**

Disable the excludes defined in *yum.conf*. The possible options are **all** to disable all excludes, **main** to disable only the excludes defined in [**main**] in *yum.conf*, or *repoid* to disable any excludes defined for the specified repository.

**--disableplugin=plugin**

Run with the specified plugins disabled, where *plugin* is a comma-separated list of plugins.

**--disablerepo=repoid**

Disable the repository specified by *repoid* so **yum** won't use it for this operation. The configuration option is **enabled**.

**-e [num]**

Set the error level to *num*, where *num* is a number, generally between 0 and 10. If the value is 0, print only critical errors. If it is 1, print all errors. Values greater than 1 mean print more errors, if there are any. The configuration option is **errorlevel**.

**--enablerepo=repoid**

Enable the specified repository that is marked as disabled (**enable=0**) in the configuration file. This allows the repository to be used for this operation. The configuration option is **enabled**.

**-h [command], --help [command]**

Display a help message and exit. With a command, display help for that command.

- installroot=*root***  
Specify an alternative root for package installation. All packages will be installed relative to *root*. The configuration option is **installroot**.
- nogpgcheck**  
Disable GPG signature checking. The configuration option is **gpgcheck**.
- nopugins**  
Disable all plugins. The configuration option is **plugins**.
- obsoletes**  
Enable obsoletes processing logic, taking into consideration packages that are obsoleted by other packages in the repository. Meaningful only with the **yum update** command. The configuration option is **obsoletes**.
- q, --quiet**  
Run without producing output. See also **-y**.
- R [minutes]**  
Set the maximum amount of time in minutes that **yum** will wait before performing a command.
- showduplicates**  
For the **info**, **list**, and **search** commands, show all matching packages, not just the latest versions.
- skipbroken**  
If **yum** finds dependency problems in a transaction, resolve them by removing the packages causing problems. The configuration option is **skip\_broken**.
- t, --tolerant**  
Currently does nothing. This option was originally intended to allow **yum** to keep going (be tolerant) in spite of any package errors on the command line. The configuration option is **tolerant**.
- v, --verbose**  
Display debugging information.
- version**  
Display the version of **yum** and exit.
- x [package], --exclude=package**  
Exclude the specified package from updates on all repositories. *package* can be given as a name or a glob. The configuration option is **exclude**.
- y** Assume that the answer to any question is yes. The configuration option is **assumeyes**.

## yum Command Summary

The individual **yum** commands are listed here.

---

### **check-update** check-update

Determine if updates are available, without running **yum** interactively. If any package updates are available, returns an exit value of

100 and a list of packages. If there are no updates, returns 0. Returns 1 on error.

---

|                     |                                                                                                                                                                                                                                     |
|---------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>clean</b>        | <code>clean [options]</code><br>Clean up the <b>yum</b> cache directory.                                                                                                                                                            |
|                     | <b>Options</b>                                                                                                                                                                                                                      |
|                     | <b>all</b><br>Clean everything: headers, packages, metadata, and the cache.                                                                                                                                                         |
|                     | <b>dbcache</b><br>Clean up the <b>sqlite</b> database cache, forcing <b>yum</b> to recreate it the next time it runs.                                                                                                               |
|                     | <b>headers</b><br>Remove all header files, forcing <b>yum</b> to download new headers the next time it runs.                                                                                                                        |
|                     | <b>metadata</b><br>Remove the metadata files, which maintain information about the packages such as package name, file size, description, dependencies, etc. The metadata will be downloaded again the next time <b>yum</b> is run. |
|                     | <b>packages</b><br>Remove cached packages from the system.                                                                                                                                                                          |
| <b>deplist</b>      | <code>deplist packages</code><br>Generate a list of dependencies for the specified packages, including what packages satisfy the dependencies.                                                                                      |
| <b>groupinfo</b>    | <code>groupinfo groups</code><br>Like <b>info</b> , but operates on package groups instead of individual packages.                                                                                                                  |
| <b>groupinstall</b> | <code>groupinstall groups</code><br><code>groupupdate groups</code><br>Like <b>install</b> , but operates on package groups instead of individual packages.                                                                         |
| <b>grouplist</b>    | <code>grouplist</code><br>Generate a list of installed and available groups to standard output. You can use these groups as input parameters to the other <b>group</b> commands, with their names in quotes (" ").                  |
| <b>groupremove</b>  | <code>groupremove groups</code><br>Like <b>remove</b> , but operates on package groups instead of individual packages.                                                                                                              |

---

|                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|---------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>help</b>         | <code>help [command]</code><br>Display help information for the specified command, or if no command is given, for all commands.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| <b>info</b>         | <code>info [options] [packages]</code><br>Display version information, a summary, and a description for each package, or for all packages if none is specified. See the <code>list</code> command for a description of the options.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| <b>install</b>      | <code>install packages</code><br>Install the latest version of a package or packages, ensuring that all dependencies are met. If no package matches the name as specified, the name is treated as a shell glob and any matches are installed.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| <b>list</b>         | <code>list [options] [packages]</code><br>Display a list of packages that match the <i>packages</i> specification and that are installed or available for installation.<br><br><b>Options</b><br><b>all</b> List all installed or available packages.<br><b>available</b> List packages on the repository that are available for installation.<br><b>extras</b> List packages on the system that are not available on any repository specified in the configuration file.<br><b>installed</b> List installed packages.<br><b>obsoletes</b> List installed packages that are made obsolete by any packages in any repository in the configuration file.<br><b>recent</b> List packages that have been recently added to any repository in the configuration file.<br><b>updates</b> List packages that have updates available for installation. |
| <b>localinstall</b> | <code>localinstall packages</code><br>Install the specified packages, which reside on the local system, rather than downloading them from a repository.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |

---

---

|                    |                                                                                                                                                                                                                                                                                                                                                       |
|--------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>localupdate</b> | <code>localupdate packages</code><br>Update the specified packages, which reside on the local system, rather than downloading them from a repository.                                                                                                                                                                                                 |
| <b>makecache</b>   | <code>makecache</code><br>Download and cache the metadata files from the repository. Once the cache has been built, you can use the <code>-C</code> option to run the commands that use the metadata ( <code>check-update</code> , <code>info</code> , <code>list</code> , <code>provides</code> , and <code>search</code> ) directly from the cache. |
| <b>provides</b>    | <code>provides feature1 [feature2 ...]</code><br><code>whatprovides feature1 [feature2 ...]</code><br>List packages that are available or installed that provide the specified features. The features can be specified as a name or as a wildcard in file-glob syntax format, and Perl or Python regular expressions can be used.                     |
| <b>reinstall</b>   | <code>reinstall package1 [package2 ...]</code><br>Reinstall the specified packages. The packages must already exist on the system.                                                                                                                                                                                                                    |
| <b>remove</b>      | <code>remove package1 [package2 ...]</code><br><code>erase package1 [package2 ...]</code><br>Remove the specified packages from the system. Also remove any packages that depend on the specified packages.                                                                                                                                           |
| <b>repolist</b>    | <code>repolist [option]</code><br>Generate a list of configured repositories. With no option or if the option is <code>all</code> , list all repositories. The other options are <code>disabled</code> to list disabled repositories, and <code>enabled</code> to list enabled repositories.                                                          |
| <b>resolvedep</b>  | <code>resolvedep dep1 [dep2 ...]</code><br>List the packages that provide the specified dependencies.                                                                                                                                                                                                                                                 |
| <b>search</b>      | <code>search string1 [string2 ...]</code><br>Find packages matching the specified string or strings in the description, summary, packager, or package name fields. Perl or Python regular expressions can be used for the strings. Useful for finding a package if you don't know the name.                                                           |
| <b>shell</b>       | <code>shell [filename]</code><br>Run an interactive <code>yum</code> shell, allowing multiple commands to be run within one <code>yum</code> execution. A filename can be specified that contains the commands to be run.                                                                                                                             |

---

---

|               |                                                                                                                                                                                                                                                                                          |
|---------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>update</b> | <code>update [packages]</code>                                                                                                                                                                                                                                                           |
|               | With no packages specified, update all installed packages. Otherwise, update the specified packages. In either case, <b>yum</b> makes sure that all dependencies are satisfied. If no package matches, the names specified are assumed to be shell globs, and any matches are installed. |
|               | With the <b>--obsoletes</b> option, <b>yum</b> includes package obsoletes in its calculations.                                                                                                                                                                                           |

---

|                |                                           |
|----------------|-------------------------------------------|
| <b>upgrade</b> | <code>upgrade [packages]</code>           |
|                | Equivalent to <b>update --obsoletes</b> . |

---

## Plugins and yum-utils

You can install plugins to extend the capabilities of **yum**. Plugins are Python programs that are installed in a directory specified with the **pluginpath** option in */etc/yum.conf*. In addition, to enable plugins, set the **plugins** option in */etc/yum.conf* to 1.

One set of such plugins is **yum-utils**, a collection of tools for managing packages and repositories. The following list briefly describes each tool. For more information on a tool, run it with the **-h** or **--help** option.

### **debuginfo-install**

Install **debuginfo** packages, which contain debugging information for programs.

### **package-cleanup**

Clean up packages, handling duplicates, orphaned packages, and dependency problems.

### **repoclosure**

Read the metadata of one or more repositories, check dependencies, and display a list of any unresolved dependencies.

### **repodiff**

Compare two or more repositories and display a list of added, removed, or changed packages.

### **repo-graph**

Display a full package dependency list in dot format.

### **repomanage**

Manage a directory of **rpm** packages, returning a list of the newest or oldest packages in a directory.

### **repoquery**

Query **yum** repositories for additional information.

### **repo-rss**

Generate an RSS feed from one or more repositories.

**reposync**

Synchronize a remote repository to a local directory, using **yum** to retrieve packages.

**repotrack**

Keep track of packages and their dependencies, and download them.

**yum-builddep**

Install missing dependencies to build specified packages.

**yum-complete-transaction**

Find incomplete or aborted transactions and try to complete them.

**yumdownloader**

Download binary and source packages from **yum** repositories for specified packages.

## The Red Hat Package Manager

The Red Hat Package Manager (RPM) is a freely available packaging system for software distribution and installation. In addition to the Red Hat Enterprise Linux and Fedora distributions, both SUSE and Mandriva are among the Linux distributions that use RPM.

Using RPM is straightforward. A single command, **rpm**, has options to perform all package-management functions except building packages.\* For example, to find out if the Emacs editor is installed on your system, you could enter:

```
$ rpm -q emacs
emacs-22.2-5.fc9.x86_64
```

This command prints the full package name, confirming its presence.

The **rpmbuild** command is used to build both binary and source packages.

## RPM Package Concepts

This section provides an overview of some of the parts of an RPM package. Much of the information is of primary use to developers, but because some of the terms are referenced in the RPM command descriptions, they are explained briefly here.

An RPM package has three primary components. The *header* contains all the information about the package, such as its name and version, a description, a list of included files, the copyright terms, and where the source file can be found. The *signature* contains information used to verify the integrity and authenticity of the package. The *archive* contains the actual files that make up the package.

When a package is being built, one of the requirements for its developers is to create a *spec* file. If you download the source **rpm** for a package, you can look at the spec file; it has a filename of *package.spec* (e.g., *yum.spec* for the **yum** spec file). The spec file contains all the information required to build a package,

\* In older versions of RPM, the build options were part of the **rpm** command.

including a description of the software, instructions telling the **rpmbuild** command how to build the package, and a list of the files included and where they get installed. Some other features of spec files include the following:

#### Macros

Macros are sequences of commands stored together and executed by invoking the macro name. The RPM build process provides two standard macros: **%setup** to unpack the original sources and **%patch** to apply patches. Other macros appear later in this chapter in the command descriptions and are described there.

#### Scripts

Scripts are used to control the build process. Some of the scripts RPM uses include **%prep** to begin the build process, **%build** primarily to run **make** and perhaps do some configuration, **%install** to do a **make install** and **%clean** to clean up afterward. Four additional scripts may be created to run when a package is actually installed on a system. These scripts are **%pre** for scripts run before package installation, **%post** for scripts run after package installation, **%preun** for scripts run before a package is uninstalled, and **%postun** for scripts run after a package is uninstalled.

#### Trigger scriptlets

Trigger scriptlets are extensions of the normal install and uninstall scripts. They provide for interaction between packages. A trigger scriptlet provided with one package will be triggered to run by the installation or removal of some other package. For example, a newly installed RPM package may cause an existing application to run or restart once installation is complete. In many cases, a newly installed package requires services to be restarted.

## The rpm Command

RPM packages are built, installed, and queried with the **rpm** command. RPM package filenames usually end with a *.rpm* extension. **rpm** has a set of modes, each with its own options. The format of the **rpm** command is:

```
rpm [options] [packages]
```

With a few exceptions, as noted in the lists of options that follow, the first option specifies the **rpm** mode (install, query, update, etc.), and any remaining options affect that mode.

Options that refer to packages are sometimes specified as *package-name* and sometimes as *package-file*. The package name is the name of the program or application, such as **xpdf**. The package file is the name of the RPM file, such as *xpdf-3.00-10.1.i386.rpm*.

RPM provides a configuration file for specifying frequently used options. The default global configuration file is usually */usr/lib/rpm/rpmrc*, the local system configuration file is */etc/rpmrc*, and users can set up their own *\$HOME/.rpmrc* files. You can use the **--showrc** option to show the values RPM will use by default for all the options that may be set in an *rpmrc* file:

```
rpm --showrc
```

The **rpm** command includes FTP and HTTP clients, so you can specify an *ftp://* or *http://* URL to install or query a package across the Internet. You can use an FTP or HTTP URL wherever *package-file* is specified in the commands presented here. Be careful, however, when downloading packages from the Internet. Always verify package contents by checking MD5 hashes and signatures. Whenever possible, install from trusted sites.

Any user can query the RPM database. Most of the other functions, such as installing and removing packages, require superuser privileges.

## General options

The following options can be used with all modes:

### **--dbpath** *path*

Use *path* as the path to the RPM database instead of the default */var/lib/rpm*.

### **-?, --help**

Print a long usage message (run **rpm** with no options for a shorter usage message).

### **--pipe** *command*

Pipe the query output to the specified command.

### **--quiet**

Display only error messages.

### **--rcfile** *filelist*

Get configuration from the files in the colon-separated *filelist*. If **--rcfile** is specified, the list must contain at least one file and that file must exist. *filelist* defaults to */usr/lib/rpm/rpmrc:/usr/lib/rpm/redhat/rpmrc:/etc/rpmrc:~/.rpmrc*. Use with **--showrc** to see what options will be used if alternate configuration files are specified.

### **--root** *dir*

Perform all operations within the directory tree rooted at *dir*.

### **-v** Verbose. Print progress messages.

### **--version**

Print the **rpm** version number.

### **-vv** Print debugging information.

## Install, upgrade, and freshen options

Use the **install** command to install or upgrade an RPM package. Upgrading with **install** leaves any existing versions on the system. The **install** syntax is:

```
rpm -i [install-options] package_file ...
rpm --install [install-options] package_file ...
```

To install a new version of a package and remove an existing version at the same time, use the **upgrade** command instead:

```
rpm -U [install-options] package_file ...
rpm --upgrade [install-options] package_file ...
```

If the package doesn't already exist on the system, `-U` acts like `-i` and installs it. To prevent that behavior, you can **freshen** a package instead; in that case, **rpm** upgrades the package only if an earlier version is already installed. The **freshen** syntax is:

```
rpm -F [install-options] package_file ...
rpm --freshen [install-options] package_file ...
```

For all forms, *package-file* can be specified as an FTP or HTTP URL to download the file before installing it. See “FTP/HTTP options” on page 563.

The installation and upgrade options are:

**--aid**

If **rpm** suggests additional packages, add them to the list of package files.

**--allfiles**

Install or upgrade all files.

**--badreloc**

Used with **--relocate** to force relocation even if the package is not relocatable.

**--excludedocs**

Don't install any documentation files.

**--excludepath** *path*

Don't install any file whose filename begins with *path*.

**--force**

Force the installation. Equivalent to **--replacepkgs** **--replacefiles** **--oldpackage**.

**-h, --hash**

Print 50 hash marks as the package archive is unpacked. Use this option with **-v** or **--verbose** for a nicer display.

**--ignorearch**

Install even if the binary package is intended for a different architecture.

**--ignoreos**

Install binary package even if the operating systems don't match.

**--ignoresize**

Don't check disk space availability before installing.

**--includedocs**

Install documentation files. This is needed only if **excludedocs: 1** is specified in an *rpmrc* file.

**--justdb**

Update the database only; don't change any files.

**--nodeps**

Don't check whether this package depends on the presence of other packages.

**--nodigest**

Don't verify package or header digests.

**--nomanifest**

Don't process nonpackage files as manifests.

**--noorder**

Don't reorder packages to satisfy dependencies before installing.

- nopost**  
Don't execute any post-install script.
- nopostun**  
Don't execute any post-uninstall script.
- nopre**  
Don't execute any pre-install script.
- nopreun**  
Don't execute any pre-uninstall script.
- noscripts**  
Don't execute any pre-install or post-install scripts. Equivalent to specifying --nopre --nopost --nopreun --nopostun.
- nosignature**  
Don't verify package or header signatures.
- nosuggest**  
Don't suggest packages that provide a missing dependency.
- nottriggerin**  
Don't execute any install trigger scriptlet.
- nottriggerun**  
Don't execute any uninstall trigger scriptlet.
- nottriggerpostun**  
Don't execute any post-uninstall trigger scriptlet.
- notriggers**  
Don't execute any scripts triggered by package installation. Equivalent to specifying --nottriggerin --nottriggerun --nottriggerpostun.
- oldpackage**  
Allow an upgrade to replace a newer package with an older one.
- percent**  
Print percent-completion messages as files are unpacked. Useful for running **rpm** from other tools.
- prefix *path***  
Set the installation prefix to *path* for relocatable binary packages.
- relocate *oldpath=newpath***  
For relocatable binary files, change all file paths from *oldpath* to *newpath*.  
Can be specified more than once to relocate multiple paths.
- replacefiles**  
Install the packages even if they replace files from other installed packages.
- replacepkgs**  
Install the packages even if some of them are already installed.
- test**  
Go through the installation to see what it would do, but don't actually install the package. This option lets you test for problems before doing the installation.

## Query options

The syntax for the **query** command is:

```
rpm -q [package-selection-options] [package-query-options]
rpm --query [package-selection-options] [package-query-options]
```

There are two subsets of query options. *Package-selection options* determine which packages to query, and *package-query options* determine which information to provide.

### Package-selection options

**package\_name**

Query the installed package *package\_name*.

**-a, --all**

Query all installed packages.

**-f file, --file file**

Find out which package owns *file*.

**--fileid md5**

Query package with the specified MD5 digest.

**-g group, --group group**

Find out which packages have group *group*.

**-hdrid sha1**

Query package with the specified SHA1 digest in the package header.

**-p package\_file, --package package\_file**

Query the uninstalled package *package\_file*, which can be a URL. If *package\_file* is not a binary package, it is treated as a text file containing a package manifest, with each line of the manifest containing a path or one or more whitespace-separated glob expressions to be expanded to paths. These paths are then used instead of *package\_file* as the query arguments. The manifest can contain comments that begin with a hash mark (#).

**--pkgid md5**

Query the package with a package identifier that is the given MD5 digest of the combined header and contents.

**--querybynumber num**

Query the *num*th database entry. Useful for debugging.

**-qf, --queryformat string**

Specify the format for displaying the query output, using tags to represent different types of data (e.g., NAME, FILENAME, DISTRIBUTION). The format specification is a variation of the standard **printf** formatting, with the type specifier omitted and replaced by the name of the header tag inclosed in brackets ({}). For example:

```
%{NAME}
```

The tag names are case-insensitive. Use `--querytags` (see “Miscellaneous options” on page 563) to view a list of available tags. The tag can be followed by `:type` to get a different output format type. The possible types are:

**:armor**

Wrap a public key in ASCII armor.

**:arraysize**

Display number of elements in array tags.

**:base64**

Encode binary data as base64.

**:date**

Use `%c` format as in `strftime(3)` to display the preferred date and time format for this locale.

**:day**

Use `%a %b %d %Y` format as in the function `strftime(3)`. This format displays the day, the month, the month as a decimal number, and the four-digit year.

**:depflags**

Format dependency flags.

**:fflags**

Format file flags.

**:hex**

Use hexadecimal format.

**:octal**

Use octal format.

**:perms**

Format file permissions.

**:pgpsig**

Display the PGP signature and time.

**:shescape**

Escape single quotes for use in a script.

**:triggertype**

Display trigger suffix (i.e., `in`, `un`, or `postun`, indicating whether it’s an install, uninstall, or post-uninstall trigger).

**:xml**

Wrap data in simple XML markup.

**--specfile** *specfile*

Query *specfile* as if it were a package. Useful for extracting information from a spec file.

**-tid** *tid*

List packages with the specified transaction identifier (*tid*), which is a Unix timestamp. All packages installed or erased in a single transaction have the same tid.

**-triggeredby *pkg***

List packages containing triggers that are run when the installation status of package *pkg* changes. For example:

```
$ rpm -q --triggeredby glibc
redhat-lsb-3.2-2.fc10.x86_64
```

In this example, the package `redhat-lsb-3.2-2.fc10.x86_64` contains a `triggerpostun` scriptlet that runs after `glibc` is uninstalled.

**--whatrequires *capability***

List packages that require the given capability to function. For example:

```
$ rpm -q --whatrequires popt
popt-devel-1.13-4.fc10.x86_64
logrotate-3.7.7-1.fc10.x86_64
nash-6.0.71-2.fc10.x86_64
initscripts-8.86-1.x86_64
rpm-4.6.0-0.rc1.7.x86_6
```

**--whatprovides *capability***

List packages that provide the given capability. For example:

```
$ rpm -q --whatprovides popt
popt-1.13-4.fc10.x86_64
```

**Package-query options****-c, --configfiles**

List configuration files in the package. Implies `-l`.

**--changelog**

Display the log of change information for the package.

**-d, --docfiles**

List documentation files in the package. Implies `-l`.

**--dump**

Dump information for each file in the package. The output includes the following information in this order:

```
path size mtime md5sum mode owner group isconfig isdoc rdev symlink
```

**--filesbypkg**

List all files in each package.

**-i, --info**

Display package information, including the name, version, and description. The results are formatted according to `--queryformat` if specified.

**-l, --list**

List all files in the package.

**--last**

List packages by install time, with the latest packages listed first.

**--provides**

List the capabilities this package provides.

**-R, --requires**

List any packages this package depends on.

#### **-s, --state**

List each file in the package and its state. The possible states are **normal**, **not installed**, or **replaced**. Implies **-l**.

#### **--scripts**

List any package-specific shell scripts used during installation and uninstallation of the package.

#### **--triggers, --triggerscript**

Display any trigger scripts in the package.

## **Uninstall options**

The syntax for **erase**, the uninstall command, is:

```
rpm -e [uninstall-options] package_name ...
rpm --erase [uninstall-options] package_name ...
```

The uninstall options are:

#### **--allmatches**

Remove all versions of the package. Only one package should be specified; otherwise, an error results.

#### **--nodeps**

Don't check dependencies before uninstalling the package.

#### **--nopostun**

Don't run any post-uninstall scripts.

#### **--nopreun**

Don't run any pre-uninstall scripts.

#### **--noscripts**

Don't execute any pre-uninstall or post-uninstall scripts. This option is equivalent to **--nopreun --nopostun**.

#### **--notriggerpostun**

Don't execute any post-uninstall scripts triggered by the removal of this package.

#### **--notriggers**

Don't execute any scripts triggered by the removal of this package. Equivalent to **--notriggerun --notriggerpostun**.

#### **--notriggerun**

Don't execute any uninstall scripts triggered by the removal of this package.

#### **--test**

Don't really uninstall anything; just go through the motions. Use with **-vv** for debugging.

## **Verify options**

The syntax for the **verify** command is:

```
rpm -V | --verify [package-selection-options] [verify-options]
```

Verify mode compares information about the installed files in a package with information about the files that came in the original package and displays any

discrepancies. The information compared includes the size, MD5 sum, permissions, type, owner, and group of each file. Uninstalled files are ignored.

The package selection options include those available for query mode. In addition, the following **verify** options are available:

- nodeps**  
Ignore package dependencies.
- nодigest**  
Ignore package or header digests.
- nofiles**  
Ignore attributes of package files.
- нogroup**  
Ignore group ownership errors.
- nolinkto**  
Ignore symbolic-link errors.
- nomd5**  
Ignore MD5 checksum errors.
- номode**  
Ignore file mode (permissions) errors.
- nordev**  
Ignore major and minor device number errors.
- nmtime**  
Ignore modification time errors.
- noscripts**  
Ignore any verify script.
- nosignature**  
Ignore package or header signatures.
- nosize**  
Ignore file size errors.
- nouser**  
Ignore user ownership errors.

The output is formatted as an eight-character string, possibly followed by an attribute marker, and then the filename. Each of the eight characters in the string represents the result of comparing one file attribute to the value of that attribute from the RPM database. A period (.) indicates that the file passed that test. The following characters indicate failure of the corresponding test:

- 5** MD5 sum
- D** Device
- G** Group
- L** Symlink
- M** Mode (includes permissions and file type)
- S** File size
- T** Mtime
- U** User

The possible attribute markers are:

- c Configuration file
- d Documentation file
- g Ghost file (contents not included in package)
- l License file
- r Readme file

### Database rebuild options

The syntax of the command to rebuild the RPM database is:

```
rpm --rebuilddb [options]
```

You also can build a new database:

```
rpm --initdb [options]
```

The options available with the database rebuild mode are the **--dbpath**, **--root**, and **-v** options described earlier under “General options” on page 546.

### Signature-check options

RPM packages may have a GPG signature built into them. There are three types of digital signature options: you can check signatures, add signatures to packages, and import signatures.

The syntax of the signature check mode is:

```
rpm --checksig [options] package_file...
rpm -K [options] package_file...
```

The signature-checking options **-K** and **--checksig** check the digests and signatures contained in the specified packages to insure the integrity and origin of the packages. Note that RPM now automatically checks the signature of any package when it is read; these options are still useful, however, for checking all headers and signatures associated with a package.

The **--nosignature** and **--nodigest** options described earlier, under “Verify options” on page 560, are available for use with signature check mode.

The syntax for adding signatures to binary packages is:

```
rpm --addsign binary-pkgfile...
rpm --resign binary-pkgfile...
```

Both **--addsign** and **--resign** generate and insert new signatures, replacing any that already exist in the specified binary packages.\*

The syntax for importing signatures is:

```
rpm --import public-key
```

\* In older versions of RPM, **--addsign** was used to add new signatures without replacing existing ones, but currently both options work the same way and replace any existing signatures.

The **--import** option is used to import an ASCII public key to the RPM database so that digital signatures for packages using that key can be verified. Imported public keys are carried in headers, and keys are kept in a ring, which can be queried and managed like any package file.

### Miscellaneous options

Several additional **rpm** options are available:

#### **--querytags**

Print the tags available for use with the **--queryformat** option in query mode.

#### **--setperms** *packages*

Set file permissions of the specified packages to those in the database.

#### **--setuids** *packages*

Set file owner and group of the specified packages to those in the database.

#### **--showrc**

Show the values **rpm** will use for all options that can be set in an *rpmrc* file.

### FTP/HTTP options

The following options are available for use with FTP and HTTP URLs in install, update, and query modes.

#### **-ftpport** *port*

Use *port* for making an FTP connection on the proxy FTP server instead of the default port. Same as specifying the macro **%\_ftpport**.

#### **-ftpproxy** *host*

Use *host* as the proxy server for FTP transfers through a firewall that uses a proxy. Same as specifying the macro **%\_ftpproxy**.

#### **-httpport** *port*

Use *port* for making an HTTP connection on the proxy HTTP server instead of the default port. Same as specifying the macro **%\_httpport**.

#### **-httpproxy** *host*

Use *host* as the proxy server for HTTP transfers. Same as specifying the macro **%\_httpproxy**.

## RPM Examples

Query the RPM database to find Emacs-related packages:

```
$ rpm -q -a | grep emacs
```

Query an uninstalled package, printing information about the package and listing the files it contains:

```
$ rpm -qpil ~/downloads/bash-3.2-29.fc10.x86_64.rpm
```

Install a package (assumes superuser privileges):

```
$ rpm -i sudo-1.6.9p17-2.fc10.x86_64.rpm
```

Do the same thing, but report on the progress of the installation:

```
$ rpm -ivh sudo-1.6.9p17-2.fc10.x86_64.rpm
```

## The rpmbuild Command

The **rpmbuild** command is used to build RPM packages. The syntax for **rpmbuild** is:

```
rpmbuild -[b|t] stage [build-options] spec-file ...
```

Specify **-b** to build a package directly from a spec file, or **-t** to open a **tarred, gzipped** file and use its spec file.

Both forms take the following single-character *stage* arguments, which specify the stages, or steps, required to build a package. The stages are listed in the order they would be performed:

- p** Perform the prep stage, unpacking source files and applying patches.
- l** Do a list check, expanding macros in the files section of the spec file and verifying that each file exists.
- c** Perform the prep and build stages; generally equivalent to doing a **make**.
- i** Perform the prep, build, and install stages; generally equivalent to doing a **make install**.
- b** Perform the prep, build, and install stages, then build a binary package.
- s** Build a source package.
- a** Perform the prep, build, and install stages, then build both binary and source packages.

The difference between the build stage, which is one of the early steps, and building a binary package in **b** or **a** is the difference between building a working binary for the software and putting all the pieces together into a final **rpm** package.

### rpmbuild options

The general **rpm** options described under “General options” on page 546 can be used with **rpmbuild**.

The following additional options can also be used when building an **rpm** file with **rpmbuild**:

#### **--buildroot** *dir*

Override the **BuildRoot** tag with *dir* when building the package.

#### **--clean**

Clean up (remove) the build files after the package has been made.

#### **--nobuild**

Go through the motions, but don’t execute any build stages. Used for testing spec files.

#### **--rmsource**

Remove the source files when the build is done. Can be used as a standalone option to clean up files separately from creating the packages.

#### **--rmspec**

Remove the spec file when the build is done. Can be used as a standalone option.

**--short-circuit**

Can be used with **-bc** and **-bi** to skip previous stages that already ran successfully. With **--short-circuit**, **-bc** starts directly at the build stage and **-bi** starts with the install stage.

**--sign**

Add a GPG signature to the package for verifying its integrity and origin.

**--target *platform***

When building the package, set the macros **%\_target**, **%\_target\_arch**, and **%\_target\_os** to the value indicated by *platform*.

Two other options can be used standalone with **rpmbuild** to recompile or rebuild a package:

**--rebuild *source-pkgfile...***

Like **--recompile**, but also build a new binary package. Remove the build directory, the source files, and the spec file once the build is complete.

**--recompile *source-pkgfile...***

Install the named source package, and prep, compile, and install the package.

Finally, the **--showrc** option is used to show the current **rpmbuild** configuration:

```
rpmbuild --showrc
```

This option shows the values that will be used for all options that can be set in *rpmrc* and *macros* files.

## The Debian Package Manager

Debian GNU/Linux provides several package-management tools, primarily intended to facilitate the building, installation, and management of binary packages. In addition, the tools described here also work on other Debian-based systems such as Ubuntu, Xandros, Knoppix, and numerous others.

Debian package names generally end in *.deb*. The Debian package-management tools we describe include the **apt** commands, **aptitude**, **dpkg**, **dpkg-deb**, **dselect**, and **synaptic**. Each of these tools is described in detail in “Debian Package Manager Command Summary” on page 569.

## Files

Some important files used by the Debian package-management tools are described briefly here:

***control***

Comes with each package. Documents dependencies; contains the name and version of the package, a description, maintainer, installed size, the package priority, etc.

***conffiles***

Comes with each package. Contains a list of the configuration files associated with the package.

### *preinst, postinst, prerm, postrm*

Scripts developers can include in a package to be run before installation, after installation, before removal, or after removal of the package.

### */var/lib/dpkg/available*

Contains information about packages available on the system.

### */var/lib/dpkg/status*

Contains information about the status of packages available on the system.

### */etc/apt/sources.list*

A list for APT of package sources, used to locate packages. The sources are listed one per line, in order of preference.

### */etc/apt/apt.conf*

The main APT configuration file.

### */etc/apt/preferences*

A preferences file that controls various aspects of APT, such as letting a user select the version or release of a package to install.

### */etc/dpkg/dpkg.cfg*

A configuration file containing default options for **dpkg**.

For a user, the important file is */etc/apt/sources.list*. This file is where you set up the paths to the package archives, telling **apt** where to go to find packages. **apt** is installed with a default file. You aren't required to modify the sources in the file, but you'll probably want to change some sources or add additional ones at some point. You might also want to change some of the options in the configuration files *apt.conf*, *preferences*, and *dpkg.config* if you aren't satisfied with the defaults. The *control*, *conffiles*, and the pre- and post-install and removal script files are created by the package developers and used internally by the package-management system.

## Package Priorities

Every Debian package has a priority associated with it, indicating how important the package is to the system. The priorities are:

### **required**

The package is essential to the proper functioning of the system.

### **important**

The package provides important functionality that enables the system to run well.

### **standard**

The package is included in a standard system installation.

### **optional**

The package is one that you might want to install, but you can omit it if you are short on disk space, for example.

### **extra**

The package either conflicts with other packages that have a higher priority, has specialized requirements, or is one that you would want to install only if you need it.

The control file for `dpkg`, for example, shows that `dpkg` itself has a priority of **required**; `dpkg-dev` (which provides tools for building Debian packages) has a priority of **standard**; and `dpkg-doc` is **optional**.

## Package and Selection States

The possible states that a package can be in are:

### **config-files**

Only the configuration files for the package are present on the system.

### **half-configured**

The package is unpacked, and configuration was started but not completed.

### **half-installed**

Installation was started but not completed.

### **installed**

The package is unpacked and configured.

### **not-installed**

The package is not installed.

### **unpacked**

The package is unpacked but not configured.

The possible package selection states are:

### **deinstall**

The package has been selected for deinstallation (i.e., for removal of everything but the configuration files).

### **install**

The package has been selected for installation.

### **purge**

The package has been selected to be purged (i.e., for removal of everything including the configuration files).

## Package Flags

Two possible package flags can be set for a package:

### **hold**

The package should not be handled by `dpkg` unless forced with the `--force-hold` option. Holding a package keeps it at the current version, preventing it from being updated. You might hold a package, for example, if the latest version is broken and you want to stay with the version you have until a newer one is released.

### **reinst-required**

The package is broken and needs to be reinstalled. Such a package cannot be removed unless forced with the `--force-reinstreq` option.

## Scripts

In addition to the commands described in the next section, a number of shell and Perl scripts are included with the package manager for use in managing and building packages:

### **apt-file**

Search for packages, specifying an action and a pattern to search for. (Perl script)

### **apt-rdepends**

Recursively list dependencies. (Perl script)

### **dpkg-architecture**

Determine and set the build and host architecture for package building. (Perl script)

### **dpkg-checkbuilddeps**

Check installed packages against the build dependencies and build conflicts listed in the control file. (Perl script)

### **dpkg-buildpackage**

A control script to help automate package building. (Shell script)

### **dpkg-distaddfile**

Add an entry for a file to *debian/files*. (Perl script)

### **dpkg-divert**

Create and manage the list of diversions, used to override the default location for installing files. (Perl script)

### **dpkg-genchanges**

Generate an upload control file from the information in an unpacked, built source tree and the files it has generated. (Perl script)

### **dpkg-gencontrol**

Read information from an unpacked source tree, generate a binary package control file (by default, *debian/tmp/DEBIAN/control*), and add an entry for the binary file to *debian/files*. (Perl script)

### **dpkg-name**

Rename Debian packages to their full package names. (Shell script)

### **dpkg-parsechangelog**

Read and parse the changelog from an unpacked source tree and write the information to standard output in machine-readable form. (Perl script)

### **dpkg-preconfigure**

Let packages ask questions prior to installation. (Perl script)

### **dpkg-reconfigure**

Reconfigure a package that is already installed. (Perl script)

### **dpkg-scanpackages**

Create a *Packages* file from a tree of binary packages. The *Packages* file is used by **dselect** to provide a list of packages available for installation. (Perl script)

### **dpkg-shlibdeps**

Calculate shared library dependencies for named executables. (Perl script)

**dpkg-source**

Pack and unpack Debian source archives. (Perl script)

**dpkg-statoverride**

Manage the list of stat overrides, which let **dpkg** override file ownership and mode when a package is installed. (Perl script)

## Debian Package Manager Command Summary

For the **apt-** commands, options can be specified on the command line or set in the configuration file. Boolean options set in the configuration file can be overridden on the command line in a number of different ways, such as **-no-opt** and **-opt=no**, where *opt* is the single-character or full name of the option.

---

**apt-cache**

*apt-cache [options] command*

Perform low-level operations on the APT binary cache, including the ability to perform searches and produce output reports from package metadata.

**Commands****add** *files*

Add the specified package index files to the source cache.  
Useful for debugging.

**depends** *pkgs*

For each specified package, show a list of dependencies and packages that can fulfill them.

**dotty** *pkgs*

Graph the relationships between the specified packages. The default is to trace out all dependent packages; turn this behavior off by setting the **APT::Cache::GivenOnly** configuration option.

**dump**

List every package in the cache. Used for debugging.

**dumpavail**

Print a list of available packages to standard output, suitable for use with **dpkg**.

**gencaches**

Build source and package caches from the sources in *sources.list* and from */var/lib/dpkg/status*. Equivalent to running **apt-get check**.

**madison** *[pkgs]*

Display a table showing the available versions of each specified package. Similar to **madison**, a Debian tool that checks for package versions and reports their status. This option works locally and doesn't require access to the Debian project's internal archive.

**pkgnames** [*prefix*]

Print a list of packages in the system. If *prefix* is specified, print only packages whose names begin with that prefix. Most useful with the **--generate** option.

**policy** [*pkgs*]

Print detailed information about the priority selection of each specified package. With no arguments, print the priorities of all sources. Useful for debugging issues related to the *preferences* file.

**rdepends** [*pkgs*]

Show a list of reverse dependencies for each specified package—i.e., list any packages that depend on the specified packages.

**search** *regex*

Search package names and descriptions of all available package files for the specified regular expression and print the name and short description of each matching package. With **--full**, the output is identical to that from the **show** command. With **--names-only**, only the package name is searched. Multiple regular expressions can be specified. Useful for finding packages when you don't know the actual package name.

**show** *pkgs*

Display the package records for each specified package. See the **-a** option for more details.

**showpkg** *pkgs*

Display information about the specified packages. For each package, the output includes the available versions, packages that depend on this package, and packages that this package depends on.

**showsrd** *pkgs*

Display source package records for each specified package.

**stats**

Display statistics about the cache.

**unmet**

Display the unmet dependencies in the package cache.

**Options****-a, --all-versions**

Print full records for all available versions. For use with the **show** command. The default is to show all versions; use with **--no-all-versions** to display only the version that would be installed. The configuration option is **APT::Cache::AllVersions**.

**--all-names**

Cause **pkgnames** to print all names, including virtual packages and missing dependencies. The configuration option is **APT::Cache::AllNames**.

- c *file*, --config-file=*file*  
Specify a configuration file to be read after the default configuration file.
- f, --full  
Print full package records when searching. The configuration option is APT::Cache::ShowFull.
- g, --generate  
Automatically regenerate the package cache rather than using the current cache. Default is to regenerate; turn it off with --no-generate. The configuration option is APT::Cache::Generate.
- h, --help  
Print usage information and exit.
- i, --important  
Print only important dependencies (Depends and Pre-Depends relations). For use with unmet. The configuration option is APT::Cache::Important.
- installed  
Only produce output for currently installed packages. For use with depends and rdepends. The configuration option is APT::Cache::Installed.
- n, --names-only  
Search only on package names, not long descriptions. The configuration option is APT::Cache::NamesOnly.
- o, --option  
Set a configuration option. Syntax is -o *group::tool=option*.
- p *file*, --pkg-cache=*file*  
Use the specified file for the package cache, the primary cache used by all operations. The configuration option is Dir::Cache::pkgcache.
- q, --quiet  
Operate quietly, producing output for logging but no progress indicators. Use -qq for even quieter operation. The configuration option is quiet.
- recurse  
Run depends or rdepends recursively, so all specified packages are printed once. The configuration option is APT::Cache::RecurseDepends.
- s *file*, --src-cache=*file*  
Specify the source cache file used by gencaches. The configuration option is Dir::Cache::srccache.
- v, --version  
Print version information and exit.

---

**apt-cdrom**

apt-cdrom [*options*] *command*

Add a new CD or DVD to apt's list of available sources. The database of CD-ROM IDs that apt maintains is /var/lib/apt/cdroms.list.

## Commands

### add

Add a disk to the source list.

### ident

Print the identity of the current disk and the stored filename.  
Used for debugging.

## Options

### -a, --thorough

Do a thorough package scan. May be needed with some old Debian CDs to find all package locations.

### -c file, --config-file=file

Specify a configuration file to be read after the default configuration file.

### -d mount-point, --cdrom=mount-point

Specify the CD-ROM mount point, which must be listed in */etc/fstab*. The configuration option is **Acquire::cdrom::mount**.

### -f, --fast

Do a fast copy, assuming the files are valid and don't all need checking. Specify this only if the disk has been run before without error. The configuration option is **APT::CDROM::Fast**.

### -h, --help

Print help message and exit.

### -m, --no-mount

Don't mount or unmount the mount point. The configuration option is **APT::CDROM::NoMount**.

### -n, --just-print, --recon, --no-act

Check everything, but don't actually make any changes. The configuration option is **APT::CDROM::NoAct**.

### -o, --option

Set a configuration option. Syntax is **-o group::tool=option**.

### -r, --rename

Prompt for a new label and rename the disk to the new value.  
The configuration option is **APT::CDROM::Rename**.

### -v, --version

Print the version information and exit.

---

## apt-config

`apt-config [options] shell args`

`apt-config [options] dump`

An internal program for querying configuration information, accessing the main configuration file */etc/apt/apt.conf*.

## Commands

### dump

Display the contents of the configuration space.

**shell**

Access the configuration information from a shell script. The arguments are in pairs, specifying the name of a shell variable and a configuration value to query. The value may be postfixed with /*x*, where *x* is one of the following letters:

- b** Return true or false.
- d** Return directories.
- f** Return filenames.
- i** Return an integer.

**Options**

**-c file, --config-file=file**

Specify a configuration file to be read after the default configuration file.

**-h, --help**

Print help message and exit.

**-o, --option**

Set a configuration option. Syntax is **-o group::tool=option**.

**-v, --version**

Print the version information and exit.

**apt-****extracttemplates**

`apt-extracttemplates [options] files`

Extract configuration scripts and templates from the specified Debian package files. For each specified file, a line of output is generated with the following information:

*package* *version* *template-file config-script*

and the template files and configuration scripts are written to the directory specified with **-t** or **--temp-dir**, or by the configuration option **APT::ExtractTemplates::TempDir**. The filenames are in the form *package.template.xxxx* and *package.config.xxxx*.

**Options**

**-c file, --config-file=file**

Specify a configuration file to be read after the default configuration file.

**-h, --help**

Print help message and exit.

**-o, --option**

Set a configuration option. Syntax is **-o group::tool=option**.

**-t dir, --tempdir=dir**

Write the extracted template files and configuration scripts to the specified directory. The configuration option is **APT::ExtractTemplates::TempDir**.

**-v, --version**

Print the version information and exit.

---

|                                                            |                                                                                                                                                                                                                                                                                                                                                                                                                    |
|------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>apt-ftparchive</b>                                      | <code>apt-ftparchive [options] command</code><br>Generate package and other index files used to access a distribution source. The files should be generated on the source's origin site.                                                                                                                                                                                                                           |
| <b>Commands</b>                                            |                                                                                                                                                                                                                                                                                                                                                                                                                    |
| <b>clean</b> <i>config-file</i>                            | Clean the databases used by the specified configuration file by removing obsolete records.                                                                                                                                                                                                                                                                                                                         |
| <b>contents</b> <i>path</i>                                | Search the specified directory tree recursively to generate a contents file. For each .deb file found, read the file list, sort the files by package, and write the results to standard output. Use with <b>--db</b> to specify a binary caching database.                                                                                                                                                         |
| <b>generate</b> <i>config-file sections</i>                | Build indexes according to the specified configuration file.                                                                                                                                                                                                                                                                                                                                                       |
| <b>packages</b> <i>path [override [pathprefix]]</i>        | Generate a package file from the specified directory tree. The optional override file contains information describing how the package fits into the distribution, and the optional path prefix is a string prepended to the filename fields. Similar to <b>dpkg-scangpkages</b> . Use with <b>--db</b> to specify a binary caching database.                                                                       |
| <b>release</b> <i>path</i>                                 | Generate a release file from the specified directory tree.                                                                                                                                                                                                                                                                                                                                                         |
| <b>sources</b> <i>paths [override [pathprefix]]</i>        | Generate a source index file from the specified directory tree. The optional override file contains information used to set priorities in the index file and to modify maintainer information. The optional path prefix is a string prepended to the directory field in the generated source index. Use <b>--source-override</b> to specify a different source override file. Similar to <b>dpkg-scansources</b> . |
| <b>Options</b>                                             |                                                                                                                                                                                                                                                                                                                                                                                                                    |
| <b>-c</b> <i>file</i> , <b>--config-file</b> = <i>file</i> | Specify a configuration file to be read after the default configuration file.                                                                                                                                                                                                                                                                                                                                      |
| <b>--contents</b>                                          | Perform contents generation. If set and if package indexes are being generated with a cache database, the file listing is extracted and stored in the database. If used with <b>generate</b> , allows the creation of any contents files. The default is on. The configuration option is <b>APT::FTPArchive::Contents</b> .                                                                                        |
| <b>-d, --db</b>                                            | Use a binary caching database. This option has no effect on <b>generate</b> . The configuration option is <b>APT::FTPArchive::DB</b> .                                                                                                                                                                                                                                                                             |

**--delink**  
Enable delinking of files when used with the **External-Links** setting. The default is on; turn off with **--no-delink**. The configuration option is **APT::FTPArchive::DeLinkAct**.

**-h, --help**  
Print help message and exit.

**--md5**  
Generate MD5 sums for the index files. The default is on. The configuration option is **APT::FTPArchive::MD5**.

**-o, --option**  
Set a configuration option. Syntax is **-o group::tool=option**.

**-q, --quiet**  
Run quietly, producing logging information but no progress indicators. Use **-qq** for quieter operation. The configuration option is **quiet**.

**--read-only**  
Make the caching databases read-only. The configuration option is **APT::FTPArchive::ReadOnlyDB**.

**-s file, --source-override=file**  
Specify a source override file. Use with the **sources** command. See the **sources** command description for more information. The configuration option is **APT::FTPArchive::SourceOverride**.

**-v, --version**  
Print the version information and exit.

---

**apt-get** *options* *command [package...]*  
A command-line tool for handling packages. Also serves as a backend to other APT tools such as **dselect**, **synaptic**, and **aptitude** (all described later in this section).

## Commands

### autoclean

Like **clean**, but remove only package files that can no longer be downloaded. Set the configuration option **APT::Clean-Installed** to **off** to prevent installed packages from being erased.

### autoremove

Remove packages that were automatically installed to satisfy a dependency and are no longer needed.

### build-dep

Install or remove packages to satisfy the build dependencies for a source package.

### clean

Clear the local repository of retrieved package files. Useful for freeing up disk space.

### check

Update the package cache and check for broken dependencies.

### **dist-upgrade**

Like **upgrade**, but also handle changing dependencies for new package versions intelligently. See the **-f** option for more information.

### **dselect-upgrade**

Used with **dselect**. Track the changes made by **dselect** to the **Status** field of available packages and take actions necessary to realize that status.

### **install packages**

Install one or more packages. Specify the package name, not the full filename. Other required packages are also retrieved and installed. With a hyphen appended to the package name, the package is removed if it is already installed. Select a version to install by appending an equals sign and the version. Select a distribution to install by appending a slash and the distribution.

### **purge packages**

Like **remove**, but also purge the specified packages from the system.

### **remove packages**

Remove one or more packages. Specify the package name, not the full filename. With a plus sign appended to the name, the package is installed.

### **source packages**

Find source packages and download them into the current directory. If specified with **--compile**, the source packages are compiled into binary packages. With **--download-only**, the source packages are not unpacked. Select a specific version by appending an equals sign and the version.

### **update**

Resynchronize the package overview files from their sources. Must be done before an **upgrade** or **dist-upgrade**.

### **upgrade**

Install the latest versions of all packages currently installed. Run **update** first.

## **Options**

### **--arch-only**

Process only architecture-dependent build dependencies. Configuration option is **APT::Get::Arch-Only**.

### **--auto-remove**

With **install** or **remove**, remove unused dependencies; like running the **autoremove** command. Configuration option is **APT::Get::AutomaticRemove**.

### **-b, --compile, --build**

Compile source packages after download. The configuration option is **APT::Get::Compile**.

### **-c file, --config-file=file**

Specify a configuration file to read after the default.

**-d, --download-only**

Retrieve package files, but don't unpack or install them. The configuration option is **APT::Get::Download-only**.

**--diff-only**

Download only the *diff* file from a source archive. The configuration option is **APT::Get::Diff-Only**.

**--dsc-only**

Download only the *dsc* file from a source archive. The configuration option is **APT::Get::Dsc-Only**.

**-f, --fix-broken**

Try to fix a system with broken dependencies. Can be used alone or with a command. Run with the **install** command if you have problems installing packages. You can run the sequence:

```
apt-get -f install
apt-get dist-upgrade
```

several times to clean up interlocking dependency problems. The configuration option is **APT::Get::Fix-Broken**.

**--force-yes**

Force yes. Cause **apt** to continue without prompting even if it is doing something that could damage your system. Use with great caution and only if absolutely necessary. The configuration option is **APT::Get::force-yes**.

**-h, --help**

Display a help message and exit.

**--ignore-hold**

Ignore a hold placed on a package, which would normally prevent the package from being upgraded. Use with **dist-upgrade** to override many undesired holds. The configuration option is **APT::Get::Ignore-Hold**.

**--list-cleanup**

Erase obsolete files from */var/lib/apt/lists*. The default is on; use **--no-list-cleanup** to turn it off, which you would normally do only if you frequently modify your list of sources. The configuration option is **APT::Get::List-Cleanup**.

**-m, --ignore-missing, --fix-missing**

Ignore missing or corrupted packages or packages that cannot be retrieved. Can cause problems when used with **-f**. The configuration option is **APT::Get::Fix-Missing**.

**--no-download**

Disable package downloading; use with **--ignore-missing** to force APT to use only the packages that have already been downloaded. The configuration option is **APT::Get::Download**.

**--no-remove**

Do not remove any packages; instead, abort without prompting. The configuration option is **APT::Get::Remove**.

**--no-upgrade**

Do not upgrade packages. Use with **install** to prevent upgrade of packages that are already installed. The configuration option is **APT::Get::Upgrade**.

**-o, --option**

Set a configuration option. Syntax is **-o group::tool=option**.

**--only-source**

Do not map the names specified with the **source** or **build-dep** commands through the binary table. With this option, only source package names can be specified. The configuration option is **APT::Get::Only-Source**.

**--print-uris**

Print Uniform Resource Indicators (URIs) of files instead of fetching them. Prints path, destination filename, size, and expected MD5 hash. The configuration option is **APT::Get::Print-URIs**.

**--purge**

Tell **dpkg** to do a purge instead of a remove for items that would be removed. Purging removes packages completely, including any configuration files. The configuration option is **APT::Get::Purge**.

**-q, --quiet**

Quiet mode. Omit progress indicators and produce only logging output. Use **-qq** to make even quieter. The configuration option is **quiet**.

**--reinstall**

Reinstall packages that are already installed, upgrading them to the latest version. The configuration option is **APT::Get::ReInstall**.

**-s, --simulate, --just-print, --dry-run, --recon, --no-act**

Go through the motions, but don't actually make any changes to the system. The configuration option is **APT::Get::Simulate**.

**-t rel, --target-release=rel, --default-release=rel**

Retrieve packages only from the specified release. The value of *rel* can be a release number or a value such as "unstable." The configuration option is **APT::Default-Release**.

**--tar-only**

Download only the *tar* file from a source archive. The configuration option is **APT::Get::Tar-Only**.

**--trivial-only**

Perform only operations that are considered trivial—i.e., ones that won't harm your system, by, say, removing needed files. Unlike **--assume-yes**, which always answers "yes" to any prompts, **--trivial-only** always answers "no." The configuration option is **APT::Get::Trivial-Only**.

**-u, --show-upgraded**

Print a list of all packages to be upgraded. The configuration option is **APT::Get::Show-Upgraded**.

- v, --version**  
Display the version and exit.
- V, --verbose-versions**  
Show full versions for upgraded and installed packages. The configuration option is **APT::Get::Show-Versions**.
- y, --yes, --assume-yes**  
Automatically reply “yes” to prompts and run noninteractively. Abort if there is an error. The configuration option is **APT::Get::Assume-Yes**.

**apt-sortpkgs**`apt-sortpkgs [options] indexfiles`

Sort the records in a source or package index file by package name and write the results to standard output. **apt-sortpkgs** also sorts the internal fields of each record.

**Options**

- c file, --config-file=file**  
Specify a configuration file to read after the default.
- h, --help**  
Display a help message and exit.
- o, --option**  
Set a configuration option. Syntax is **-o group::tool=option**.
- s, --source**  
Order by source index field. The configuration option is **APT::SortPkgs::Source**.
- v, --version**  
Display the version and exit.

**aptitude**`aptitude [options] [action [arguments]]`

A text-based frontend to **apt**, which can be run either directly from the command line or from a visual mode that runs in a terminal window.

**Actions**

The following actions are supported. Running **aptitude** with no action invokes the visual mode. Package names can be entered individually or as search patterns. A search pattern consists of terms starting with a tilde (~), followed by a character indicating the type of term, followed by the text to be searched for. The most common usage is to use **~n** to search for a package name (e.g., **~nemacs**, to search for packages that have *emacs* in their name). You can find the full list of term types in the *Aptitude User’s Manual*. The manual can be found in `/usr/share/doc/aptitude/README` on a Debian-based system. On an RPM-based system with **aptitude** installed, the `README` file may be in `/usr/share/aptitude` or `/usr/share/doc/aptitude`.

**autoclean**

Clean out the cache by removing only packages that can no longer be downloaded.

**changelog package[=version | /archive] ...**

Download and display the Debian changelog for each specified package.

**clean**

Clean out the cache by removing all previously downloaded .deb files.

**download package[=version | /archive] ...**

Download the .deb file for each specified package to the current directory. With a version, install that version; with an archive, install the version from that archive.

**forbid-version package[=version] ...**

Don't allow **aptitude** to upgrade the package to a particular version. If no version is specified, it is assumed to be the version that would normally be used. To override later, use the **install** action.

**forget-new**

Remove internal information about what packages are "new."

**full-upgrade**

Upgrade as many installed packages as possible, installing and removing packages as needed to satisfy dependencies. Formerly called **dist-upgrade**, which is now deprecated.

**help**

Display help information and exit.

**hold packages**

Place a hold on each specified package.

**install [package[=version | /archive] ...]**

Install the specified packages. With a version, install that version. With an archive, install the version in that archive. With no arguments, install any stored or pending actions. You can also use **install** to perform different actions on multiple packages with a single command. Append - to the package name to **remove**, + to **install**, \_ to **purge**, = to **hold** a package, or : to leave the package at the current version.

**keep packages**

Cancel any scheduled action on the specified packages.

**keep-all**

Cancel all scheduled actions on all packages.

**markauto packages**

Mark the specified packages as automatically installed.

**purge [package[=version] ...]**

Remove the specified packages and their configuration files.

**remove [package[=version] ...]**

Remove the specified packages.

**safe-upgrade**

Upgrade as many packages as possible; if a package has dependency problems, avoid upgrading that package (but don't remove it).

**search patterns**

Search for packages matching each of the specified patterns and display a list of matches. The full list of search terms can be found in the *Aptitude User's Manual*.

**show patterns**

Search for packages matching each of the specified patterns and display detailed information for every match found.

**unhold packages**

Remove the hold on each specified package.

**unmarkauto packages**

Mark the specified packages as manually installed.

**update**

Update the list of available packages by downloading the names of new and upgradeable packages.

**why, why-not packages**

Display reasons why the specified packages can or cannot be installed.

## Options

Most of the `aptitude` options have corresponding configuration options that can be set in the configuration file.

**-d, --download-only**

Download packages to the cache but do not install them. Configuration option is `Aptitude::CmdLine::Download-Only`.

**-D, --show-deps**

Show summaries of why packages will be automatically installed or removed. Configuration option is `Aptitude::CmdLine::Show-Deps`.

**-f**

Attempt to fix dependencies of broken packages. Configuration option is `Aptitude::CmdLine::Fix-Broken`.

**-F format, --display-format format**

Specify the output format for `search`. See the *Aptitude User's Manual* for details on specifying the format. Configuration option is `Aptitude::CmdLine::Package-Display-Format`.

**-h, --help**

Print help message and exit.

**-O order, --sort order**

Specify the sort order for `search` output. See the *Aptitude User's Manual* for details.

**-P, --prompt**

Always display a prompt even for actions that were explicitly requested. The corresponding configuration option is `Aptitude::CmdLine::Always-Prompt`.

**--purge-unused**

Purge packages that are no longer required by any installed package.

**-q[=n], --quiet[=n]**

Run in quiet mode, suppressing progress indicators. Use multiple *qs* to run even quieter, or specify a number *n* to indicate directly the degree of quietness.

**-r, --with-recommends**

Treat recommendations as dependencies when installing new packages. The corresponding configuration option is **Aptitude::CmdLine::Recommends-Important**.

**-R, --without-recommends**

Do not treat recommendations as dependencies when installing new packages. The corresponding configuration option is **Aptitude::CmdLine::Recommends-Important**.

**-s, --simulate**

Go through the motions, but do not actually perform the actions. Print the actions that would be performed. Configuration option is **Aptitude::Simulate**.

**--schedule-only**

Schedule actions to be performed later, but don't perform them now. Works with actions that modify package states; to run them later, use **aptitude install** with no arguments.

**-t release, --target-release release**

Specify the release to use for installing packages. Equivalent to adding */release* to package names for the **changelog**, **download**, and **show** actions. The corresponding configuration option is **Aptitude::CmdLine::Default-Release**.

**-v, --verbose**

Operate verbosely, displaying additional information. Specify multiple times to get even more information displayed. The corresponding configuration option is **Aptitude::CmdLine::Verbose**.

**-V, --show-versions**

Display the version for packages being installed. Configuration option is **Aptitude::CmdLine::Show-Versions**.

**--version**

Display the version information for **aptitude** and exit.

**--visual-preview**

Start the visual interface and display the preview screen.

**-w width, --width width**

Specify the output display width for **search**. The default is the terminal width. The corresponding configuration option is **Aptitude::CmdLine::Package-Display-Width**.

**-y, --assume-yes**

Assume a yes response to a yes/no prompt and don't display the prompt. Prompts for dangerous actions are still shown. This option overrides **-P**. The corresponding configuration option is **Aptitude::CmdLine::Assume-Yes**.

- Z Display the disk space that will be used or freed by the packages being acted upon. The corresponding configuration option is `Aptitude::CmdLine::Show-Size-Changes`.

### Internal options

The following options are used internally for `aptitude`'s visual mode. You shouldn't need to issue them directly.

- i Display a download preview when the program starts. Cannot be used with -u.

### -S *filename*

Load extended state information from the specified file, not the default state file.

- u Begin updating the package lists when the program starts. Cannot be used with -i.

---

## dpkg

### dpkg [*options*] *action*

A tool for installing, managing, and building packages. Also serves as a frontend to `dpkg-deb` and `dpkg-query`.

### dpkg actions

These actions are carried out by `dpkg` itself:

#### -A *pkgfile*, --record-avail *pkgfile*

Update the record of available files kept in `/var/lib/dpkg/available` with information from *pkgfile*. This information is used by `dpkg` and `dselect` to determine which packages are available. With -R or --recursive, *pkgfile* must be a directory.

#### -C, --audit

Search for partially installed packages and suggest how to get them working.

#### --clear-avail

Remove existing information about which packages are available.

#### --clear-selections

Set the state of every nonessential package to `deinstall` to deselect them before running `dpkg --set-selections`.

#### --command-fd *n*

Accept commands passed on the file descriptor given by *n*. Note that any additional options set through this file descriptor or on the command line are not reset, but remain for other commands issued during the same session.

#### --compare-versions *ver1 op ver2*

Perform a binary comparison of two version numbers. The operators `lt` `le` `eq` `ne` `ge` `gt` treat a missing version as earlier. The operators `lt-nl` `le-nl` `ge-nl` `gt-nl` treat a missing version as later (where `nl` is “not later”). A third set of operators (`<` `<<` `<=` `=` `>` `>>`) is provided for compatibility with control-file syntax. `dpkg` returns zero for success (i.e., the condition is satisfied) and nonzero otherwise.

**--configure [packages | -a | --pending]**

Reconfigure one or more unpacked *packages*. If **-a** or **--pending** is given instead of *packages*, configure all packages that are unpacked but not configured. Configuring a package involves unpacking the configuration files, backing up the old configuration files, and running the **postinst** script if one is present.

**-Dh, --debug=help**

Print debugging help message and exit.

**--force-help**

Print help message about the **--force-list** options and exit. See the **--force-list** option description for the possible values of *list*.

**--forget-old-unavail**

Forget about uninstalled, unavailable packages.

**--get-selections [pattern]**

Get list of package selections and write to standard output. With *pattern* specified, write selections that match the pattern.

**--help**

Print help message and exit.

**-i pkgfile, --install pkgfile**

Install the package specified as *pkgfile*. With **-R** or **--recursive**, *pkgfile* must be a directory.

**--license, --licence**

Print **dpkg** license information and exit.

**--print-architecture**

Print the target architecture.

**--print-installation-architecture**

Print the host architecture for installation.

**-r, --remove [packages | -a | --pending]**

**-P, --purge [packages | -a | --pending]**

Remove or purge one or more installed *packages*. Removal gets rid of everything except the configuration files listed in *debian/conffiles*; purging also removes the configuration files. If **-a** or **--pending** is given instead of *packages*, **dpkg** removes or purges all packages that are unpacked and marked (in */var/lib/dpkg/status*) for removing or purging.

**--set-selections**

Set package selections based on input file read from standard input.

**--unpack pkgfile**

Unpack the package, but don't configure it. When used with **-R** or **--recursive**, *pkgfile* must be a directory.

**--update-avail pkgs-file**

**--merge-avail pkgs-file**

Update the record of available files kept in */var/lib/dpkg/available*. This information is used by **dpkg** and **dselect** to determine what packages are available. Update replaces the information with the contents of the *pkgs-file*, distributed as

*Packages*. Merge combines the information from *Packages* with the existing information. You can also use **dselect update** to do the same thing.

#### **--version**

Print **dpkg** version information and exit.

#### **--yet-to-unpack**

Search for uninstalled packages that have been selected for installation.

### **dpkg-deb actions**

The following actions can be specified for **dpkg** and are passed to **dpkg-deb** for execution. Also see **dpkg-deb**.

#### **-b dir [archive], --build dir [archive]**

Build a package.

#### **-c archive, --contents archive**

List the contents of a package.

#### **-e archive [dir], --control archive [dir]**

Extract control information from a package.

#### **-f archive [control-fields], --field archive [control-fields]**

Display the control field or fields of a package.

#### **-I archive [control-files], --info archive [control-files]**

Show information about a package.

#### **--fsys-tarfile archive**

Write the filesystem tree contained in a package to standard output in *tar* format.

#### **-x archive dir, --extract archive dir**

Extract the files from a package.

#### **-X archive dir, --vextract archive dir**

Extract the files and display the filenames from a package.

### **dpkg-query actions**

The following actions can be specified for **dpkg** and are passed to **dpkg-query** for execution. Also see **dpkg-query**.

#### **-l, --list [pkg-name-pattern]**

List all packages whose names match the specified pattern.

With no pattern, list all packages in */var/lib/dpkg/available*.

#### **-L packages, --listfiles packages**

List installed files that came from the specified package or packages.

#### **-p, --print-avail package**

Print the details about *package* from */var/lib/dpkg/available*.

#### **-s packages, --status packages**

Report the status of one or more *packages*.

#### **-S filename-pattern, --search filename-pattern**

Search installed packages for a filename.

## Options

**dpkg** options can be specified on the command line or set in the configuration file. Each line in the configuration file contains a single option, specified without the leading dash (-).

### **--abort-after=num**

Abort processing after *num* errors. Default is 50.

### **--admindir= dir, --instdir= dir, --root=dir**

Change default directories. **admindir** contains administrative files with status and other information about packages; it defaults to */var/lib/dpkg*. **instdir** is the directory into which packages are installed; it defaults to */*. Changing the **root** directory to *dir* automatically changes **instdir** to *dir* and **admindir** to */dir/var/lib/dpkg*.

### **-B, --auto-deconfigure**

When a package is removed, automatically deconfigure any other package that depended on it.

### **-Doctal, --debug=octal**

Turn on debugging, with the *octal* value specifying the desired level of debugging information. Use **-Dh** or **--debug-help** to display the possible values. You can OR the values to get the desired output.

### **-E, --skip-same-version**

Don't install the package if this version is already installed.

### **--force-list, --no-force-list, --refuse-list**

Force or refuse to force an operation. *list* is specified as a comma-separated list of options. With **--force**, a warning is printed, but processing continues. **--refuse** and **--no-force** cause processing to stop with an error. Use **--force-help** to display a message describing the options. The force/refuse options are:

#### **all**

Turn all force options on or off.

#### **architecture**

Process even if intended for a different architecture.

#### **bad-path**

Some programs are missing from the path.

#### **bad-verify**

Install package even if it fails to verify.

#### **confdef**

Always choose the default action for modified configuration files. If there is no default and **confnew** or **confold** is also specified, use that to decide; otherwise, ask the user.

#### **configure-any**

Configure any unpacked but unconfigured package that the package depends on.

**conflicts**

Permit installation of conflicting packages. Can result in problems from files being overwritten.

**confmiss**

Always install a missing configuration file. Be careful using this option, since it means overriding the removal of the file.

**confnew**

Always install the new version of a modified configuration file, unless **confdef** is also specified. In that case, use the default action if there is one.

**confold**

Keep the old version of a modified configuration file, unless **confdef** is also specified. In that case, use the default action if there is one.

**depends**

Turn dependency problems into warnings.

**depends-version**

Warn of version problems when checking dependencies, but otherwise ignore.

**downgrade**

Install even if a newer version is already installed. Forced by default. Note that no dependency checking is done, so use of this option can cause serious system problems.

**hold**

Process packages even if they are marked to be held.

**not-root**

Try to install or remove even when not logged on as root.

**overwrite**

Overwrite a file from one package with the same file from another package.

**overwrite-dir**

Overwrite one package's directory with a file from another package.

**overwrite-diverted**

Overwrite a diverted file with an undiverted version.

**remove-essential**

Remove a package even if it is essential. Note that this can cause your system to stop working.

**remove-reinstreq**

Remove a package even if it is broken and is marked to require reinstallation.

**-G, --refuse-downgrade**

Don't install a package if a newer version is already installed.

**--ignore-dependents=pkglist**

Dependency problems result only in a warning for the packages in *pkglist*.

**--log=filename**  
Log status updates and actions to the specified file instead of the default `/var/log/dpkg.log`.

**--new**  
New binary package format. This is a **dpkg-deb** option.

**--no-act, --dry-run, --simulate**  
Go through the motions, but don't actually write any changes.  
Used for testing. Be sure to specify before the action; otherwise, changes might be written.

**--nocheck**  
Ignore the contents of the control file when building a package.  
This is a **dpkg-deb** option.

**--no-debsig**  
Do not verify package signatures.

**-O, --selected-only**  
Process only packages that are marked as selected for installation.

**--old**  
Old binary package format. This is a **dpkg-deb** option.

**-R, --recursive**  
Recursively handle `.deb` files found in the directories and their subdirectories specified with `-A`, `-i`, `--install`, `--unpack`, and `--avail`.

**--status-fd n**  
Send the package status information to the specified file descriptor. Can be given more than once.

---

**dpkg-deb** `dpkg-deb action [options]`  
Backend command for building and managing Debian package archives. Also see **dpkg**; you'll often want to use **dpkg** to pass commands through to **dpkg-deb**, rather than call **dpkg-deb** directly.

### Actions

**-b dir [archive], --build dir [archive]**  
Create an *archive* from the filesystem tree starting with directory *dir*. The directory must have a *DEBIAN* subdirectory containing the control file and any other control information. If *archive* is specified and is a filename, the package is written to that file; if no *archive* is specified, the package is written to *dir.deb*. If the archive already exists, it is replaced. If *archive* is the name of a directory, **dpkg-deb** looks in the control file for the information it needs to generate the package name. (Note that for this reason, you cannot use **--nocheck** with a directory name.)

**-c archive, --contents archive**  
List the filesystem-tree portion of *archive*.

- e archive [dir], --control archive [dir]**  
Extract control information from *archive* into the directory *dir*, which is created if it doesn't exist. If *dir* is omitted, a *DEBIAN* subdirectory in the current directory is used.
- f archive [control-fields], --field archive [control-fields]**  
Extract information about one or more fields in the control file for *archive*. If no fields are provided, print the entire control file.
- h, --help**  
Print help information and exit.
- I archive [control-files], --info archive [control-files]**  
Write information about binary package *archive* to standard output. If no control files are provided, print a summary of the package contents; otherwise, print the control files in the order they were specified. An error message is printed to standard error for any missing components.
- fsys-tarfile archive**  
Extract the filesystem tree from *archive*, and send it to standard output in **tar** format. Can be used with **tar** to extract individual files from an archive.
- license, --licence**  
Print the license information and exit.
- version**  
Print the version number and exit.
- W archive, --show archive**  
Show information about the specified archive. Display package name and version on one line or customize with the **--show-format** option.
- x archive dir, --extract archive dir**  
**-X archive dir, --vextract archive dir**  
Extract the filesystem tree from *archive* into the specified directory, creating *dir* if it doesn't already exist. **-x** (**--extract**) works silently, while **-X** (**--vextract**) lists the files as it extracts them. Do not use this action to install packages; use **dpkg** instead.

## Options

- D, --debug**  
Turn on debugging.
- new**  
Build a new-style archive format (this is the default).
- nocheck**  
Don't check the control file before building an archive. This lets you build a broken archive.
- old**  
Build an old-style archive format; obsolete.

**--showformat=format**

Specify the output format for **-W**/**--show**. The format can include the standard escape sequences **\n** (newline), **\r** (carriage return), or **\\\** (backslash). Specify package fields with the syntax  **\${var[;width]}**. Fields are right-aligned by default, or left-aligned if **width** is negative.

**-z#**

Set the compression level to the value specified by # when building an archive.

**-Z type**

Set the type of compression to use when building an archive. Possible values are **gzip**, **bzip2**, and **none**.

---

**dpkg-query**

**dpkg-query [option] command**

Display information about packages listed in the **dpkg** database. You can also use **dpkg-query** as a backend for **dpkg**, instead of calling **dpkg-query** directly.

**Commands**

**--help**

Print help information and exit.

**-l [patterns], --list [patterns]**

List packages whose names match any of the specified patterns. With no pattern specified, list all packages in **/var/lib/dpkg/status**. The pattern may need to be in quotes to avoid expansion by the shell.

**-L packages, --listfiles packages**

List files installed on your system from each of the specified packages. This command does not list files created by package-specific installation scripts.

**--license, --licence**

Print the license information and exit.

**-p package, --print-avail package**

Display details for the specified package, as found in **/var/lib/dpkg/available**.

**-s package, --status package**

Report on the status of the specified package.

**-S patterns, --search patterns**

Search the installed packages for filenames matching one of the specified patterns. At least one pattern must be specified.

**-W [patterns], --show [patterns]**

Like **-l**, but the output can be customized with the **--showformat** option.

**--version**

Print version information and exit.

## Options

**--admindir=dir**

Use *dir* as the location of the **dpkg** database. The default is */var/lib/dpkg*.

**-f format, --showformat=format**

Specify the output format for **-W**/**--show**. The format can include the standard escape sequences **\n** (newline), **\r** (carriage return), or **\\\** (backslash). Specify package fields with the syntax  **\${var[;width]}**. Fields are right-aligned by default, or left-aligned if *width* is negative.

## dpkg-split

**dpkg-split [action] [options]**

Split a binary package into smaller pieces and reassemble the pieces, either manually or in automatic mode. The automatic mode maintains a queue of parts for reassembling.

## Actions

**-a -o output part, --auto -o output part**

Add *part* to the queue for automatic reassembly, and if all the parts are available, reassemble the package as *output*. Requires the use of the **-o** (or **--output**) option, as shown.

**-d [packages], --discard [packages]**

Discard parts from the automatic-assembly queue. If any *packages* are specified, discard only parts from those packages. Otherwise, empty the queue.

**-I parts, --info parts**

Print information about the specified part file or files to standard output.

**-j parts, --join parts**

Join the parts of a package file together from the *parts* specified. The default output file is *package-version.deb*.

**-l, --listq**

List the contents of the queue of parts waiting for reassembly, giving the package name, the parts that are on the queue, and the number of bytes.

**-s full-package [prefix], --split full-package [prefix]**

Split the package *full-package* into parts *N* of *M*, named *prefix-NofM.deb*. The prefix defaults to the *full-package* name without the *.deb* extension.

**-h, --help**

Print help message and exit.

**--license, --licence**

Print license information and exit.

**--version**

Print version information and exit.

## Options

**--depotdir dir**

Specify an alternate directory *dir* for the queue of parts waiting for reassembly. Default is */var/lib/dpkg*.

**--msdos**

Force **--split** output filenames to be MS-DOS-compatible.

**-Q, --npquiet**

Do not print an error message for a part that doesn't belong to a binary package when doing automatic queuing or reassembly.

**-o output, --output output**

Use *output* as the filename for a reassembled package.

**-S num, --partsize num**

When splitting, specify the maximum part size (*num*) in kilobytes. Default is 450 KB.

---

## dselect

**dselect [options] [action]**

A screen-oriented user frontend to **dpkg**, used to install and manage packages. See **dpkg** and **dpkg-deb** for information on building packages.

## Actions

If **dselect** is run with no action specified on the command line, it displays the following menu:

- \* 0. [A]ccess Choose the access method to use.
- 1. [U]pdate Update list of available packages, if possible.
- 2. [S]elect Request which packages you want on your system.
- 3. [I]nstall Install and upgrade wanted packages.
- 4. [C]onfig Configure any packages that are unconfigured.
- 5. [R]emove Remove unwanted software.
- 6. [Q]uit Quit **dselect**.

The asterisk (on the first line) shows the currently selected option. Any of the menu items can be specified directly on the command line as an action (**access**, **update**, **select**, **install**, **config**, **remove**, **quit**) to go directly to the desired activity. For example:

```
$ dselect access
```

If you enter **quit** on the command line, **dselect** exits immediately without doing anything. An additional command-line action is **menu**, which displays the menu and is equivalent to running **dselect** with no action.

## Options

Options can be specified both on the command line and in the **dselect** configuration file, */etc/dpkg/dselect.cfg*.

**--admindir** *dir*

Change the directory that holds internal datafiles to *dir*. Default is */var/lib/dpkg*.

**--color** *colorspec*, **--colour** *colorspec*

Set colors for different parts of the screen, as specified by *colorspec* as follows:

*screenpart*: [*fgcolor*], [*bgcolor*] [:*attr*[+*attr*+...]]

This option can be specified multiple times, to override the default colors for different *screenparts*. Rather than having to specify the colors on the command line each time you run **dselect**, you might prefer to set them in the configuration file. The possible screen parts (going from the top of the screen to the bottom) are:

**title**

The screen title.

**listhead**

The header line above the package list.

**list**

The scrolling list of packages and some help text.

**listsel**

The selected item in the list.

**pkgstate**

The text showing the current state of each package.

**pkgstatesel**

The text showing the current state of the selected package.

**infohead**

The header line showing the state of the selected package.

**infodesc**

The short description of the package.

**info**

The text that displays information such as the package description.

**infofoot**

The last line of the screen when selecting packages.

**query**

Query lines.

**helpscreen**

The color of help screens.

Either the foreground color, the background color, or both can be specified for each screen part. The colors are given as the standard **curses** colors. After the color specification, you can specify a list of attributes separated by plus signs (+). The possible attributes are **normal**, **standout**, **underline**, **reverse**, **blink**, **bright**, **dim**, and **bold**. Not all attributes work on all terminals.

**--expert**  
Run in expert mode; don't print help messages.

**-D [file], --debug [file]**  
Turn on debugging. Send output to *file* if specified.

**--help**  
Print help message and exit.

**--license, licence**  
Print license information and exit.

**--version**  
Print version information and exit.

---

## **synaptic**

**synaptic [options]**

Graphical frontend for APT. Use in place of **apt-get** to install, upgrade, or remove packages from your system. With **synaptic**, you can view a list of all available packages, or you can break the list down in various ways to make it more manageable. From the **synaptic** window, you can select from a list of categories. The categories are section (e.g., view only development-related packages), package status, origin, search history, or filter.

If you choose to display by filter, there is a set of predefined filters, or you can define your own. The predefined filters include ones to display all packages, packages marked for a status change, packages that can be configured with **debconf** (Debian systems only), packages with broken dependencies, and packages that can be upgraded to a later version. You can edit the existing filters or define your own, by selecting Filters from the Settings menu.

Once you've used the selection criteria to find the list of packages, you can select a single package, or you can select multiple packages by holding down the Shift or Ctrl key. Like **apt-get**, first do an **update** to update the package lists, then you can do an **install** or **upgrade**.

To start **synaptic** from Gnome, select Administration → Synaptic Package Manager from the System menu. From the KDE menu, select System → Synaptic Package Manager. You can also start the graphical interface from the command line, with the command:

**synaptic [options]**

### **Options**

In addition to the following options, **synaptic** accepts the standard GTK+ toolkit command-line options.

**-f filename, --filter-file=filename**  
Use the specified file as an alternative filter settings file.

**-h, --help**  
Print help message and exit.

**-i num, --initial-filter=num**  
Start up with the filter numbered *num* as the initial filter.

**--non-interactive**

Run without prompting for user input.

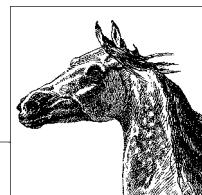
**-o $option$ , --option= $option$** 

Set an internal option. Don't use this option unless you are sure you know what you are doing.

- r** Open with the file repository window displayed. This window lists the repositories and shows which are active.
-

# 6

## The Bash Shell



The *shell* is a program that acts as a buffer between you and the operating system. In its role as a command interpreter, it should (for the most part) act invisibly. There are three main uses for the shell: interactive use; customizing your Linux session by defining *variables* and startup files; and programming, by writing and executing *shell scripts*.

The original Bourne shell became the standard shell for writing shell scripts. The Bourne shell is still found in `/bin/sh` on Linux systems but is now usually a symbolic link to Bash. Because the Berkeley C shell (`csh` and later `tcsch`) offered better features for interactive use, such as command history and job control, for a long time the standard practice was to use the Bourne shell for programming and the C shell for daily use. David Korn at Bell Labs enhanced the Bourne shell by adding `csh`-like features; his shell is known as the Korn shell (`ksh`).

The Free Software Foundation developed a clone of the Bourne shell, written from scratch, named “Bash,” the Bourne-Again SHell. Over time, Bash has become a POSIX-compliant version of the shell, incorporating many popular features from other shells, such as `csh`, `tcsch`, and `ksh`. Bash is the primary shell for Linux.

Another popular shell is the Z Shell, `zsh`, which is similar to `ksh` but with many extensions. `zsh` differs from Bash both in being based on `ksh` and because it does not attempt to be POSIX-compliant the way Bash does.

This chapter covers Bash. All references are to Bash version 4, which among numerous other changes includes new features such as associative arrays and coprocesses. The following topics are presented:

- Overview of features
- Invoking the shell
- Syntax
- Functions
- Variables

- Arithmetic expressions
- Command history
- Job control
- Command execution
- Restricted shells
- Built-in commands

<http://www.gnu.org/software/bash/bash.html> provides information about the Bash shell, as does <http://tiswww.case.edu/php/chet/bash/bash-top.html>. See also *Classic Shell Scripting* and *Learning the bash Shell* (both from O'Reilly).

## Overview of Features

The Bash shell provides the following features:

- Input/output redirection
- Wildcard characters (metacharacters) for filename abbreviation
- Shell variables and options for customizing your environment
- A built-in command set for writing shell programs
- Shell functions, for modularizing tasks within a shell program
- Job control
- Command-line editing (using the command syntax of either **vi** or **emacs**)
- Access to previous commands (command history)
- Integer arithmetic
- Arrays and arithmetic expressions
- Command-name abbreviation (aliasing)
- Upward compliance with POSIX
- Internationalization facilities
- An arithmetic **for** loop
- More ways to substitute variables

## Invoking the Shell

The command interpreter for the Bash shell (**bash**) can be invoked as follows:

```
bash [options] [arguments]
```

Bash can execute commands from a terminal, from a file (when the first *argument* is an executable script), or from standard input (if no arguments remain or if **-s** is specified). Bash automatically prints prompts if standard input is a terminal, or if **-i** is given on the command line.

On Linux systems, **/bin/sh** is generally a link to Bash. When invoked as **sh**, Bash acts more like the traditional Bourne shell: Login shells read **/etc/profile** and **~/.profile**, and regular shells read **\$ENV**, if it's set. Full details are available on the **bash** manpage.

## Options

If both single- and multi-character options appear on the command line, the multi-character options must appear first.

- ,** -- End option processing.
- c str** Read commands from string *str*.
- D, --dump-strings** Print all \$"..." strings in the program.
- debugger** Read the debugging profile at startup, turn on the **extdebug** option to **shopt**, and enable function tracing. For use by the Bash debugger.
- dump-po-strings** Same as **-D**, but output in GNU **gettext** po (portable object) format.
- help** Print a usage message and exit successfully
- i** Create an interactive shell (prompt for input).
- init-file file, --rcfile file** Use *file* as the startup file instead of *~/.bashrc* for interactive shells.
- login** Shell is a login shell.
- noediting** Do not use the *readline* library for input, even in an interactive shell.
- noprofile** Do not read */etc/profile* or any of the personal startup files.
- norc** Do not read *~/.bashrc*. Enabled automatically when invoked as **sh**.
- O option** Enable the **shopt** built-in command option *option*.
- p** Start up as a privileged user. Don't read \$ENV or \$BASH\_ENV, don't import functions from the environment, and ignore the value of \$SHELLOPTS.
- posix** Turn on POSIX mode.
- r, --restricted** Create a restricted shell.
- s** Read commands from standard input; output from built-in commands goes to file descriptor 1 (standard output); all other shell output goes to file descriptor 2 (standard error).
- verbose** Same as **set -v**; the shell prints lines as it reads them.
- version** Print a version message and exit.

The remaining options to Bash are listed under the **set** built-in command.

## Arguments

Arguments are assigned in order to the positional parameters **\$1**, **\$2**, etc. If the first argument is an executable script, commands are read from it, and the remaining arguments are assigned to **\$1**, **\$2**, etc. The name of the script is available as **\$0**.

## Syntax

This section describes the many symbols peculiar to the Bash shell. The topics are arranged as follows:

- Special files
- Filename metacharacters
- Quoting
- Command forms
- Redirection forms
- Coprocesses

## Special Files

Bash reads one or more startup files. Some of the files are read only when a shell is a login shell.

The startup files are, in the order they are read:

1. */etc/profile*. Executed automatically at login.
2. The first file found from this list: *~/.bash\_profile*, *~/.bash\_login*, or *~/.profile*. Executed automatically at login.
3. *~/.bashrc* is read by every shell, after the login files. However, if invoked as **sh**, Bash instead reads **\$ENV**.

The **getpwnam()** and **getpwuid()** functions are the sources of home directories for *~name* abbreviations. (On single-user systems, the user database is stored in */etc/passwd*. However on networked systems, this information may come from NIS, NIS+, or LDAP—not your workstation password file.)

## Filename Metacharacters

| Characters           | Meaning                                                                                                               |
|----------------------|-----------------------------------------------------------------------------------------------------------------------|
| *                    | Match any string of zero or more characters.                                                                          |
| ?                    | Match any single character.                                                                                           |
| [abc...]             | Match any one of the enclosed characters; a hyphen can specify a range (e.g., <b>a-z</b> , <b>A-Z</b> , <b>0-9</b> ). |
| [!abc...], [^abc...] | Match any character <i>not</i> enclosed as above.                                                                     |
| ~                    | Home directory of the current user.                                                                                   |
| <i>~name</i>         | Home directory of user <i>name</i> .                                                                                  |
| <i>~-</i>            | Current working directory (\$PWD).                                                                                    |
| <i>~-</i>            | Previous working directory (\$OLDPWD).                                                                                |

With the `extglob` option on:

| Characters              | Meaning                                             |
|-------------------------|-----------------------------------------------------|
| <code>?(pattern)</code> | Match zero or one instance of <i>pattern</i> .      |
| <code>*(pattern)</code> | Match zero or more instances of <i>pattern</i> .    |
| <code>+(pattern)</code> | Match one or more instances of <i>pattern</i> .     |
| <code>@(pattern)</code> | Match exactly one instance of <i>pattern</i> .      |
| <code>!(pattern)</code> | Match any strings that don't match <i>pattern</i> . |

This *pattern* can be a sequence of patterns separated by |, meaning that the match applies to any of the patterns. This extended syntax resembles that available in `egrep` and `awk`.

Bash supports the POSIX [[=c=]] notation for matching characters that have the same weight, and [[.c.]] for specifying collating sequences. In addition, character classes, of the form [[:class:]], allow you to match the following classes of characters.

| Class              | Characters matched      | Class               | Characters matched     |
|--------------------|-------------------------|---------------------|------------------------|
| <code>alnum</code> | Alphanumeric characters | <code>graph</code>  | Nonspace characters    |
| <code>alpha</code> | Alphabetic characters   | <code>print</code>  | Printable characters   |
| <code>blank</code> | Space or tab            | <code>punct</code>  | Punctuation characters |
| <code>cntrl</code> | Control characters      | <code>space</code>  | Whitespace characters  |
| <code>digit</code> | Decimal digits          | <code>upper</code>  | Uppercase characters   |
| <code>lower</code> | Lowercase characters    | <code>xdigit</code> | Hexadecimal digits     |

Bash also accepts the [:word:] character class, which is not in POSIX. [[:word:]] is equivalent to [[:alnum:]\_].

## Examples

```
$ ls new*
$ cat ch?
$ vi [D-R]*
$ pr !(*.o|core) | lp
```

*List new and new.1*  
*Match ch9 but not ch10*  
*Match files that begin with uppercase D through R*  
*Print files that are not object files or core dumps*



On modern systems, ranges such as [D-R] are not portable; the system's locale may include more than just the uppercase letters from D to R in the range.

## Quoting

Quoting disables a character's special meaning and allows it to be used literally, as itself. The following table displays characters that have special meaning to the Bash shell.

| Character         | Meaning                                                       |
|-------------------|---------------------------------------------------------------|
| ;                 | Command separator                                             |
| &                 | Background execution                                          |
| ()                | Command grouping                                              |
|                   | Pipe                                                          |
| < > &             | Redirection symbols                                           |
| *?[]~+-@!         | Filename metacharacters                                       |
| ''\`              | Used in quoting other characters                              |
| `                 | Command substitution                                          |
| \$                | Variable substitution (or command or arithmetic substitution) |
| space tab newline | Word separators                                               |

These characters can be used for quoting:

- " " Everything between " and " is taken literally, except for the following characters that keep their special meaning:
  - \$ Variable (or command and arithmetic) substitution will occur.
  - ' Command substitution will occur.
  - " This marks the end of the double quote.
- ' ' Everything between ' and ' is taken literally except for another '. You cannot embed another ' within such a quoted string.
- \ The character following a \ is taken literally. Use within " " to escape ", \$, and '. Often used to escape itself, spaces, or newlines.
- \$" " Just like " ", except that locale translation is done.
- \$' ' Similar to ' ', but the quoted text is processed for the following escape sequences.

| Sequence | Value               | Sequence | Value                |
|----------|---------------------|----------|----------------------|
| \a       | Alert               | \t       | Tab                  |
| \b       | Backspace           | \v       | Vertical tab         |
| \cx      | Control character X | \nnn     | Octal value nnn      |
| \e       | Escape              | \nnn     | Hexadecimal value nn |
| \E       | Escape              | '        | Single quote         |
| \f       | Form feed           | "        | Double quote         |
| \n       | Newline             | \        | Backslash            |
| \r       | Carriage return     |          |                      |

## Examples

```
$ echo 'Single quotes "protect" double quotes'
Single quotes "protect" double quotes
$ echo "Well, isn't that \"special\"?"
Well, isn't that "special"?
```

```
$ echo "You have `ls | wc -l` files in `pwd`"
You have 43 files in /home/bob
$ echo "The value of \$x is $x"
The value of $x is 100
```

## Command Forms

| Syntax                              | Effect                                                                                                                                                                            |
|-------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>cmd &amp;</code>              | Execute <i>cmd</i> in background.                                                                                                                                                 |
| <code>cmd1 ; cmd2</code>            | Command sequence; execute multiple <i>cmds</i> on the same line.                                                                                                                  |
| <code>{ cmd1 ; cmd2 ; }</code>      | Execute commands as a group in the current shell.                                                                                                                                 |
| <code>(cmd1 ; cmd2)</code>          | Execute commands as a group in a subshell.                                                                                                                                        |
| <code>cmd1   cmd2</code>            | Pipe; use output from <i>cmd1</i> as input to <i>cmd2</i> .                                                                                                                       |
| <code>cmd1 'cmd2'</code>            | Command substitution; use <i>cmd2</i> output as arguments to <i>cmd1</i> .                                                                                                        |
| <code>cmd1 \$(cmd2)</code>          | POSIX shell command substitution; nesting is allowed.                                                                                                                             |
| <code>cmd \${((expression))}</code> | POSIX shell arithmetic substitution. Use the result of <i>expression</i> as argument to <i>cmd</i> .                                                                              |
| <code>cmd1 &amp;&amp; cmd2</code>   | AND; execute <i>cmd1</i> and then (if <i>cmd1</i> succeeds) <i>cmd2</i> . This is a “short-circuit” operation; <i>cmd2</i> is never executed if <i>cmd1</i> fails.                |
| <code>cmd1    cmd2</code>           | OR; execute either <i>cmd1</i> or (if <i>cmd1</i> fails) <i>cmd2</i> . This is a “short-circuit” operation; <i>cmd2</i> is never executed if <i>cmd1</i> succeeds.                |
| <code>!cmd</code>                   | NOT; execute <i>cmd</i> , and produce a zero exit status if <i>cmd</i> exits with a nonzero status. Otherwise, produce a nonzero status when <i>cmd</i> exits with a zero status. |

## Examples

```
$ nroff file > file.txt & Format in the background
$ cd; ls Execute sequentially
$ (date; who; pwd) > logfile All output is redirected
$ sort file | pr -3 | lp Sort file, page output, then print
$ vi `grep -l ifdef *.c` Edit files found by grep
$ egrep '(yes|no)' `cat list` Specify a list of files to search
$ egrep '(yes|no)' $(cat list) POSIX version of previous
$ egrep '(yes|no)' $(< list) Faster, not in POSIX
$ grep XX file && lp file Print file if it contains the pattern;
$ grep XX file || echo "XX not found" Otherwise, echo an error message
```

## Redirection Forms

| File descriptor | Name            | Common abbreviation | Typical default |
|-----------------|-----------------|---------------------|-----------------|
| 0               | Standard input  | stdin               | Keyboard        |
| 1               | Standard output | stdout              | Screen          |
| 2               | Standard error  | stderr              | Screen          |

The usual input source or output destination can be changed, as seen in the following sections.

## Simple redirection

*cmd > file*

Send output of *cmd* to *file* (overwrite).

*cmd >> file*

Send output of *cmd* to *file* (append).

*cmd < file*

Take input for *cmd* from *file*.

*cmd << text*

The contents of the shell script up to a line identical to *text* become the standard input for *cmd* (*text* can be stored in a shell variable). This command form is sometimes called a *Here document*. Input is usually typed at the keyboard or in the shell program. Commands that typically use this syntax include **cat**, **ex**, and **sed**. (If <<- is used, leading tabs are stripped from the contents of the here document, and the tabs are ignored when comparing input with the end-of-input *text* marker.) If any part of *text* is quoted, the input is passed through verbatim. Otherwise, the contents are processed for variable, command, and arithmetic substitutions.

*cmd <<< word*

Supply text of *word*, with trailing newline, as input to *cmd*. (This is known as a *here string*, from the free version of the **rc** shell.)

*cmd <> file*

Open *file* for reading *and* writing on the standard input. The contents are not destroyed.\*

*cmd >| file*

Send output of *cmd* to *file* (overwrite), even if the shell's **noclobber** option is set.

## Redirection using file descriptors

| Syntax                 | Effect                                                                                                                      |
|------------------------|-----------------------------------------------------------------------------------------------------------------------------|
| <i>cmd &gt;&amp;n</i>  | Send <i>cmd</i> output to file descriptor <i>n</i> .                                                                        |
| <i>cmd m&gt;&amp;n</i> | Same, except that output that would normally go to file descriptor <i>m</i> is sent to file descriptor <i>n</i> instead.    |
| <i>cmd &gt;&amp;-</i>  | Close standard output.                                                                                                      |
| <i>cmd &lt;&amp;n</i>  | Take input for <i>cmd</i> from file descriptor <i>n</i> .                                                                   |
| <i>cmd m&lt;&amp;n</i> | Same, except that input that would normally come from file descriptor <i>m</i> comes from file descriptor <i>n</i> instead. |
| <i>cmd &lt;&amp;-</i>  | Close standard input.                                                                                                       |
| <i>cmd &lt;&amp;n-</i> | Move input file descriptor <i>n</i> instead of duplicating it.                                                              |
| <i>cmd &gt;&amp;n-</i> | Move output file descriptor <i>n</i> instead of duplicating it.                                                             |

\* With <, the file is opened read-only, and writes on the file descriptor will fail. With <>, the file is opened read-write; it is up to the application to actually take advantage of this.

## Multiple redirection

| Syntax                                   | Effect                                                                                                                                      |
|------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------|
| <code>cmd 2&gt;file</code>               | Send standard error to <i>file</i> ; standard output remains the same (e.g., the screen).                                                   |
| <code>cmd &gt; file 2&gt;&amp;1</code>   | Send both standard error and standard output to <i>file</i> .                                                                               |
| <code>cmd &amp;&gt;&gt;file</code>       | Append both standard error and standard output to <i>file</i> .                                                                             |
| <code>cmd &amp;&gt; file</code>          | Same. Preferred form.                                                                                                                       |
| <code>cmd &gt;&amp; file</code>          | Same.                                                                                                                                       |
| <code>cmd &gt; f1 2&gt;f2</code>         | Send standard output to file <i>f1</i> , standard error to file <i>f2</i> .                                                                 |
| <code>cmd   tee files</code>             | Send output of <i>cmd</i> to standard output (usually the terminal) and to <i>files</i> . (See the example in Chapter 3 under <b>tee</b> .) |
| <code>cmd 2&gt;&amp;1   tee files</code> | Send standard output and error output of <i>cmd</i> to standard output (usually the terminal) and to <i>files</i> .                         |
| <code>cmd  &amp;</code>                  | Same as <code>cmd 2&gt;&amp;1  </code> to send standard error through a pipe.                                                               |

No space should appear between file descriptors and a redirection symbol; spacing is optional in the other cases.

## Examples

```
$ cat part1 > book
$ cat part2 part3 >> book
$ mail tim < report
$ sed 's/^/XX /g' << END_ARCHIVE
> This is often how a shell archive is "wrapped",
> bundling text for distribution. You would normally
> run sed from a shell program, not from the command line.
> END_ARCHIVE
XX This is often how a shell archive is "wrapped",
XX bundling text for distribution. You would normally
XX run sed from a shell program, not from the command line.
```

To redirect standard output to standard error:

```
$ echo "Usage error: see administrator" 1>&2
```

The following command sends output (files found) to *filelist* and error messages (inaccessible files) to file *no\_access*:

```
$ find / -print > filelist 2>no_access
```

## Coprocesses

A coprocess is a shell command that runs asynchronously in a subshell, connected to the originating shell by a two-way pipe. Set up a coprocess with the **coproc** reserved word:

```
coproc [NAME] command [redirections]
```

The reserved word **coproc** sets up the pipe to communicate with *command* and runs *command* in the background. *NAME* is the name of the coprocess (**COPROC** by default). If *command* is a complex command, *NAME* is optional; with a simple

command, **NAME** must not be given and the default is used. If any redirections are specified, they are set up after the pipe has been set up.

The coprocess establishes an array with two values: **NAME[0]** contains the file descriptor for command output and **NAME[1]** contains the file descriptor for command input. The variable **NAME\_PID** contains the process id of the coprocess. The return status is the exit status of *command*. You can use the **wait** built-in command to wait for the output of *command*.

## Functions

A shell *function* is a grouping of commands within a shell script. Shell functions let you modularize your program by dividing it up into separate tasks. This way the code for each task need not be repeated every time you need to perform the task. The POSIX shell syntax for defining a function follows the Bourne shell:

```
name () {
 function body's code come here
}
```

Functions are invoked just as are regular shell built-in commands or external commands. The command line parameters **\$1**, **\$2**, and so on receive the function's arguments, temporarily hiding the global values of **\$1**, etc. For example:

```
fatal --- print an error message and die:

fatal () { # defining function fatal
 echo "$0: fatal error:" "$@" >&2 # messages to standard error
 exit 1
}
...
if [$# = 0] # not enough arguments
then
 fatal "not enough arguments" # call function with message
fi
```

A function may use the **return** command to return an exit value to the calling shell program. Be careful *not* to use **exit** from within a function unless you really wish to terminate the entire program.

Bash allows you to define functions using an additional keyword, **function**, as follows:

```
function fatal {
 echo "$0: fatal error:" "$@" >&2 # messages to standard error
 exit 1
}
```

All functions share traps with the “parent” shell (except the **DEBUG** trap, if function tracing has been turned on). With the **errtrace** option enabled (either **set -E** or **set -o errtrace**), functions also inherit the **ERR** trap. If function tracing has been enabled, functions inherit the **RETURN** trap. Functions may have local variables, and they may be recursive. The syntax used to define a function is irrelevant.

# Variables

This section describes the following:

- Variable substitution
- Built-in shell variables
- Other shell variables
- Arrays
- Special prompt strings

## Variable Substitution

No spaces should be used in the following expressions. The colon (:) is optional; if it's included, *var* must be nonnull, as well as set.

| Variable expression                        | Description                                                                                                                                                                                     |
|--------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>var=value ...</code>                 | Set each variable <i>var</i> to a <i>value</i> .                                                                                                                                                |
| <code> \${var}</code>                      | Use value of <i>var</i> ; braces are optional if <i>var</i> is separated from the following text. They are required for array variables.                                                        |
| <code> \${var:-value}</code>               | Use <i>var</i> if set; otherwise, use <i>value</i> .                                                                                                                                            |
| <code> \${var:=value}</code>               | Use <i>var</i> if set; otherwise, use <i>value</i> and assign <i>value</i> to <i>var</i> .                                                                                                      |
| <code> \${var:?value}</code>               | Use <i>var</i> if set; otherwise, print <i>value</i> and exit (if not interactive). If <i>value</i> isn't supplied, print the phrase "parameter null or not set."                               |
| <code> \${var:+value}</code>               | Use <i>value</i> if <i>var</i> is set; otherwise, use nothing.                                                                                                                                  |
| <code> \${#var}</code>                     | Use the length of <i>var</i> .                                                                                                                                                                  |
| <code> \${#*}, \${#@}</code>               | Use the number of positional parameters.                                                                                                                                                        |
| <code> \${var#pattern}</code>              | Use value of <i>var</i> after removing <i>pattern</i> from the left. Remove the shortest matching piece.                                                                                        |
| <code> \${var##pattern}</code>             | Same as <code>#pattern</code> , but remove the longest matching piece.                                                                                                                          |
| <code> \${var%pattern}</code>              | Use value of <i>var</i> after removing <i>pattern</i> from the right. Remove the shortest matching piece.                                                                                       |
| <code> \${var%%pattern}</code>             | Same as <code>%pattern</code> , but remove the longest matching piece.                                                                                                                          |
| <code> \${!prefix*}, \${!prefix@}</code>   | List of variables whose names begin with <i>prefix</i> .                                                                                                                                        |
| <code> \${var:pos}, \${var:pos:len}</code> | Starting at position <i>pos</i> (0-based) in variable <i>var</i> , extract <i>len</i> characters, or rest of string if no <i>len</i> . <i>pos</i> and <i>len</i> may be arithmetic expressions. |
| <code> \${var/pat/repl}</code>             | Use value of <i>var</i> , with first match of <i>pat</i> replaced with <i>repl</i> .                                                                                                            |
| <code> \${var/pat/}</code>                 | Use value of <i>var</i> , with first match of <i>pat</i> deleted.                                                                                                                               |
| <code> \${var//pat/repl}</code>            | Use value of <i>var</i> , with every match of <i>pat</i> replaced with <i>repl</i> .                                                                                                            |
| <code> \${var/#pat/repl}</code>            | Use value of <i>var</i> , with match of <i>pat</i> replaced with <i>repl</i> . Match must occur at beginning of the value.                                                                      |
| <code> \${var/%pat/repl}</code>            | Use value of <i>var</i> , with match of <i>pat</i> replaced with <i>repl</i> . Match must occur at end of the value.                                                                            |

You can indirectly reference a variable by “aliasing” one variable name to affect the value of the other:

```
$ greet="hello, world"
$ friendly_message=greet
$ echo ${friendly_message}
hello, world
```

*Create initial variable*  
*Aliasing variable*  
*Use the alias*

## Examples

|                                           |                                                                                                                                               |
|-------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| <code>\$ u=up d=down blank=</code>        | Assign values to three variables (last is null)                                                                                               |
| <code>\$ echo \${u}root</code>            | Braces are needed here                                                                                                                        |
| <code>uproot</code>                       |                                                                                                                                               |
| <code>\$ echo \${u-\$d}</code>            | Display value of <i>u</i> or <i>d</i> ; since <i>u</i> is set, it's printed                                                                   |
| <code>up</code>                           |                                                                                                                                               |
| <code>\$ echo \${tmp-`date`}</code>       | If <i>tmp</i> is not set, the <i>date</i> command is executed                                                                                 |
| <code>Thu May 7 02:09:09 EDT 2009</code>  |                                                                                                                                               |
| <code>\$ echo \${blank="no data"}</code>  | <i>blank</i> is set, so it is printed (a blank line)                                                                                          |
| <code>\$ echo \${blank:="no data"}</code> | <i>blank</i> is set but null, so the string is printed                                                                                        |
| <code>no data</code>                      |                                                                                                                                               |
| <code>\$ echo \$blank</code>              | <i>blank</i> now has a new value                                                                                                              |
| <code>no data</code>                      |                                                                                                                                               |
| <code>\$ tail=\${PWD##*/}</code>          | Take the current directory name and remove the longest character string ending with /, which removes the leading pathname and leaves the tail |

## Built-in Shell Variables

Built-in variables are automatically set by the shell and are typically used inside shell scripts. Built-in variables can make use of the variable substitution patterns shown previously. Note that the \$ is not actually part of the variable name, although the variable is always referenced this way.

| Variable | Description                                                                                                                                                                                      |
|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| \$#      | Number of command-line arguments.                                                                                                                                                                |
| \$-      | Options currently in effect (arguments supplied on command line or to <i>set</i> ).                                                                                                              |
| \$?      | Exit value of last executed command.                                                                                                                                                             |
| \$\$     | Process number of current process.                                                                                                                                                               |
| \$!      | Process number of last background command.                                                                                                                                                       |
| \$0      | First word; that is, command name. This will have the full pathname if it was found via a PATH search.                                                                                           |
| \$n      | Individual arguments on command line (positional parameters). If <i>n</i> is greater than 9, it must be specified as \${n}.                                                                      |
| \$*, \$@ | All arguments on command line (\$1 \$2 ...).                                                                                                                                                     |
| "\$*"    | All arguments on command line as one string ("\$1 \$2 ..."). The values are separated by the first character in the IFS special variable.                                                        |
| "\$@"    | All arguments on command line, individually quoted ("\$1" "\$2" ...).                                                                                                                            |
| \$_      | Temporary variable; initialized to pathname of script or program being executed. Later, stores the last argument of previous command. Also stores name of matching MAIL file during mail checks. |
| HISTCMD  | The history number of the current command.                                                                                                                                                       |
| LINENO   | Current line number within the script or function.                                                                                                                                               |
| OLDPWD   | Previous working directory (set by <i>cd</i> ).                                                                                                                                                  |
| OPTARG   | Name of last option processed by <i>getopts</i> .                                                                                                                                                |
| OPTIND   | Numerical index of OPTARG.                                                                                                                                                                       |
| PPID     | Process number of this shell's parent.                                                                                                                                                           |
| PWD      | Current working directory (set by <i>cd</i> ).                                                                                                                                                   |

| Variable    | Description                                                                                                                    |
|-------------|--------------------------------------------------------------------------------------------------------------------------------|
| RANDOM[=n]  | Generate a new random number with each reference; start with integer <i>n</i> , if given.                                      |
| REPLY       | Default reply, used by <code>select</code> and <code>read</code> .                                                             |
| SECONDS[=n] | Number of seconds since the shell was started, or, if <i>n</i> is given, number of seconds + <i>n</i> since the shell started. |

In addition, Bash sets the following variables. Many of these variables are for use by the Bash debugger (see <http://bashdb.sourceforge.net>) or for providing programmable completion (see the section “Programmable Completion” on page 615).

| Variable              | Description                                                                                                                                                                                                                                                                                                                                              |
|-----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| BASH                  | The full pathname used to invoke this instance of Bash.                                                                                                                                                                                                                                                                                                  |
| BASH_ARGC             | Array variable. Each element holds the number of arguments for the corresponding function or dot-script invocation. Set only in extended debug mode, with <code>shopt -s extdebug</code> .                                                                                                                                                               |
| BASH_ARGV             | An array variable similar to BASH_ARGC. Each element is one of the arguments passed to a function or dot-script. It functions as a stack, with values being pushed on at each call. Thus, the last element is the last argument to the most recent function or script invocation. Set only in extended debug mode, with <code>shopt -s extdebug</code> . |
| BASH_COMMAND          | The command currently executing or about to be executed. Inside a trap handler, it is the command running when the trap was invoked.                                                                                                                                                                                                                     |
| BASH_EXECUTION_STRING | The string argument passed to the <code>-c</code> option.                                                                                                                                                                                                                                                                                                |
| BASH_LINENO           | Array variable, corresponding to BASH_SOURCE and FUNCNAME. For any given function number <i>i</i> (starting at 0), \${FUNCNAME[i]} was invoked in file \${BASH_SOURCE[i]} on line \${BASH_LINENO[i]}. The information is stored with the most recent function invocation first.                                                                          |
| BASH_REMATCH          | Array variable, assigned by the <code>=~</code> operator of the <code>[[ ]]</code> construct. Index 0 is the text that matched the entire pattern. The other indices are the text matched by parenthesized subexpressions. This variable is read-only.                                                                                                   |
| BASH_SOURCE           | Array variable, containing source filenames. Each element corresponds with those in FUNCNAME and BASH_LINENO.                                                                                                                                                                                                                                            |
| BASH_SUBSHELL         | This variable is incremented by one each time a subshell or subshell environment is created.                                                                                                                                                                                                                                                             |
| BASH_VERSINFO[0]      | The major version number, or release, of Bash.                                                                                                                                                                                                                                                                                                           |
| BASH_VERSINFO[1]      | The minor version number, or version, of Bash.                                                                                                                                                                                                                                                                                                           |
| BASH_VERSINFO[2]      | The patch level.                                                                                                                                                                                                                                                                                                                                         |
| BASH_VERSINFO[3]      | The build version.                                                                                                                                                                                                                                                                                                                                       |
| BASH_VERSINFO[4]      | The release status.                                                                                                                                                                                                                                                                                                                                      |
| BASH_VERSINFO[5]      | The machine type; same value as in MACHTYPE.                                                                                                                                                                                                                                                                                                             |
| BASH_VERSION          | A string describing the version of Bash.                                                                                                                                                                                                                                                                                                                 |
| COMP_CWORD            | For programmable completion. Index into COMP_WORDS, indicating the current cursor position.                                                                                                                                                                                                                                                              |
| COMP_LINE             | For programmable completion. The current command line.                                                                                                                                                                                                                                                                                                   |
| COMP_POINT            | For programmable completion. The position of the cursor as a character index in COMP_LINE.                                                                                                                                                                                                                                                               |
| COMP_WORDBREAKS       | For programmable completion. The characters that the readline library treats as word separators when doing word completion.                                                                                                                                                                                                                              |
| COMP_WORDS            | For programmable completion. Array variable containing the individual words on the command line.                                                                                                                                                                                                                                                         |

| Variable                | Description                                                                                                                                                                                                                                    |
|-------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>DIRSTACK</code>   | Array variable, containing the contents of the directory stack as displayed by <code>dirs</code> . Changing existing elements modifies the stack, but only <code>pushd</code> and <code>popd</code> can add or remove elements from the stack. |
| <code>EUID</code>       | Read-only variable with the numeric effective UID of the current user.                                                                                                                                                                         |
| <code>FUNCNAME</code>   | Array variable, containing function names. Each element corresponds with those in <code>BASH_SOURCE</code> and <code>BASH_lineno</code> .                                                                                                      |
| <code>GROUPS</code>     | Array variable containing the list of numeric group IDs in which the current user is a member.                                                                                                                                                 |
| <code>HISTCMD</code>    | The history number of the current command.                                                                                                                                                                                                     |
| <code>HOSTNAME</code>   | The name of the current host.                                                                                                                                                                                                                  |
| <code>HOSTTYPE</code>   | A string that describes the host system.                                                                                                                                                                                                       |
| <code>MACHTYPE</code>   | A string that describes the host system in the GNU <i>cpu-company-system</i> format.                                                                                                                                                           |
| <code>OSTYPE</code>     | A string that describes the operating system.                                                                                                                                                                                                  |
| <code>PIPESTATUS</code> | An array variable containing the exit statuses of the commands in the most recent foreground pipeline.                                                                                                                                         |
| <code>SHELLOPTS</code>  | A colon-separated list of shell options (for <code>set -o</code> ). If set in the environment at startup, Bash enables each option present in the list.                                                                                        |
| <code>SHLVL</code>      | Incremented by one every time a new Bash starts up.                                                                                                                                                                                            |
| <code>UID</code>        | Read-only variable with the numeric real UID of the current user.                                                                                                                                                                              |

## Other Shell Variables

The following variables are not automatically set by the shell, although many of them can influence the shell's behavior. They are typically used in your `.profile` file, where you can define them to suit your needs. Variables can be assigned values by issuing commands of the form:

`variable=value`

This list includes the type of value expected when defining these variables.

| Variable expression                | Description                                                                                                                                     |
|------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>CDPATH=dirs</code>           | DIRECTORIES searched by <code>cd</code> ; allows shortcuts in changing directories; unset by default.                                           |
| <code>COLUMNS=n</code>             | Screen's column width; used in line edit modes and <code>select</code> lists.                                                                   |
| <code>COMPREPLY=(words ...)</code> | ARRAY variable from which Bash reads the possible completions generated by a completion function.                                               |
| <code>EDITOR=file</code>           | Pathname of line-edit mode to turn on (can end in <code>emacs</code> or <code>vi</code> ); used when <code>VISUAL</code> is not set.            |
| <code>EMACS</code>                 | If the value starts with <code>t</code> , Bash assumes it's running in an Emacs buffer and disables line editing.                               |
| <code>ENV=file</code>              | Name of script that gets executed at startup; useful for storing alias and function definitions. For example, <code>ENV=\$HOME/.bashrc</code> . |
| <code>FCEDIT=file</code>           | Editor used by <code>fc</code> command (default is <code>/bin/ed</code> ).                                                                      |
| <code>IGNORE=pattern</code>        | Colon-separated list of patterns describing filenames to ignore when doing filename completion.                                                 |
| <code>GLOBIGNORE=patlist</code>    | Colon-separated list of patterns describing the set of filenames to ignore during pattern matching.                                             |

| Variable expression                 | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|-------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>HISTCONTROL=list</code>       | Colon-separated list of values controlling how commands are saved in the history file. Recognized values are: <code>ignoredups</code> , <code>ignorespace</code> , <code>ignoreboth</code> , and <code>erasedups</code> .                                                                                                                                                                                                                                                           |
| <code>HISTFILE=file</code>          | File in which to store command history. Default is <code>~/.bash_history</code> .                                                                                                                                                                                                                                                                                                                                                                                                   |
| <code>HISTFILESIZE=n</code>         | Number of lines to be kept in the history file. This may be different than the number of commands.                                                                                                                                                                                                                                                                                                                                                                                  |
| <code>HISTIGNORE=list</code>        | A colon-separated list of patterns that must match the entire command line. Matching lines are <i>not</i> saved in the history file. An unescaped & in a pattern matches the previous history line.                                                                                                                                                                                                                                                                                 |
| <code>HISTSIZE=n</code>             | Number of history commands to be kept in the history file.                                                                                                                                                                                                                                                                                                                                                                                                                          |
| <code>HISTTIMEFORMAT=string</code>  | A format string for <code>strftime(3)</code> to use for printing timestamps along with commands from the <code>history</code> command. If set (even if null), Bash saves timestamps in the history file along with the commands.                                                                                                                                                                                                                                                    |
| <code>HOME=dir</code>               | Home directory; set by <code>login</code> (from <code>/etc/passwd</code> file).                                                                                                                                                                                                                                                                                                                                                                                                     |
| <code>HOSTFILE=file</code>          | Name of a file in the same format as <code>/etc/hosts</code> that Bash should use to find hostnames for hostname completion.                                                                                                                                                                                                                                                                                                                                                        |
| <code>IFS='chars'</code>            | Input field separators; default is space, tab, and newline.                                                                                                                                                                                                                                                                                                                                                                                                                         |
| <code>IGNOREEOF=n</code>            | Numeric value indicating how many successive EOF characters must be typed before Bash exits. If null or nonnumeric value, default is 10.                                                                                                                                                                                                                                                                                                                                            |
| <code>INPUTRC=file</code>           | Initialization file for the <code>readline</code> library. This overrides the default value of <code>~/.inputrc</code> .                                                                                                                                                                                                                                                                                                                                                            |
| <code>LANG=dir</code>               | Default value for locale, used if no <code>LC_*</code> variables are set.                                                                                                                                                                                                                                                                                                                                                                                                           |
| <code>LC_ALL=locale</code>          | Current locale; overrides <code>LANG</code> and the other <code>LC_*</code> variables.                                                                                                                                                                                                                                                                                                                                                                                              |
| <code>LC_COLLATE=locale</code>      | Locale to use for character collation (sorting order).                                                                                                                                                                                                                                                                                                                                                                                                                              |
| <code>LC_CTYPE=locale</code>        | Locale to use for character class functions.                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| <code>LC_MESSAGES=locale</code>     | Locale to use for translating \$"..." strings.                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| <code>LC_NUMERIC=locale</code>      | Locale to use for the decimal-point character.                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| <code>LINES=n</code>                | Screen's height; used for <code>select</code> lists.                                                                                                                                                                                                                                                                                                                                                                                                                                |
| <code>MAIL=file</code>              | Default file to check for incoming mail; set by <code>login</code> .                                                                                                                                                                                                                                                                                                                                                                                                                |
| <code>MAILCHECK=n</code>            | Number of seconds between mail checks; default is 600 (10 minutes).                                                                                                                                                                                                                                                                                                                                                                                                                 |
| <code>MAILPATH=files</code>         | One or more files, delimited by a colon, to check for incoming mail. Along with each file, you may supply an optional message that the shell prints when the file increases in size. Messages are separated from the filename by a ? character, and the default message is <code>You have mail in \$_</code> . <code>\$_</code> is replaced with the name of the file. For example, you might have:<br><code>MAILPATH="\$MAIL?Ring! Candygram!: /etc/motd?New Login Message"</code> |
| <code>OPTERR=n</code>               | When set to 1 (the default value), Bash prints error messages from the built-in <code>getopts</code> command.                                                                                                                                                                                                                                                                                                                                                                       |
| <code>PATH=dirlist</code>           | One or more pathnames, delimited by colons, in which to search for commands to execute.                                                                                                                                                                                                                                                                                                                                                                                             |
| <code>POSIXLY_CORRECT=string</code> | When set at startup or while running, Bash enters POSIX mode, disabling behavior and modifying features that conflict with the POSIX standard.                                                                                                                                                                                                                                                                                                                                      |
| <code>PROMPT_COMMAND=command</code> | If set, Bash executes this command each time before printing the primary prompt.                                                                                                                                                                                                                                                                                                                                                                                                    |
| <code>PS1=string</code>             | Primary prompt string; default is \$.                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| <code>PS2=string</code>             | Secondary prompt (used in multiline commands); default is >.                                                                                                                                                                                                                                                                                                                                                                                                                        |
| <code>PS3=string</code>             | Prompt string in <code>select</code> loops; default is #?.                                                                                                                                                                                                                                                                                                                                                                                                                          |

| Variable expression            | Description                                                                                                                                                                                                                                    |
|--------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>PS4=string</code>        | Prompt string for execution trace ( <code>bash -x</code> or <code>set -x</code> ); default is <code>+</code> .                                                                                                                                 |
| <code>SHELL=file</code>        | Name of default shell (e.g., <code>/bin/bash</code> ). Bash sets this if it's not in the environment at startup.                                                                                                                               |
| <code>TERM=string</code>       | Terminal type.                                                                                                                                                                                                                                 |
| <code>TIMEFORMAT=string</code> | A format string for the output for the <code>time</code> keyword.                                                                                                                                                                              |
| <code>TMOUT=n</code>           | If no command is typed after <i>n</i> seconds, exit the shell. Also affects the <code>read</code> command and the <code>select</code> loop.                                                                                                    |
| <code>VISUAL=path</code>       | Same as <code>EDITOR</code> , but <code>VISUAL</code> is checked first.                                                                                                                                                                        |
| <code>auto_resume=list</code>  | Enables the use of simple strings for resuming stopped jobs. With a value of <code>exact</code> , the string must match a command name exactly. With a value of <code>substring</code> , it can match a substring of the command name.         |
| <code>histchars=chars</code>   | Two or three characters that control history expansion. The first character signals a history event. The second is the “quick substitution” character, and the third indicates the start of a comment. The default value is <code>!^#</code> . |

## Arrays

Bash supports one-dimensional arrays. Elements are referenced by an index; the first element is numbered 0 and there is no upper limit on the number of elements. Arrays are initialized with a special form of assignment:

```
message=(hi there how are you today)
```

where each value (in this example, each word) becomes an element of the array.

Elements may also be assigned individually:

```
message[0]=hi This is the hard way
message[1]=there
message[2]=how
message[3]=are
message[4]=you
message[5]=today
```

Declaring arrays is not required. Any valid reference to a subscripted variable can create an array.

Bash also provides associative arrays, where the indices are strings instead of numbers. In this case, [ and ] act like double quotes. Associative arrays are created with `declare -A arrayname`. Unlike indexed arrays, when assigning a value to an associative array, a subscript is always required.

When referencing arrays, use the `$( ... )` syntax. This isn't needed when referencing arrays inside `(( ))` (the form of `let` that does automatic quoting). Note that [ and ] are typed literally (i.e., they don't stand for optional syntax).

| Syntax                                  | Effect                                                                                                                      |
|-----------------------------------------|-----------------------------------------------------------------------------------------------------------------------------|
| <code>\$(name[i])</code>                | Use element <i>i</i> of array <i>name</i> . <i>i</i> can be any arithmetic expression as described under <code>let</code> . |
| <code>\$(name)</code>                   | Use element 0 of indexed array <i>name</i> .                                                                                |
| <code>\$(name[*]), \${name[@]}</code>   | Use all elements of array <i>name</i> .                                                                                     |
| <code>#\${name[*]}, \${#name[@]}</code> | Use the number of elements in array <i>name</i> .                                                                           |

The built-in commands **declare**, **local**, and **readonly** accept the **-a** option for an indexed array and the **-A** option for an associative array. Use the **unset** built-in to remove arrays or array elements. The built-ins are described in detail later in this chapter.

## Special Prompt Strings

Bash processes the values of the built-in shell variables **PS1**, **PS2**, and **PS4** for the following special escape sequences:

| Escape sequence | Description                                                                                            |
|-----------------|--------------------------------------------------------------------------------------------------------|
| \a              | An ASCII BEL character (octal 07).                                                                     |
| \A              | The current time in 24-hour <i>HH:MM</i> format.                                                       |
| \d              | The date in “weekday month day” format.                                                                |
| \D{format}      | The date as specified by the <i>strftime(3)</i> format <i>format</i> . The braces are required.        |
| \e              | An ASCII Escape character (octal 033).                                                                 |
| \h              | The hostname, up to the first period.                                                                  |
| \H              | The full hostname.                                                                                     |
| \j              | The current number of jobs.                                                                            |
| \l              | The basename of the shell’s terminal device.                                                           |
| \n              | A newline character.                                                                                   |
| \r              | A carriage-return character.                                                                           |
| \s              | The name of the shell (basename of \$0).                                                               |
| \t              | The current time in 24-hour <i>HH:MM:SS</i> format.                                                    |
| \T              | The current time in 12-hour <i>HH:MM:SS</i> format.                                                    |
| \u              | The current user’s username.                                                                           |
| \v              | The version of Bash.                                                                                   |
| \V              | The release (version plus patch level) of Bash.                                                        |
| \w              | The current directory, with \$HOME abbreviated as ~.                                                   |
| \W              | The basename of the current directory, with \$HOME abbreviated as ~.                                   |
| \!              | The history number of this command.                                                                    |
| \#              | The command number of this command.                                                                    |
| \\$             | If the effective UID is 0, a #; otherwise a \$.                                                        |
| \@              | The current time in 12-hour a.m./p.m. format.                                                          |
| \nnn            | The character represented by octal value <i>nnn</i> .                                                  |
| \`              | A literal backslash.                                                                                   |
| \[              | Start a sequence of nonprinting characters, such as for highlighting or changing colors on a terminal. |
| \]              | End a sequence of nonprinting characters.                                                              |

In Bash, the escape sequences are processed first. After that, variable, command, and arithmetic substitutions are performed if the **promptvars** shell option is enabled via the **shopt** command (the default).

# Arithmetic Expressions

The `let` built-in command performs integer arithmetic. You can substitute arithmetic values (for use as command arguments or in variables); base conversion is also possible.

| Expression                | Meaning                                                                                                                             |
|---------------------------|-------------------------------------------------------------------------------------------------------------------------------------|
| <code>\$(( expr ))</code> | Use the value of the enclosed arithmetic expression.                                                                                |
| <code>B#n</code>          | Interpret integer <i>n</i> in numeric base <i>B</i> . For example, <code>8#100</code> specifies the octal equivalent of decimal 64. |

## Operators

The shell uses arithmetic operators from the C programming language; the following table lists the operators in decreasing order of precedence.

| Operator                           | Description                                                                     |
|------------------------------------|---------------------------------------------------------------------------------|
| <code>++--</code>                  | Auto-increment and auto-decrement, both prefix and postfix.                     |
| <code>+ - ! ~</code>               | Unary plus and minus, logical negation and binary inversion (one's complement). |
| <code>**</code>                    | Exponentiation.                                                                 |
| <code>* / %</code>                 | Multiplication; division; modulus (remainder).                                  |
| <code>+ -</code>                   | Addition; subtraction.                                                          |
| <code>&lt;&lt; &gt;&gt;</code>     | Bitwise left shift; bitwise right shift.                                        |
| <code>&lt; &lt;= &gt; &gt;=</code> | Less than; less than or equal to; greater than; greater than or equal to.       |
| <code>== !=</code>                 | Equality; inequality (both evaluated left to right).                            |
| <code>&amp;</code>                 | Bitwise AND.                                                                    |
| <code>^</code>                     | Bitwise exclusive OR.                                                           |
| <code> </code>                     | Bitwise OR.                                                                     |
| <code>&amp;&amp;</code>            | Logical AND (short-circuit).                                                    |
| <code>  </code>                    | Logical OR (short-circuit).                                                     |
| <code>?:</code>                    | Inline conditional evaluation.                                                  |
| <code>= += -=</code>               | Assignment.                                                                     |
| <code>*=/=%=</code>                |                                                                                 |
| <code>&lt; &lt;= &gt; &gt;=</code> |                                                                                 |
| <code>&amp;= ^=  =</code>          |                                                                                 |
| <code>'</code>                     | Sequential expression evaluation.                                               |

## Examples

See the `let` command for more information and examples:

```
let "count=0" "i = i + 1" Assign i and count
let "num % 2" Test for an even number
((percent >= 0 && percent <= 100)) Test the range of a value
```

# Command History

Bash lets you display or modify previous commands. Commands in the history list can be modified using:

- Line-edit mode
- The **fc** and **hist** commands

Bash also supports a command-history mechanism very similar to that of the C shell. This mechanism uses a history expansion character (! by default) to select a line from the history. Portions of the line are then selected to be included in the current line. Because the interactive line-editing features are considerably superior we do not cover those features here. See the Bash manpage for more information.

## Line-Edit Mode

Line-edit mode emulates many features of the **vi** and **emacs** editors. The history list is treated like a file. When the editor is invoked, you type editing keystrokes to move to the command line you want to execute. You can also change the line before executing it. When you're ready to issue the command, press the Enter key.

Select an editor with either **set -o vi** or **set -o emacs**; assignment to the **VISUAL** or **EDITOR** variables has no effect. Note that **vi** starts in input mode; to type a **vi** command, press the Escape key first.

### Common editing keystrokes

| vi      | emacs         | Result                                          |
|---------|---------------|-------------------------------------------------|
| k       | Ctrl-P        | Get previous command.                           |
| j       | Ctrl-N        | Get next command.                               |
| /string | Ctrl-R string | Get previous command containing <i>string</i> . |
| h       | Ctrl-B        | Move back one character.                        |
| l       | Ctrl-F        | Move forward one character.                     |
| b       | ESC-B         | Move back one word.                             |
| w       | ESC-F         | Move forward one word.                          |
| X       | DEL           | Delete previous character.                      |
| x       | Ctrl-D        | Delete character under cursor.                  |
| dw      | ESC-D         | Delete word forward.                            |
| db      | ESC-H         | Delete word backward.                           |
| xp      | Ctrl-R        | Transpose two characters.                       |

## The fc Command

**fc** stands for either “find command” or “fix command,” since it does both jobs. Use **fc -l** to list history commands and **fc -e** to edit them. See the **fc** entry in the section “Built-in Commands” on page 619 for more information.

## Examples

|                                    |                                                                    |
|------------------------------------|--------------------------------------------------------------------|
| <code>\$ history</code>            | <i>List the last 16 commands</i>                                   |
| <code>\$ fc -1 20 30</code>        | <i>List commands 20 through 30</i>                                 |
| <code>\$ fc -l -5</code>           | <i>List the last five commands</i>                                 |
| <code>\$ fc -l cat</code>          | <i>List all commands since the last command beginning with cat</i> |
| <code>\$ fc -l 50</code>           | <i>List all commands since command 50</i>                          |
| <code>\$ fc -ln 5 &gt; doit</code> | <i>Save command 5 to file doit</i>                                 |
| <code>\$ fc -e vi 5 20</code>      | <i>Edit commands 5 through 20 using vi</i>                         |
| <code>\$ fc -e emacs</code>        | <i>Edit previous command using emacs</i>                           |

Interactive line editing is easier to use than `fc`, since you can move up and down in the saved command history using your favorite editor commands (as long as your favorite editor is either `vi` or Emacs!). You can also use the Up and Down arrow keys to traverse the command history and the right and left arrow keys to move around in the command line.

## Programmable Completion

Bash and the *readline* library provide *completion* facilities, whereby you can type part of a command name, press the Tab key, and have Bash fill in part or all of the rest of the command or filename. *Programmable completion* lets you, as a shell programmer, write code to customize the list of possible completions that Bash will present for a particular, partially entered word. This is accomplished through the combination of several facilities:

- The `complete` command allows you provide a completion specification, or *compspec*, for individual commands. You specify, via various options, how to tailor the list of possible completions for the particular command. This is simple, but adequate for many needs. (See the `complete` entry in the section “Built-in Commands” on page 619.)
- For more flexibility, you may use `complete -F funcname command`. This tells Bash to call *funcname* to provide the list of completions for *command*. You write the *funcname* function.
- Within the code for a -F function, the COMP\* shell variables provide information about the current command line. COMPREPLY is an array into which the function places the final list of completion results.
- Also within the code for a -F function, you may use the `compgen` command to generate a list of results, such as “usernames that begin with a” or “all set variables.” The intent is that such results would be used with an array assignment:

```
...
COMPREPLY=($(compgen options arguments))
...
```

Compspecs may be associated with either a full pathname for a command or, more commonly, with an unadorned command name (*/usr/bin/man* versus plain **man**). In the list that follows, completions are attempted based on the options provided to the `complete` command.

1. Bash first identifies the command. If a pathname is used, Bash looks to see if a compspec exists for the full pathname. Otherwise, it sets the command name to the last component of the pathname, and searches for a compspec for the command name.
2. If a compspec exists, Bash uses it. If not, Bash falls back to the default built-in completions.
3. Bash performs the action indicated by the compspec to generate a list of possible matches. Of this list, only those that have the word being completed as a prefix are used for the list of possible completions. For the **-d** and **-f** options, the variable **IGNORE** is used to filter out undesirable matches.
4. Bash generates filenames as specified by the **-G** option. **GLOBIGNORE** is not used to filter the results, but **IGNORE** is.
5. Bash processes the argument string provided to **-W**. The string is split using the characters in **\$IFS**. The resulting list provides the candidates for completion. This is often used to provide a list of options that a command accepts.
6. Bash runs functions and commands as specified by the **-F** and **-C** options. For both, Bash sets **COMP\_LINE** and **COMP\_POINT** as described previously. For a shell function, **COMP\_WORDS** and **COMP\_CWORD** are also set. Also for both, **\$1** is the name of the command whose arguments are being completed, **\$2** is the word being completed, and **\$3** is the word in front of the word being completed. Bash does *not* filter the results of the command or function.
  - a. Functions named with **-F** are run first. The function should set the **COMPREPLY** array to the list of possible completions. Bash retrieves the list from there.
  - b. Commands provided with **-C** are run next, in an environment equivalent to command substitution. The command should print the list of possible completions, one per line. An embedded newline should be escaped with a backslash.
7. Once the list is generated, Bash filters the results according to the **-X** option. The argument to **-X** is a pattern specifying files to exclude. By prefixing the pattern with a **!**, the sense is reversed, and the pattern instead specifies that only matching files should be retained in the list.  
An **&** in the pattern is replaced with the text of the word being completed. Use **\&** to produce a literal **&**.
8. Finally, Bash prepends or appends any prefixes or suffixes supplied with **-P** or **-S** options.
9. In the case that no matches were generated, if **-o dirnames** was used, Bash attempts directory name completion.
10. On the other hand, if **-o plusdirs** was provided, Bash *adds* the result of directory completion to the previously generated list.
11. Normally, when a compspec is provided, Bash's default completions are not attempted, nor are the *readline* library's default filename completions.
  - a. If the compspec produces no results and **-o bashdefault** was provided, then Bash attempts its default completions.

- b. If neither the compspec, nor the Bash default completions with **-o bash-default** produced any results, and **-o default** was provided, then Bash has the *readline* library attempt its filename completions.

Ian Macdonald has collected a large set of useful compspecs, often distributed as the file */etc/bash\_completion*.

## Examples

Restrict files for the C compiler to C, C++ and assembler source files, and relocatable object files:

```
complete -f -X '!*.[CcOs]' gcc cc
```

For the **man** command, restrict expansions to things that have manpages:

```
Simple example of programmable completion for manual pages.
A more elaborate example appears in the bash_completion file.
Assumes man [num] command command syntax.

shopt -s extglob Enable extended pattern matching
_coman () { Local variables
 local dir mandir=/usr/share/man
 COMPREPLY=() Clear reply list
 if [[${COMP_WORDS[1]} = +([0-9])]] Section number provided
 then
 # section provided: man 3 foo
 dir=$mandir/man${COMP_WORDS[COMP_CWORD-1]} Look in that directory
 else
 # no section, default to commands
 dir=$mandir/'man[18]' Look in command directories
 fi
 COMPREPLY=($(find $dir -type f |
 sed 's;..*/;;' | Generate raw file list
 sed 's/[.][0-9].*$/ /' | Remove leading directories
 grep "^\${COMP_WORDS[$COMP_CWORD]}"' | Remove trailing suffixes
 sort) Keep those that match
 given prefix
 Sort final list
}
complete -F _man man Associate function with command
```

## Job Control

Job control lets you place foreground jobs in the background, bring background jobs to the foreground, or suspend (temporarily stop) running jobs. Many job-control commands take a *jobID* as an argument. This argument can be specified as follows:

**%n** Job number *n*.

**%s** Job whose command line starts with string *s*.

**%?s**  
Job whose command line contains string *s*.  
**%%**  
Current job.  
**%+**  
Current job (same as above).  
**%-**  
Previous job.

The following job-control commands are described more completely in the section “Built-in Commands” on page 619.

**bg** Put a job in the background.

**fg** Put a job in the foreground.

**jobs**

List active jobs.

**kill**

Terminate a job.

**stty tostop**

Stop background jobs if they try to send output to the terminal. (Note that **stty** is not a built-in command.)

**suspend**

Suspend a job-control shell (such as one created by **su**).

**wait**

Wait for background jobs to finish.

**Ctrl-Z**

Suspend a foreground job. Then use **bg** or **fg**. (Your terminal may use something other than **Ctrl-Z** as the suspend character.)

## Command Execution

When you type a command to Bash, it looks in the following places until it finds a match:

1. Keywords such as **if** and **for**.
2. Aliases. You can’t define an alias whose name is a shell keyword, but you can define an alias that expands to a keyword, e.g., **alias aslongas=while**. (In non-POSIX mode, Bash does allow you to define an alias for a shell keyword.)
3. Special built-ins like **break** and **continue**. The list of POSIX special built-ins is . (dot), :, **break**, **continue**, **eval**, **exec**, **exit**, **export**, **readonly**, **return**, **set**, **shift**, **source**, **times**, **trap**, and **unset**.
4. Functions. When in non-POSIX mode, Bash finds functions before built-in commands.
5. Nonspecial built-ins like **cd** and **test**.
6. Scripts and executable programs, for which the shell searches in the directories listed in the PATH environment variable.

The distinction between “special” built-in commands and nonspecial ones comes from POSIX. This distinction, combined with the **command** command, makes it possible to write functions that override shell built-ins, such as **cd**. For example:

```
cd () {
 command cd "$@"
 echo now in $PWD
}

Shell function, found before built-in cd
Use real cd to change directory
Other stuff we want to do
```

## Restricted Shells

A *restricted shell* is one that disallows certain actions, such as changing directory, setting PATH, or running commands whose names contain a / character. See the Bash manpage for the full list of restrictions.

To run a restricted shell, enter the command **bash -r**. Depending on your Linux distribution, you may also be able to enter the command as **rbash**.

You can still run shell scripts, since in that case the restricted shell calls the unrestricted version of the shell to run the script after it reads */etc/profile*, *\$HOME/.profile*, and other startup files.

Restricted shells are not used much in practice, as they are difficult to set up correctly.

## Built-in Commands

Examples to be entered as a command line are shown with the \$ prompt. Otherwise, examples should be treated as code fragments that might be included in a shell script. For convenience, some of the reserved words used by multiline commands are also included.

---

!

### *! pipeline*

Negate the sense of a pipeline. Returns an exit status of 0 if the pipeline exited nonzero, and an exit status of 1 if the pipeline exited zero. Typically used in **if** and **while** statements.

### **Example**

This code prints a message if user **jane** is not logged on:

```
if ! who | grep jane > /dev/null
then
 echo jane is not currently logged on
fi
```

---

#

Ignore all text that follows on the same line. # is used in shell scripts as the comment character and is not really a command.

---

|                |                                                                                                                                                                                                                                                                                                                                                                                            |
|----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>#!shell</b> | <code>#!/shell [option]</code>                                                                                                                                                                                                                                                                                                                                                             |
|                | Used as the first line of a script to invoke the named <i>shell</i> . Anything given on the rest of the line is passed as a single argument to the named <i>shell</i> . This feature is typically implemented by the kernel, but may not be supported on some older systems. Some systems have a limit of approximately 32 characters on the maximum length of <i>shell</i> . For example: |
|                | <code>#!/bin/sh</code>                                                                                                                                                                                                                                                                                                                                                                     |
| <b>:</b>       | <b>:</b>                                                                                                                                                                                                                                                                                                                                                                                   |
|                | Null command. Returns an exit status of 0. The line is still processed for side effects, such as variable and command substitutions, or I/O redirection. See the following Example and the Example under <b>case</b> .                                                                                                                                                                     |
|                | <b>Example</b>                                                                                                                                                                                                                                                                                                                                                                             |
|                | Check whether someone is logged in:                                                                                                                                                                                                                                                                                                                                                        |
|                | <pre>if who   grep \$1 &gt; /dev/null then : # Do nothing if user is found else echo "User \$1 is not logged in" fi</pre>                                                                                                                                                                                                                                                                  |
| <b>.</b>       | <b>.</b> <i>file</i> [ <i>arguments</i> ]                                                                                                                                                                                                                                                                                                                                                  |
|                | Read and execute lines in <i>file</i> . <i>file</i> does not have to be executable, but must reside in a directory searched by PATH. The <i>arguments</i> are stored in the positional parameters. If Bash is not in POSIX mode and <i>file</i> is not found in PATH, Bash looks in the current directory for the <i>file</i> .                                                            |
| <b>[[ ]]</b>   | <b>[[ expression ]]</b>                                                                                                                                                                                                                                                                                                                                                                    |
|                | Same as <b>test expression</b> or <b>[ expression ]</b> , except that <b>[[ ]]</b> allows additional operators. Word splitting and filename expansion are disabled. Note that the brackets ( <b>[ ]</b> ) are typed literally and that they must be surrounded by whitespace.                                                                                                              |
|                | <b>Additional operators</b>                                                                                                                                                                                                                                                                                                                                                                |
|                | && Logical AND of test expressions (short circuit).                                                                                                                                                                                                                                                                                                                                        |
|                | Logical OR of test expressions (short circuit).                                                                                                                                                                                                                                                                                                                                            |
|                | < First string is lexically “less than” the second.                                                                                                                                                                                                                                                                                                                                        |
|                | > First string is lexically “greater than” the second.                                                                                                                                                                                                                                                                                                                                     |
| <b>alias</b>   | <b>alias [option] [name[='cmd']...]</b>                                                                                                                                                                                                                                                                                                                                                    |
|                | Assign a shorthand <i>name</i> as a synonym for <i>cmd</i> . If <b>='cmd'</b> is omitted, print the alias for <i>name</i> ; if <i>name</i> is also omitted, print all aliases. By itself or with <b>-p</b> , <b>alias</b> prints one alias per line on standard output as                                                                                                                  |

---

**alias** *name=value*. If the value contains a trailing space, the next word on the command line also becomes a candidate for alias expansion. See also **unalias**.

### Option

**-p** Print the word **alias** before each alias.

### Example

```
alias dir='echo ${PWD##*/}'
```

## bg

**bg** [*jobIDs*]

Put current job or *jobIDs* in the background. See the section “Job Control” on page 617.

## bind

```
bind [-m map] [options]
bind [-m map] [-q function] [-r sequence] [-u function]
bind [-m map] -f file
bind [-m map] -x sequence:command
bind [-m map] sequence:function
bind readline-command
```

Manage the *readline* library. Nonoption arguments have the same form as in an *.inputrc* file.

### Options

#### **-f** *file*

Read key bindings from *file*.

**-l** List the names of all the *readline* functions.

#### **-m** *map*

Use *map* as the keymap. Available keymaps are: **emacs**, **emacs-standard**, **emacs-meta**, **emacs-ctlx**, **vi**, **vi-move**, **vi-command**, and **vi-insert**. **vi** is the same as **vi-command**, and **emacs** is the same **emacs-standard**.

**-p** Print the current *readline* bindings such that they can be reread from a *.inputrc* file.

**-P** Print the current *readline* bindings.

#### **-q** *function*

Query which keys invoke the *readline* function *function*.

#### **-r** *sequence*

Remove the binding for key sequence *sequence*.

**-s** Print the current *readline* key sequence and macro bindings such that they can be reread from a *.inputrc* file.

**-S** Print the current *readline* key sequence and macro bindings.

#### **-u** *function*

Unbind all keys that invoke the *readline* function *function*.

**-v** Print the current *readline* variables such that they can be reread from a *.inputrc* file.

**-V** Print the current *readline* variables.

**-x sequence:command**

Execute the shell command *command* whenever *sequence* is entered.

---

**break**

`break [n]`

Exit from a **for**, **while**, **select**, or **until** loop (or break out of *n* loops).

---

**builtin**

`builtin command [ arguments ... ]`

Run the shell built-in command *command* with the given arguments. This allows you to bypass any functions that redefine a built-in command's name. The **command** command is more portable.

**Example**

This function lets you do your own tasks when you change directory:

```
cd () {
 builtin cd "$@" Actually change directory
 pwd Report location
}
```

**caller**

`caller [expression]`

Print the line number and source filename of the current function call or dot file. With nonzero *expression*, prints that element from the call stack. The most recent is zero. This command is for use by the Bash debugger.

---

**case**

```
case value in
 pattern1) cmd1;;
 pattern2) cmd2;;
 .
 .
 .
esac
```

Execute the first set of commands (*cmd1*) if *value* matches *pattern1*, execute the second set of commands (*cmd2*) if *value* matches *pattern2*, etc. The last command in each set ends with `;;` and no further matches are attempted. If the set ends in `;.&` instead, execution continues with the commands for the next set of patterns; if it ends in `;&.`, the next pattern in the list is tested. *value* is typically a positional parameter or other shell variable. *cmds* are typically Linux commands, shell programming commands, or variable assignments. Patterns can use file-generation metacharacters. Multiple patterns (separated by `|`) can be specified on the same line; in this case, the associated *cmds* are executed whenever *value*

matches any of these patterns. See the Examples here and under eval.

A pattern may be preceded by an optional open parenthesis, as in *(pattern*, necessary for balancing parentheses inside a `$( )` construct.

## Examples

Check first command-line argument and take appropriate action:

```
case $1 in # Match the first arg
 no|yes) response=1;;
 -[tT]) table=TRUE;;
 *) echo "unknown option"; exit 1;;
esac
```

Read user-supplied lines until user exits:

```
while : # Null command; always true
do
 printf "Type . to finish ==> "
 read line
 case "$line" in
 .) echo "Message done"
 break ;;
 *) echo "$line" >> $message ;;
 esac
done
```

## cd

```
cd [-LP] [dir]
cd [-LP] [-]
```

With no arguments, change to the user's home directory. Otherwise, change the working directory to *dir*. If *dir* is a relative pathname but is not in the current directory, the CDPATH variable is searched. A directory of - stands for the previous directory.

## Options

- L Use the logical path (what the user typed, including any symbolic links) for cd .. and the value of PWD. This is the default.
- P Use the actual filesystem physical path for cd .. and the value of PWD.

## command

```
command [-pvV] name [arg ...]
```

Without -v or -V, execute *name* with given arguments. This command bypasses any aliases or functions that may be defined for *name*. When used with a special built-in, it prevents the built-in from exiting the script if it fails.

## Options

- p Use a predefined, default search path, not the current value of PATH.
- v Print a description of how the shell interprets *name*.

- V Print a more verbose description of how the shell interprets *name*.

### Example

Create an alias for **rm** that gets the system's version, and runs it with the **-i** option:

```
$ alias 'rm=command -p rm -i'
```

---

|                       |                                                                                                                                                                                                                                                                                                  |
|-----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>compgen</b>        | <code>compgen [options] [string]</code><br>Generate possible completions for <i>string</i> according to the options and write to standard output. Options are those accepted by <b>complete</b> , except for <b>-p</b> and <b>-r</b> . For more information, see the entry for <b>complete</b> . |
| <b>complete</b>       | <code>complete [options] command ...</code><br>Specify the way to complete arguments for each <i>command</i> . See “Programmable Completion” on page 615 for more discussion.                                                                                                                    |
| <b>Options</b>        |                                                                                                                                                                                                                                                                                                  |
| <b>-A type</b>        | Use <i>type</i> to specify a list of possible completions. The <i>type</i> may be one of the following. Options in parentheses are alternative specifications for <b>-A type</b> .                                                                                                               |
| <b>alias (-a)</b>     | Alias names.                                                                                                                                                                                                                                                                                     |
| <b>arrayvar</b>       | Array variable names.                                                                                                                                                                                                                                                                            |
| <b>binding</b>        | Bindings from the <i>readline</i> library.                                                                                                                                                                                                                                                       |
| <b>builtin (-b)</b>   | Shell built-in command names.                                                                                                                                                                                                                                                                    |
| <b>command (-c)</b>   | Command names.                                                                                                                                                                                                                                                                                   |
| <b>directory (-d)</b> | Directory names.                                                                                                                                                                                                                                                                                 |
| <b>disabled</b>       | Names of disabled shell built-in commands.                                                                                                                                                                                                                                                       |
| <b>enabled</b>        | Names of enabled shell built-in commands.                                                                                                                                                                                                                                                        |
| <b>export (-e)</b>    | Names of exported shell variables.                                                                                                                                                                                                                                                               |
| <b>file (-f)</b>      | Filenames.                                                                                                                                                                                                                                                                                       |
| <b>function</b>       | Names of shell functions.                                                                                                                                                                                                                                                                        |

- group (-g)**  
Group names.
- helptopic**  
Help topics as allowed by the **help** built-in command.
- hostname**  
Hostnames, as found in the file named by \$HOSTFILE.
- job (-j)**  
Job names.
- keyword (-k)**  
Shell reserved keywords.
- running**  
Names of running jobs.
- service (-s)**  
Service names (from */etc/services*).
- setopt**  
Valid arguments for **set -o**.
- shopt**  
Valid option names for the **shopt** built-in command.
- signal**  
Signal names.
- stopped**  
Names of stopped jobs.
- user (-u)**  
Usernames.
- variable (-v)**  
Shell variable names.
- C command**  
Run *command* in a subshell and use its output as the list of completions.
- E** Remaining options and actions apply to completion attempts on a blank line.
- F function**  
Run shell function *function* in the current shell. Upon its return, retrieve the list of completions from the COMPREPLY array.
- G pattern**  
Expand *pattern* to generate completions.
- o option**  
Control the behavior of the completion specification. The value for *option* is one of the following:
- bashdefault**  
Fall back to the normal Bash completions if no matches are produced.
  - default**  
Use the default *readline* completions if no matches are produced.

**dirnames**

Do directory name completion if no matches are produced.

**filenames**

Inform the *readline* library that the intended output is filenames, so that the library can do any filename-specific processing, such as adding a trailing slash for directories or removing trailing spaces.

**nospace**

Inform the *readline* library that it should not append a space to words completed at the end of a line.

**plusdirs**

Attempt directory completion and add any results to the list of completions already generated.

- p With no commands, print all completion settings in a way that can be reread.

**-P prefix**

The *prefix* is added to each resulting string as a prefix after all the other options have been applied.

- r Remove the completion settings for the given commands, or all settings if no commands.

**-S suffix**

The *suffix* is added to each resulting string as a suffix after all the other options have been applied.

**-W wordlist**

Split *wordlist* (a single shell word) using \$IFS. The generated list contains the members of the split list that matched the word being completed. Each member is expanded using brace expansion, tilde expansion, parameter and variable expansion, command substitution, and arithmetic expansion. Shell quoting is respected.

**-X pattern**

Exclude filenames matching *pattern* from the filename completion list. With a leading ! in the pattern, the sense is reversed, and only filenames matching *pattern* are retained.

---

**continue**

`continue [n]`

Skip remaining commands in a **for**, **while**, **select**, or **until** loop, resuming with the next iteration of the loop (or skipping *n* loops).

---

**declare**

`declare [options] [name[=value]]`  
`typeset [options] [name[=value]]`

Declare variables and manage their attributes. In function bodies, variables are local, as if declared with the **local** command.

**Options**

- a Each *name* is an indexed array variable.
- A Each *name* is an associative array variable.
- f Each *name* is a function.

- F For functions, print just the function name and attributes, not the function definition (body). Implies -f.
  - i Each variable is an integer; in an assignment, the value is evaluated as an arithmetic expression.
  - l Assign all values as lowercase only; convert uppercase to lowercase.
  - p With no *names*, print all variables and their values. With *names*, print the names, attributes, and values of the given variables. Used with -f, print all function names and attributes. This option causes all other options to be ignored.
  - r Mark *names* as read-only. Subsequent assignments will fail.
  - t Apply the *trace* attribute to each name. Traced functions inherit the DEBUG and RETURN traps from the shell. This attribute has no meaning for variables.
  - u Assign all values as uppercase only; convert lowercase to uppercase.
  - x Mark *names* for export into the environment of child processes.
- With a + instead of a -, the given attribute is disabled. With no variable names, all variables having the given attribute(s) are printed in a form that can be reread as input to the shell.

### Examples

```
$ declare -i val Make val an integer
$ val=4+7 Evaluate value
$ echo $val Show result
11
```

```
$ declare -r z=42 Make z read-only
$ z=31 Try to assign to it
bash: z: readonly variable Assignment fails
$ echo $z
42
```

```
$ declare -p val z Show attributes and values
declare -i val=val="11"
declare -r z=z="42"
```

## dirs

dirs [-clpv] [+n] [-n]

Print the directory stack, which is managed with **pushd** and **popd**.

### Options

- +n Print the *n*th entry from the left; first entry is zero.
- n Print the *n*th entry from the right; first entry is zero.
- c Remove all entries from (clear) the directory stack.
- l Produce a longer listing, one that does not replace \$HOME with ~.
- p Print the directory stack, one entry per line.

- v Print the directory stack, one entry per line, with each entry preceded by its index in the stack.
- 

**disown** disown [-ahr] [job ...]  
Remove *job* from the list of jobs managed by Bash.

### Options

- a Remove all jobs. With -h, mark all jobs.
  - h Instead of removing jobs from the list of known jobs, mark them to *not* receive SIGHUP when Bash exits.
  - r With no jobs, remove (or mark) only running jobs.
- 

**do** do  
Reserved word that precedes the command sequence in a **for**, **while**, **until**, or **select** statement.

**done** done  
Reserved word that ends a **for**, **while**, **until**, or **select** statement.

---

**echo** echo [-eEn] [string]  
Write *string* to standard output. (See also **echo** in Chapter 3.)

### Options

- e Enable interpretation of the following escape sequences, which must be quoted (or escaped with a \) to prevent interpretation by the shell:

- \a Alert (ASCII BEL).
- \b Backspace.
- \c Suppress the terminating newline (same as -n).
- \e ASCII Escape character.
- \f Formfeed.
- \n Newline.
- \r Carriage return.
- \t Tab character.
- \v Vertical-tab character.
- \\\ Backslash.

#### \0nnn

ASCII character represented by octal number *nnn*, where *nnn* is zero, one, two, or three digits and is preceded by a 0.

#### \nnn

ASCII character represented by octal number *nnn*, where *nnn* is one, two, or three digits.

\xHH

ASCII character represented by hexadecimal number *HH*, where *HH* is one or two hexadecimal digits.

- E Do not interpret escape sequences, even on systems where the default behavior of the built-in **echo** is to interpret them.
- n Do not print the terminating newline.

### Examples

```
$ echo "testing printer" | lp
$ echo "Warning: ringing bell \a"
```

## enable

**enable** [-adnps] [-f *file*] [*command* ...]

Enable or disable shell built-in commands. Disabling a built-in lets you use an external version of a command that would otherwise use a built-in version, such as **echo** or **test**.

### Options

- a For use with -p, print information about all built-in commands, disabled and enabled.
- d Remove (delete) a built-in previously loaded with -f.
- f*file* Load a new built-in command *command* from the shared library file *file*.
- n Disable the named built-in commands.
- p Print a list of enabled built-in commands.
- s Print only the POSIX special built-in commands. When combined with -f, the new built-in command becomes a POSIX special built-in.

## esac

**esac**

Reserved word that ends a **case** statement.

## eval

**eval** *args*

Typically, **eval** is used in shell scripts, and *args* is a line of code that contains shell variables. **eval** forces variable expansion to happen first and then runs the resulting command. This “double-scanning” is useful any time shell variables contain input/output redirection symbols, aliases, or other shell variables. (For example, redirection normally happens before variable expansion, so a variable containing redirection symbols must be expanded first using **eval**; otherwise, the redirection symbols remain uninterpreted.)

### Example

This fragment of a shell script shows how **eval** constructs a command that is interpreted in the right order:

```
for option
do
```

```
 case "$option" in
 save) out=' > $newfile' ;;
 show) out=' | more' ;;
 esac
 done

 eval sort $file $out
```

---

## exec

`exec [-a name] [-cl] [command args ...]`

Execute *command* in place of the current process (instead of creating a new process). **exec** is also useful for opening, closing, or copying file descriptors.

### Options

**-a *name***

Use *name* for the value of `argv[0]`.

**-c** Clear the environment before executing the program.

**-l** Place a minus sign at the front of `argv[0]`, just as `login(1)` does.

### Examples

`trap 'exec 2>&-' 0`

*Close standard error when shell script exits (signal 0)*

`$ exec /bin/csh`

*Replace shell with C shell*

`$ exec < infile`

*Reassign standard input to infile*

---

## exit

`exit [n]`

Exit a shell script with status *n* (e.g., **exit 1**). *n* can be 0 (success) or nonzero (failure). If *n* is not given, exit status is that of the most recent command. **exit** can be issued at the command line to close a window (log out). Exit statuses can range in value from 0 to 255.

### Example

```
if [$# -eq 0]
then
 echo "Usage: $0 [-c] [-d] file(s)" 1>&2
 exit 1 # Error status
fi
```

## export

`export [-fn] [name=[value] ...]
export -p`

Pass (**export**) the value of one or more shell variables, specified by *name*, giving them global meaning (they are local by default). For example, a variable defined in one shell script must be exported if its value is used in other programs called by the script. If a *value* is specified, the variable is set to that value. If no *names* are given, or with **-p**, **export** lists the variables exported by the current shell.

## Options

- f Names refer to functions; the functions are exported in the environment.
- n Remove the named variables or functions from the environment.
- p Print names and values of exported variables.

## Example

In the original Bourne shell, you would type:

```
TERM=vt100
export TERM
```

In Bash, type this instead:

```
export TERM=vt100
```

## false

false

Built-in command that exits with a false return value.

## fc

```
fc [-lnr] [-e editor] [first [last]]
fc -s [old=new] [command]
```

Display or edit commands in the history list. *first* and *last* are numbers or strings specifying the range of commands to display or edit. If *last* is omitted, fc applies to a single command (specified by *first*). If both *first* and *last* are omitted, fc edits the previous command or lists the last 16. The second form of fc takes a history *command*, replaces *old* with *new*, and executes the modified command. If no strings are specified, *command* is just reexecuted. *command* is a number or string like *first*. See the examples in the earlier section “Command History” on page 614.

## Options

Use only one of -e, -l or -s.

### -e editor

Invoke *editor* to edit the specified history commands. If no *editor* is specified, Bash defaults first to the value of FCEDIT, then to the value of EDITOR, then to vi.

**-l** List the specified command or range of commands, or list the last 16.

**-n** Suppress command numbering from the -l listing.

**-r** Reverse the order of the -l listing.

### -s [old=new]

Replace the string *old* with *new* in the specified command and execute the modified command.

## fg

```
fg [jobIDs]
```

Bring current job or *jobIDs* to the foreground. See the section “Job Control” on page 617.

**fi**      fi  
Reserved word that ends an if statement. (Don't forget to use it!)

**for**            *for x [in list]*  
          *do*  
          *commands*  
          *done*

For variable *x* (in optional *list* of values) do *commands*. If **in** *list* is omitted, "\$@" (the positional parameters) is assumed.

## Examples

Paginate files specified on the command line; save each result:

```
for file; do
 pr $file > $file.tmp
done
```

Search chapters for a list of words (like **fgrep -f**):

```
for item in `cat program_list`
do
 echo "Checking chapters for"
 echo "references to program $item..."
 grep -c "$item.[co]" chap*
done
```

Extract a one-word title from each file and use as new filename:

```
for file
do
 name=`sed -n 's/NAME: //p' $file`
 mv $file $name
done
```

```
for ((init; cond; incr)
do
 commands
done
```

Arithmetic **for** loop, similar to C's. Evaluate *init*. While *cond* is true, execute the body of the loop. Evaluate *incr* before retesting *cond*. Any one of the expressions may be omitted; a missing *cond* is treated as being true.

## Example

Search for a phrase in each odd chapter:

```
for ((x=1; x <= 20; x += 2))
do
 grep $1 chap$x
done
```

---

|                 |                                                                                                                                                                                                                           |
|-----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>function</b> | <code>[function] name () { commands; }</code><br><code>function name { commands; }</code>                                                                                                                                 |
|                 | Define <i>name</i> as a shell function. See the description of semantic issues in the section “Functions” on page 605. If the reserved word <b>function</b> is given, the parentheses following <i>name</i> are optional. |

**Example**

Define a function to count files:

```
$ function fcount {
 > ls | wc -l
 >}
```

---

|                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>getopts</b> | <code>getopts string name [args]</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|                | Process command-line arguments (or <i>args</i> , if specified) and check for legal options. <b>getopts</b> is used in shell-script loops and is intended to ensure standard syntax for command-line options. Standard syntax dictates that command-line options begin with <code>-</code> . Options can be stacked: i.e., consecutive letters can follow a single <code>-</code> . End processing of options by specifying <code>--</code> on the command line. <i>string</i> contains the option letters to be recognized by <b>getopts</b> when running the shell script. Valid options are processed in turn and stored in the shell variable <i>name</i> . If an option is followed by a colon, the option must be followed by one or more arguments. (Multiple arguments must be given to the command as one shell <i>word</i> . This is done by quoting the arguments or separating them with commas. The application must be written to expect multiple arguments in this format.) <b>getopts</b> uses the shell variables <code>OPTARG</code> , <code>OPTIND</code> , and <code>OPTERR</code> . |

---

|             |                                                                                                                                                                                                                             |
|-------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>hash</b> | <code>hash [-drlt] [-p file] [commands]</code>                                                                                                                                                                              |
|             | As the shell finds commands along the search path ( <code>\$PATH</code> ), it remembers the found location in an internal hash table. The next time you enter a command, the shell uses the value stored in its hash table. |

With no arguments, **hash** lists the current hashed commands. The display shows hits (the number of times the command is called by the shell) and the command name.

With *commands*, the shell adds those commands to the hash table.

**Options**

- d Remove (delete) just the specified commands from the hash table.
- l Produce output in a format that can be reread to rebuild the hash table.

**-p file**

Associate *file*, assumed to be the full pathname, with *command* in the hash table rather than searching `$PATH`.

- r Remove all commands from the hash table.
- t With one name, print the full pathname of the command. With more than one name, print the command name and the full path, in two columns.

Besides the -r option, the hash table is also cleared when PATH is assigned. Use **PATH=\$PATH** to clear the hash table without affecting your search path. This is most useful if you have installed a new version of a command in a directory that is earlier in \$PATH than the current version of the command.

---

|             |                                                                                                                                                                                                                         |
|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>help</b> | <b>help [-s] [pattern]</b>                                                                                                                                                                                              |
|             | Print usage information on standard output for each Bash command that matches <i>pattern</i> . The information includes descriptions of each command's options. With the -s option, print only brief usage information. |

### Examples

```
$ help -s cd Short help
cd: cd [-L|-P] [dir]
```

```
$ help true Full help
true: true
Return a successful result.
```

```
Exit Status:
Always succeeds.
```

---

|                |                                                                                                                                                                                                                               |
|----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>history</b> | <b>history [count]</b><br><b>history [options]</b>                                                                                                                                                                            |
|                | Print commands in the history list or manage the history file. With no options or arguments, display the history list with command numbers. With a <i>count</i> argument, print only that number of the most recent commands. |

### Options

- a Append new history lines (those executed since the beginning of the session) to the history file.
- c Clear the history list (remove all entries).
- d *position* Delete the history item at position *position*.
- n Read unread history lines from the history file into the history list.
- p *argument* ... Perform history substitution on each *argument*, printing the results to standard output. The results are not saved in the history list. Each argument must be quoted.
- r Read the history file and replace the history list with its contents.

- s *argument* ...
    - Store the *arguments* in the history list as a single entry.
  - w Write the current history list to the history file, overwriting the file.
- 

```
if if condition1
 then commands1
 [elif condition2
 then commands2]
 .
 .
 .
 [else commands3]
 fi
```

If *condition1* is met, do *commands1*; otherwise, if *condition2* is met, do *commands2* ; if neither is met, do *commands3*. Conditions are often specified with the **test** and [[ ]] commands. See **test** and [[ ]] for a full list of conditions, and see additional Examples under : and **exit**.

### Examples

Insert a 0 before numbers less than 10:

```
if [$counter -lt 10]
then number=0$counter
else number=$counter
fi
```

Make a directory if it doesn't exist:

```
if [! -d $dir]; then
 mkdir $dir
 chmod 775 $dir
fi
```

---

## jobs [options] [*jobIDs*]

List all running or stopped jobs, or list those specified by *jobIDs*. For example, you can check whether a long compilation or text format is still running. Also useful before logging out. See the section “Job Control” on page 617.

### Options

- l List job IDs and process group IDs.
- n List only jobs whose status changed since last notification.
- p List process group IDs only.
- r List running jobs only.
- s List stopped jobs only.
- x *cmd*
  - Replace each job ID found in *cmd* with the associated process ID and then execute *cmd*.

---

|             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|-------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>kill</b> | <code>kill [options] IDs</code><br>Terminate each specified process <i>ID</i> or job <i>ID</i> . You must own the process or be a privileged user. This built-in command is similar to <code>/usr/bin/kill</code> , described in Chapter 3. See the section “Job Control” on page 617.                                                                                                                                                                                                                  |
|             | <b>Options</b><br><b>-l</b> List the signal names and numbers. (Used by itself.)<br><b>-n num</b> Send the given signal number.<br><b>-s name</b> Send the given signal name.<br><b>-signal</b> The signal number or name (from <code>kill -l</code> or <code>/usr/include/sys/signal.h</code> ). With a signal number of 9, the kill is absolute.                                                                                                                                                      |
| <b>let</b>  | <code>let expressions</code><br>or<br><code>((expressions))</code><br>Perform arithmetic as specified by one or more <i>expressions</i> . <i>expressions</i> consist of numbers, operators, and shell variables (which don’t need a preceding \$). Expressions must be quoted if they contain spaces or other special characters. The <code>(( ))</code> form does the quoting for you. For more information and examples, see “Arithmetic Expressions” on page 613. See also <b>expr</b> in Chapter 3. |
|             | <b>Examples</b><br>Each of these examples adds 1 to variable i:<br><code>i=`expr \$i + 1`</code><br><code>let i=i+1</code><br><code>let "i = i + 1"</code><br><code>(( i = i + 1 ))</code><br><code>(( i += 1 ))</code><br><code>(( i++ ))</code>                                                                                                                                                                                                                                                       |

---

|               |                                                                                                                                                                                                                                                                     |
|---------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>local</b>  | <code>local [option] [name[=value]]</code><br>Declares local variables for use inside functions. The <i>option</i> can be any option accepted by <b>declare</b> ; see <b>declare</b> for the full list. It is an error to use <b>local</b> outside a function body. |
| <b>logout</b> | <code>logout</code><br>Exit a login shell. The command fails if the current shell is not a login shell.                                                                                                                                                             |

---

|                           |                                                                                                                                                                                                                               |
|---------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>mapfile</b>            | <code>mapfile [options] [array]</code><br><code>readarray [options] [array]</code>                                                                                                                                            |
|                           | Populate an array by reading lines from standard input and placing them into the specified array. The default array is MAPFILE. Without the <b>-O</b> option, <b>mapfile</b> clears the array before assigning entries to it. |
|                           | <b>Options</b>                                                                                                                                                                                                                |
| <b>-c</b> <i>count</i>    | Used with <b>-C</b> to specify the number of lines read between <i>callback</i> calls.                                                                                                                                        |
| <b>-C</b> <i>callback</i> | Evaluate the <i>callback</i> code every time <i>count</i> lines are read, where <i>count</i> is specified by <b>-c</b> (default is 5000).                                                                                     |
| <b>-n</b> <i>num</i>      | Read in at most <i>num</i> lines, or all lines if <i>num</i> is 0.                                                                                                                                                            |
| <b>-O</b> <i>origin</i>   | Begin assigning entries to the array at index <i>origin</i> (default index is 0).                                                                                                                                             |
| <b>-s</b> <i>count</i>    | Discard the first <i>count</i> lines read.                                                                                                                                                                                    |
| <b>-t</b>                 | Remove a trailing line from each input line.                                                                                                                                                                                  |
| <b>-u</b> <i>fd</i>       | Read from the given file descriptor instead of standard input.                                                                                                                                                                |

---

|               |                                                                                                                                                                                                                              |
|---------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>name()</b> | <code>[function] name () {commands; }</code>                                                                                                                                                                                 |
|               | Define <i>name</i> as a function. POSIX syntax. The reserved word <b>function</b> is optional. The function definition can be written on one line or across many. See “Functions” on page 605 for more detailed information. |

**Example**

```
$ count () {
> ls | wc -l
> }
```

When issued at the command line, **count** displays the number of files in the current directory.

---

|             |                                                                                                                                                          |
|-------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>popd</b> | <code>popd [-n] [+count] [-count]</code>                                                                                                                 |
|             | Pop the top directory off the directory stack (as shown by the <b>dirs</b> command), and change to the new top directory, or manage the directory stack. |

**Options**

- n** Don’t change to the new top directory; just manipulate the stack.

*+count*

Remove the item *count* entries from the left, as shown by **dirs**. Counting starts at zero. No directory change occurs.

*-count*

Remove the item *count* entries from the right, as shown by **dirs**. Counting starts at zero. No directory change occurs.

---

## printf

**printf** [**-v** *var*] *format* [*val* ...]

Format the specified values according to the format *format* and write them to standard output. The possible format character strings are those of the ANSI C **printf** function plus several additional strings.

### Option

**-v** *var*

Write output to the variable *var* instead of standard output.

### Additional format strings

**%b** Expand escape sequences in strings (e.g., \t to tab, and so on).

**%q** Print a quoted string that can be reread later on.

### Example

```
$ date Reformat date/time
Fri May 15 15:39:42 EDT 2009
$ printf "%(It is now %m/%d/%Y %H:%M:%S)T\n" "$(date)"
It is now 05/15/2009 15:40:10
```

---

## pushd

**pushd** [-n] [*directory*]

**pushd** [-n] [+*count*] [-*count*]

Add *directory* to the directory stack, or rotate the directory stack. With no arguments, swap the top two entries on the stack, and change to the new top entry.

### Options

**-n** Don't change to the new top directory, just manipulate the stack.

*+count*

Rotate the stack so that the *count*'th item from the left, as shown by **dirs**, is the new top of the stack. Counting starts at zero. The new top becomes the current directory.

*-count*

Rotate the stack so that the *count*'th item from the right, as shown by **dirs**, is the new top of the stack. Counting starts at zero. The new top becomes the current directory.

---

## pwd

**pwd** [-LP]

Print your present working directory on standard output.

## Options

Options give control over the use of logical versus physical treatment of the printed path. See also the entry for `cd`, earlier in this section.

- L Use logical path (what the user typed, including any symbolic links) and the value of PWD for the current directory. This is the default.
- P Use the actual filesystem physical path for the current directory with no symbolic links.

## read

`read [options] [variable1] [variable2 ...]`

Read one line of standard input and assign each word to the corresponding *variable*, with all leftover words assigned to the last variable. If only one variable is specified, the entire line is assigned to that variable. See the examples here and under `case`. The return status is 0 unless EOF is reached. If no variables are given, input is stored in the REPLY variable.

## Options

-a *array*

Read into indexed array *array*.

-d *delim*

Read up to first occurrence of *delim*, instead of newline.

-e Use the *readline* library if reading from a terminal.

-i *text*

If *readline* is being used, put *text* into the editing buffer.

-n *count*

Read at most *count* bytes.

-p *prompt*

Print *prompt* before reading input.

-r Raw mode; ignore \ as a line-continuation character.

-s Read silently; do not echo characters.

-t *timeout*

When reading from a terminal or pipe, if no data is entered after *timeout* seconds, return 1. This prevents an application from hanging forever, waiting for user input.

-u [*n*]

Read input from file descriptor *n* (default is 0).

## Example

Read three variables:

```
$ read first last address
Sarah Caldwell 123 Main Street
```

```
$ echo "$last, $first\n$address"
Caldwell, Sarah
123 Main Street
```

---

**readonly** `readonly [-afp] [variable[=value] ...]`  
Prevent the specified shell variables from being assigned new values. An initial value may be supplied using the assignment syntax, but that value may not be changed subsequently. If no variables are specified, **readonly** displays a list of variables marked read-only.

### Options

- a Each *variable* must refer to an indexed array.
- A Each *variable* must refer to an associative array.
- f Each *variable* must refer to a function.
- p Display the output in a format that allows the list of read-only variables to be saved for rereading later.

---

**return** `return [n]`  
Use inside a function definition. Exit the function with status *n* or with the exit status of the previously executed command.

---

**select** `select x [in list]  
do  
 commands  
done`  
Display a list of menu items on standard error, numbered in the order they are specified in *list*. If no **in** *list* is given, items are taken from the command line (via "\$@"). Following the menu is a prompt string (set by the variable PS3). At the prompt, the user selects a menu item by typing its line number, or redisplays the menu by pressing the Enter key. User input is stored in the shell variable REPLY and the value selected is stored in *x*. If a valid item number is typed, the *commands* associated with the value in *x* are executed and the prompt is redisplayed for the user to select a new value. Typing EOF terminates the loop.

### Example

```
PS3="Select the item number: "
select event in Format Page View Exit
do
 case "$event" in
 Format) nroff $file | lp;;
 Page) pr $file | lp;;
 View) more $file;;
 Exit) exit 0;;
 *) echo "Invalid selection";;
 esac
done
```

The output of this script looks like this:

1. Format
2. Page
3. View
4. Exit

Select the item number:

**set** [options arg1 arg2 ...]

With no arguments, **set** prints the values of all variables known to the current shell. Options can be enabled (*-option*) or disabled (*+option*). Options can also be set when the shell is invoked. (See the section “Invoking the Shell” on page 597.) Arguments are assigned in order to \$1, \$2, etc.

### Options

- a** From now on, automatically mark variables for export after defining or changing them.
- b** Print job completion messages as soon as jobs terminate; don’t wait until the next prompt.
- B** Enable brace expansion. On by default.
- C** Prevent overwriting via > redirection; use >| to overwrite files.
- e** Exit if a command yields a nonzero exit status. The **ERR** trap executes before the shell exits.
- E** Cause shell functions, command substitutions, and subshells to inherit the **ERR** trap.
- f** Ignore filename metacharacters (e.g., \* ? [ ]).
- h** Locate and remember commands as they are defined. On by default.
- H** Enable **csh**-style (!-style) history substitution. On by default.
- k** Assignment of environment variables (*var=value*) takes effect regardless of where they appear on the command line. Normally, assignments must precede the command name.
- m** Enable job control; background jobs execute in a separate process group. **-m** is usually on by default.
- n** Read commands but don’t execute; useful for checking syntax. Ignored if the shell is interactive.
- +o [mode]**  
With *mode*, disable the given shell option. Plain **set +o** prints the settings of all the current options in a form that can be reread by the shell later.
- o [mode]**  
List shell modes, or turn on mode *mode*. Many modes can be set by other options. Modes are:
  - allexport**

Same as **-a**.

**braceexpand**  
Same as **-B**.

**emacs**  
Set command-line editor to **emacs**.

**errexit**  
Same as **-e**.

**errtrace**  
Same as **-E**.

**functrace**  
Same as **-T**.

**hashall**  
Same as **-h**.

**histexpand**  
Same as **-H**.

**history**  
Enable command history. On by default.

**ignoreeof**  
Don't process *EOF* signals. To exit the shell, type **exit**.

**keyword**  
Same as **-k**.

**monitor**  
Same as **-m**.

**noclobber**  
Same as **-C**.

**noexec**  
Same as **-n**.

**noglob**  
Same as **-f**.

**notify**  
Same as **-b**.

**nounset**  
Same as **-u**.

**onecmd**  
Same as **-t**.

**physical**  
Same as **-P**.

**pipefail**  
Change pipeline exit status to be that of the rightmost command that failed, or zero if all exited successfully.

**posix**  
Change to POSIX mode.

**privileged**  
Same as **-p**.

**verbose**  
Same as **-v**.

- vi** Set command-line editor to vi.
- xtrace**
  - Same as -x.
- +p** Reset effective UID to real UID.
- p** Start up as a privileged user. Don't read \$ENV or \$BASH\_ENV, don't import functions from the environment, and ignore the value of \$SHELLOPTS.
- P** Always use physical paths for cd and pwd; do not follow symbolic links.
- t** Exit after one command is executed.
- T** Cause shell functions, command substitutions, and subshells to inherit any DEBUG and RETURN traps.
- u** In substitutions, treat unset variables as errors.
- v** Show each shell command line when read.
- x** Show commands and arguments when executed, preceded by the value of PS4. This provides step-by-step tracing of shell scripts.
- Turn off -v and -x, and turn off option processing. Included for compatibility with older versions of the Bourne shell.
- Used as the last option; -- turns off option processing so that arguments beginning with - are not misinterpreted as options. (For example, you can set \$1 to -1.) If no arguments are given after --, unset the positional parameters.

### Examples

|                        |                                                                                |
|------------------------|--------------------------------------------------------------------------------|
| set -- "\$num" -20 -30 | Set \$1 to \$num, \$2 to -20, \$3 to -30                                       |
| set -vx                | Read each command line; show it;<br>execute it; show it again (with arguments) |
| set +x                 | Stop command tracing                                                           |
| set -o noclobber       | Prevent file overwriting                                                       |
| set +o noclobber       | Allow file overwriting again                                                   |

## shift

shift [*n*]

Shift positional arguments (e.g., \$2 becomes \$1). If *n* is given, shift to the left *n* places; otherwise *n* is assumed to be 1. Used in while loops to iterate through command-line arguments.

## shopt

shopt [-opqsu] [*options*]

Set or unset shell options. With no options or just -p, print the option names and whether they are set or not.

### Options

- o** Each *option* must be one of the shell option names for set -o, instead of the options listed in the next section.
- p** Print the option settings as shopt commands that can be reread later.

- q Quiet mode. The exit status is zero if the given option is set, nonzero otherwise. With multiple options, all of them must be set for a zero exit status.
- s Set the given *options*. With no *options*, print only those that are set.
- u Unset the given *options*. With no *options*, print only those that are unset.

### Settable shell options

The following list describes the behavior when set. Options marked with an asterisk (\*) are enabled by default.

#### **autocd**

Attempt to **cd** to a directory that is given as a command name.  
Allowed in interactive shells only.

#### **cdable\_vars**

Treat a nondirectory argument to **cd** as a variable whose value is the directory to go to.

#### **cdspell**

Attempt spelling correction on each directory component of an argument to **cd**. Allowed in interactive shells only.

#### **checkhash**

Check that commands found in the hash table still exist before attempting to use them. If not, perform a normal PATH search.

#### **checkjobs**

Display the status of any running or stopped jobs before exiting an interactive shell.

#### **checkwinsize**

Check the window size after each command and update LINES and COLUMNS if the size has changed.

#### **cmdhist \***

Save all lines of a multiline command in one history entry.  
This permits easy reediting of multiline commands.

#### **compat31**

Behave like version 3.1 with respect to quoted arguments to the conditional command's `=~` operator.

#### **dirspell**

During filename completion, attempt to correct the spelling of directory names if the name as given is not found.

#### **dotglob**

Include filenames starting with a period in the results of filename expansion.

#### **execfail**

Do not exit a noninteractive shell if the command given to **exec** cannot be executed. Interactive shells do not exit in such a case, no matter the setting of this option.

|                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|-------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>expand_aliases</b> * | Expand aliases created with <b>alias</b> . Disabled in noninteractive shells.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| <b>extdebug</b>         | Enable behavior needed for debuggers:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|                         | <ul style="list-style-type: none"> <li>• <b>declare -F</b> displays the source filename and line number for each function name argument.</li> <li>• When a command run by the <b>DEBUG</b> trap fails, the next command is skipped.</li> <li>• When a command run by the <b>DEBUG</b> trap inside a shell function or script sourced with . (dot) or <b>source</b> returns with an exit status of 2, the shell simulates a call to <b>return</b>.</li> <li>• <b>BASH_ARGC</b> and <b>BASH_ARGV</b> are set as described earlier.</li> <li>• Function tracing is enabled. Command substitutions, shell functions, and subshells invoked via (...) inherit the <b>DEBUG</b> and <b>RETURN</b> traps.</li> <li>• Error tracing is enabled. Command substitutions, shell functions, and subshells invoked via (...) inherit the <b>ERROR</b> trap.</li> </ul> |
| <b>extglob</b>          | Enable extended pattern-matching facilities such as +(...).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| <b>extquote</b> *       | Allow \$'...' and \$"..." within \${variable} expansions inside double quotes.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| <b>failglob</b>         | Cause patterns that do not match filenames to produce an error.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| <b>force_fignore</b> *  | When doing completion, ignore words matching the list of suffixes in <b>IGNORE</b> , even if such words are the only possible completions.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| <b>globstar</b>         | During filename expansion, the pattern ** matches all directories and subdirectories, and filenames in directories, recursively. Only directories and subdirectories match if the pattern ends in /.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| <b>gnu_errfmt</b>       | Print error messages in the standard GNU format.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| <b>histappend</b>       | Append the history list to the file named by <b>HISTFILE</b> upon exit, instead of overwriting the file.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| <b>histreedit</b>       | Allow a user to reedit a failed history substitution with the <i>readline</i> library.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |

**histverify**

Place the results of history substitution into the *readline* library's editing buffer, in case the user wishes to modify it further, instead of executing it directly.

**hostcomplete \***

If using *readline*, attempt hostname completion when a word containing an @ is being completed.

**huponexit**

Send a SIGHUP to all running jobs upon exiting an interactive shell.

**interactive\_comments \***

Allow words beginning with # to start a comment in an interactive shell.

**lithist**

If **cmdhist** is also set, save multiline commands to the history file with newlines instead of semicolons.

**login\_shell**

Set by the shell when it is a login shell. This is a read-only option.

**mailwarn**

Print the message "The mail in *mailfile* has been read" when a file being checked for mail has been accessed since the last time Bash checked it.

**no\_empty\_cmd\_completion**

If using *readline*, do not search \$PATH when a completion is attempted on an empty line.

**nocaseglob**

Ignore letter case when doing filename matching.

**nocasematch**

Ignore case when pattern-matching in **case** or [[ ]] conditional commands.

**nullglob**

Expand patterns that do not match any files to the null string, instead of using the literal pattern as an argument.

**progcomp \***

Enable programmable completion.

**promptvars \***

Perform variable, command, and arithmetic substitution on the values of PS1, PS2 and PS4.

**restricted\_shell**

Set by the shell when it is a restricted shell. This is a read-only option.

**shift\_verbose**

Causes **shift** to print an error message when the shift count is greater than the number of positional parameters.

- sourcepath \***  
 Causes the . (dot) and **source** commands to search \$PATH in order to find the file to read and execute.
- xpg\_echo**  
 Causes **echo** to expand escape sequences, even without the -e or -E options.

**source**      *source file [arguments]*  
 Identical to the . (dot) command; see that entry.

**suspend**      *suspend [-f]*  
 Suspend the current shell. Often used to stop an **su** command.

**Option**  
**-f**    Force the suspension, even if the shell is a login shell.

**test**      *test condition*  
 or  
*[ condition ]*  
 or  
*[[ condition ]]*  
 Evaluate a *condition* and, if its value is true, return a zero exit status; otherwise, return a nonzero exit status. An alternate form of the command uses [ ] rather than the word **test**. An additional alternate form uses [[ ]], in which case word splitting and pathname expansion are not done. (See the [[ ]] entry.) *condition* is constructed using the following expressions. Conditions are true if the description holds true.

#### File conditions

- a *file*, -e *file*  
*file* exists.
- b *file*  
*file* exists and is a block special file.
- c *file*  
*file* exists and is a character special file.
- d *file*  
*file* exists and is a directory.
- f *file*  
*file* exists and is a regular file.
- g *file*  
*file* exists, and its set-group-id bit is set.
- G *file*  
*file* exists, and its group is the effective group ID.
- h *file*, -L *file*  
*file* exists and is a symbolic link.

**-k** *file*  
    *file* exists, and its sticky bit is set.

**-N** *file*  
    *file* exists and was modified after it was last read.

**-O** *file*  
    *file* exists, and its owner is the effective user ID.

**-p** *file*  
    *file* exists and is a named pipe (FIFO).

**-r** *file*  
    *file* exists and is readable.

**-s** *file*  
    *file* exists and has a size greater than zero.

**-S** *file*  
    *file* exists and is a socket.

**-t** [*n*]  
    The open file descriptor *n* is associated with a terminal device;  
    default *n* is 1.

**-u** *file*  
    *file* exists, and its set-user-id bit is set.

**-w** *file*  
    *file* exists and is writable.

**-x** *file*  
    *file* exists and is executable.

***f1 -ef f2***  
    Files *f1* and *f2* are linked (refer to same file).

***f1 -nt f2***  
    File *f1* is newer than *f2*.

***f1 -ot f2***  
    File *f1* is older than *f2*.

### String conditions

***string***  
    *string* is not null.

**-n** *s1*  
    String *s1* has nonzero length.

**-z** *s1*  
    String *s1* has zero length.

***s1 == s2***  
    Strings *s1* and *s2* are identical. *s2* can be a wildcard pattern.  
    Quote *s2* to treat it literally. Preferred over =.

***s1 != s2***  
    Strings *s1* and *s2* are *not* identical. *s2* can be a wildcard  
    pattern. Quote *s2* to treat it literally.

*s1*  $\sim$  *s2*

String *s1* matches extended regular expression *s2*. Quote *s2* to keep the shell from expanding embedded shell metacharacters. Strings matched by parenthesized subexpressions are placed into elements of the BASH\_REMATCH array. See the description of BASH\_REMATCH earlier in this chapter.

*s1* < *s2*

ASCII value of *s1* precedes that of *s2*. (Valid only within [[ ]]) construct.)

*s1* > *s2*

ASCII value of *s1* follows that of *s2*. (Valid only within [[ ]]) construct.)

### Internal shell conditions

-o *opt*

Shell option *opt* for set -o is on.

### Integer comparisons

*n1* -eq *n2*

*n1* equals *n2*.

*n1* -ge *n2*

*n1* is greater than or equal to *n2*.

*n1* -gt *n2*

*n1* is greater than *n2*.

*n1* -le *n2*

*n1* is less than or equal to *n2*.

*n1* -lt *n2*

*n1* is less than *n2*.

*n1* -ne *n2*

*n1* does not equal *n2*.

### Combined forms

(*condition*)

True if *condition* is true (used for grouping). For test and [ ], the ()s should be quoted by a \. The form using [[ ]] doesn't require quoting the parentheses.

! *condition*

True if *condition* is false.

*condition1* -a *condition2*

True if both conditions are true.

*condition1* && *condition2*

True if both conditions are true. (Valid only within [[ ]]) construct.)

*condition1* -o *condition2*

True if either condition is true.

*condition1* || *condition2*

True if either condition is true. (Valid only within [[ ]]) construct.)

## Examples

The following examples show the first line of various statements that might use a test condition:

|                                 |                                                                   |
|---------------------------------|-------------------------------------------------------------------|
| while test \$# -gt 0            | While there are arguments...                                      |
| while [ -n "\$1" ]              | While there are nonempty arguments..                              |
| if [ \$count -lt 10 ]           | If \$count is less than 10...                                     |
| if [ -d RCS ]                   | If the RCS directory exists...                                    |
| if [ "\$answer" != "y" ]        | If the answer is not y...                                         |
| if [ ! -r "\$1" -o ! -f "\$1" ] | If the first argument is not a readable file or a regular file... |

---

## time

`time command`

Execute *command* and print the total elapsed time, user time, and system time (in seconds). Same as the Linux command **time** (see Chapter 3), except that the built-in version can also time other built-in commands as well as all commands in a pipeline.

---

## times

`times`

Print accumulated process times for user and system.

---

## trap

`trap [options] [[commands] signals]`

Execute *commands* if any *signals* are received. Common signals include **EXIT** (0), **HUP** (1), **INT** (2), and **TERM** (15). Multiple commands must be quoted as a group and separated by semicolons internally. If the command is the null string (i.e., `trap "" signals`), the signals are ignored by the shell. If the commands are omitted entirely, processing of the specified signals is reset to the default action. If the command is `-`, the signals are reset to their initial defaults.

If both *commands* and *signals* are omitted, list current trap assignments. See the Examples here and in **exec**.

### Options

- l List all signals and their numbers, like `kill -l`.
- p Print the current trap settings in a form suitable for rereading later.

### Signals

A list of signal names, numbers, and meanings were given earlier in the **kill** entry; see the listing there. The shell allows you to use either the signal number or the signal name (without the **SIG** prefix). In addition, the shell supports “pseudo-signals,” signal names or numbers that aren’t real operating system signals but which direct the shell to perform a specific action. These signals are:

#### DEBUG

Execution of any command.

#### ERR

Nonzero exit status.

**EXIT**

Exit from shell (usually when shell script finishes).

- 0 Same as **EXIT**, for historical compatibility with the Bourne shell.

**RETURN**

A **return** is executed, or a script run with . (dot) or **source** finishes.

**Examples**

|                    |                                     |
|--------------------|-------------------------------------|
| <b>trap "" INT</b> | <i>Ignore interrupts (signal 2)</i> |
| <b>trap INT</b>    | <i>Obey interrupts again</i>        |

Remove a \$tmp file when the shell program exits, or if the user logs out, presses Ctrl-C, or does a **kill**:

|                                                   |                                     |
|---------------------------------------------------|-------------------------------------|
| <b>trap "rm -f \$tmp; exit" EXIT HUP INT TERM</b> | <i>POSIX style</i>                  |
| <b>trap "rm -f \$tmp; exit" 0 1 2 15</b>          | <i>Pre-POSIX Bourne shell style</i> |

Print a “clean up” message when the shell program receives signals SIGHUP, SIGINT, or SIGTERM:

```
trap 'echo Interrupt! Cleaning up...' HUP INT TERM
```

**true**

true

Built-in command that exits with a true return value.

**type**

```
type [-afP]t commands
```

Show whether each command name is a Linux command, a built-in command, an alias, a shell keyword, or a defined shell function.

**Options**

- a Print all locations in \$PATH that include *command*, including aliases and functions. Use -p together with -a to suppress aliases and functions.
- f Suppress function lookup, as with **command**.
- p If **type -t** would print *file* for a given *command*, this option prints the full pathname for the executable files. Otherwise, it prints nothing.
- P Like -p, but force a PATH search, even if **type -t** would not print *file*.
- t Print a word describing each *command*. The word is one of **alias**, **builtin**, **file**, **function**, or **keyword**, depending upon the type of each *command*.

**Example**

```
$ type mv read if
mv is /bin/mv
read is a shell builtin
if is a shell keyword
```

---

**ulimit**      `ulimit [options] [n]`  
Print the value of one or more resource limits, or, if *n* is specified, set a resource limit to *n*. Resource limits can be either hard (-H) or soft (-S). By default, **ulimit** sets both limits or prints the soft limit. The options determine which resource is acted on.

### Options

- H Hard limit. Anyone can lower a hard limit; only privileged users can raise it.
- S Soft limit. Anyone can raise a soft limit up to the value of the hard limit.
- a Print all limits.
- b Maximum socket buffer size.
- c Maximum size of core files.
- d Maximum kilobytes of data segment or heap.
- e Maximum scheduling priority (“nice”).
- f Maximum size of files (the default option).
- i Maximum number of pending signals.
- l Maximum size of address space that can be locked in memory.
- m Maximum kilobytes of physical memory. (Not effective on all Linux systems.)
- n Maximum number of file descriptors.
- p Maximum size of pipe buffers in 512-byte blocks. (May not be set.)
- q Maximum number of bytes in POSIX message queues.
- r Maximum real-time scheduling priority.
- s Maximum stack size.
- t Maximum CPU seconds.
- T Maximum number of threads.
- u Maximum number of processes a single user can have.
- v Maximum kilobytes of virtual memory.
- x Maximum number of file locks

---

**umask**      `umask [-pS] [mask]`  
Display or set file creation mask. If *mask* begins with a digit, it is treated as an octal number; otherwise it is treated as a symbolic mask. The file creation mask determines which permission bits are turned off (e.g., **umask 002** produces **rw-rw-r--**). The mask is similar to that accepted by the **chmod** command.

### Options

- p Output is in a form that can be reread later by the shell.
- S Print the current mask using symbolic notation.

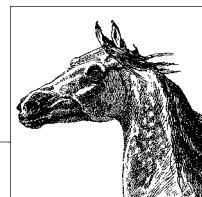
---

|                |                                                                                                                                                                                                                                                                                                                        |
|----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>unalias</b> | <code>unalias names</code><br><code>unalias -a</code>                                                                                                                                                                                                                                                                  |
|                | Remove <i>names</i> from the alias list. See also <b>alias</b> .                                                                                                                                                                                                                                                       |
|                | <b>Option</b>                                                                                                                                                                                                                                                                                                          |
|                | -a Remove all aliases.                                                                                                                                                                                                                                                                                                 |
| <b>unset</b>   | <code>unset [options] names</code>                                                                                                                                                                                                                                                                                     |
|                | Erase definitions of functions or variables listed in <i>names</i> .                                                                                                                                                                                                                                                   |
|                | <b>Options</b>                                                                                                                                                                                                                                                                                                         |
|                | -f Unset functions <i>names</i> .<br>-v Unset shell variables <i>names</i> (default).                                                                                                                                                                                                                                  |
| <b>until</b>   | <code>until condition</code><br>do<br><i>commands</i><br>done                                                                                                                                                                                                                                                          |
|                | Until <i>condition</i> is met, do <i>commands</i> . <i>condition</i> is often specified with the <b>test</b> command. See the Examples under <b>case</b> and <b>test</b> .                                                                                                                                             |
| <b>wait</b>    | <code>wait [ID]</code>                                                                                                                                                                                                                                                                                                 |
|                | Pause in execution until all background jobs complete (exit status 0 is returned), or pause until the specified background process <i>ID</i> or job <i>ID</i> completes (exit status of <i>ID</i> is returned). Note that the shell variable <b>\$!</b> contains the process ID of the most recent background process. |
|                | <b>Example</b>                                                                                                                                                                                                                                                                                                         |
|                | <code>wait \$!</code> <i>Wait for most recent background process to finish</i>                                                                                                                                                                                                                                         |
| <b>while</b>   | <code>while condition</code><br>do<br><i>commands</i><br>done                                                                                                                                                                                                                                                          |
|                | While <i>condition</i> is met, do <i>commands</i> . <i>condition</i> is often specified with the <b>test</b> command. See the Examples under <b>case</b> and <b>test</b> .                                                                                                                                             |

---

# 7

## Pattern Matching



A number of Linux text-processing utilities let you search for, and in some cases change, text patterns rather than fixed strings. These utilities include the editing programs `ed`, `ex`, `vi`, and `sed`; the `gawk` programming language; and the commands `grep` and `egrep`. Text patterns (called *regular expressions* in computer science literature) contain normal characters mixed with special characters (called *metacharacters*).

Perl's regular expression support is so rich that it does not fit into this book; you can find a description in the O'Reilly books *Mastering Regular Expressions* by Jeffrey E.F. Friedl, *Regular Expression Pocket Reference* by Tony Stubblebine, *Perl in a Nutshell* by Nathan Patwardhan et al., or *Perl 5 Pocket Reference* by Johan Vromans. The Emacs editor also provides regular expressions similar to those shown in this chapter. See the O'Reilly books *Learning GNU Emacs* by Debra Cameron et al., or *GNU Emacs Pocket Reference*, also by Debra Cameron, for details.

This chapter presents the following topics:

- Filenames versus patterns
- Description of metacharacters
- List of metacharacters available to each program
- Examples

For more information on regular expressions, see the aforementioned O'Reilly book *Mastering Regular Expressions*.

### Filenames Versus Patterns

Metacharacters used in pattern matching are different from metacharacters used for filename expansion (see Chapter 6). However, several metacharacters have meaning for both regular expressions and for filename expansion. This can lead to

a problem: the shell sees the command line first, and can potentially interpret an unquoted regular expression metacharacter as a filename expansion. For example, the command:

```
$ grep [A-Z]* chap[12]
```

could be transformed by the shell into:

```
$ grep Array.c Bug.c Comp.c chap1 chap2
```

and **grep** would then try to find the pattern *Array.c* in files *Bug.c*, *Comp.c*, *chap1*, and *chap2*. To bypass the shell and pass the special characters to **grep**, use quotes as follows:

```
$ grep "[A-Z]*" chap[12]
```

Double quotes suffice in most cases, but single quotes are the safest bet, since the shell does absolutely no expansions on single-quoted text.

Note also that in pattern matching, ? matches zero or one instance of a regular expression; in filename expansion, ? matches a single character.

## Metacharacters

Different metacharacters have different meanings, depending upon where they are used. In particular, regular expressions used for searching through text (matching) have one set of metacharacters, while the metacharacters used when processing replacement text (such as in a text editor) have a different set. These sets also vary somewhat per program. This section covers the metacharacters used for searching and replacing, with descriptions of the variants in the different utilities.

## Search Patterns

The characters in the following table have special meaning only in search patterns.

| Character | Pattern                                                                                                                                                                                                                                                                                                                                                                                                             |
|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| .         | Match any <i>single</i> character except newline. Can match newline in <b>gawk</b> .                                                                                                                                                                                                                                                                                                                                |
| *         | Match any number (or none) of the single character that immediately precedes it. The preceding character can also be a regular expression. For example, since . (dot) means any character, .* means “match any number of any character.”                                                                                                                                                                            |
| ^         | Match the following regular expression at the beginning of the line or string.                                                                                                                                                                                                                                                                                                                                      |
| \$        | Match the preceding regular expression at the end of the line or string.                                                                                                                                                                                                                                                                                                                                            |
| []        | Match any <i>one</i> of the enclosed characters. A hyphen (-) indicates a range of consecutive characters. A circumflex (^) as the first character in the brackets reverses the sense: it matches any one character <i>not</i> in the list. A hyphen or close bracket (]) as the first character is treated as a member of the list. All other metacharacters are treated as members of the list (i.e., literally). |
| {n,m}     | Match a range of occurrences of the single character that immediately precedes it. The preceding character can also be a regular expression. {n} matches exactly n occurrences, {n,} matches at least n occurrences, and {n,m} matches any number of occurrences between n and m. n and m must be between 0 and 255, inclusive. (The GNU programs on Linux allow a range of 0 to 32767.)                            |
| \{n,m\}   | Just like {n,m}, earlier, but with backslashes in front of the braces.                                                                                                                                                                                                                                                                                                                                              |
| \         | Turn off the special meaning of the following character.                                                                                                                                                                                                                                                                                                                                                            |

| Character | Pattern                                                                                                                                                                                                                                         |
|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| \(\)      | Save the subpattern enclosed between \(` and \)` into a special holding space. Up to nine subpatterns can be saved on a single line. The text matched by the subpatterns can be “replayed” in substitutions by the escape sequences \`1 to \`9. |
| \n        | Replay the <i>n</i> th subpattern enclosed in \(` and \)` into the pattern at this point. <i>n</i> is a number from 1 to 9, with 1 starting on the left. See the following Examples.                                                            |
| \<\>      | Match characters at beginning (\<) or end (\>) of a word.                                                                                                                                                                                       |
| +         | Match one or more instances of preceding regular expression.                                                                                                                                                                                    |
| ?         | Match zero or one instances of preceding regular expression.                                                                                                                                                                                    |
|           | Match one or the other of the regular expressions specified before and after the vertical bar. (This is known as alternation.)                                                                                                                  |
| ( )       | Apply a match to the enclosed group of regular expressions.                                                                                                                                                                                     |

Linux allows the use of POSIX “character classes” within the square brackets that enclose a group of characters. They are typed enclosed in [: and :]. For example, [:alnum:] matches a single alphanumeric character.

| Class | Characters matched      | Class  | Characters matched     |
|-------|-------------------------|--------|------------------------|
| alnum | Alphanumeric characters | lower  | Lowercase characters   |
| alpha | Alphabetic characters   | print  | Printable characters   |
| blank | Space or Tab            | punct  | Punctuation characters |
| cntrl | Control characters      | space  | Whitespace characters  |
| digit | Decimal digits          | upper  | Uppercase characters   |
| graph | Non-space characters    | xdigit | Hexadecimal digits     |

Finally, the GNU utilities on Linux accept additional escape sequences that act like metacharacters. (Because \b can also be interpreted as the sequence for the ASCII Backspace character, different utilities treat it differently. Check each utility’s documentation.)

| Sequence | Meaning                                                                                                     |
|----------|-------------------------------------------------------------------------------------------------------------|
| \b       | Word boundary, either beginning or end of a word, as for the \< and \> metacharacters described earlier.    |
| \B       | Interword match; matches between two word-constituent characters.                                           |
| \w       | Matches any word-constituent character; equivalent to [:alnum:]._.                                          |
| \W       | Matches any non-word-constituent character; equivalent to [^[:alnum:]].                                     |
| \`       | Beginning of an Emacs buffer. Used by most other GNU utilities to mean unambiguously “beginning of string.” |
| \`       | End of an Emacs buffer. Used by most other GNU utilities to mean unambiguously “end of string.”             |

## Replacement Patterns

The characters in the following table have special meaning only in replacement patterns, used for example in editing, when searching for and replacing text.

| Character | Pattern                                                                                                                                                                                |
|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| \         | Turn off the special meaning of the following character.                                                                                                                               |
| \n        | Reuse the text matched by the <i>n</i> th subpattern previously saved by \(` and `) as part of the replacement pattern. <i>n</i> is a number from 1 to 9, with 1 starting on the left. |
| &         | Reuse the text matched by the search pattern as part of the replacement pattern.                                                                                                       |
| ~         | Reuse the previous replacement pattern in the current replacement pattern. Must be the only character in the replacement pattern (ex and vi).                                          |
| %         | Reuse the previous replacement pattern in the current replacement pattern. Must be the only character in the replacement pattern (ed).                                                 |
| \u        | Convert the first character of replacement pattern to uppercase.                                                                                                                       |
| \U        | Convert the entire replacement pattern to uppercase.                                                                                                                                   |
| \l        | Convert the first character of replacement pattern to lowercase.                                                                                                                       |
| \L        | Convert the entire replacement pattern to lowercase.                                                                                                                                   |
| \e        | Turn off previous \u or \L.                                                                                                                                                            |
| \E        | Turn off previous \U or \L.                                                                                                                                                            |

## Metacharacters, Listed by Program

Some metacharacters are valid for one program but not for another. Those that are available are marked by a bullet (•) in the following table. Items marked with a “P” are specified by POSIX. Full descriptions were provided in the previous section.

| Symbol | ed | ex | vi | sed | gawk           | grep | egrep          | Action                                       |
|--------|----|----|----|-----|----------------|------|----------------|----------------------------------------------|
| .      | •  | •  | •  | •   | •              | •    | •              | Match any character.                         |
| *      | •  | •  | •  | •   | •              | •    | •              | Match zero or more preceding characters.     |
| ^      | •  | •  | •  | •   | •              | •    | •              | Match beginning of line/string.              |
| \$     | •  | •  | •  | •   | •              | •    | •              | Match end of line/string.                    |
| \      | •  | •  | •  | •   | •              | •    | •              | Escape following character.                  |
| []     | •  | •  | •  | •   | •              | •    | •              | Match one from a set.                        |
| \(\)   | •  | •  | •  | •   | •              | •    | •              | Store pattern for later replay. <sup>a</sup> |
| \n     | •  | •  | •  | •   | •              | •    | •              | Replay subpattern in match.                  |
| {}     |    |    |    |     | • <sup>P</sup> |      | • <sup>P</sup> | Match a range of instances.                  |
| \{\}   | •  |    |    | •   |                | •    |                | Match a range of instances.                  |
| \<\>   | •  | •  | •  |     |                |      |                | Match word's beginning or end.               |
| +      |    |    |    |     | •              |      | •              | Match one or more preceding characters.      |
| ?      |    |    |    |     | •              |      | •              | Match zero or one preceding characters.      |
|        |    |    |    |     | •              |      | •              | Separate choices to match.                   |
| ( )    |    |    |    |     | •              |      | •              | Group expressions to match.                  |

<sup>a</sup> Stored subpatterns can be “replayed” during matching. See the following table.

Note that in **ed**, **ex**, **vi**, and **sed**, you specify both a search pattern (on the left) and a replacement pattern (on the right). The metacharacters listed above are meaningful only in a search pattern.

In **ed**, **ex**, **vi**, and **sed**, the metacharacters in the following table are valid only in a replacement pattern.

| Symbol | ex | vi | sed | ed | Action                                |
|--------|----|----|-----|----|---------------------------------------|
| \      | .  | .  | .   | .  | Escape following character.           |
| \n     | .  | .  | .   | .  | Text matching pattern stored in \(\). |
| &      | .  | .  | .   | .  | Text matching search pattern.         |
| ~      | .  | .  |     |    | Reuse previous replacement pattern.   |
| %      |    |    | .   |    | Reuse previous replacement pattern.   |
| \u\U   | .  | .  |     |    | Change character(s) to uppercase.     |
| \l\L   | .  | .  |     |    | Change character(s) to lowercase.     |
| \e     | .  | .  |     |    | Turn off previous \u or \l.           |
| \E     | .  | .  |     |    | Turn off previous \U or \L.           |

## Examples of Searching

When used with **grep** or **egrep**, regular expressions should be surrounded by quotes. (If the pattern contains a \$, you must use single quotes; e.g., '*pattern*'). When used with **ed**, **ex**, **sed**, and **gawk**, regular expressions are usually surrounded by /, although (except for **gawk**) any delimiter works. The following tables show some sample patterns.

| Pattern       | What does it match?                                                    |
|---------------|------------------------------------------------------------------------|
| bag           | The string <i>bag</i> .                                                |
| ^bag          | <i>bag</i> at the beginning of the line.                               |
| bag\$         | <i>bag</i> at the end of the line.                                     |
| ^bag\$        | <i>bag</i> as the only word on the line.                               |
| [Bb]ag        | <i>Bag</i> or <i>bag</i> .                                             |
| b[aeiou]g     | Second letter is a vowel.                                              |
| b[^aeiou]g    | Second letter is a consonant (or uppercase or symbol).                 |
| b.g           | Second letter is any character.                                        |
| ^.{\$}        | Any line containing exactly three characters.                          |
| ^.{}          | Any line that begins with a dot.                                       |
| ^a.[a-z][a-z] | Same, followed by two lowercase letters (e.g., <b>troff</b> requests). |
| ^a.[a-z]{2\}  | Same as previous; <b>ed</b> , <b>grep</b> , and <b>sed</b> only.       |
| ^[^.]         | Any line that doesn't begin with a dot.                                |
| bugs*         | <i>bug</i> , <i>bugs</i> , <i>bugss</i> , etc.                         |
| "word"        | A word in quotes.                                                      |
| "*word"**     | A word, with or without quotes.                                        |
| [A-Z][A-Z]*   | One or more uppercase letters.                                         |
| [A-Z]+        | Same; <b>egrep</b> or <b>gawk</b> only.                                |

| Pattern                                | What does it match?                                             |
|----------------------------------------|-----------------------------------------------------------------|
| [[:upper:]]+                           | Same as previous, egrep or gawk.                                |
| [A-Z].*                                | An uppercase letter, followed by zero or more characters.       |
| [A-Z]*                                 | Zero or more uppercase letters.                                 |
| [a-zA-Z]                               | Any letter, either lower- or uppercase.                         |
| [^0-9A-Za-z]                           | Any symbol or space (not a letter or a number).                 |
| [^[:alnum:]]                           | Same, using POSIX character class.                              |
| egrep or gawk pattern                  | What does it match?                                             |
| [567]                                  | One of the numbers 5, 6, or 7.                                  |
| five six seven                         | One of the words <i>five</i> , <i>six</i> , or <i>seven</i> .   |
| 80[2-4]?86                             | 8086, 80286, 80386, or 80486.                                   |
| 80[2-4]?86 (Pentium(-III)??)           | 8086, 80286, 80386, 80486, Pentium, Pentium-II, or Pentium-III. |
| compan(y)ies                           | <i>company</i> or <i>companies</i> .                            |
| ex or vi pattern                       | What does it match?                                             |
| \<the                                  | Words like <i>theater</i> or <i>the</i> .                       |
| the\>                                  | Words like <i>breathe</i> or <i>the</i> .                       |
| \<the\>                                | The word <i>the</i> .                                           |
| ed, sed, or grep pattern               | What does it match?                                             |
| 0\{5,\}                                | Five or more zeros in a row.                                    |
| [0-9]\{3\}-[0-9]\{2\}-[0-9]\{4\}       | U.S. Social Security number ( <i>nnn-nn-nnnn</i> ).             |
| \(why\).*1                             | A line with two occurrences of <i>why</i> .                     |
| \(([:alpha:]\_) ([[:alnum:]\_]*)=\)\1; | C/C++ simple assignment statements.                             |

## Examples of Searching and Replacing

The examples in the following table show the metacharacters available to **sed** or **ex**. Note that **ex** commands begin with a colon. A space is marked by a □; a tab is marked by a *tab*.

| Command           | Result                                                         |
|-------------------|----------------------------------------------------------------|
| s/.*/(&)/         | Redo the entire line, but add spaces and parentheses.          |
| s/.*/mv & &.old/  | Change a wordlist (one word per line) into <b>mv</b> commands. |
| /^\\$/d           | Delete blank lines.                                            |
| :g/^\$/d          | Same as previous, in <b>ex</b> editor.                         |
| /^\[\t\]*\$/\$d   | Delete blank lines, plus lines containing only spaces or tabs. |
| :g/\[\t\]*\$/\$d  | Same as previous, in <b>ex</b> editor.                         |
| s/\t\t\t\t/g      | Turn one or more spaces into one space.                        |
| :%s/\t\t\t\t/g    | Same as previous, in <b>ex</b> editor.                         |
| :s/[0-9]/Item &:/ | Turn a number into an item label (on the current line).        |
| :s                | Repeat the substitution on the first occurrence.               |
| :&                | Same as previous.                                              |
| :sg               | Same as previous, but for all occurrences on the line.         |
| :&g               | Same as previous.                                              |
| :%&g              | Repeat the substitution globally (i.e., on all lines).         |

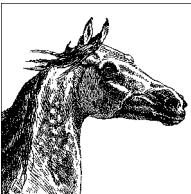
| Command                                  | Result                                                                               |
|------------------------------------------|--------------------------------------------------------------------------------------|
| <code>:,\$s/Fortran/U&amp;/g</code>      | On current line to last line, change word to uppercase.                              |
| <code>:,\$s/(F) (ORTRAN)/\1\L\2/g</code> | On current line to last line, change spelling of "FORTRAN" to correct, modern usage. |
| <code>:%s/*\L&amp;/</code>               | Lowercase entire file.                                                               |
| <code>:s/^.*/u&amp;/g</code>             | Uppercase first letter of each word on current line. (Useful for titles.)            |
| <code>:%s/yes/No/g</code>                | Globally change a word to <i>No</i> .                                                |
| <code>:%s/Yes/~/g</code>                 | Globally change a different word to <i>No</i> (previous replacement).                |

Finally, some **sed** examples for transposing words. A simple transposition of two words might look like this:

`s/die or do/do or die/` *Transpose words*

The real trick is to use hold buffers to transpose variable patterns. For example:

`s/([Dd]ie) or ([Dd]o)/\2 or \1/` *Transpose, using hold buffers*



# 8

## The Emacs Editor

The Emacs editor is found on many Unix systems, including Linux, because it is a popular alternative to **vi**. Many versions are available, but this book documents the most popular one, GNU Emacs (version 22.3), which is available from the Free Software Foundation (<http://www.gnu.org/software/emacs>).

Emacs is much more than “just an editor”—in fact, it provides a fully integrated user environment. From within Emacs, you can issue individual shell commands or open a window where you can work in the shell, read and send mail, read news, access the Internet, write and test programs, and maintain a calendar. To fully describe Emacs would require more space than we have available. In this chapter, therefore, we focus on the editing capabilities of Emacs.

This chapter presents the following topics:

- Conceptual overview
- Command-line syntax
- Summary of **emacs** commands by group
- Summary of **emacs** commands by key
- Summary of **emacs** commands by name

For more information about Emacs, see *Learning GNU Emacs*, by Debra Cameron et al. (O'Reilly).

## Conceptual Overview

This section describes some Emacs terminology that may be unfamiliar if you haven't used Emacs before.

## Modes

One of the features that makes Emacs popular is its editing modes. The modes set up an environment designed for the type of editing you are doing, with features

such as having appropriate key bindings available and automatically indenting according to standard conventions for that type of document. There are two types of modes: major and minor. The major modes include modes for various programming languages such as C or Perl, for text processing (e.g., XML, or even straight text), and many more. One particularly useful major mode is *Dired* (Directory Editor), which has commands that let you manage directories. Minor modes set or unset features that are independent of the major mode, such as auto-fill (which controls word wrapping), insert versus overwrite, and auto-save. For a full discussion of modes, see *Learning GNU Emacs* (O'Reilly) or the Emacs Info documentation system (**C-h i**).

## Buffer and Window

When you open a file in Emacs, the file is put into a *buffer* so you can edit it. If you open another file, that file goes into another buffer. The view of the buffer contents that you have at any point in time is called a *window*. For a small file, the window might show the entire file; for a large file, it shows only a portion of a file. Emacs allows multiple windows to be open at the same time, to display the contents of different buffers or different portions of a single buffer.

## Point and Mark

When you are editing in Emacs, the position of the cursor is known as *point*. You can set a *mark* at another place in the text to operate on the region between point and mark. This is a very useful feature for such operations as deleting or moving an area of text.

## Kill and Yank

Emacs uses the terms *kill* and *yank* for the concepts more commonly known today as *cut* and *paste*. You cut text in Emacs by killing it, and paste it by yanking it back. If you do multiple kills in a row, you can yank them back all at once.

## Notes on the Tables

Emacs commands use the Ctrl key and the Meta key (Meta is usually the Alt key or the Escape key). In this chapter, the notation C- indicates that the Ctrl key is pressed at the same time as the character that follows. Similarly, M- indicates the use of the Meta key. When using Escape for Meta, press and release the Escape key, then type the next key. If you use Alt (or Option on the Mac) for Meta, it is just like Ctrl or Shift, and you should press it simultaneously with the other key(s).

In the command tables that follow, the first column lists the keystroke and the last column describes it. When there is a middle column, it lists the command name. If there are no keystrokes for a given command, you'll see (**none**) in the first column. Access these commands by typing **M-x** followed by the command name. If you're unsure of the name, you can type a tab or a carriage return, and Emacs lists possible completions of what you've typed so far.

Because Emacs is such a comprehensive editor, containing literally thousands of commands, some commands must be omitted for the sake of preserving a "quick"

reference. You can browse the command set from within Emacs by typing **C-h** (for help) or **M-x Tab** (for command names).

## Absolutely Essential Commands

If you're just getting started with Emacs, here's a short list of the most important commands.

| Keystrokes | Description                           |
|------------|---------------------------------------|
| C-h        | Enter the online help system.         |
| C-x C-s    | Save the file.                        |
| C-x C-c    | Exit Emacs.                           |
| C-x u      | Undo last edit (can be repeated).     |
| C-g        | Get out of current command operation. |
| C-p        | Move up to the previous line.         |
| C-n        | Move down to the next line.           |
| C-f        | Move forward one character.           |
| C-b        | Move backward one character.          |
| C-v        | Move forward by one screen.           |
| M-v        | Move backward by one screen           |
| C-s        | Search forward for characters.        |
| C-r        | Search backward for characters.       |
| C-d        | Delete the current character.         |
| Del        | Delete the previous character.        |

## Command-Line Syntax

To start an Emacs editing session, type:

```
emacs [file]
```

## Summary of Commands by Group

Reminder: C- indicates the Ctrl key; M- indicates the Meta key.

### File-Handling Commands

| Keystrokes | Command name            | Description                                           |
|------------|-------------------------|-------------------------------------------------------|
| C-x C-f    | find-file               | Find file and read it.                                |
| C-x C-v    | find-alternate-file     | Read another file; replace the one read with C-x C-f. |
| C-x i      | insert-file             | Insert file at cursor position.                       |
| C-x C-s    | save-buffer             | Save file (may hang terminal; use C-q to restart).    |
| C-x C-w    | write-file              | Write buffer contents to file.                        |
| C-x C-c    | save-buffers-kill-emacs | Exit Emacs.                                           |
| C-z        | suspend-emacs           | Suspend Emacs (use exit or fg to restart).            |

## Cursor-Movement Commands

| Keystrokes       | Command name        | Description                                    |
|------------------|---------------------|------------------------------------------------|
| C-f              | forward-char        | Move <i>forward</i> one character (right).     |
| C-b              | backward-char       | Move <i>backward</i> one character (left).     |
| C-p              | previous-line       | Move to <i>previous</i> line (up).             |
| C-n              | next-line           | Move to <i>next</i> line (down).               |
| M-f              | forward-word        | Move one word <i>forward</i> .                 |
| M-b              | backward-word       | Move one word <i>backward</i> .                |
| C-a              | beginning-of-line   | Move to beginning of line.                     |
| C-e              | end-of-line         | Move to <i>end</i> of line.                    |
| M-a              | backward-sentence   | Move backward one sentence.                    |
| M-e              | forward-sentence    | Move forward one sentence.                     |
| M-{              | backward-paragraph  | Move backward one paragraph.                   |
| M-}              | forward-paragraph   | Move forward one paragraph.                    |
| C-v              | scroll-up           | Move forward one screen.                       |
| M-v              | scroll-down         | Move backward one screen.                      |
| C-x [            | backward-page       | Move backward one page.                        |
| C-x ]            | forward-page        | Move forward one page.                         |
| M->              | end-of-buffer       | Move to end of file.                           |
| M-<              | beginning-of-buffer | Move to beginning of file.                     |
| M-g g or M-g M-g | goto-line           | Go to line <i>n</i> of file.                   |
| (none)           | goto-char           | Go to character <i>n</i> of file.              |
| C-l              | recenter            | Redraw screen with current line in the center. |
| M-n              | digit-argument      | Repeat the next command <i>n</i> times.        |
| C-u <i>n</i>     | universal-argument  | Repeat the next command <i>n</i> times.        |

## Deletion Commands

| Keystrokes | Command name            | Description                                         |
|------------|-------------------------|-----------------------------------------------------|
| Del        | backward-delete-char    | Delete previous character.                          |
| C-d        | delete-char             | Delete character under cursor.                      |
| M-Del      | backward-kill-word      | Delete previous word.                               |
| M-d        | kill-word               | Delete the word the cursor is on.                   |
| C-k        | kill-line               | Delete from cursor to end of line.                  |
| M-k        | kill-sentence           | Delete sentence the cursor is on.                   |
| C-x Del    | backward-kill-sentence  | Delete previous sentence.                           |
| C-y        | yank                    | Restore what you've deleted.                        |
| C-w        | kill-region             | Delete a marked region (see next section).          |
| (none)     | backward-kill-paragraph | Delete previous paragraph.                          |
| (none)     | kill-paragraph          | Delete from the cursor to the end of the paragraph. |

## Paragraphs and Regions

| Keystrokes | Command name            | Description                                     |
|------------|-------------------------|-------------------------------------------------|
| C-@        | set-mark-command        | Mark the beginning (or end) of a region.        |
| C-Space    | (same as above)         | (same as above)                                 |
| C-x C-p    | mark-page               | Mark page.                                      |
| C-x C-x    | exchange-point-and-mark | Exchange location of cursor and mark.           |
| C-x h      | mark-whole-buffer       | Mark buffer.                                    |
| M-q        | fill-paragraph          | Reformat paragraph.                             |
| (none)     | fill-region             | Reformat individual paragraphs within a region. |
| M-h        | mark-paragraph          | Mark paragraph.                                 |

## Stopping and Undoing Commands

| Keystrokes | Command name    | Description                                                                         |
|------------|-----------------|-------------------------------------------------------------------------------------|
| C-g        | keyboard-quit   | Abort current command.                                                              |
| C-x u      | advertised-undo | Undo last edit (can be done repeatedly).                                            |
| (none)     | revert-buffer   | Restore buffer to the state it was in when the file was last saved (or auto-saved). |

## Transposition Commands

| Keystrokes | Command name         | Description               |
|------------|----------------------|---------------------------|
| C-t        | transpose-chars      | Transpose two letters.    |
| M-t        | transpose-words      | Transpose two words.      |
| C-x C-t    | transpose-lines      | Transpose two lines.      |
| (none)     | transpose-sentences  | Transpose two sentences.  |
| (none)     | transpose-parapraphs | Transpose two paragraphs. |

## Search Commands

| Keystrokes  | Command name       | Description                        |
|-------------|--------------------|------------------------------------|
| C-s         | isearch-forward    | Incremental search forward.        |
| C-r         | isearch-backward   | Incremental search backward        |
| M-%         | query-replace      | Search and replace.                |
| C-M-s Enter | re-search-forward  | Regular expression search forward. |
| C-M-r Enter | re-search-backward | Regular expression search backward |

## Capitalization Commands

| Keystrokes | Command name                       | Description                      |
|------------|------------------------------------|----------------------------------|
| M-c        | capitalize-word                    | Capitalize first letter of word. |
| M-u        | upcase-word                        | Uppercase word.                  |
| M-l        | downcase-word                      | Lowercase word.                  |
| M- M-c     | negative-argument; capitalize-word | Capitalize previous word.        |
| M- M-u     | negative-argument; upcase-word     | Uppercase previous word.         |
| M- M-l     | negative-argument; downcase-word   | Lowercase previous word.         |
| (none)     | capitalize-region                  | Capitalize region.               |
| C-x C-u    | upcase-region                      | Uppercase region                 |
| C-x C-l    | downcase-region                    | Lowercase region.                |

## Word-Abbreviation Commands

| Keystrokes | Command name              | Description                                |
|------------|---------------------------|--------------------------------------------|
| (none)     | abbrev-mode               | Enter (or exit) word abbreviation mode.    |
| C-x a i g  | inverse-add-global-abbrev | Type global abbreviation, then definition. |
| C-x a i l  | inverse-add-local-abbrev  | Type local abbreviation, then definition.  |
| (none)     | unexpand-abbrev           | Undo the last word abbreviation.           |
| (none)     | write-abbrev-file         | Write the word abbreviation file.          |
| (none)     | edit-abbrevs              | Edit the word abbreviations.               |
| (none)     | list-abbrevs              | View the word abbreviations.               |
| (none)     | kill-all-abbrevs          | Kill abbreviations for this session.       |

## Buffer-Manipulation Commands

| Keystrokes | Command name      | Description                               |
|------------|-------------------|-------------------------------------------|
| C-x b      | switch-to-buffer  | Move to specified buffer.                 |
| C-x C-b    | list-buffers      | Display buffer list.                      |
| C-x k      | kill-buffer       | Delete specified buffer.                  |
| (none)     | kill-some-buffers | Ask about deleting each buffer.           |
| (none)     | rename-buffer     | Change buffer name to specified name.     |
| C-x s      | save-some-buffers | Ask whether to save each modified buffer. |

## Window Commands

| Keystrokes | Command name              | Description                                                  |
|------------|---------------------------|--------------------------------------------------------------|
| C-x 2      | split-window-vertically   | Divide the current window into two, one on top of the other. |
| C-x 3      | split-window-horizontally | Divide the current window into two, side by side.            |
| C-x >      | scroll-right              | Scroll the window right.                                     |

| Keystrokes | Command name                  | Description                                 |
|------------|-------------------------------|---------------------------------------------|
| C-x <      | scroll-left                   | Scroll the window left.                     |
| C-x o      | other-window                  | Move to the other window.                   |
| C-x 0      | delete-window                 | Delete current window.                      |
| C-x 1      | delete-other-windows          | Delete all windows but this one.            |
| (none)     | delete-windows-on             | Delete all windows on a given buffer.       |
| C-x ^      | enlarge-window                | Make window taller.                         |
| (none)     | shrink-window                 | Make window shorter.                        |
| C-x }      | enlarge-window-horizontally   | Make window wider.                          |
| C-x {      | shrink-window-horizontally    | Make window narrower.                       |
| C-M-v      | scroll-other-window           | Scroll other window.                        |
| C-x 4 f    | find-file-other-window        | Find a file in the other window.            |
| C-x 4 b    | switch-to-buffer-other-window | Select a buffer in the other window.        |
| C-x 5 f    | find-file-other-frame         | Find a file in a new frame.                 |
| C-x 5 b    | switch-to-buffer-other-frame  | Select a buffer in another frame.           |
| (none)     | compare-windows               | Compare two buffers; show first difference. |

## Special Shell Characters

| Keystrokes | Command Name            | Description                                  |
|------------|-------------------------|----------------------------------------------|
| M-!        | shell-command           | Run a shell command and display the results. |
| (none)     | shell                   | Start a shell buffer.                        |
| C-c C-c    | comint-interrupt-subjob | Terminate the current job.                   |
| C-c C-d    | comint-send-eof         | End of file character.                       |
| C-c C-u    | comint-kill-input       | Erase current line.                          |
| C-c C-w    | backward-kill-word      | Erase the previous word.                     |
| C-c C-z    | comint-stop-subjob      | Suspend the current job.                     |

## Indentation Commands

| Keystrokes | Command name               | Description                                                                                                                                                                                               |
|------------|----------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| C-x .      | set-fill-prefix            | Use characters from the beginning of the line up to the cursor column as the “fill prefix.” This prefix is prepended to each line in the paragraph. Cancel the prefix by typing this command in column 1. |
| (none)     | indented-text-mode         | Major mode: each tab defines a new indent for subsequent lines.                                                                                                                                           |
| (none)     | text-mode                  | Exit indented text mode; return to text mode.                                                                                                                                                             |
| C-M-\      | indent-region              | Indent a region to match first line in region.                                                                                                                                                            |
| M-m        | back-to-indentation        | Move cursor to first nonblank character on line.                                                                                                                                                          |
| M-^        | delete-indentation         | Join this line to the previous one.                                                                                                                                                                       |
| C-M-o      | split-line                 | Split line at cursor; indent to column of cursor.                                                                                                                                                         |
| (none)     | fill-individual-paragraphs | Reformat indented paragraphs, keeping indentation.                                                                                                                                                        |

## Centering Commands

| Keystrokes | Command name     | Description                         |
|------------|------------------|-------------------------------------|
| M-s        | center-line      | Center line that cursor is on.      |
| M-S        | center-paragraph | Center paragraph that cursor is on. |
| (none)     | center-region    | Center currently defined region.    |

## Macro Commands

| Keystrokes      | Command name                           | Description                                      |
|-----------------|----------------------------------------|--------------------------------------------------|
| C-x ( or F3 key | start-kbd-macro                        | Start macro definition.                          |
| C-x ) or F4 key | end-kbd-macro                          | End macro definition.                            |
| C-x e or F4 key | call-last-kbd-macro                    | Execute last macro defined.                      |
| M-n C-x e       | digit-argument and call-last-kbd-macro | Execute last macro defined <i>n</i> times.       |
| C-u C-x (       | universal-argument and start-kbd-macro | Execute last macro defined, then add keystrokes. |
| (none)          | name-last-kbd-macro                    | Name last macro you created (before saving it).  |
| (none)          | insert-kbd-macro                       | Insert the macro you named into a file.          |
| (none)          | load-file                              | Load macro files you've saved and loaded.        |
| (none)          | <i>macroname</i>                       | Execute a keyboard macro you've saved.           |
| C-x q           | kbd-macro-query                        | Insert a query in a macro definition.            |
| C-u C-x q       | (none)                                 | Insert a recursive edit in a macro definition.   |
| C-M-c           | exit-recursive-edit                    | Exit a recursive edit.                           |

## Detail Information Help Commands

| Keystrokes | Command name            | Description                                                         |
|------------|-------------------------|---------------------------------------------------------------------|
| C-h a      | command-apropos         | What commands involve this concept?                                 |
| C-h d      | apropos                 | What functions and variables involve this concept?                  |
| C-h c      | describe-key-briefly    | What command does this keystroke sequence run?                      |
| C-h b      | describe-bindings       | What are all the key bindings for this buffer?                      |
| C-h k      | describe-key            | What command does this keystroke sequence run, and what does it do? |
| C-h l      | view-lossage            | What are the last 100 characters I typed?                           |
| C-h e      | view-echo-area-messages | Display the *Messages* buffer.                                      |
| C-h w      | where-is                | What is the key binding for this command?                           |
| C-h f      | describe-function       | What does this function do?                                         |
| C-h v      | describe-variable       | What does this variable mean, and what is its value?                |
| C-h m      | describe-mode           | Tell me about the mode the current buffer is in.                    |
| C-h s      | describe-syntax         | What is the syntax table for this buffer?                           |

## Help Commands

| Keystrokes | Command name          | Description                                      |
|------------|-----------------------|--------------------------------------------------|
| C-h t      | help-with-tutorial    | Run the Emacs tutorial.                          |
| C-h i      | info                  | Start the Info documentation reader.             |
| C-h r      | info-emacs-command    | View the Emacs documentation in the Info reader. |
| C-h n      | view-emacs-news       | View news about updates to Emacs.                |
| C-h C-c    | describe-copying      | View the Emacs General Public License.           |
| C-h C-d    | describe-distribution | View information on ordering Emacs from the FSF. |
| C-h C-w    | describe-no-warranty  | View the (non)warranty for Emacs.                |

## Summary of Commands by Key

Emacs commands are presented below in two alphabetical lists. Reminder: C- indicates the Ctrl key; M- indicates the Meta key.

## Control-Key Sequences

| Keystrokes | Command name            | Description                                        |
|------------|-------------------------|----------------------------------------------------|
| C-@        | set-mark-command        | Mark the beginning (or end) of a region.           |
| C-Space    | (same as previous)      |                                                    |
| C-]        | (none)                  | Exit recursive edit and exit query-replace.        |
| C-a        | beginning-of-line       | Move to beginning of line.                         |
| C-b        | backward-char           | Move <i>backward</i> one character (left).         |
| C-c C-c    | comint-interrupt-subjob | Terminate the current job.                         |
| C-c C-d    | comint-send-eof         | End-of-file character.                             |
| C-c C-u    | comint-kill-input       | Erase current line.                                |
| C-c C-w    | backward-kill-word      | Erase the previous word.                           |
| C-c C-z    | comint-stop-subjob      | Suspend the current job.                           |
| C-d        | delete-char             | Delete character under cursor.                     |
| C-e        | end-of-line             | Move to <i>end</i> of line.                        |
| C-f        | forward-char            | Move <i>forward</i> one character (right).         |
| C-g        | keyboard-quit           | Abort current command.                             |
| C-h        | help-command            | Enter the online help system.                      |
| C-h a      | command-apropos         | What commands involve this concept?                |
| C-h b      | describe-bindings       | What are all the key bindings for this buffer?     |
| C-h C-c    | describe-copying        | View the Emacs General Public License.             |
| C-h C-d    | describe-distribution   | View information on ordering Emacs from FSF.       |
| C-h C-w    | describe-no-warranty    | View the (non)warranty for Emacs.                  |
| C-h c      | describe-key-briefly    | What command does this keystroke sequence run?     |
| C-h d      | apropos                 | What functions and variables involve this concept? |
| C-h e      | view-echo-area-messages | Display the *Messages* buffer.                     |
| C-h f      | describe-function       | What does this function do?                        |

| Keystrokes | Command name                           | Description                                                                                                                                                                                               |
|------------|----------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| C-h i      | info                                   | Start the Info documentation reader.                                                                                                                                                                      |
| C-h k      | describe-key                           | What command does this keystroke sequence run, and what does it do?                                                                                                                                       |
| C-h l      | view-lossage                           | What are the last 100 characters I typed?                                                                                                                                                                 |
| C-h m      | describe-mode                          | Tell me about the mode the current buffer is in.                                                                                                                                                          |
| C-h n      | view-emacs-news                        | View news about updates to Emacs.                                                                                                                                                                         |
| C-h r      | info-emacs-manual                      | View the Emacs documentation in the Info reader.                                                                                                                                                          |
| C-h s      | describe-syntax                        | What is the syntax table for this buffer?                                                                                                                                                                 |
| C-h t      | help-with-tutorial                     | Run the Emacs tutorial.                                                                                                                                                                                   |
| C-h v      | describe-variable                      | What does this variable mean, and what is its value?                                                                                                                                                      |
| C-h w      | where-is                               | What is the key binding for this command?                                                                                                                                                                 |
| C-k        | kill-line                              | Delete from cursor to end of line.                                                                                                                                                                        |
| C-l        | recenter                               | Redraw screen with current line in the center.                                                                                                                                                            |
| C-M-\      | indent-region                          | Indent a region to match first line in region.                                                                                                                                                            |
| C-M-c      | exit-recursive-edit                    | Exit a recursive edit.                                                                                                                                                                                    |
| C-M-o      | split-line                             | Split line at cursor; indent to column of cursor.                                                                                                                                                         |
| C-M-v      | scroll-other-window                    | Scroll other window.                                                                                                                                                                                      |
| C-n        | next-line                              | Move to <i>next</i> line (down).                                                                                                                                                                          |
| C-p        | previous-line                          | Move to <i>previous</i> line (up).                                                                                                                                                                        |
| C-r        | isearch-backward                       | Start incremental search backward.                                                                                                                                                                        |
| C-s        | isearch-forward                        | Start incremental search forward.                                                                                                                                                                         |
| C-t        | transpose-chars                        | Transpose two letters.                                                                                                                                                                                    |
| C-u n      | universal-argument                     | Repeat the next command <i>n</i> times.                                                                                                                                                                   |
| C-u C-x (  | universal-argument and start-kbd-macro | Execute last macro defined, then add keystrokes.                                                                                                                                                          |
| C-u C-x q  | (none)                                 | Insert recursive edit in a macro definition.                                                                                                                                                              |
| C-v        | scroll-up                              | Move forward one screen.                                                                                                                                                                                  |
| C-w        | kill-region                            | Delete a marked region.                                                                                                                                                                                   |
| C-x (      | start-kbd-macro                        | Start macro definition.                                                                                                                                                                                   |
| C-x )      | end-kbd-macro                          | End macro definition.                                                                                                                                                                                     |
| C-x [      | backward-page                          | Move backward one page.                                                                                                                                                                                   |
| C-x ]      | forward-page                           | Move forward one page.                                                                                                                                                                                    |
| C-x ^      | enlarge-window                         | Make window taller.                                                                                                                                                                                       |
| C-x {      | shrink-window-horizontally             | Make window narrower.                                                                                                                                                                                     |
| C-x }      | enlarge-window-horizontally            | Make window wider.                                                                                                                                                                                        |
| C-x <      | scroll-left                            | Scroll the window left.                                                                                                                                                                                   |
| C-x >      | scroll-right                           | Scroll the window right.                                                                                                                                                                                  |
| C-x .      | set-fill-prefix                        | Use characters from the beginning of the line up to the cursor column as the “fill prefix.” This prefix is prepended to each line in the paragraph. Cancel the prefix by typing this command in column 1. |
| C-x 0      | delete-window                          | Delete current window.                                                                                                                                                                                    |
| C-x 1      | delete-other-windows                   | Delete all windows but this one.                                                                                                                                                                          |

| Keystrokes | Command name                  | Description                                                  |
|------------|-------------------------------|--------------------------------------------------------------|
| C-x 2      | split-window-vertically       | Divide the current window into two, one on top of the other. |
| C-x 3      | split-window-horizontally     | Divide the current window into two, side by side.            |
| C-x 4 b    | switch-to-buffer-other-window | Select a buffer in the other window.                         |
| C-x 4 f    | find-file-other-window        | Find a file in the other window.                             |
| C-x 5 b    | switch-to-buffer-other-frame  | Select a buffer in another frame.                            |
| C-x 5 f    | find-file-other-frame         | Find a file in a new frame.                                  |
| C-x C-b    | list-buffers                  | Display the buffer list.                                     |
| C-x C-c    | save-buffers-kill-emacs       | Exit Emacs.                                                  |
| C-x C-f    | find-file                     | Find file and read it.                                       |
| C-x C-l    | downcase-region               | Lowercase region.                                            |
| C-x C-p    | mark-page                     | Mark page.                                                   |
| C-x C-q    | (none)                        | Toggle read-only status of buffer.                           |
| C-x C-s    | save-buffer                   | Save file (may hang terminal; use C-q to restart).           |
| C-x C-t    | transpose-lines               | Transpose two lines.                                         |
| C-x C-u    | upcase-region                 | Uppercase region                                             |
| C-x C-v    | find-alternate-file           | Read an alternate file, replacing the one read with C-x C-f. |
| C-x C-w    | write-file                    | Write buffer contents to file.                               |
| C-x C-x    | exchange-point-and-mark       | Exchange location of cursor and mark.                        |
| C-x DEL    | backward-kill-sentence        | Delete previous sentence.                                    |
| C-x a g    | inverse-add-global-abbrev     | Type global abbreviation, then definition.                   |
| C-x a l    | inverse-add-local-abbrev      | Type local abbreviation, then definition.                    |
| C-x b      | switch-to-buffer              | Move to the buffer specified.                                |
| C-x e      | call-last-kbd-macro           | Execute last macro defined.                                  |
| C-x h      | mark-whole-buffer             | Mark buffer.                                                 |
| C-x i      | insert-file                   | Insert file at cursor position.                              |
| C-x k      | kill-buffer                   | Delete the buffer specified.                                 |
| C-x o      | other-window                  | Move to the other window.                                    |
| C-x q      | kbd-macro-query               | Insert a query in a macro definition.                        |
| C-x s      | save-some-buffers             | Ask whether to save each modified buffer.                    |
| C-x u      | advertised-undo               | Undo last edit (can be done repeatedly).                     |
| C-y        | yank                          | Restore what you've deleted.                                 |
| C-z        | suspend-emacs                 | Suspend Emacs (use exit or fg to restart).                   |

## Meta-Key Sequences

| Keystrokes | Command name                       | Description                                |
|------------|------------------------------------|--------------------------------------------|
| Meta       | (none)                             | Exit a query-replace or successful search. |
| M- - M-c   | negative-argument; capitalize-word | Capitalize previous word.                  |
| M- - M-l   | negative-argument; downcase-word   | Lowercase previous word.                   |
| M- - M-u   | negative-argument; upcase-word     | Uppercase previous word.                   |

| Keystrokes       | Command name                           | Description                                      |
|------------------|----------------------------------------|--------------------------------------------------|
| M-\$             | spell-word                             | Check spelling of word after cursor.             |
| M-<              | beginning-of-buffer                    | Move to beginning of file.                       |
| M->              | end-of-buffer                          | Move to end of file.                             |
| M-{              | backward-paragraph                     | Move backward one paragraph.                     |
| M-}              | forward-paragraph                      | Move forward one paragraph.                      |
| M-^              | delete-indentation                     | Join this line to the previous one.              |
| M-n              | digit-argument                         | Repeat the next command <i>n</i> times.          |
| M-n C-x e        | digit-argument and call-last-kbd-macro | Execute the last defined macro <i>n</i> times.   |
| M-a              | backward-sentence                      | Move backward one sentence.                      |
| M-b              | backward-word                          | Move one word <i>backward</i> .                  |
| M-c              | capitalize-word                        | Capitalize first letter of word.                 |
| M-d              | kill-word                              | Delete word that cursor is on.                   |
| M-DEL            | backward-kill-word                     | Delete previous word.                            |
| M-e              | forward-sentence                       | Move forward one sentence.                       |
| M-f              | forward-word                           | Move one word <i>forward</i> .                   |
| M-g g or M-g M-g | goto-line                              | Go to line <i>n</i> of file.                     |
| M-h              | mark-paragraph                         | Mark paragraph.                                  |
| M-k              | kill-sentence                          | Delete sentence the cursor is on.                |
| M-l              | downcase-word                          | Lowercase word.                                  |
| M-m              | back-to-indentation                    | Move cursor to first nonblank character on line. |
| M-q              | fill-paragraph                         | Reformat paragraph.                              |
| M-s              | center-line                            | Center line that cursor is on.                   |
| M-S              | center-paragraph                       | Center paragraph that cursor is on.              |
| M-t              | transpose-words                        | Transpose two words.                             |
| M-u              | upcase-word                            | Uppercase word.                                  |
| M-v              | scroll-down                            | Move backward one screen.                        |
| M-x              | (none)                                 | Access command by command name.                  |

## Summary of Commands by Name

The Emacs commands below are presented alphabetically by command name. Use **M-x** to access the command name. Reminder: C- indicates the Ctrl key; M- indicates the Meta key.

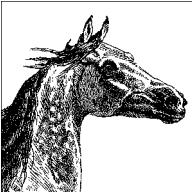
| Command name         | Keystrokes | Description                                        |
|----------------------|------------|----------------------------------------------------|
| macroname            | (none)     | Execute a keyboard macro you've saved.             |
| abbrev-mode          | (none)     | Enter (or exit) word abbreviation mode.            |
| advertised-undo      | C-x u      | Undo last edit (can be done repeatedly).           |
| apropos              | (none)     | What functions and variables involve this concept? |
| back-to-indentation  | M-m        | Move cursor to first nonblank character on line.   |
| backward-char        | C-b        | Move <i>backward</i> one character (left).         |
| backward-delete-char | Del        | Delete previous character.                         |

| Command name                           | Keystrokes | Description                                                         |
|----------------------------------------|------------|---------------------------------------------------------------------|
| backward-kill-paragraph                | (none)     | Delete previous paragraph.                                          |
| backward-kill-sentence                 | C-x Del    | Delete previous sentence.                                           |
| backward-kill-word                     | C-c C-w    | Erase previous word.                                                |
| backward-kill-word                     | M-Del      | Delete previous word.                                               |
| backward-page                          | C-x [      | Move backward one page.                                             |
| backward-paragraph                     | M-{        | Move backward one paragraph.                                        |
| backward-sentence                      | M-a        | Move backward one sentence.                                         |
| backward-word                          | M-b        | Move backward one word.                                             |
| beginning-of-buffer                    | M-<        | Move to beginning of file.                                          |
| beginning-of-line                      | C-a        | Move to beginning of line.                                          |
| call-last-kbd-macro                    | C-x e      | Execute last macro defined.                                         |
| capitalize-region                      | (none)     | Capitalize region.                                                  |
| capitalize-word                        | M-c        | Capitalize first letter of word.                                    |
| center-line                            | M-s        | Center line that cursor is on.                                      |
| center-paragraph                       | M-S        | Center paragraph that cursor is on.                                 |
| center-region                          | (none)     | Center currently defined region.                                    |
| comint-interrupt-subjob                | C-c C-c    | Terminate the current job.                                          |
| comint-kill-input                      | C-c C-u    | Erase current line.                                                 |
| comint-send-eof                        | C-c C-d    | End of file character.                                              |
| comint-stop-subjob                     | C-c C-z    | Suspend current job.                                                |
| command-apropos                        | C-h a      | What commands involve this concept?                                 |
| compare-windows                        | (none)     | Compare two buffers; show first difference.                         |
| delete-char                            | C-d        | Delete character under cursor.                                      |
| delete-indentation                     | M-^        | Join this line to previous one.                                     |
| delete-other-windows                   | C-x 1      | Delete all windows but this one.                                    |
| delete-window                          | C-x 0      | Delete current window.                                              |
| delete-windows-on                      | (none)     | Delete all windows on a given buffer.                               |
| describe-bindings                      | C-h b      | What are all the key bindings for in this buffer?                   |
| describe-copying                       | C-h C-c    | View the Emacs General Public License.                              |
| describe-distribution                  | C-h C-d    | View information on ordering Emacs from the FSF.                    |
| describe-function                      | C-h f      | What does this function do?                                         |
| describe-key                           | C-h k      | What command does this keystroke sequence run, and what does it do? |
| describe-key-briefly                   | C-h c      | What command does this keystroke sequence run?                      |
| describe-mode                          | C-h m      | Tell me about the mode the current buffer is in.                    |
| describe-no-warranty                   | C-h C-w    | View the (non)warranty for Emacs.                                   |
| describe-syntax                        | C-h s      | What is the syntax table for this buffer?                           |
| describe-variable                      | C-h v      | What does this variable mean, and what is its value?                |
| digit-argument and call-last-kbd-macro | M-n C-x e  | Execute the last defined macro <i>n</i> times.                      |
| digit-argument                         | M-n        | Repeat next command <i>n</i> times.                                 |
| downcase-region                        | C-x C-l    | Lowercase region.                                                   |

| Command name                | Keystrokes          | Description                                                     |
|-----------------------------|---------------------|-----------------------------------------------------------------|
| downcase-word               | M-l                 | Lowercase word.                                                 |
| edit-abbrevs                | (none)              | Edit word abbreviations.                                        |
| end-kbd-macro               | C-x) or F4 key      | End macro definition.                                           |
| end-of-buffer               | M->                 | Move to end of file.                                            |
| end-of-line                 | C-e                 | Move to end of line.                                            |
| enlarge-window              | C-x ^               | Make window taller.                                             |
| enlarge-window-horizontally | C-x }               | Make window wider.                                              |
| exchange-point-and-mark     | C-x C-x             | Exchange location of cursor and mark.                           |
| exit-recursive-edit         | C-M-c               | Exit a recursive edit.                                          |
| fill-individual-paragraphs  | (none)              | Reformat indented paragraphs, keeping indentation.              |
| fill-paragraph              | M-q                 | Reformat paragraph.                                             |
| fill-region                 | (none)              | Reformat individual paragraphs within a region.                 |
| find-alternate-file         | C-x C-v             | Read an alternate file, replacing the one read with C-x C-f.    |
| find-file                   | C-x C-f             | Find file and read it.                                          |
| find-file-other-frame       | C-x 5 f             | Find a file in a new frame.                                     |
| find-file-other-window      | C-x 4 f             | Find a file in the other window.                                |
| forward-char                | C-f                 | Move <i>forward</i> one character (right).                      |
| forward-page                | C-x ]               | Move forward one page.                                          |
| forward-paragraph           | M-{                 | Move forward one paragraph.                                     |
| forward-sentence            | M-e                 | Move forward one sentence.                                      |
| forward-word                | M-f                 | Move forward one word.                                          |
| goto-char                   | (none)              | Go to character <i>n</i> of file.                               |
| goto-line                   | M-g g or M-g<br>M-g | Go to line <i>n</i> of file.                                    |
| help-command                | C-h                 | Enter the online help system.                                   |
| help-with-tutorial          | C-h t               | Run the Emacs tutorial.                                         |
| indent-region               | C-M-\               | Indent a region to match first line in region.                  |
| indented-text-mode          | (none)              | Major mode: each tab defines a new indent for subsequent lines. |
| info                        | C-h i               | Start the Info documentation reader.                            |
| info-emacs-manual           | C-h r               | View the Emacs documentation in the Info reader.                |
| insert-file                 | C-x i               | Insert file at cursor position.                                 |
| insert-kbd-macro            | (none)              | Insert the macro you named into a file.                         |
| inverse-add-global-abbrev   | C-x a i g           | Type global abbreviation, then definition.                      |
| inverse-add-local-abbrev    | C-x a i l           | Type local abbreviation, then definition.                       |
| isearch-backward            | C-r                 | Start incremental search backward.                              |
| isearch-backward-regexp     | C-r                 | Same, but search for regular expression.                        |
| isearch-forward             | C-s                 | Start incremental search forward.                               |
| isearch-forward-regexp      | C-r                 | Same, but search for regular expression.                        |
| kbd-macro-query             | C-x q               | Insert a query in a macro definition.                           |
| keyboard-quit               | C-g                 | Abort current command.                                          |
| kill-all-abbrevs            | (none)              | Kill abbreviations for this session.                            |
| kill-buffer                 | C-x k               | Delete the buffer specified.                                    |

| Command name                          | Keystrokes        | Description                                                                                                                                                                                               |
|---------------------------------------|-------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| kill-line                             | C-k               | Delete from cursor to end of line.                                                                                                                                                                        |
| kill-paragraph                        | (none)            | Delete from cursor to end of paragraph.                                                                                                                                                                   |
| kill-region                           | C-w               | Delete a marked region.                                                                                                                                                                                   |
| kill-sentence                         | M-k               | Delete sentence the cursor is on.                                                                                                                                                                         |
| kill-some-buffers                     | (none)            | Ask about deleting each buffer.                                                                                                                                                                           |
| kill-word                             | M-d               | Delete word the cursor is on.                                                                                                                                                                             |
| list-abbrevs                          | (none)            | View word abbreviations.                                                                                                                                                                                  |
| list-buffers                          | C-x C-b           | Display buffer list.                                                                                                                                                                                      |
| load-file                             | (none)            | Load macro files you've saved.                                                                                                                                                                            |
| mark-page                             | C-x C-p           | Mark page.                                                                                                                                                                                                |
| mark-paragraph                        | M-h               | Mark paragraph.                                                                                                                                                                                           |
| mark-whole-buffer                     | C-x h             | Mark buffer.                                                                                                                                                                                              |
| name-last-kbd-macro                   | (none)            | Name last macro you created (before saving it).                                                                                                                                                           |
| negative-argument;<br>capitalize-word | M- - M-c          | Capitalize previous word.                                                                                                                                                                                 |
| negative-argument;<br>downcase-word   | M- - M-l          | Lowercase previous word.                                                                                                                                                                                  |
| negative-argument;<br>upcase-word     | M- - M-u          | Uppercase previous word.                                                                                                                                                                                  |
| next-line                             | C-n               | Move to <i>next</i> line (down).                                                                                                                                                                          |
| other-window                          | C-x o             | Move to the other window.                                                                                                                                                                                 |
| previous-line                         | C-p               | Move to <i>previous</i> line (up).                                                                                                                                                                        |
| query-replace                         | M-%               | Search and replace.                                                                                                                                                                                       |
| query-replace-regexp                  | C-% Meta          | Query-replace a regular expression.                                                                                                                                                                       |
| recenter                              | C-l               | Redraw screen, with current line in center.                                                                                                                                                               |
| rename-buffer                         | (none)            | Change buffer name to specified name.                                                                                                                                                                     |
| replace-regexp                        | (none)            | Replace a regular expression unconditionally.                                                                                                                                                             |
| re-search-backward                    | (none)            | Simple regular expression search backward.                                                                                                                                                                |
| re-search-forward                     | (none)            | Simple regular expression search forward.                                                                                                                                                                 |
| revert-buffer                         | (none)            | Restore buffer to the state it was in when the file was last saved (or auto-saved).                                                                                                                       |
| save-buffer                           | C-x C-s           | Save file (may hang terminal; use C-q to restart).                                                                                                                                                        |
| save-buffers-kill-emacs               | C-x C-c           | Exit Emacs.                                                                                                                                                                                               |
| save-some-buffers                     | C-x s             | Ask whether to save each modified buffer.                                                                                                                                                                 |
| scroll-down                           | M-v               | Move backward one screen.                                                                                                                                                                                 |
| scroll-left                           | C-x <             | Scroll the window left.                                                                                                                                                                                   |
| scroll-other-window                   | C-M-v             | Scroll other window.                                                                                                                                                                                      |
| scroll-right                          | C-x >             | Scroll the window right.                                                                                                                                                                                  |
| scroll-up                             | C-v               | Move forward one screen.                                                                                                                                                                                  |
| set-fill-prefix                       | C-x .             | Use characters from the beginning of the line up to the cursor column as the "fill prefix." This prefix is prepended to each line in the paragraph. Cancel the prefix by typing this command in column 1. |
| set-mark-command                      | C-@ or<br>C-Space | Mark the beginning (or end) of a region.                                                                                                                                                                  |

| Command name                           | Keystrokes      | Description                                                  |
|----------------------------------------|-----------------|--------------------------------------------------------------|
| shell                                  | (none)          | Start a shell buffer.                                        |
| shell-command                          | M-!             | Run a shell command and display the results.                 |
| shrink-window                          | (none)          | Make window shorter.                                         |
| shrink-window-horizontally             | C-x {           | Make window narrower.                                        |
| spell-buffer                           | (none)          | Check spelling of current buffer.                            |
| spell-region                           | (none)          | Check spelling of current region.                            |
| spell-string                           | (none)          | Check spelling of string typed in minibuffer.                |
| spell-word                             | M-\$            | Check spelling of word after cursor.                         |
| split-line                             | C-M-o           | Split line at cursor; indent to column of cursor.            |
| split-window-vertically                | C-x 2           | Divide the current window into two, one on top of the other. |
| split-window-horizontally              | C-x 3           | Divide the current window into two, side by side.            |
| start-kbd-macro                        | C-x ( or F3 key | Start macro definition.                                      |
| suspend-emacs                          | C-z             | Suspend Emacs (use exit or fg to restart).                   |
| switch-to-buffer                       | C-x b           | Move to the buffer specified.                                |
| switch-to-buffer-other-frame           | C-x 5 b         | Select a buffer in another frame.                            |
| switch-to-buffer-other-window          | C-x 4 b         | Select a buffer in the other window.                         |
| text-mode                              | (none)          | Exit indented text mode; return to text mode.                |
| transpose-chars                        | C-t             | Transpose two letters.                                       |
| transpose-lines                        | C-x C-t         | Transpose two lines.                                         |
| transpose-paragraphs                   | (none)          | Transpose two paragraphs.                                    |
| transpose-sentences                    | (none)          | Transpose two sentences.                                     |
| transpose-words                        | M-t             | Transpose two words.                                         |
| unexpand-abbrev                        | (none)          | Undo the last word abbreviation.                             |
| universal-argument                     | C-u n           | Repeat the next command <i>n</i> times.                      |
| universal-argument and start-kbd-macro | C-u C-x (       | Execute last macro defined, then add keystrokes to it.       |
| upcase-region                          | C-x C-u         | Uppercase region.                                            |
| upcase-word                            | M-u             | Uppercase word.                                              |
| view-emacs-news                        | C-h n           | View news about updates to Emacs.                            |
| view-lossage                           | C-h l           | What are the last 100 characters I typed?                    |
| where-is                               | C-h w           | What is the key binding for this command?                    |
| write-abbrev-file                      | (none)          | Write the word abbreviation file.                            |
| write-file                             | C-x C-w         | Write buffer contents to file.                               |
| yank                                   | C-y             | Restore what you've deleted.                                 |



# 9

## The vi, ex, and vim Editors

The **vi** and **ex** editors are the “standard” editors on Unix systems. You can count on there being some version of them, no matter what Unix flavor you are using. The two editors are in fact the same program; based on how it was invoked, the editor enters full-screen mode or line mode.

**vim** is a popular extended version of **vi**. On some Linux distributions, the **vi** command invokes **vim** in a **vi**-compatible mode.

This chapter presents the following topics:

- Conceptual overview
- Command-line syntax
- Review of **vi** operations
- Alphabetical list of keys in command mode
- **vi** commands
- **vi** configuration
- **ex** basics
- Alphabetical summary of **ex** commands

**vi** is pronounced “vee eye.”

Besides the original Unix **vi**, there are a number of freely available **vi** clones (including **vim**). Both the original **vi** and the clones are covered in *Learning the vi and Vim Editors* by Arnold Robbins et al. (O’Reilly).

### Conceptual Overview

**vi** is the classic screen-editing program for Unix. A number of enhanced versions exist, including **nvi**, **vim**, **vile**, and **elvis**. On GNU/Linux systems, the **vi** command is usually one of these programs (either a copy or a link). The Emacs

editor, covered in Chapter 8, has several **vi** modes that allow you to use many of the same commands covered in this chapter.

The **vi** editor operates in two modes: command mode and insert mode. The dual modes make **vi** an attractive editor for users who separate text entry from editing. For users who edit as they type, the modeless editing of Emacs can be more comfortable. However, **vim** supports both ways of editing, through the **insert-mode** option.

**vi** is based on an older line editor called **ex**. (**ex**, in turn, was developed by Bill Joy at the University of California, Berkeley, from the primordial Unix line editor, **ed**.) A user can invoke powerful editing capabilities within **vi** by typing a colon (:), entering an **ex** command, and pressing the Enter key. Furthermore, you can place **ex** commands in a startup file called `~/.exrc`, which **vi** reads at the beginning of your editing session. Because **ex** commands are such an important part of **vi**, they are also described in this chapter.

One of the most common versions of **vi** found on Linux systems is Bram Moolenaar's Vi IMproved, or **vim**. On some Linux distributions, **vim** is the default version of **vi** and runs when you invoke **vi**. **vim** offers many extra features, and optionally changes some of the basic features of **vi**, most notoriously changing the undo key to support multiple levels of undo.

Fully documenting **vim** is beyond the scope of this chapter, but we do cover some of its most commonly used options and features. Beyond what we cover here, **vim** offers enhanced support to programmers through an integrated build and debugging process, syntax highlighting, extended **ctags** support, and support for Perl and Python, as well as GUI fonts and menus, function-key mapping, independent mapping for each mode, and more. Fortunately, **vim** comes with a powerful internal help system that you can use to learn more about the things we just couldn't fit into this chapter. See <http://www.vim.org> for more information.

## Command-Line Syntax

The three most common ways of starting a **vi** session are:

```
vi [options] file
vi [options] +num file
vi [options] +/pattern file
```

You can open *file* for editing, optionally at line *num* or at the first line matching *pattern*. If no *file* is specified, **vi** opens with an empty buffer.

## Command-Line Options

Because **vi** and **ex** are the same program, they share the same options. However, some options only make sense for one version of the program. Options specific to **vim** are so marked.

+[*num*]

Start editing at line number *num*, or the last line of the file if *num* is omitted.

## +/pattern

Start editing at the first line matching *pattern*. (For **ex**, fails if **nowrapscan** is set in your *.exrc* startup file, since **ex** starts editing at the last line of a file.)

## -b Edit the file in binary mode. {vim}

## -c *command*

Run the given **ex** command upon startup. Only one **-c** option is permitted for **vi**; **vim** accepts up to 10. An older form of this option, **+command**, is still supported.

## --cmd *command*

Like **-c**, but execute the command before any resource files are read. {vim}

## -C vim: Start the editor in **vi**-compatible mode.

## -d Run in diff mode. Can also be invoked by running the command **vimdiff**. {vim}

## -D Debugging mode for use with scripts. {vim}

## -e Run as **ex** (line editing rather than full-screen mode).

## -h Print help message, then exit. {vim}

## -i *file*

Use the specified *file* instead of the default (*~/.viminfo*) to save or restore **vim**'s state. {vim}

## -l Enter Lisp mode for running Lisp programs (not supported in all versions).

## -L List files that were saved due to an aborted editor session or system crash (not supported in all versions). For **vim**, this option is the same as **-r**.

## -m Start the editor with the **write** option turned off so the user cannot write to files. {vim}

## -M Do not allow text in files to be modified. {vim}

## -n Do not use a swapfile; record changes in memory only. {vim}

## --nopugin

Do not load any plug-ins. {vim}

## -N Run **vim** in a non-**vi**-compatible mode. {vim}

## -o[*num*]

Start **vim** with *num* open windows. The default is to open one window for each file. {vim}

## -O[*num*]

Start **vim** with *num* open windows arranged horizontally (split vertically) on the screen. {vim}

## -r [*file*]

Recovery mode; recover and resume editing on *file* after an aborted editor session or system crash. Without *file*, list files available for recovery.

## -R Edit files read-only.

## -s Silent; do not display prompts. Useful when running a script. This behavior also can be set through the older **-** option. For **vim**, only applies when used together with **-e**.

**-s** *scriptfile*

Read and execute commands given in the specified *scriptfile* as if they were typed in from the keyboard. {vim}

**-S** *commandfile*

Read and execute commands given in *commandfile* after loading any files for editing specified on the command line. Shorthand for the option **vim -c 'source commandfile'**. {vim}

**-t** *tag*

Edit the file containing *tag* and position the cursor at its definition. (See **ctags** in Chapter 3 for more information.)

**-T** *type*

Set the terminal type. This value overrides the \$TERM environment variable. {vim}

**-u** *file*

Read configuration information from the specified resource file instead of default *.vimrc* resource file. If the *file* argument is **NONE**, **vim** will read no resource files, load no plug-ins, and run in compatible mode. If the argument is **NORC**, it will read no resource files, but it will load plug-ins. {vim}

**-v** Run in full-screen mode (default for **vi**).

**--version**

Print version information, then exit. {vim}

**-V**[*num*]

Verbose mode; print messages about what options are being set and what files are being read or written. You can set a level of verbosity to increase or decrease the number of messages received. The default value is 10 for high verbosity. {vim}

**-w** *rows*

Set the window size so *rows* lines at a time are displayed; useful when editing over a slow dial-up line (or long distance Internet connection). Older versions of **vi** do not permit a space between the option and its argument. **vim** does not support this option.

**-W** *scriptfile*

Write all typed commands from the current session to the specified *scriptfile*. The file created can be used with the **-s** command. {vim}

**-x** Prompt for a key that will be used to try to encrypt or decrypt a file using **crypt** (not supported in all versions).\*

**-y** Modeless **vi**; run **vim** in insert mode only, without a command mode. {vim}

**-Z** Start **vim** in restricted mode. Do not allow shell commands or suspension of the editor. {vim}

While most people know **ex** commands only by their use within **vi**, the editor also exists as a separate program and can be invoked from the shell (for instance, to edit files as part of a script). Within **ex**, you can enter the **vi** or **visual** command to start **vi**. Similarly, within **vi**, you can enter **Q** to quit the **vi** editor and enter **ex**.

\* The **crypt** command's encryption is weak. Don't use it for serious secrets.

You can exit **ex** in several ways:

- :x** Exit (save changes and quit).
- :q!** Quit without saving changes.
- :vi** Enter the **vi** editor.

## Review of vi Operations

This section provides a review of the following:

- **vi** modes
- Syntax of **vi** commands
- Status-line commands

### Command Mode

Once the file is opened, you are in command mode. From command mode, you can:

- Invoke insert mode
- Issue editing commands
- Move the cursor to a different position in the file
- Invoke **ex** commands
- Invoke a Unix shell
- Save the current version of the file
- Exit **vi**

### Insert Mode

In insert mode, you can enter new text in the file. You normally enter insert mode with the **i** command. Press the Escape key to exit insert mode and return to command mode. The full list of commands that enter insert mode is provided later, in the section “Insert Commands” on page 686.

### Syntax of vi Commands

In **vi**, editing commands have the following general form:

**[n] operator [m] motion**

The basic editing operators are:

- c** Begin a change.
- d** Begin a deletion.
- y** Begin a yank (or copy).

If the current line is the object of the operation, the *motion* is the same as the operator: **cc**, **dd**, **yy**. Otherwise, the editing operators act on objects specified by cursor-movement commands or pattern-matching commands. (For example, **cf**.

changes up to the next period.) *n* and *m* are the number of times the operation is performed, or the number of objects the operation is performed on. If both *n* and *m* are specified, the effect is  $n \times m$ .

An object of operation can be any of the following text blocks:

*word*

Includes characters up to a whitespace character (space or tab) or punctuation mark. A capitalized object is a variant form that recognizes only whitespace.

*sentence*

Is up to ., !, or ?, followed by two spaces.

*paragraph*

Is up to the next blank line or paragraph macro defined by the **para=** option.

*section*

Is up to the next **nroff/troff** section heading defined by the **sect=** option.

*motion*

Is up to the character or other text object as specified by a motion specifier, including pattern searches.

## Examples

**2cw**

Change the next two words.

**d}** Delete up to next paragraph.

**d^** Delete back to beginning of line.

**5yy**

Copy the next five lines.

**y]]**

Copy up to the next section.

**cG** Change to the end of the edit buffer.

More commands and examples may be found in the section “Changing and deleting text” on page 687.

## Visual mode (**vim** only)

**vim** provides an additional facility, “visual mode.” This allows you to highlight blocks of text which then become the object of edit commands such as deletion or saving (yanking). Graphical versions of **vim** allow you to use the mouse to highlight text in a similar fashion. See the **vim** help file *visual.txt* for the full story.

**v** Select text in visual mode one character at a time.

**V** Select text in visual mode one line at a time.

**Ctrl-V**

Select text in visual mode in blocks.

## Status-Line Commands

Most commands are not echoed on the screen as you input them. However, the status line at the bottom of the screen is used to edit these commands:

- / Search forward for a pattern.
- ? Search backward for a pattern.
- :
- Invoke an **ex** command.
- ! Invoke a Unix command that takes as its input an object in the buffer and replaces it with output from the command. You type a motion command after the ! to describe what should be passed to the Unix command. The command itself is entered on the status line.

Commands that are entered on the status line must be entered by pressing the Enter key. In addition, error messages and output from the **Ctrl-G** command are displayed on the status line.

## vi Commands

**vi** supplies a large set of single-key commands when in command mode. **vim** supplies additional multi-key commands.

### Movement Commands

Some versions of **vi** do not recognize extended keyboard keys (e.g., arrow keys, Page Up, Page Down, Home, Insert, and Delete); some do. All, however, recognize the keys in this section. Many users of **vi** prefer to use these keys, as it helps them keep their fingers on the home row of the keyboard. A number preceding a command repeats the movement. Movement commands are also used after an operator. The operator works on the text that is moved.

#### Character

| Command    | Action                                                                             |
|------------|------------------------------------------------------------------------------------|
| h, j, k, l | Left, down, up, right ( $\leftarrow$ , $\downarrow$ , $\uparrow$ , $\rightarrow$ ) |
| Space      | Right                                                                              |
| Backspace  | Left                                                                               |
| Ctrl-H     | Left                                                                               |

#### Text

| Command | Action                                                                        |
|---------|-------------------------------------------------------------------------------|
| w, b    | Forward, backward by “word” (letters, numbers, and underscore make up words). |
| W, B    | Forward, backward by “WORD” (only whitespace separates items).                |
| e       | End of word.                                                                  |
| E       | End of WORD.                                                                  |
| ge      | End of previous word. {vim}                                                   |

| Command | Action                                |
|---------|---------------------------------------|
| gE      | End of previous WORD. {vim}           |
| ), (    | Beginning of next, current sentence.  |
| }, {    | Beginning of next, current paragraph. |
| ]], [[  | Beginning of next, current section.   |
| ]], [ ] | End of next, current section. {vim}   |

## Lines

Long lines in a file may show up on the screen as multiple lines. (They *wrap* around from one screen line to the next.) While most commands work on the lines as defined in the file, a few commands work on lines as they appear on the screen. The **vim** option **wrap** allows you to control how long lines are displayed.

| Command | Action                                           |
|---------|--------------------------------------------------|
| 0, \$   | First, last position of current line.            |
| ^, _    | First nonblank character of current line.        |
| +, -    | First nonblank character of next, previous line. |
| Enter   | First nonblank character of next line.           |
| num     | Column <i>num</i> of current line.               |
| g0, g\$ | First, last position of screen line. {vim}       |
| g^      | First nonblank character of screen line. {vim}   |
| gm      | Middle of screen line. {vim}                     |
| gk, gj  | Move up, down one screen line. {vim}             |
| H       | Top line of screen (Home position).              |
| M       | Middle line of screen.                           |
| L       | Last line of screen.                             |
| numH    | <i>num</i> lines after top line.                 |
| numL    | <i>num</i> lines before last line.               |

## Screens

| Command        | Action                                                                  |
|----------------|-------------------------------------------------------------------------|
| Ctrl-F, Ctrl-B | Scroll forward, backward one screen.                                    |
| Ctrl-D, Ctrl-U | Scroll down, up one-half screen.                                        |
| Ctrl-E, Ctrl-Y | Show one more line at bottom, top of screen.                            |
| z Enter        | Reposition line with cursor to top of screen.                           |
| z.             | Reposition line with cursor to middle of screen.                        |
| z-             | Reposition line with cursor to bottom of screen.                        |
| Ctrl-L         | Redraw screen (without scrolling).                                      |
| Ctrl-R         | vi: Redraw screen (without scrolling).<br>vim: Redo last undone change. |

## Searches

| Command       | Action                                                                                                           |
|---------------|------------------------------------------------------------------------------------------------------------------|
| /pattern      | Search forward for <i>pattern</i> . End with Enter.                                                              |
| /pattern/+num | Go to line <i>num</i> after <i>pattern</i> .                                                                     |
| ?pattern      | Search backward for <i>pattern</i> . End with Enter.                                                             |
| ?pattern?-num | Go to line <i>num</i> before <i>pattern</i> .                                                                    |
| :noh          | Suspend search highlighting until next search. {vim}.                                                            |
| n             | Repeat previous search.                                                                                          |
| N             | Repeat search in opposite direction.                                                                             |
| /             | Repeat previous search forward.                                                                                  |
| ?             | Repeat previous search backward.                                                                                 |
| *             | Search forward for word under cursor. Matches only exact words. {vim}                                            |
| #             | Search backward for word under cursor. Matches only exact words. {vim}                                           |
| g*            | Search backward for word under cursor. Matches the characters of this word when embedded in a longer word. {vim} |
| g#            | Search backward for word under cursor. Matches the characters of this word when embedded in a longer word. {vim} |
| %             | Find match of current parenthesis, brace, or bracket.                                                            |
| fx            | Move cursor forward to <i>x</i> on current line.                                                                 |
| Fx            | Move cursor backward to <i>x</i> on current line.                                                                |
| tx            | Move cursor forward to character before <i>x</i> in current line.                                                |
| Tx            | Move cursor backward to character after <i>x</i> in current line.                                                |
| ,             | Reverse search direction of last f, F, t, or T.                                                                  |
| ;             | Repeat last f, F, t, or T.                                                                                       |

## Line numbering

| Command | Action                            |
|---------|-----------------------------------|
| Ctrl-G  | Display current line number.      |
| gg      | Move to first line in file. {vim} |
| numG    | Move to line number <i>num</i> .  |
| G       | Move to last line in file.        |
| :num    | Move to line number <i>num</i> .  |

## Marks

| Command | Action                                                                |
|---------|-----------------------------------------------------------------------|
| mx      | Place mark <i>x</i> at current position.                              |
| 'x      | (backquote) Move cursor to mark <i>x</i> .                            |
| 'x      | (apostrophe) Move to start of line containing <i>x</i> .              |
| ``      | (backquotes) Return to position before most recent jump.              |
| ''      | (apostrophes) Like preceding, but return to start of line.            |
| '''     | (apostrophe quote) Move to position when last editing the file. {vim} |

| Command | Action                                                                                           |
|---------|--------------------------------------------------------------------------------------------------|
| `[, `]  | (backquote bracket) Move to beginning/end of previous text operation. {vim}                      |
| '[, ']  | (apostrophe bracket) Like preceding, but return to start of line where operation occurred. {vim} |
| `.      | (backquote period) Move to last change in file. {vim}                                            |
| '.      | (apostrophe period) Like preceding, but return to start of line. {vim}                           |
| `0      | Position where you last exited vim. {vim}                                                        |
| :marks  | List active marks. {vim}                                                                         |

## Insert Commands

| Command | Action                             |
|---------|------------------------------------|
| a       | Append after cursor.               |
| A       | Append to end of line.             |
| c       | Begin change operation.            |
| C       | Change to end of line.             |
| gi      | Insert at beginning of line. {vim} |
| i       | Insert before cursor.              |
| l       | Insert at beginning of line.       |
| o       | Open a line below cursor.          |
| O       | Open a line above cursor.          |
| R       | Begin overwriting text.            |
| s       | Substitute a character.            |
| S       | Substitute entire line.            |
| ESC     | Terminate insert mode.             |

The following commands work in insert mode.

| Command   | Action                                                                     |
|-----------|----------------------------------------------------------------------------|
| Backspace | Delete previous character.                                                 |
| Delete    | Delete current character.                                                  |
| Tab       | Insert a tab.                                                              |
| Ctrl-A    | Repeat last insertion. {vim}                                               |
| Ctrl-D    | Shift line left to previous shift width. {vim}                             |
| Ctrl-E    | Insert character found just below cursor. {vim}                            |
| Ctrl-H    | Delete previous character (same as Backspace).                             |
| Ctrl-I    | Insert a tab.                                                              |
| Ctrl-K    | Begin insertion of multi-keystroke character.                              |
| Ctrl-N    | Insert next completion of the pattern to the left of the cursor. {vim}     |
| Ctrl-P    | Insert previous completion of the pattern to the left of the cursor. {vim} |
| Ctrl-T    | Shift line right to next shift width. {vim}                                |
| Ctrl-U    | Delete current line.                                                       |
| Ctrl-V    | Insert next character verbatim.                                            |

| Command | Action                                          |
|---------|-------------------------------------------------|
| Ctrl-W  | Delete previous word.                           |
| Ctrl-Y  | Insert character found just above cursor. {vim} |
| Ctrl-[  | (Escape) Terminate insert mode.                 |

Some of the control characters listed in the previous table are set by `stty`. Your terminal settings may differ.

## Edit Commands

Recall that `c`, `d`, and `y` are the basic editing operators.

### Changing and deleting text

The following table is not exhaustive, but it illustrates the most common operations.

| Command            | Action                                                                   |
|--------------------|--------------------------------------------------------------------------|
| <code>cw</code>    | Change word.                                                             |
| <code>cc</code>    | Change line.                                                             |
| <code>c\$</code>   | Change text from current position to end of line.                        |
| <code>c</code>     | Same as <code>c\$</code> .                                               |
| <code>dd</code>    | Delete current line.                                                     |
| <code>numdd</code> | Delete <code>num</code> lines.                                           |
| <code>d\$</code>   | Delete text from current position to end of line.                        |
| <code>D</code>     | Same as <code>d\$</code> .                                               |
| <code>dw</code>    | Delete a word.                                                           |
| <code>d}</code>    | Delete up to next paragraph.                                             |
| <code>d^</code>    | Delete back to beginning of line.                                        |
| <code>d/pat</code> | Delete up to first occurrence of pattern.                                |
| <code>dn</code>    | Delete up to next occurrence of pattern.                                 |
| <code>dfa</code>   | Delete up to and including <code>a</code> on current line.               |
| <code>dta</code>   | Delete up to (but not including) <code>a</code> on current line.         |
| <code>dL</code>    | Delete up to last line on screen.                                        |
| <code>dG</code>    | Delete to end of file.                                                   |
| <code>ggap</code>  | Reformat current paragraph to <code>textwidth</code> . {vim}             |
| <code>g~w</code>   | Switch case of word. {vim}                                               |
| <code>guw</code>   | Change word to lowercase. {vim}                                          |
| <code>gUw</code>   | Change word to uppercase. {vim}                                          |
| <code>p</code>     | Insert last deleted or yanked text after cursor.                         |
| <code>gp</code>    | Same as <code>p</code> , but leave cursor at end of inserted text. {vim} |
| <code>lp</code>    | Same as <code>p</code> , but match current indentation. {vim}            |
| <code>[p</code>    | Same as <code>P</code> , but match current indentation. {vim}            |
| <code>P</code>     | Insert last deleted or yanked text before cursor.                        |
| <code>gP</code>    | Same as <code>P</code> , but leave cursor at end of inserted text. {vim} |

| Command       | Action                                                                                   |
|---------------|------------------------------------------------------------------------------------------|
| r             | Replace character with <i>x</i> .                                                        |
| R <i>text</i> | Replace with new <i>text</i> (overwrite), beginning at cursor. Escape ends replace mode. |
| s             | Substitute character.                                                                    |
| 4s            | Substitute four characters.                                                              |
| S             | Substitute entire line.                                                                  |
| u             | Undo last change.                                                                        |
| Ctrl-R        | Redo last change. {vim}                                                                  |
| U             | Restore current line.                                                                    |
| x             | Delete current cursor position.                                                          |
| X             | Delete back one character.                                                               |
| 5X            | Delete previous five characters.                                                         |
| .             | Repeat last change.                                                                      |
| ~             | Reverse case and move cursor right.                                                      |
| Ctrl-A        | Increment number under cursor. {vim}                                                     |
| Ctrl-X        | Decrement number under cursor. {vim}                                                     |

## Copying and moving

Register names are the letters **a–z**. Uppercase names append text to the corresponding register.

| Command | Action                                                      |
|---------|-------------------------------------------------------------|
| Y       | Copy current line.                                          |
| yy      | Copy current line.                                          |
| "xyy    | Copy current line to register <i>x</i> .                    |
| ye      | Copy text to end of word.                                   |
| yw      | Like <b>ye</b> , but include the whitespace after the word. |
| y\$     | Copy rest of line.                                          |
| "xdd    | Delete current line into register <i>x</i> .                |
| "xd     | Delete into register <i>x</i> .                             |
| "xp     | Put contents of register <i>x</i> .                         |
| y]]     | Copy up to next section heading.                            |
| ye      | Copy to end of word.                                        |
| "xp     | Put contents of register <i>x</i> .                         |
| J       | Join current line to next line.                             |
| gJ      | Same as <b>J</b> , but without inserting a space. {vim}     |
| :j      | Same as <b>J</b> .                                          |
| :jl     | Same as <b>gJ</b> .                                         |

## Saving and Exiting

Writing a file means overwriting the file with the current text.

| Command              | Action                                                      |
|----------------------|-------------------------------------------------------------|
| ZZ                   | Quit vi, writing the file only if changes were made.        |
| :x                   | Same as ZZ.                                                 |
| :wq                  | Write file and quit.                                        |
| :w                   | Write file.                                                 |
| :w <i>file</i>       | Save copy to <i>file</i> .                                  |
| :n,mw <i>file</i>    | Write lines <i>n</i> to <i>m</i> to new <i>file</i> .       |
| :n,mw >> <i>file</i> | Append lines <i>n</i> to <i>m</i> to existing <i>file</i> . |
| :w!                  | Write file (overriding protection).                         |
| :w! <i>file</i>      | Overwrite <i>file</i> with current text.                    |
| :w %.new             | Write current buffer named <i>file</i> as <i>file.new</i> . |
| :q                   | Quit vi (fails if changes were made).                       |
| :q!                  | Quit vi (discarding edits).                                 |
| Q                    | Quit vi and invoke ex.                                      |
| :vi                  | Return to vi after Q command.                               |
| %                    | Replaced with current filename in editing commands.         |
| #                    | Replaced with alternate filename in editing commands.       |

## Accessing Multiple Files

| Command                     | Action                                                     |
|-----------------------------|------------------------------------------------------------|
| :e <i>file</i>              | Edit another <i>file</i> ; current file becomes alternate. |
| :e!                         | Return to version of current file at time of last write.   |
| :e + <i>file</i>            | Begin editing at end of <i>file</i> .                      |
| :e + <i>num</i> <i>file</i> | Open <i>file</i> at line <i>num</i> .                      |
| :e #                        | Open to previous position in alternate file.               |
| :ta <i>tag</i>              | Edit file at location <i>tag</i> .                         |
| :n                          | Edit next file in the list of files.                       |
| :n!                         | Force next file.                                           |
| :n <i>files</i>             | Specify new list of <i>files</i> .                         |
| :rewind                     | Edit first file in the list.                               |
| Ctrl-G                      | Show current file and line number.                         |
| :args                       | Display list of files to be edited.                        |
| :prev                       | Edit previous file in the list of files.                   |

## Window Commands

The following table lists common commands for controlling windows in **vim**. See also the **split**, **vsplit**, and **resize** commands in the “Alphabetical Summary of ex Commands” on page 697. For brevity, control characters are marked in the following list by ^.

| Command                  | Action                                                                         |
|--------------------------|--------------------------------------------------------------------------------|
| :new                     | Open a new window.                                                             |
| :new <i>file</i>         | Open <i>file</i> in a new window.                                              |
| :sp [ <i>file</i> ]      | Split the current window. With <i>file</i> , edit that file in the new window. |
| :sv [ <i>file</i> ]      | Same as :sp, but make new window read-only.                                    |
| :sn [ <i>file</i> ]      | Edit next file in file list in new window.                                     |
| :vsp [ <i>file</i> ]     | Like :sp, but split vertically instead of horizontally.                        |
| :clo                     | Close current window.                                                          |
| :hid                     | Hide current window, unless it is the only visible window.                     |
| :on                      | Make current window the only visible one.                                      |
| :res <i>num</i>          | Resize window to <i>num</i> lines.                                             |
| :wa                      | Write all changed buffers to file.                                             |
| :qa                      | Close all buffers and exit.                                                    |
| $\wedge W s$             | Same as :sp.                                                                   |
| $\wedge W n$             | Same as :new.                                                                  |
| $\wedge W ^$             | Open new window with alternate (previously edited) file.                       |
| $\wedge W c$             | Same as :clo.                                                                  |
| $\wedge W o$             | Same as :only.                                                                 |
| $\wedge W j, \wedge W k$ | Move cursor to next/previous window.                                           |
| $\wedge W p$             | Move cursor to previous window.                                                |
| $\wedge W h, \wedge W l$ | Move cursor to window on left/right.                                           |
| $\wedge W t, \wedge W b$ | Move cursor to window on top/bottom of screen.                                 |
| $\wedge W K, \wedge W B$ | Move current window to top/bottom of screen.                                   |
| $\wedge W H, \wedge W L$ | Move current window to far left/right of screen.                               |
| $\wedge W r, \wedge W R$ | Rotate windows down/up.                                                        |
| $\wedge W +, \wedge W -$ | Increase/decrease current window size.                                         |
| $\wedge W =$             | Make all windows same height.                                                  |

## Interacting with the System

| Command                     | Action                                                                               |
|-----------------------------|--------------------------------------------------------------------------------------|
| :r <i>file</i>              | Read in contents of <i>file</i> after cursor.                                        |
| :r ! <i>command</i>         | Read in output from <i>command</i> after current line.                               |
| :num! <i>command</i>        | Like above, but place after line <i>num</i> (0 for top of file).                     |
| :! <i>command</i>           | Run <i>command</i> , then return.                                                    |
| ! <i>motion command</i>     | Send the text covered by <i>motion</i> to Unix <i>command</i> ; replace with output. |
| :n,m! <i>command</i>        | Send lines <i>n-m</i> to <i>command</i> ; replace with output.                       |
| <i>num</i> ! <i>command</i> | Send <i>num</i> lines to Unix <i>command</i> ; replace with output.                  |
| ::!                         | Repeat last system command.                                                          |
| :sh                         | Create subshell; return to file with EOF.                                            |
| Ctrl-Z                      | Suspend editor, resume with fg.                                                      |
| :so <i>file</i>             | Read and execute ex commands from <i>file</i> .                                      |

## Macros

| Command                      | Action                                                                                                                 |
|------------------------------|------------------------------------------------------------------------------------------------------------------------|
| :ab <i>in out</i>            | Use <i>in</i> as abbreviation for <i>out</i> in insert mode.                                                           |
| :unab <i>in</i>              | Remove abbreviation for <i>in</i> .                                                                                    |
| :ab                          | List abbreviations.                                                                                                    |
| :map <i>string sequence</i>  | Map characters <i>string</i> as <i>sequence</i> of commands. Use #1, #2, etc., for the function keys.                  |
| :unmap <i>string</i>         | Remove map for characters <i>string</i> .                                                                              |
| :map                         | List character strings that are mapped.                                                                                |
| :map! <i>string sequence</i> | Map characters <i>string</i> to input mode <i>sequence</i> .                                                           |
| :unmap! <i>string</i>        | Remove input mode map (you may need to quote the character with <b>Ctrl-V</b> ).                                       |
| :map!                        | List character strings that are mapped for input mode.                                                                 |
| qx                           | Record typed characters into register specified by letter <i>x</i> . If letter is uppercase, append to register. {vim} |
| q                            | Stop recording. {vim}                                                                                                  |
| @ <i>x</i>                   | Execute the register specified by letter <i>x</i> . Use @@ to repeat the last @ command.                               |

In **vi**, the following characters are unused in command mode and can be mapped as user-defined commands:

### Letters

g K q V v

### Control keys

^A ^K ^O ^W ^X ^\_ ^\

### Symbols

\_ \* \ = #



The = is used by vi if Lisp mode is set. Different versions of vi may use some of these characters, so test them before using.

vim does not use ^K, ^\_, \_, or \.

## Miscellaneous Commands

| Command | Action                                                                                             |
|---------|----------------------------------------------------------------------------------------------------|
| <       | Shift text described by following motion command left by one shiftwidth. {vim}                     |
| >       | Shift text described by following motion command right by one shiftwidth. {vim}                    |
| <<      | Shift line left one shift width (default is eight spaces).                                         |
| >>      | Shift line right one shift width (default is eight spaces).                                        |
| >}      | Shift right to end of paragraph.                                                                   |
| <%      | Shift left until matching parenthesis, brace, or bracket. (Cursor must be on the matching symbol.) |
| ==      | Indent line in C-style, or using program specified in equalprg option. {vim}                       |

| Command    | Action                                                                                  |
|------------|-----------------------------------------------------------------------------------------|
| <b>g</b>   | Start many multiple character commands in <b>vim</b> .                                  |
| <b>K</b>   | Look up word under cursor in manpages (or program defined in <b>keywordprg</b> ). {vim} |
| <b>^0</b>  | Return to previous jump. {vim}                                                          |
| <b>q</b>   | Record keystrokes. {vim}                                                                |
| <b>^Q</b>  | Same as <b>^V</b> . {vim} (On some terminals, resume data flow.)                        |
| <b>^T</b>  | Return to the previous location in the tag stack. {vim}                                 |
| <b>^]</b>  | Perform a tag lookup on the text under the cursor.                                      |
| <b>^\\</b> | Enter <b>ex</b> line-editing mode.                                                      |
| <b>^^</b>  | (Caret key with Ctrl key pressed) Return to previously edited file.                     |

## vi Configuration

This section describes the following:

- The **:set** command
- Options available with **:set**
- Sample *.exrc* file

### The **:set** Command

The **:set** command allows you to specify options that change characteristics of your editing environment. Options may be put in the *~/.exrc* file or set during a **vi** session.

The colon does not need to be typed if the command is put in *.exrc*:

| Command             | Action                                                        |
|---------------------|---------------------------------------------------------------|
| <b>:set x</b>       | Enable Boolean option <i>x</i> , show value of other options. |
| <b>:set nox</b>     | Disable option <i>x</i> .                                     |
| <b>:set x=value</b> | Give <i>value</i> to option <i>x</i> .                        |
| <b>:set</b>         | Show changed options.                                         |
| <b>:set all</b>     | Show all options.                                             |
| <b>:set x?</b>      | Show value of option <i>x</i> .                               |

### Options Used by **:set**

Table 9-1 contains brief descriptions of the important **set** command options. In the first column, options are listed in alphabetical order; if the option can be abbreviated, that abbreviation is shown in parentheses. The second column shows the default setting. The last column describes what the option does, when enabled.

This table lists **set** options for **vi**, with the addition of important **vim** options. Other versions of **vi** may have more or fewer or different options. See your local documentation, or use **:set all** to see the full list. Options that receive a value are marked with an =.

Table 9-1. `:set` options

| Option                         | Default                     | Description                                                                                                                                                                                                 |
|--------------------------------|-----------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>autoindent (ai)</code>   | <code>noai</code>           | In insert mode, indent each line to the same level as the line above or below. Use with the <code>shiftwidth</code> option.                                                                                 |
| <code>autoprint (ap)</code>    | <code>ap</code>             | Display changes after each editor command. (For global replacement, display last replacement.)                                                                                                              |
| <code>autowrite (aw)</code>    | <code>noaw</code>           | Automatically write (save) the file if changed before opening another file with a command such as <code>:n</code> , or before giving Unix command with <code>!.</code>                                      |
| <code>background (bg)</code>   |                             | Describe the background so the editor can choose appropriate highlighting colors. Default value of <code>dark</code> or <code>light</code> depends on the environment in which the editor is invoked. {vim} |
| <code>backup (bk)</code>       | <code>nobackup</code>       | Create a backup file when overwriting an existing file. {vim}                                                                                                                                               |
| <code>backupdir= (bdir)</code> | <code>.,~/tmp/,~/</code>    | Name directories in which to store backup files if possible. The list of directories is comma-separated and in order of preference. {vim}                                                                   |
| <code>beautify (bf)</code>     | <code>nobf</code>           | Ignore all control characters during input (except tab, newline, or formfeed).                                                                                                                              |
| <code>backupext= (bex)</code>  | <code>~</code>              | String to append to filenames for backup files. {vim}                                                                                                                                                       |
| <code>cindent (cin)</code>     | <code>nocindent</code>      | In insert mode, indent each line relative to the one above it, as is appropriate for C or C++ code. {vim}                                                                                                   |
| <code>compatible (cp)</code>   | <code>cp</code>             | Make vim behave more like vi. Default is <code>nocp</code> when a <code>~/.vimrc</code> file is found. {vim}                                                                                                |
| <code>directory (dir)</code>   | <code>/tmp</code>           | Name of directory in which ex/vi stores buffer files. (Directory must be writable.) This can be a comma-separated list for vim.                                                                             |
| <code>edcompatible</code>      | <code>noedcompatible</code> | Remember the flags used with the most recent substitute command (global, confirming) and use them for the next substitute command. Despite the name, no version of ed actually does this.                   |
| <code>equalprg= (ep)</code>    |                             | Use the specified program for the = command. When the option is blank (the default), the key invokes the internal C indentation function or the value of the <code>indentexpr</code> option. {vim}          |
| <code>errorbells (eb)</code>   | <code>errorbells</code>     | Sound bell when an error occurs.                                                                                                                                                                            |
| <code>exrc (ex)</code>         | <code>noexrc</code>         | Allow the execution of .exrc files that reside outside the user's home directory.                                                                                                                           |
| <code>formatprg= (fp)</code>   |                             | The <code>gq</code> command will invoke the named external program to format text. It will call internal formatting functions when this option is empty (the default). {vim}                                |
| <code>gdefault (gd)</code>     | <code>nogdefault</code>     | Set the g flag on for substitutions by default. {vim}                                                                                                                                                       |
| <code>hardtabs= (ht)</code>    | <code>8</code>              | Define boundaries for terminal hardware tabs.                                                                                                                                                               |
| <code>hidden (hid)</code>      | <code>nohidden</code>       | Hide buffers rather than unload them when they are abandoned. {vim}                                                                                                                                         |
| <code>hlsearch (hls)</code>    | <code>hlsearch</code>       | Highlight all matches of most recent search pattern. Use <code>:nohlsearch</code> to remove highlighting. {vim}                                                                                             |
| <code>history= (hi)</code>     | <code>20</code>             | Number of ex commands to store in the history table. {vim}                                                                                                                                                  |
| <code>ignorecase (ic)</code>   | <code>noic</code>           | Disregard case during a search.                                                                                                                                                                             |

Table 9-1. *:set options (continued)*

| Option                   | Default                       | Description                                                                                                                                                                                                          |
|--------------------------|-------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>incsearch (is)</b>    | <b>noincsearch</b>            | Highlight matches to a search pattern as it is typed. {vim}                                                                                                                                                          |
| <b>lisp</b>              | <b>nolisp</b>                 | Insert indents in appropriate Lisp format. ( ), { }, [ [, and ] ] are modified to have meaning for Lisp.                                                                                                             |
| <b>list</b>              | <b>nolist</b>                 | Print tabs as ^I; mark ends of lines with \$. (Use list to tell if end character is a tab or a space.)                                                                                                               |
| <b>magic</b>             | <b>magic</b>                  | Wildcard characters; . (dot), * (asterisk), and [ ] (brackets) have special meaning in patterns.                                                                                                                     |
| <b>mesg</b>              | <b>mesg</b>                   | Permit system messages to display on terminal while editing in vi.                                                                                                                                                   |
| <b>mousehide (mh)</b>    | <b>mousehide</b>              | When characters are typed, hide the mouse pointer. {vim}                                                                                                                                                             |
| <b>novice</b>            | <b>nonovice</b>               | Require the use of long ex command names, such as <b>copy</b> or <b>read</b> .                                                                                                                                       |
| <b>number (nu)</b>       | <b>nonu</b>                   | Display line numbers on left of screen during editing session.                                                                                                                                                       |
| <b>open</b>              | <b>open</b>                   | Allow entry to <b>open</b> or <b>visual</b> mode from <b>ex</b> . Although not in vim, this option has traditionally been in vi, and may be in your version of vi.                                                   |
| <b>optimize (opt)</b>    | <b>noopt</b>                  | Abolish carriage returns at the end of lines when printing multiple lines; speed output on dumb terminals when printing lines with leading whitespace (spaces or tabs).                                              |
| <b>paragraphs (para)</b> | <b>IPLPPPQPP Llpplpipnpbp</b> | Define paragraph delimiters for movement by { or }. The pairs of characters in the value are the names of troff macros that begin paragraphs.                                                                        |
| <b>paste</b>             | <b>nopaste</b>                | Change the defaults of various options to make pasting text into a terminal window work better. All options are returned to their original value when the <b>paste</b> option is reset. {vim}                        |
| <b>prompt</b>            | <b>prompt</b>                 | Display the ex prompt ( : ) when vi 's Q command is given.                                                                                                                                                           |
| <b>readonly (ro)</b>     | <b>noro</b>                   | Any writes (saves) of a file fail unless you use ! after the write (works with w, ZZ, or autowrite).                                                                                                                 |
| <b>redraw (re)</b>       |                               | vi redraws the screen whenever edits are made. <b>nore-draw</b> is useful at slow speeds on a dumb terminal: the screen isn't fully updated until you press Escape. Default depends on line speed and terminal type. |
| <b>remap</b>             | <b>remap</b>                  | Allow nested map sequences.                                                                                                                                                                                          |
| <b>report=</b>           | <b>5</b>                      | Display a message on the status line whenever you make an edit that affects at least a certain number of lines. For example, 6dd reports the message "6 lines deleted."                                              |
| <b>ruler (ru)</b>        | <b>ruler</b>                  | Show line and column numbers for the current cursor position. {vim}                                                                                                                                                  |
| <b>scroll=</b>           | <b>[1/2 window]</b>           | Number of lines to scroll with ^D and ^U commands.                                                                                                                                                                   |
| <b>sections= (sect)</b>  | <b>SHNHH HUnhsh+c</b>         | Define section delimiters for [ [ and ] ] movement. The pairs of characters in the value are the names of troff macros that begin sections.                                                                          |
| <b>shell= (sh)</b>       | <b>/bin/sh</b>                | Pathname of shell used for shell escape (!) and shell command (:sh). Default value is derived from shell environment, which varies on different systems.                                                             |
| <b>shiftwidth= (sw)</b>  | <b>8</b>                      | Define number of spaces used when the indent is increased or decreased.                                                                                                                                              |

Table 9-1. *:set* options (continued)

| Option                  | Default                   | Description                                                                                                                                                                       |
|-------------------------|---------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>showmatch (sm)</b>   | <b>nosm</b>               | In vi, when ) or } is entered, cursor moves briefly to matching ( or {. (If no match, rings the error message bell.) Very useful for programming.                                 |
| <b>showmode</b>         | <b>noshowmode</b>         | In insert mode, display a message on the prompt line indicating the type of insert you are making. For example, "OPEN MODE" or "APPEND MODE."                                     |
| <b>slowopen (slow)</b>  |                           | Hold off display during insert. Default depends on line speed and terminal type.                                                                                                  |
| <b>smartcase (scs)</b>  | <b>nosmartcase</b>        | Override the <b>ignorecase</b> option when a search pattern contains uppercase characters. {vim}                                                                                  |
| <b>tabstop= (ts)</b>    | <b>8</b>                  | Define number of spaces a tab indents during editing session. (Printer still uses system tab of 8.)                                                                               |
| <b>taglength= (tl)</b>  | <b>0</b>                  | Define number of characters that are significant for tags. Default (zero) means that all characters are significant.                                                              |
| <b>tags=</b>            | <b>tags /usr/lib/tags</b> | Define pathname of files containing tags. (See the Unix <b>ctags</b> command.) (By default, vi searches the file <b>tags</b> in the current directory and <b>/usr/lib/tags</b> .) |
| <b>tagstack</b>         | <b>tagstack</b>           | Enable stacking of tag locations on a stack. (Solaris vi and vim.)                                                                                                                |
| <b>term=</b>            |                           | Set terminal type.                                                                                                                                                                |
| <b>terse</b>            | <b>noterse</b>            | Display shorter error messages.                                                                                                                                                   |
| <b>textwidth= (tw)</b>  | <b>0</b>                  | The maximum width of text to be inserted; longer lines are broken after whitespace. Default (zero) disables this feature, in which case <b>wrapmargin</b> is used. {vim}          |
| <b>timeout (to)</b>     | <b>timeout</b>            | Keyboard maps timeout after 1 second. <sup>a</sup>                                                                                                                                |
| <b>timeoutlen= (tm)</b> | <b>1000</b>               | Number of milliseconds after which keyboard maps timeout. Default value of 1000 provides traditional vi behavior. {vim}                                                           |
| <b>ttytype=</b>         |                           | Set terminal type. This is just another name for <b>term</b> .                                                                                                                    |
| <b>undolevels= (ul)</b> | <b>1000</b>               | Number of changes that can be undone. {vim}                                                                                                                                       |
| <b>warn</b>             | <b>warn</b>               | Display the warning message, "No write since last change."                                                                                                                        |
| <b>window (w)</b>       |                           | Show a certain number of lines of the file on the screen. Default depends on line speed and terminal type.                                                                        |
| <b>wrap</b>             |                           | When on, long lines wrap on the screen. When off, only the first part of the line is displayed. {vim}                                                                             |
| <b>wrapmargin (wm)</b>  | <b>0</b>                  | Define right margin. If greater than zero, vi automatically inserts carriage returns to break lines.                                                                              |
| <b>wrapscan (ws)</b>    | <b>ws</b>                 | Searches wrap around either end of file.                                                                                                                                          |
| <b>writeany (wa)</b>    | <b>nowa</b>               | Allow saving to any file.                                                                                                                                                         |
| <b>writebackup (wb)</b> | <b>wb</b>                 | Back up files before attempting to overwrite them. Remove the backup when the file has been successfully written, unless the <b>backup</b> option is set. {vim}                   |

<sup>a</sup> When you have mappings of several keys (for example, `:map zzz 3dw`), you probably want to use `notimeout`. Otherwise, you need to type `zzz` within 1 second. When you have an insert mode mapping for a cursor key (for example, `:map! ^[OB ^[ja`), you should use `timeout`. Otherwise, vi won't react to Escape until you type another key.

## Sample .exrc File

The following lines of code are an example of a customized `.exrc` file:

```
set nowrapscan " Searches don't wrap at end of file
set wrapmargin=7 " Wrap text at 7 columns from right margin
set sections=SeAhBhChDh nomesg " Set troff macros, disallow message
map q :w^M:n^M " Alias to move to next file
map v dwElp " Move a word
ab ORA O'Reilly Media, Inc. " Input shortcut
```



The `q` alias isn't needed for `vim`, which has the `:wn` command. The `v` alias would hide the `vim` command `v`, which enters character-at-a-time visual-mode operation.

## ex Basics

The `ex` line editor serves as the foundation for the screen editor `vi`. Commands in `ex` work on the current line or on a range of lines in a file. Most often, you use `ex` from within `vi`. In `vi`, `ex` commands are preceded by a colon and entered by pressing Enter.

You can also invoke `ex` on its own—from the command line—just as you would invoke `vi`. (You could execute an `ex` script this way.) You can also use the `vi` command `Q` to quit the `vi` editor and enter `ex`.

### Syntax of ex Commands

To enter an `ex` command from `vi`, type:

```
:[address] command [options]
```

An initial `:` indicates an `ex` command. As you type the command, it is echoed on the status line. Execute the command by pressing the Enter key. `address` is the line number or range of lines that are the object of `command`. `options` and `addresses` are described below. `ex` commands are described in the next section “Alphabetical Summary of ex Commands” on page 697.

You can exit `ex` in several ways:

- `:x` Exit (save changes and quit).
- `:q!` Quit without saving changes.
- `:vi` Switch to the `vi` editor on the current file.

### Addresses

If no address is given, the current line is the object of the command. If the address specifies a range of lines, the format is:

`x,y`

where `x` and `y` are the first and last addressed lines (`x` must precede `y` in the buffer). `x` and `y` may each be a line number or a symbol. Using `;` instead of `,` sets the current line to `x` before interpreting `y`. The notation `1,$` addresses all lines in the file, as does `%`.

## Address Symbols

| Symbol                 | Meaning                                                                 |
|------------------------|-------------------------------------------------------------------------|
| <code>1,\$</code>      | All lines in the file.                                                  |
| <code>x,y</code>       | Lines <i>x</i> through <i>y</i> .                                       |
| <code>x;y</code>       | Lines <i>x</i> through <i>y</i> , with current line reset to <i>x</i> . |
| <code>0</code>         | Top of file.                                                            |
| <code>.</code>         | Current line.                                                           |
| <code>num</code>       | Absolute line number <i>num</i> .                                       |
| <code>\$</code>        | Last line.                                                              |
| <code>%</code>         | All lines; same as <code>1,\$</code> .                                  |
| <code>x-n</code>       | <i>n</i> lines before <i>x</i> .                                        |
| <code>x+n</code>       | <i>n</i> lines after <i>x</i> .                                         |
| <code>-[num]</code>    | One or <i>num</i> lines previous.                                       |
| <code>+[num]</code>    | One or <i>num</i> lines ahead.                                          |
| <code>'x</code>        | Line marked with <i>x</i> .                                             |
| <code>"</code>         | Previous mark.                                                          |
| <code>/pattern/</code> | Forward to line matching <i>pattern</i> .                               |
| <code>?pattern?</code> | Backward to line matching <i>pattern</i> .                              |

See Chapter 7 for more information on using patterns.

## Options

- ! Indicates a variant form of the command, overriding the normal behavior. The ! must come immediately after the command.

### *count*

The number of times the command is to be repeated. Unlike in **vi** commands, *count* cannot precede the command, because a number preceding an **ex** command is treated as a line address. For example, **d3** deletes three lines beginning with the current line; **3d** deletes line 3.

### *file*

The name of a file that is affected by the command. % stands for the current file; # stands for the previous file.

## Alphabetical Summary of ex Commands

**ex** commands can be entered by specifying any unique abbreviation. In this listing, the full name appears in the margin, and the shortest possible abbreviation is used in the syntax line. Examples are assumed to be typed from **vi**, so they include the : prompt.

---

**abbreviate**      `ab [string text]`

Define *string* when typed to be translated into *text*. If *string* and *text* are not specified, list all current abbreviations.

## Examples

Note: ^M appears when you type ^V followed by Enter.

```
:ab ora O'Reilly Media, Inc.
:ab id Name:^MRank:^MPhone:
```

---

### append

[*address*] *a[!]*  
*text*

Append new *text* at specified *address*, or at present address if none is specified. Add a ! to toggle the **autoindent** setting that is used during input. That is, if **autoindent** was enabled, ! disables it. Enter new text after entering the command. Terminate input of new text by entering a line consisting of just a period.

#### Example

```
:a
Append this line
and this line too.
.
```

*Begin appending to current line.*

*Terminate input of text to append.*

---

### args

*ar*  
args *file* ...

Print the members of the argument list (files named on the command line), with the current argument printed in brackets ([ ]).

The second syntax is for **vim**, which allows you to reset the list of files to be edited.

---

### bdelete

[*num*] bd[!] [*num*]

Unload buffer *num* and remove it from the buffer list. Add a ! to force removal of an unsaved buffer. The buffer may also be specified by filename. If no buffer is specified, remove the current buffer.  
{vim}

---

### buffer

[*num*] b[!] [*num*]

Begin editing buffer *num* in the buffer list. Add a ! to force a switch from an unsaved buffer. The buffer may also be specified by filename. If no buffer is specified, continue editing the current buffer.  
{vim}

---

### buffers

buffers[!]

Print the members of the buffer list. Some buffers (e.g., deleted buffers) will not be listed. Add ! to show unlisted buffers. **ls** is another abbreviation for this command.  
{vim}

---

### cd

cd *dir*  
chdir *dir*

Change current directory within the editor to *dir*.

---

**center** [address] ce [width]  
Center line within the specified *width*. If *width* is not specified, use **textwidth**. {vim}

---

**change** [address] c[!]  
*text*  
Replace the specified lines with *text*. Add a ! to switch the **autoindent** setting during input of *text*. Terminate input by entering a line consisting of just a period.

---

**close** clo[!]  
Close current window unless it is the last window. If buffer in window is not open in another window, unload it from memory. This command will not close a buffer with unsaved changes, but you may add ! to hide it instead. {vim}

---

**copy** [address] co *destination*  
Copy the lines included in *address* to the specified *destination* address. The command t (short for “to”) is a synonym for **copy**.

#### Example

:1,10 co 50      Copy first 10 lines to just after line 50

---

**delete** [address] d [*register*]  
Delete the lines included in *address*. If *register* is specified, save or append the text to the named register. Register names are the lowercase letters a-z. Uppercase names append text to the corresponding register.

#### Examples

:/Part I/,/Part II/-1d      Delete to line above "Part II"  
:/main/+d                  Delete line below "main"  
:.,\$d x                    Delete from this line to last line into register x

---

**edit** [!] [+num] [*filename*]  
Begin editing on *filename*. If no *filename* is given, start over with a copy of the current file. Add a ! to edit the new file even if the current file has not been saved since the last change. With the +*num* argument, begin editing on line *num*. Or *num* may be a pattern, of the form /*pattern*.

#### Examples

:e file                  Edit file in current editing buffer  
:e +/^Index #            Edit alternate file at pattern match  
:e!                       Start over again on current file

---

**file** `f [filename]`  
Change the filename for the current buffer to *filename*. The next time the buffer is written, it will be written to file *filename*. When the name is changed, the buffer’s “not edited” flag is set, to indicate you are not editing an existing file. If the new filename is the same as a file that already exists on the disk, you will need to use :w! to overwrite the existing file. When specifying a filename, the % character can be used to indicate the current filename. A # can be used to indicate the alternate filename. If no *filename* is specified, print the current name and status of the buffer.

**Example**

`:f%.new`

---

**fold** `address fo`  
Fold the lines specified by *address*. A fold collapses several lines on the screen into one line, which later can be unfolded. It doesn’t affect the text of the file. {vim}

---

**foldclose** `[address] foldc[!]`  
Close folds in specified *address*, or at present address if none is specified. Add a ! to close more than one level of folds. {vim}

---

**foldopen** `[address] foldo[!]`  
Open folds in specified *address*, or at present address if none is specified. Add a ! to open more than one level of folds. {vim}

---

**global** `[address] g[!]/pattern/[commands]`  
Execute *commands* on all lines which contain *pattern* or, if *address* is specified, on all lines within that range. If *commands* are not specified, print all such lines. Add a ! to execute *commands* on all lines *not* containing *pattern*. See also v.

**Examples**

`:g/Unix/p` Print all lines containing "Unix"  
`:g/Name:/s/tom/Tom/` Change "tom" to "Tom" on all lines containing "Name:"

---

---

**hide** `hid`  
Close current window unless it is the last window, but do not remove the buffer from memory. This is a safe command to use on an unsaved buffer. {vim}

---

|               |                                                                                                                                                                                                                                                                                                            |                                                     |
|---------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------|
| <b>insert</b> | <code>[address] i[!]<br/>text</code>                                                                                                                                                                                                                                                                       |                                                     |
|               | .                                                                                                                                                                                                                                                                                                          |                                                     |
|               | Insert <i>text</i> at line before the specified <i>address</i> , or at present address if none is specified. Add a ! to switch the <b>autoindent</b> setting during input of <i>text</i> . Terminate input of new text by entering a line consisting of just a period.                                     |                                                     |
| <b>join</b>   | <code>[address] j[!][count]</code>                                                                                                                                                                                                                                                                         |                                                     |
|               | Place the text in the specified range on one line, with whitespace adjusted to provide two space characters after a period (.), no space characters before a ), and one space character otherwise. Add a ! to prevent whitespace adjustment.                                                               |                                                     |
|               | <b>Example</b>                                                                                                                                                                                                                                                                                             |                                                     |
|               | <code>:1,5j!</code>                                                                                                                                                                                                                                                                                        | <i>Join first five lines, preserving whitespace</i> |
| <b>jumps</b>  | <code>ju</code>                                                                                                                                                                                                                                                                                            |                                                     |
|               | Print jump list used with Ctrl-I and Ctrl-O commands. The jump list is a record of most movement commands that skip over multiple lines. It records the position of the cursor before each jump. {vim}                                                                                                     |                                                     |
| <b>k</b>      | <code>[address] k char</code>                                                                                                                                                                                                                                                                              |                                                     |
|               | Same as <b>mark</b> ; see <b>mark</b> , later in this list.                                                                                                                                                                                                                                                |                                                     |
| <b>left</b>   | <code>[address] le [count]</code>                                                                                                                                                                                                                                                                          |                                                     |
|               | Left-align lines specified by <i>address</i> , or current line if no address is specified. Indent lines by <i>count</i> spaces. {vim}                                                                                                                                                                      |                                                     |
| <b>list</b>   | <code>[address] l [count]</code>                                                                                                                                                                                                                                                                           |                                                     |
|               | Print the specified lines so that tabs display as ^I, and the ends of lines display as \$. l is like a temporary version of :set list.                                                                                                                                                                     |                                                     |
| <b>map</b>    | <code>map[!] [string commands]</code>                                                                                                                                                                                                                                                                      |                                                     |
|               | Define a keyboard macro named <i>string</i> as the specified sequence of <i>commands</i> . <i>string</i> is usually a single character, or the sequence #num, representing a function key on the keyboard. Use a ! to create a macro for input mode. With no arguments, list the currently defined macros. |                                                     |
|               | <b>Examples</b>                                                                                                                                                                                                                                                                                            |                                                     |
|               | <code>:map K dwwP</code>                                                                                                                                                                                                                                                                                   | <i>Transpose two words</i>                          |
|               | <code>:map q :w^M:n^M</code>                                                                                                                                                                                                                                                                               | <i>Write current file; go to next</i>               |
|               | <code>:map! + ^[bi(^[ea)</code>                                                                                                                                                                                                                                                                            | <i>Enclose previous word in parentheses</i>         |



vim has **K** and **q** commands, which the above aliases would hide.

---

|                |                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|----------------|---------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>mark</b>    | <code>[address] ma char</code>        | Mark the specified line with <i>char</i> , a single lowercase letter. Return later to the line with ' <b>x</b> ' (where <i>x</i> is the same as <i>char</i> ). vim also uses uppercase and numeric characters for marks. Lowercase letters work the same as in vi. Uppercase letters are associated with filenames and can be used between multiple files. Numbered marks, however, are maintained in a special <i>viminfo</i> file and cannot be set using this command. Same as <b>k</b> . |
| <b>marks</b>   | <code>marks [chars]</code>            | Print list of marks specified by <i>chars</i> , or all current marks if no chars specified. {vim}                                                                                                                                                                                                                                                                                                                                                                                            |
| <b>Example</b> |                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|                | <code>:marks abc</code>               | <i>Print marks a, b and c.</i>                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| <b>mkexrc</b>  | <code>mk[!] file</code>               | Create an <i>.exrc</i> file containing <b>set</b> commands for changed <b>ex</b> options and key mappings. This saves the current option settings, allowing you to restore them later.                                                                                                                                                                                                                                                                                                       |
| <b>move</b>    | <code>[address] m destination</code>  | Move the lines specified by <i>address</i> to the <i>destination</i> address.                                                                                                                                                                                                                                                                                                                                                                                                                |
| <b>Example</b> |                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|                | <code>:.,/Note/m /END/</code>         | <i>Move text block to after line containing "END"</i>                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| <b>new</b>     | <code>[count] new</code>              | Create a new window <i>count</i> lines high with an empty buffer. {vim}                                                                                                                                                                                                                                                                                                                                                                                                                      |
| <b>next</b>    | <code>num[!] [[+num] filelist]</code> | Edit the next file from the command-line argument list. Use <b>args</b> to list these files. If <i>filelist</i> is provided, replace the current argument list with <i>filelist</i> and begin editing on the first file. With the <i>+num</i> argument, begin editing on line <i>num</i> . Or <i>num</i> may be a pattern, of the form <i>/pattern</i> .                                                                                                                                     |
| <b>Example</b> |                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|                | <code>:n chap*</code>                 | <i>Start editing all "chapter" files</i>                                                                                                                                                                                                                                                                                                                                                                                                                                                     |

---

|                   |                                                                                                                                                                                                                                                                                                     |
|-------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>nohlsearch</b> | noh                                                                                                                                                                                                                                                                                                 |
|                   | Temporarily stop highlighting all matches to a search when using the <b>hlsearch</b> option. Highlighting is resumed with the next search.<br>{vim}                                                                                                                                                 |
| <b>number</b>     | [ <i>address</i> ] nu [ <i>count</i> ]                                                                                                                                                                                                                                                              |
|                   | Print each line specified by <i>address</i> , preceded by its buffer line number. Use # as an alternate abbreviation for <b>number</b> . <i>count</i> specifies the number of lines to show, starting with <i>address</i> .                                                                         |
| <b>only</b>       | on [!]                                                                                                                                                                                                                                                                                              |
|                   | Make the current window be the only one on the screen. Windows open on modified buffers are not removed from the screen (hidden), unless you also use the ! character. {vim}                                                                                                                        |
| <b>open</b>       | [ <i>address</i> ] o [/ <i>pattern</i> /]                                                                                                                                                                                                                                                           |
|                   | Enter open mode (vi) at the lines specified by <i>address</i> , or at the lines matching <i>pattern</i> . Exit open mode with Q. Open mode lets you use the regular vi commands, but only one line at a time. It can be useful on slow dial-up lines (or on very distant Internet ssh connections). |
| <b>preserve</b>   | pre                                                                                                                                                                                                                                                                                                 |
|                   | Save the current editor buffer as though the system were about to crash.                                                                                                                                                                                                                            |
| <b>previous</b>   | prev[!]                                                                                                                                                                                                                                                                                             |
|                   | Edit the previous file from the command-line argument list. {vim}                                                                                                                                                                                                                                   |
| <b>print</b>      | [ <i>address</i> ] p [ <i>count</i> ]                                                                                                                                                                                                                                                               |
|                   | Print the lines specified by <i>address</i> . <i>count</i> specifies the number of lines to print, starting with <i>address</i> . P is another abbreviation.                                                                                                                                        |
| <b>Example</b>    |                                                                                                                                                                                                                                                                                                     |
|                   | :100;+5p      Show line 100 and the next five lines                                                                                                                                                                                                                                                 |
| <b>put</b>        | [ <i>address</i> ] pu [ <i>char</i> ]                                                                                                                                                                                                                                                               |
|                   | Restore previously deleted or yanked lines from named register specified by <i>char</i> , to the line specified by <i>address</i> . If <i>char</i> is not specified, the last deleted or yanked text is restored.                                                                                   |
| <b>qall</b>       | qa[!]                                                                                                                                                                                                                                                                                               |
|                   | Close all windows and terminate current editing session. Use ! to discard changes made since the last save. {vim}                                                                                                                                                                                   |

---

---

|                                                                                                                                                                                                                                                                                                                                                                           |                                   |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------|
| <b>quit</b>                                                                                                                                                                                                                                                                                                                                                               | <code>q[!]</code>                 |
| Terminate current editing session. Use <code>!</code> to discard changes made since the last save. If the editing session includes additional files in the argument list that were never accessed, quit by typing <code>q!</code> or by typing <code>q</code> twice. <code>vim</code> only closes the editing window if there are still other windows open on the screen. |                                   |
| <b>read</b>                                                                                                                                                                                                                                                                                                                                                               | <code>[address] r filename</code> |
| Copy the text of <code>filename</code> after the line specified by <code>address</code> . If <code>filename</code> is not specified, the current filename is used.                                                                                                                                                                                                        |                                   |
| <b>Example</b>                                                                                                                                                                                                                                                                                                                                                            |                                   |
| <code>:or \$HOME/data</code> <i>Read file in at top of current file</i>                                                                                                                                                                                                                                                                                                   |                                   |
| <b>read</b>                                                                                                                                                                                                                                                                                                                                                               | <code>[address] r !command</code> |
| Read the output of shell <code>command</code> into the text after the line specified by <code>address</code> .                                                                                                                                                                                                                                                            |                                   |
| <b>Example</b>                                                                                                                                                                                                                                                                                                                                                            |                                   |
| <code>:\$r !spell %</code> <i>Place results of spell checking at end of file</i>                                                                                                                                                                                                                                                                                          |                                   |
| <b>recover</b>                                                                                                                                                                                                                                                                                                                                                            | <code>rec [file]</code>           |
| Recover <code>file</code> from the system save area.                                                                                                                                                                                                                                                                                                                      |                                   |
| <b>redo</b>                                                                                                                                                                                                                                                                                                                                                               | <code>red</code>                  |
| Restore last undone change. Same as <code>Ctrl-R</code> . {vim}                                                                                                                                                                                                                                                                                                           |                                   |
| <b>resize</b>                                                                                                                                                                                                                                                                                                                                                             | <code>res [[±]num]</code>         |
| Resize current window to be <code>num</code> lines high. If <code>+</code> or <code>-</code> is specified, increase or decrease the current window height by <code>num</code> lines. {vim}                                                                                                                                                                                |                                   |
| <b>rewind</b>                                                                                                                                                                                                                                                                                                                                                             | <code>rew[!]</code>               |
| Rewind argument list and begin editing the first file in the list. Add a <code>!</code> to rewind even if the current file has not been saved since the last change.                                                                                                                                                                                                      |                                   |
| <b>right</b>                                                                                                                                                                                                                                                                                                                                                              | <code>[address] ri [width]</code> |
| Right-align lines specified by <code>address</code> , or current line if no address is specified, to column <code>width</code> . Use <code>textwidth</code> option if no <code>width</code> is specified. {vim}                                                                                                                                                           |                                   |

---

---

|                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>sbnext</b>  | <code>[count] sbn [count]</code><br>Split the current window and begin editing the <i>count</i> 'th next buffer from the buffer list. If no count is specified, edit the next buffer in the buffer list. {vim}                                                                                                                                                                                                                                                                                                                                                                      |
| <b>sbuffer</b> | <code>[num] sb [num]</code><br>Split the current window and begin editing buffer <i>num</i> from the buffer list in the new window. The buffer to be edited may also be specified by filename. If no buffer is specified, open the current buffer in the new window. {vim}                                                                                                                                                                                                                                                                                                          |
| <b>set</b>     | <code>se parameter1 parameter2 ...</code><br>Set a value to an option with each <i>parameter</i> , or, if no <i>parameter</i> is supplied, print all options that have been changed from their defaults. For Boolean options, each <i>parameter</i> can be phrased as <i>option</i> or <b>nooption</b> ; other options can be assigned with the syntax <i>option=value</i> . Specify <b>all</b> to list current settings. The form <code>set option?</code> displays the value of <i>option</i> . See the list of <b>set</b> options in the section “The :set Command” on page 692. |
|                | <b>Examples</b><br><code>:set nows wm=10</code><br><code>:set all</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| <b>shell</b>   | <code>sh</code><br>Create a new shell. Resume editing when the shell terminates.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| <b>snext</b>   | <code>[count] sn [[+num] filelist]</code><br>Split the current window and begin editing the next file from the command-line argument list. If <i>count</i> is provided, edit the <i>count</i> 'th next file. If <i>filelist</i> is provided, replace the current argument list with <i>filelist</i> and begin editing the first file. With the <i>+n</i> argument, begin editing on line <i>num</i> . Alternately, <i>num</i> may be a pattern of the form <i>/pattern</i> . {vim}                                                                                                  |
| <b>source</b>  | <code>so file</code><br>Read (source) and execute <b>ex</b> commands from <i>file</i> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|                | <b>Example</b><br><code>:so \$HOME/.exrc</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| <b>split</b>   | <code>[count] sp [+num] [filename]</code><br>Split the current window and load <i>filename</i> in the new window, or the same buffer in both windows if no file is specified. Make the new window <i>count</i> lines high, or, if <i>count</i> is not specified, split the window into equal parts. With the <i>+n</i> argument, begin editing on line <i>num</i> . <i>num</i> may also be a pattern of the form <i>/pattern</i> . {vim}                                                                                                                                            |

---

---

|                   |                                                                                                                                                                                                                                                                                                                                                                                                     |
|-------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>sprevious</b>  | <code>[count] spr [+num]</code><br>Split the current window and begin editing the previous file from the command-line argument list in the new window. If <i>count</i> is specified, edit the <i>count</i> 'th previous file. With the <i>+num</i> argument, begin editing on line <i>num</i> . <i>num</i> may also be a pattern of the form <i>/pattern</i> . {vim}                                |
| <b>stop</b>       | <code>st</code><br>Suspend the editing session. Same as <b>Ctrl-Z</b> . Use the shell <b>fg</b> command to resume the session.                                                                                                                                                                                                                                                                      |
| <b>substitute</b> | <code>[address] s [/pattern/replacement/] [options] [count]</code><br>Replace the first instance of <i>pattern</i> on each of the specified lines with <i>replacement</i> . If <i>pattern</i> and <i>replacement</i> are omitted, repeat last substitution. <i>count</i> specifies the number of lines on which to substitute, starting with <i>address</i> . See additional examples in Chapter 7. |
|                   | <b>Options</b> <ul style="list-style-type: none"><li>c Prompt for confirmation before each change.</li><li>g Substitute all instances of <i>pattern</i> on each line (global).</li><li>p Print the last line on which a substitution was made.</li></ul>                                                                                                                                            |
|                   | <b>Examples</b> <pre>:1,10s/yes/no/g          Substitute on first 10 lines :%s/[Hh]ello/Hi/gc       Confirm global substitutions :s/Fortran/\U&amp;/ 3        Uppercase "Fortran" on next three lines :g/^[0-9][0-9]*\$/Line &amp;:/ For every line beginning with                                 one or more digits, add "Line" and a colon</pre>                                                 |
| <b>suspend</b>    | <code>su</code><br>Suspend the editing session. Same as <b>Ctrl-Z</b> . Use the shell <b>fg</b> command to resume the session.                                                                                                                                                                                                                                                                      |
| <b>sview</b>      | <code>[count] sv [+num] [filename]</code><br>Same as the <b>split</b> command, but set the <b>readonly</b> option for the new buffer. {vim}                                                                                                                                                                                                                                                         |
| <b>t</b>          | <code>[address] t destination</code><br>Copy the lines included in <i>address</i> to the specified <i>destination</i> address. <b>t</b> is equivalent to <b>copy</b> .                                                                                                                                                                                                                              |
|                   | <b>Example</b> <pre>:%t\$      Copy the file and add it to the end</pre>                                                                                                                                                                                                                                                                                                                            |

---

---

|                     |                                                                                                                                                                                                                                                                |
|---------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>tag</b>          | <code>[address] ta tag</code><br>In the <i>tags</i> file, locate the file and line matching <i>tag</i> , and start editing there.                                                                                                                              |
|                     | <b>Example</b><br>Run <b>ctags</b> , then switch to the file containing <i>myfunction</i> :                                                                                                                                                                    |
|                     | <code>:!ctags *.c<br/>:tag myfunction</code>                                                                                                                                                                                                                   |
| <b>tags</b>         | <code>tags</code><br>Print list of tags in the tag stack. {vim}                                                                                                                                                                                                |
| <b>unabbreviate</b> | <code>una word</code><br>Remove <i>word</i> from the list of abbreviations.                                                                                                                                                                                    |
| <b>undo</b>         | <code>u</code><br>Reverse the changes made by the last editing command. In <b>vi</b> , the <b>undo</b> command will undo itself, redoing what you undid. <b>vim</b> supports multiple levels of undo. Use <b>redo</b> to redo an undone change in <b>vim</b> . |
| <b>unhide</b>       | <code>[count] unh</code><br>Split screen to show one window for each active buffer in the buffer list. If specified, limit the number of windows to <i>count</i> . {vim}                                                                                       |
| <b>unmap</b>        | <code>unm[!] string</code><br>Remove <i>string</i> from the list of keyboard macros. Use <b>!</b> to remove a macro for input mode.                                                                                                                            |
| <b>v</b>            | <code>[address] v/pattern/[command]</code><br>Execute <i>command</i> on all lines <i>not</i> containing <i>pattern</i> . If <i>command</i> is not specified, print all such lines. <b>v</b> is equivalent to <b>g!</b> . See <b>global</b> .                   |
|                     | <b>Example</b><br><code>:v/#include/d</code> Delete all lines except "#include" lines                                                                                                                                                                          |
| <b>version</b>      | <code>ve</code><br>Print the editor's current version number and date of last change.                                                                                                                                                                          |
| <b>view</b>         | <code>vie[+[+num] filename]</code><br>Same as <b>edit</b> , but set file to <b>readonly</b> . When executed in <b>ex</b> mode, return to normal or visual mode. {vim}                                                                                          |

---

---

|                 |                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|-----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>visual</b>   | <code>[address] vi [type] [count]</code><br>Enter visual mode ( <b>vi</b> ) at the line specified by <i>address</i> . Return to <b>ex</b> mode with <b>Q</b> . <i>type</i> can be one of -, ^, or . (see the <b>z</b> command). <i>count</i> specifies an initial window size.                                                                                                                                                            |
| <b>visual</b>   | <code>vi [+num]file</code><br>Begin editing <i>file</i> in visual mode ( <b>vi</b> ), optionally at line <i>num</i> .                                                                                                                                                                                                                                                                                                                     |
| <b>vsplit</b>   | <code>[count] vs [+num] [filename]</code><br>Same as the <b>split</b> command, but split the screen vertically. The <i>count</i> argument can be used to specify a width for the new window.<br>{vim}                                                                                                                                                                                                                                     |
| <b>wall</b>     | <code>wa[!]</code><br>Write all changed buffers with filenames. Add ! to force writing of any buffers marked <b>readonly</b> . {vim}                                                                                                                                                                                                                                                                                                      |
| <b>wnext</b>    | <code>[count] wn[!] [[+num]] filename</code><br>Write current buffer and open next file in argument list, or the <i>count</i> 'th next file if specified. If <i>filename</i> is specified, edit it next. With the <i>+num</i> argument, begin editing on line <i>num</i> . <i>num</i> may also be a pattern of the form <i>/pattern</i> . {vim}                                                                                           |
| <b>write</b>    | <code>[address] w[!] [[&gt;&gt;]] file</code><br>Write lines specified by <i>address</i> to <i>file</i> , or write full contents of buffer if <i>address</i> is not specified. If <i>file</i> is also omitted, save the contents of the buffer to the current filename. If >> <i>file</i> is used, append lines to the end of the specified <i>file</i> . Add a ! to force the editor to write over any current contents of <i>file</i> . |
| <b>Examples</b> |                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|                 | <code>:1,10w name_list</code> Copy first 10 lines to file <i>name_list</i><br><code>:50w &gt;&gt; name_list</code> Now append line 50                                                                                                                                                                                                                                                                                                     |
| <b>write</b>    | <code>[address] w !command</code><br>Write lines specified by <i>address</i> to <i>command</i> .                                                                                                                                                                                                                                                                                                                                          |
| <b>Example</b>  |                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|                 | <code>:1,66w !pr -h myfile   lp</code> Print first page of <i>file</i>                                                                                                                                                                                                                                                                                                                                                                    |
| <b>wq</b>       | <code>wq[!]</code><br>Write and quit the file in one movement. The file is always written. The ! flag forces the editor to write over any current contents of <i>file</i> .                                                                                                                                                                                                                                                               |

---

---

**wqall** `wqa[!]`  
Write all changed buffers and quit the editor. Add ! to force writing of any buffers marked **readonly**. **xall** is another alias for this command. {vim}

---

**X** `x`  
Prompt for an encryption key. This command can be preferable to :set key, as typing the key is not echoed to the console. To remove an encryption key, just reset the **key** option to an empty value. {vim}

---

**xit** `x`  
Write the file if it was changed since the last write, then quit.

---

**yank** `[address] y [char] [count]`  
Place lines specified by *address* in named register *char*. Register names are the lowercase letters **a-z**. Uppercase names append text to the corresponding register. If no *char* is given, place lines in general register. *count* specifies the number of lines to yank, starting with *address*.

#### Example

`:101,200 ya a` Copy lines 100–200 to register "a"

---

**z** `[address] z [type] [count]`  
Print a window of text with the line specified by *address* at the top. *count* specifies the number of lines to be displayed.

#### Type

- + Place specified line at the top of the window (default).
- Place specified line at the bottom of the window.
- . Place specified line in the center of the window.
- ^ Print the previous window.
- = Place specified line in the center of the window and leave the current line at this line.

---

**!** `[address] !command`  
Execute Unix *command* in a shell. If *address* is specified, use the lines contained in *address* as standard input to *command*, and replace the lines with the output and error output. (This is called *filtering* the text through the *command*.)

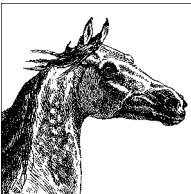
#### Examples

`:!ls` List files in the current directory  
`:11,20!sort -f` Sort lines 11–20 of current file

---

|                 |                                            |                                                                                                                                                                                                                                                                                                                                                                                                                     |
|-----------------|--------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| =               | [address] =                                | Print the line number of the line indicated by <i>address</i> . Default is line number of the last line.                                                                                                                                                                                                                                                                                                            |
| <>              | [address] < [count]<br>[address] > [count] | Shift lines specified by <i>address</i> either left (<) or right (>). Only leading spaces and tabs are added or removed when shifting lines. <i>count</i> specifies the number of lines to shift, starting with <i>address</i> . The <b>shiftwidth</b> option controls the number of columns that are shifted. Repeating the < or > increases the shift amount. For example, :>>> shifts three times as much as :>. |
| <b>address</b>  | <i>address</i>                             | Print the lines specified in <i>address</i> .                                                                                                                                                                                                                                                                                                                                                                       |
| <b>Enter</b>    |                                            | Print the next line in the file. (For <b>ex</b> only; not from the : prompt in <b>vi</b> .)                                                                                                                                                                                                                                                                                                                         |
| @               | [address] @ [char]                         | Execute contents of register specified by <i>char</i> . If <i>address</i> is given, move cursor to the specified address first. If <i>char</i> is @, repeat the last @ command.                                                                                                                                                                                                                                     |
| &               | [address] & [options] [count]              | Repeat the previous substitute ( <b>s</b> ) command. <i>count</i> specifies the number of lines on which to substitute, starting with <i>address</i> . <i>options</i> are the same as for the substitute command.                                                                                                                                                                                                   |
| <b>Examples</b> |                                            |                                                                                                                                                                                                                                                                                                                                                                                                                     |
|                 | :s/Overdue/Paid/                           | Substitute once on current line                                                                                                                                                                                                                                                                                                                                                                                     |
|                 | :g/Status/&                                | Redo substitution on all "Status" lines                                                                                                                                                                                                                                                                                                                                                                             |
| ~               | [address] ~ [count]                        | Replace the last-used regular expression (even if from a search, and not from an <b>s</b> command) with the replacement pattern from the most recent <b>s</b> (substitute) command. This is rather obscure; see Chapter 6 of <i>Learning the vi and Vim Editors</i> by Arnold Robbins et al. (O'Reilly) for details.                                                                                                |

---



# 10

## The sed Editor

The **sed** “stream editor” is one of the most prominent text-processing tools on Unix and Linux. It is most often used for performing simple substitutions on data streams going through pipelines, but **sed** scripts can be written to do much more.

This chapter presents the following topics:

- Conceptual overview of **sed**
- Command-line syntax
- Syntax of **sed** commands
- Group summary of **sed** commands
- Alphabetical summary of **sed** commands

The version of **sed** provided with Linux systems is the GNU version written by the Free Software Foundation; its home page is <http://www.gnu.org/software/sed/sed.html>. For more information on **sed**, see Dale Dougherty and Arnold Robbins’s *sed & awk* (O’Reilly).

### Conceptual Overview

The stream editor, **sed**, is a noninteractive editor. It interprets a script and performs the actions in the script. **sed** is stream-oriented because, like many Unix programs, input flows through the program and is directed to standard output. For example, **sort** is stream-oriented; **vi** is not. **sed**’s input typically comes from a file or pipe, but it can also be taken from the keyboard. Output goes to the screen by default, but it can be captured in a file or sent through a pipe instead. GNU **sed** can edit files that use multibyte character sets.

### Typical Uses of **sed**

- Editing one or more files automatically.
- Simplifying repetitive edits to multiple files.
- Writing conversion programs.

## **sed Operation**

**sed** operates as follows:

- Each line of input is copied into a *pattern space*, an internal buffer where editing operations are performed.
- All editing commands in a **sed** script are applied, in order, to each line of input.
- Editing commands are applied to all lines (globally) unless line addressing restricts the lines affected.
- If a command changes the input, subsequent commands and address tests are applied to the current line in the pattern space, not the original input line.
- The original input file is unchanged because the editing commands modify an in-memory copy of each original input line. The copy is sent to standard output (but can be redirected to a file).
- **sed** also maintains the *hold space*, a separate buffer that can be used to save data for later retrieval.

## **Command-Line Syntax**

The syntax for invoking **sed** has two forms:

```
sed [-n] [-e] 'command' file(s)
sed [-n] -f scriptfile file(s)
```

The first form allows you to specify an editing command on the command line, surrounded by single quotes. The second form allows you to specify a *scriptfile*, a file containing **sed** commands. Both forms may be used together, and they may be used multiple times. If no *file (s)* is specified, **sed** reads from standard input.

## **Standard Options**

The following options are recognized:

- n Suppress the default output; **sed** displays only those lines specified with the p command or with the p flag of the s command.
- e cmd Next argument is an editing command. Necessary if multiple scripts or commands are specified.
- f file Next argument is a file containing editing commands.

If the first line of the script is #n, **sed** behaves as if -n had been specified.

Multiple -e and -f options may be provided, and they may be mixed. The final script consists of the concatenation of all the *script* and *file* arguments.

## GNU sed Options

GNU **sed** accepts a number of additional command-line options, as well as long-option equivalents for the standard options. The GNU **sed** options are:

**-e cmd, --expression cmd**

Use *cmd* as editing commands.

**-f file, --file file**

Obtain editing commands from *file*.

**--help**

Print a usage message and exit.

**-i[suffix], --in-place[=suffix]**

Edit files in place, overwriting the original file. If optional *suffix* is supplied, use it for renaming the original file as a backup file. See the GNU **sed** online Info documentation for the details.

**-l len, --line-length len**

Set the line length for the **l** command to *len* characters.

**-n, --quiet, --silent**

Suppress the default output; **sed** displays only those lines specified with the **p** command or with the **p** flag of the **s** command.

**--posix**

Disable *all* GNU extensions. Setting `POSIXLY_CORRECT` in the environment merely disables those extensions that are incompatible with the POSIX standard.

**-r, --regex-extended**

Use Extended Regular Expressions instead of Basic Regular Expressions. See Chapter 7 for more information.

**-s, --separate**

Instead of considering the input to be one long stream consisting of the concatenation of all the input files, treat each file separately. Line numbers start over with each file, the address **\$** refers to the last line of each file, files read by the **R** command are rewound, and range addresses (**/x/,/y/**) may not cross file boundaries.

**-u, --unbuffered**

Buffer input and output as little as possible. Useful for editing the output of **tail -f** when you don't want to wait for the output.

**--version**

Print the version of GNU **sed** and a copyright notice, and then exit.

## Syntax of sed Commands

**sed** commands have the general form:

`[address[,address]][!]command [arguments]`

*commands* consist of a single letter or symbol; they are described later, by group and alphabetically. *arguments* include the label supplied to **b** or **t**, the filename supplied to **r** or **w**, and the substitution flags for **s**. *addresses* are described next.

## Pattern Addressing

A **sed** command can specify zero, one, or two addresses. In POSIX **sed**, an address has one of the forms in the following table. Regular expressions are described in Chapter 7. Additionally, \n can be used to match any newline in the pattern space (resulting from the N command), but not the newline at the end of the pattern space.

| Address                       | Meaning                                                                                                                                                             |
|-------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| /pattern/                     | Lines that match <i>pattern</i> .                                                                                                                                   |
| \;pattern;                    | Like previous, but use semicolon as the delimiter instead of slash. Any character may be used. This is useful if <i>pattern</i> contains multiple slash characters. |
| n                             | Line number <i>n</i> .                                                                                                                                              |
| \$                            | The last input line.                                                                                                                                                |
| If the command specifies:     | Then the command is applied to:                                                                                                                                     |
| No address                    | Each input line.                                                                                                                                                    |
| One address                   | Any line matching the address. Some commands accept only one address: a, i, r, q, and =.                                                                            |
| Two comma-separated addresses | First matching line and all succeeding lines up to and including a line matching the second address.                                                                |
| An address followed by !      | All lines that do <i>not</i> match the address.                                                                                                                     |

GNU **sed** allows additional address forms:

| Address      | Meaning                                                                                                                     |
|--------------|-----------------------------------------------------------------------------------------------------------------------------|
| /pattern/i   | Match pattern, ignoring case. I may be used instead of i.                                                                   |
| /pattern/m   | Match pattern, allowing ^ and \$ to match around an embedded newline. M may be used instead of m.                           |
| 0,/pattern/  | Similar to 1,/pattern/, but if line 1 matches <i>pattern</i> , it will end the range.                                       |
| address,+n   | Matches line matching <i>address</i> , and the <i>n</i> following lines.                                                    |
| address~incr | Matches line matching <i>address</i> and every <i>incr</i> lines after it. For example, 42~3 matches 42, 45, 48, and so on. |

## Pattern Addressing Examples

| Command                 | Action performed                                       |
|-------------------------|--------------------------------------------------------|
| s/xx/yy/g               | Substitute on all lines (all occurrences).             |
| /BSD/d                  | Delete lines containing BSD.                           |
| /^BEGIN/,/^END/p        | Print between BEGIN and END, inclusive.                |
| /SAVE/!d                | Delete any line that doesn't contain SAVE.             |
| /BEGIN/,/END/!s/xx/yy/g | Substitute on all lines, except between BEGIN and END. |

Braces ({} ) are used in **sed** to nest one address inside another or to apply multiple commands at a single matched address:

```
[/pattern/[,/pattern/]]{
 command1
 command2
}
```

The opening curly brace must end its line, and the closing curly brace must be on a line by itself. Be sure there are no spaces after the braces.

## GNU sed Regular Expression Extensions

With the **-r** option, GNU **sed** uses Extended Regular Expressions instead of Basic Regular expressions. (See Chapter 7 for more information.) However, even without **-r**, you can use additional escape sequences for more powerful text matching. The following escape sequences are valid only in regular expressions:

- \b Matches on a word boundary, where of the two surrounding characters (`x\b`), one is a word-constituent character and the other is not.
- \B Matches on a nonword boundary, where both of the two surrounding characters (`\Bx`) are either word-constituent or not word-constituent.
- \w Matches any word-constituent character (i.e., a letter, digit, or underscore).
- \W Matches any nonword-constituent character (i.e., anything that is *not* a letter, digit, or underscore).
- \` Matches the beginning of the pattern space. This is different from `^` when the **m** modifier is used for a pattern or the **s** command.
- \` Matches the end of the pattern space. This is different from `$` when the **m** modifier is used for a pattern or the **s** command.

The following escape sequences may be used anywhere.

- \a The ASCII BEL character.
- \f The ASCII formfeed character.
- \n The ASCII newline character.
- \r The ASCII carriage-return character.
- \v The ASCII vertical tab character.
- \dnn The character whose ASCII decimal value is *nn*.
- \onn The character whose ASCII octal value is *nn*.
- \xnn The character whose ASCII hexadecimal value is *nn*.

## Group Summary of sed Commands

In the lists that follow, the **sed** commands are grouped by function and are described tersely. Full descriptions, including syntax and examples, can be found afterward in the “Alphabetical Summary of **sed** Commands” on page 717. Commands marked with a † are specific to GNU **sed**.

## Basic Editing

| Command | Action                               |
|---------|--------------------------------------|
| a\      | Append text after a line.            |
| c\      | Replace text (usually a text block). |
| i\      | Insert text before a line.           |
| d       | Delete lines.                        |
| s       | Make substitutions.                  |
| y       | Translate characters (like Unix tr). |

## Line Information

| Command | Action                               |
|---------|--------------------------------------|
| =       | Display line number of a line.       |
| I       | Display control characters in ASCII. |
| p       | Display the line.                    |

## Input/Output Processing

| Command | Action                                                   |
|---------|----------------------------------------------------------|
| e†      | Execute commands.                                        |
| n       | Skip current line and go to the next line.               |
| r       | Read another file's contents into the output stream.     |
| R†      | Read one line from a file into the output.               |
| w       | Write input lines to another file.                       |
| W†      | Write first line in pattern space to another file.       |
| q       | Quit the sed script (no further output).                 |
| Q†      | Quit without printing the pattern space.                 |
| v†      | Require a specific version of GNU sed to run the script. |

## Yanking and Putting

| Command | Action                                                  |
|---------|---------------------------------------------------------|
| h       | Copy into hold space; wipe out what's there.            |
| H       | Copy into hold space; append to what's there.           |
| g       | Get the hold space back; wipe out the destination line. |
| G       | Get the hold space back; append to the pattern space.   |
| x       | Exchange contents of the hold and pattern spaces.       |

## Branching Commands

| Command        | Action                                                     |
|----------------|------------------------------------------------------------|
| b              | Branch to <i>label</i> or to end of script.                |
| t              | Same as b, but branch only after substitution.             |
| T              | Same as t, but branch only if no successful substitutions. |
| : <i>label</i> | Label branched to by t or b.                               |

## Multiline Input Processing

| Command | Action                                                 |
|---------|--------------------------------------------------------|
| N       | Read another line of input (creates embedded newline). |
| D       | Delete up to the embedded newline.                     |
| P       | Print up to the embedded newline.                      |

## Alphabetical Summary of sed Commands

GNU **sed** lets you use the filenames */dev/stdin*, */dev/stdout* and */dev/stderr* to refer to standard input, output, and error, respectively, for the **r**, **R**, **w**, and **W** commands and the **w** flag to the **s** command.

GNU-specific commands or extensions are noted with {G} in the command synopsis. When the GNU version allows a command to have two addresses, the command is performed for each input line within the range.

---

|   |   |
|---|---|
| # | # |
|---|---|

Begin a comment in a **sed** script. Valid only as the first character of the first line. (Some versions, including GNU **sed**, allow comments anywhere, but it is better not to rely on this.) If the first line of the script is #n, **sed** behaves as if -n had been specified.

---

|   |                |
|---|----------------|
| : | : <i>label</i> |
|---|----------------|

Label a line in the script for the transfer of control by **b** or **t**. According to POSIX, **sed** must support labels that are unique in the first eight characters. GNU **sed** has no limit, but some older versions support up to only seven characters.

---

|   |                                              |
|---|----------------------------------------------|
| = | [/pattern/] =<br>[address1[,address2]] = {G} |
|---|----------------------------------------------|

Write to standard output the line number of each line addressed by *pattern*.

- 
- a** [address]a\  
text  
[address1[,address2]]a \ {G}  
text
- Append *text* following each line matched by *address*. If *text* goes over more than one line, newlines must be “hidden” by preceding them with a backslash. The *text* is terminated by the first newline that is not hidden in this way. The *text* is not available in the pattern space, and subsequent commands cannot be applied to it. The results of this command are sent to standard output when the list of editing commands is finished, regardless of what happens to the current line in the pattern space.
- The GNU version accepts two addresses and allows you to put the first line of *text* on the same line as the **a** command.
- Example**
- ```
$ a\  
This goes after the last line in the file\  
(marked by $). This text is escaped at the\  
end of each line, except for the last one.
```
-
- b** [address1[,address2]]b[label]
- Unconditionally transfer control to *label* elsewhere in script. That is, the command following the *label* is the next command applied to the current line. If no *label* is specified, control falls through to the end of the script, so no more commands are applied to the current line.
- Example**
- ```
Ignore HTML tables; resume script after </table>:
<table/,/<\table>/b
```
- 
- c** [address1[,address2]]c\  
text
- Replace (change) the lines selected by the address(es) with *text*. (See a for details on *text*.) When a range of lines is specified, all lines are replaced as a group by a single copy of *text*. The contents of the pattern space are, in effect, deleted, and no subsequent editing commands can be applied to the pattern space (or to *text*).
- Example**
- ```
# Replace first 100 lines in a file:  
1,100c\  
\  
<First 100 names to be supplied>
```

d

[address1[,address2]]d

Delete the addressed line (or lines) from the pattern space. Thus, the line is not passed to standard output. A new line of input is read, and editing resumes with the first command in the script.

Example

```
# Delete all empty lines, including lines with just
whitespace:
/^[#tab]*$/d
```

D

[address1[,address2]]D

Delete the first part (up to embedded newline) of a multiline pattern space created by the N command, and resume editing with the first command in the script. If this command empties the pattern space, then a new line of input is read, as if the d command had been executed.

Example

```
# Strip multiple blank lines, leaving only one:
/^$/{
N
/^\\n$/D
}
```

e

[address1[,address2]]e [command]{G}

With *command*, execute the command and send the result to standard output. Without *command*, execute the contents of the pattern space as a command, and replace the pattern space with the results.

g

[address1[,address2]]g

Paste the contents of the hold space (see h and H) back into the pattern space, wiping out the previous contents of the pattern space. The Example shows a simple way to copy lines.

Example

This script collects all lines containing the word Item: and copies them to a place marker later in the file. The place marker is overwritten:

```
/Item:/H
/<Replace this line with the item list>/g
```

G

[address1[,address2]]G

Same as g, except that a newline and the hold space are pasted to the end of the pattern space instead of overwriting it. The Example shows a simple way to “cut and paste” lines.

Example

This script collects all lines containing the word `Item:` and moves them after a place marker later in the file. The original `Item:` lines are deleted.

```
/Item:/{
H
d
}
/Summary of items:/G
```

h

`[address1[,address2]]h`

Copy the pattern space into the hold space, a special temporary buffer. The previous contents of the hold space are obliterated. You can use `h` to save a line before editing it.

Example

```
# Edit a line; print the change; replay the original
/Linux/{
h
s/.*/ Linux \(.*) .*/\1:/
p
x
}
```

Sample input:

```
This describes the Linux ls command.
This describes the Linux cp command.
```

Sample output:

```
ls:
This describes the Linux ls command.
cp:
This describes the Linux cp command.
```

H

`[address1[,address2]]H`

Append a newline and then the contents of the pattern space to the contents of the hold space. Even if the hold space is empty, `H` still appends a newline. `H` is like an incremental copy. See Examples under `g` and `G`.

i

```
[address]i\
text
[address1[,address2]]i \
{G}
text
```

Insert `text` before each line matched by `address`. (See `a` for details on `text`.)

The GNU version accepts two addresses and allows you to put the first line of `text` on the same line as the `i` command.

Example

```
/Item 1/i\
The five items are listed below:
```

I [address1[,address2]]1
[address1[,address2]]1 [len]{G}

List the contents of the pattern space, showing nonprinting characters as ASCII codes. Long lines are wrapped. With GNU sed, len is the character position at which to wrap long lines. A value of 0 means to never break lines.

n [address1[,address2]]n

Read the next line of input into pattern space. The current line is sent to standard output, and the next line becomes the current line. Control passes to the command following n instead of resuming at the top of the script.

Example

In DocBook/XML, titles follow section tags. Suppose you are using a convention where each opening section tag is on a line by itself, with the title on the following line. To print all the section titles, invoke this script with sed -n:

```
<sect[1-4]/{  
n  
p  
}
```

N [address1[,address2]]N

Append the next input line to contents of pattern space; the new line is separated from the previous contents of the pattern space by a newline. (This command is designed to allow pattern matches across two lines.) By using \n to match the embedded newline, you can match patterns across multiple lines. See the Example under D.

Examples

Like the Example in n, but print the section tag line as well as header title:

```
<sect[1-4]/{  
N  
p  
}
```

Join two lines (replace newline with space):

```
<sect[1-4]/{  
N  
s/\n/ /  
p  
}
```

p *[address1[,address2]]p*
Print the addressed line(s). Note that this can result in duplicate output unless default output is suppressed by using #n or the -n command-line option. Typically used before commands that change control flow (d, n, b), which might prevent the current line from being output. See the Examples under h, n, and N.

P *[address1[,address2]]P*
Print first part (up to embedded newline) of multiline pattern space created by N command. Same as p if N has not been applied to a line.

Example

Suppose you have function references in two formats:

```
function(arg1, arg2)
function(arg1,
        arg2)
```

The following script changes argument arg2, regardless of whether it appears on the same line as the function name:

```
s/function(arg1, arg2)/function(arg1, XX)/
/function(/{
N
s/arg2/XX/
P
D
}
```

q *[address]q*
[address]q [value]{G}
Quit when address is encountered. The addressed line is first written to the output (if default output is not suppressed), along with any text appended to it by previous a or r commands. GNU sed allows you to provide value, which is used as the exit status.

Examples

Delete everything after the addressed line:

```
/Garbled text follows:/q
```

Print only the first 50 lines of a file:

```
50q
```

Q *[address]Q [value]{G}*
Quits processing, but without printing the pattern space. If value is provided, it is used as sed's exit status.

r *[address]r file*
[address1[,address2]]r file{G}

Read contents of *file* and append to the output after the contents of the pattern space. There must be exactly one space between the **r** and the filename. The GNU version accepts two addresses.

Example

/The list of items follows:/r item_file

R *[address1[,address2]]R file{G}*

Read one line of *file* and append to the output after the contents of the pattern space. Successive **R** commands read successive lines from *file*.

s *[address1[,address2]]s/pattern/replacement/[flags]*

Substitute *replacement* for *pattern* on each addressed line. If pattern addresses are used, the pattern // represents the last pattern address specified. Any delimiter may be used. Use \ within *pattern* or *replacement* to escape the delimiter. The following flags can be specified (those marked with a † are specific to GNU **sed**):

n Replace *n*th instance of *pattern* on each addressed line. *n* is any number in the range 1 to 512, and the default is 1.

e† If the substitution was made, execute the contents of the pattern space as a shell command and replaces the pattern space with the results.

g Replace all instances of *pattern* on each addressed line, not just the first instance.

i or *I*†

Do a case-insensitive regular expression match.

m or *M*†

Allow ^ and \$ to match around a newline embedded in the pattern space.

p Print the line if the substitution is successful. If several successful substitutions are successful, **sed** prints multiple copies of the line.

w file

Write the line to *file* if a replacement was done. In the traditional Unix **sed**, a maximum of 10 different *files* can be opened.

GNU **sed** allows you to use the special filenames /dev/stdout and /dev/stderr to write to standard output or standard error, respectively.

Within the *replacement*, GNU **sed** accepts special escape sequences, with the following meanings:

\L Lowercase the replacement text until a terminating \E or \U.

\l Lowercase the following character only.

- \U Uppercase the replacement text until a terminating \E or \L.
- \u Uppercase the following character only.
- \E Terminate case conversion from \L or \U.

Examples

Here are some short, commented scripts:

```
# Change third and fourth quote to ( and ):  
/function/{  
    s/"/)/4  
    s/"/(/3  
}  
  
# Remove all quotes on a given line:  
/Title/s/"//g  
  
# Remove first colon and all quotes; print resulting lines:  
s/://p  
s://"gp  
  
# Change first "if" but leave "ifdef" alone:  
/ifdef!s/if/    if/
```

t

[address1[,address2]]t [label]

Test if successful substitutions have been made on addressed lines, and if so, branch to the line marked by :label. (See **b** and :) If label is not specified, control branches to the bottom of the script. The t command is like a case statement in the C programming language or the various shell programming languages. You test each case; when it's true, you exit the construct.

Example

Suppose you want to fill empty fields of a database. You have this:

```
ID: 1  Name: greg  Rate: 45  
ID: 2  Name: dale  
ID: 3
```

You want this:

```
ID: 1  Name: greg  Rate: 45  Phone: ??  
ID: 2  Name: dale  Rate: ??  Phone: ??  
ID: 3  Name: ????  Rate: ??  Phone: ??
```

You need to test the number of fields already there. Here's the script (fields are tab-separated):

```
#n  
/ID/{  
    s/ID: .* Name: .* Rate: .*/&  Phone: ??/p  
    t  
    s/ID: .* Name: .* /&  Rate: ??  Phone: ??/p  
    t  
    s/ID: .* /&  Name: ????  Rate: ??  Phone: ??/p  
}
```

T**[address1[,address2]]T [label]{G}**

Like **t**, but only branches to *label* if there *not* any successful substitutions. (see **b**, **t**, and **:**). If *label* is not specified, control branches to the bottom of the script.

V**[address1[,address2]]V [version]{G}**

This command doesn't do anything. You use it to require GNU **sed** for your script. This works, because non-GNU versions of **sed** don't implement the command at all, and will therefore fail. If you supply a specific *version*, GNU **sed** fails if the required version is newer than the one executing the script.

W**[address1[,address2]]W file**

Append contents of pattern space to *file*. This action occurs when the command is encountered rather than when the pattern space is output. Exactly one space must separate the **w** and the filename. This command will create the file if it does not exist; if the file exists, its contents will be overwritten each time the script is executed. Multiple write commands that direct output to the same file append to the end of the file.

GNU **sed** allows you to use the special filenames **/dev/stdout** and **/dev/stderr** to write to standard output or standard error, respectively.

Example

```
# Store HTML tables in a file
/<table>/,<\/table>/w tables.html
```

W**[address1[,address2]]W file**

Like **w**, but only writes the contents of the first line in the pattern space to the file.

X**[address1[,address2]]X**

Exchange the contents of the pattern space with the contents of the hold space. See **h** for an example.

y**[address1[,address2]]y/abc/xyz/**

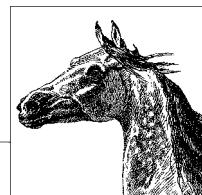
Translate characters. Change every instance of *a* to *x*, *b* to *y*, *c* to *z*, etc.

Example

```
# Change item 1, 2, 3 to Item A, B, C ...
/^item [1-9]/y/i123456789/IABCDEFGHI/
```

11

The gawk Programming Language



gawk is the GNU version of **awk**, a powerful utility often used for text and string manipulation within shell scripts, particularly when input data may be viewed as records and fields. **awk** is also an elegant and capable programming language that allows you to accomplish a lot with very little work.

This chapter presents the following topics:

- Conceptual overview
- Command-line syntax
- Patterns and procedures
- Built-in variables
- Operators
- Variables and array assignment
- User-defined functions
- **gawk**-specific facilities
- Implementation limits
- Group listing of **awk** functions and commands
- Alphabetical summary of **awk** functions and commands
- Source code

Conceptual Overview

awk is a pattern-matching program for processing files, especially when each line has a simple field-oriented layout. Linux provides the GNU version of **awk**, called **gawk**, which provides a number of additional features. This utility can be invoked either through the standard name **awk** or through **gawk**.

This chapter describes functionality that is found in **gawk**. Most of this discussion applies to all versions of **awk**, but items marked “gawk-specific” or as an “extension” may not apply to versions of **awk** other than GNU’s. If portability to older, non-Linux systems is important, do not use gawk-specific features.

With **gawk**, you can:

- Think of a text file as made up of records and fields in a textual database.
- Perform arithmetic and string operations.
- Use programming constructs, such as loops and conditionals.
- Produce formatted reports.
- Define your own functions.
- Execute Unix commands from a script.
- Process the results of Unix commands.
- Process command-line arguments gracefully.
- Work easily with multiple input streams.
- Flush open output files and pipes.
- Sort arrays.
- Retrieve and format system time values.
- Do bit manipulation.
- Internationalize your **gawk** programs, allowing strings to be translated into a local language at runtime.
- Perform two-way I/O to a coprocess.
- Open a two-way TCP/IP connection to a socket.
- Dynamically add built-in functions.
- Profile your **gawk** programs.

Command-Line Syntax

The syntax for invoking **awk** has two forms:

```
awk [options] 'script' var=value file(s)  
awk [options] -f scriptfile var=value file(s)
```

You can specify a *script* directly on the command line, or you can store a script in a *scriptfile* and specify it with **-f**. **gawk** allows multiple **-f** scripts. Variables can be assigned a value on the command line. The value can be a string or numeric constant, a shell variable (`$name`), or a command substitution (``cmd``), but the value is available only after the **BEGIN** statement is executed.

awk operates on one or more *files*. If none are specified (or if **-** is specified), **awk** reads from standard input.

Standard Options

The standard options are:

-Ffs

Set the field separator to *fs*. This is the same as setting the built-in variable **FS**. **gawk** allows *fs* to be a regular expression. Each input line, or *record*, is divided into fields by whitespace (spaces or tabs) or by some other user-definable field separator. Fields are referred to by the variables **\$1**, **\$2**, ..., **\$n**. **\$0** refers to the entire record.

-v var=value

Assign a *value* to variable *var*. This allows assignment before the script begins execution.

For example, to print the first three (colon-separated) fields of each record on separate lines:

```
awk -F: '{ print $1; print $2; print $3 }' /etc/passwd
```

Numerous examples are shown in “Simple Pattern-Procedure Examples” on page 730.

Important gawk Options

Besides the standard command line options, **gawk** has a large number of additional options. This section lists those of most value in day-to-day use. Any unique abbreviation of these options is acceptable.

--dump-variables[=file]

When the program has finished running, print a sorted list of global variables, their types, and their final values to *file*. The default file is *awkvars.out*.

--gen-po

Read the **awk** program and print all strings marked as translatable to standard output in the form of a GNU **gettext** Portable Object file. See the section “Internationalization” on page 736 for more information.

--help

Print a usage message to standard error and exit.

--lint[=fatal]

Enable checking of nonportable or dubious constructs, both when the program is read and as it runs. With an argument of **fatal**, lint warnings become fatal errors.

--non-decimal-data

Allow octal and hexadecimal data in the input to be recognized as such. This option is not recommended; use **strtonum()** in your program, instead.

--profile[=file]

With **gawk**, put a “prettyprinted” version of the program in *file*. Default is *awkprof.out*. With **pgawk** (see “Profiling” on page 735), put the profiled listing of the program in *file*.

--posix

Turn on strict POSIX compatibility, in which all common and **gawk**-specific extensions are disabled.

--source='program text'

Use *program text* as the **awk** source code. Use this option with **-f** to mix command-line programs with **awk** library files.

--traditional

Disable all **gawk**-specific extensions, but allow common extensions (e.g., the ****** operator for exponentiation).

--version

Print the version of **gawk** on standard error and exit.

Patterns and Procedures

awk scripts consist of patterns and procedures:

```
pattern {procedure}
```

Both *pattern* and *{ procedure }* are optional. If *pattern* is missing, *{ procedure }* is applied to all lines. If *{ procedure }* is missing, the matched line is printed.

Patterns

A pattern can be any of the following:

```
general expression
/regular expression/
relational expression
pattern-matching expression
BEGIN
END
```

- General expressions can be composed of quoted strings, numbers, operators, function calls, user-defined variables, or any of the predefined variables described in “Built-in Variables” on page 731.
- Regular expressions use the extended set of metacharacters, as described in Chapter 7.
- The ^ and \$ metacharacters refer to the beginning and end of a string (such as the fields), respectively, rather than the beginning and end of a line. In particular, these metacharacters will *not* match at a newline embedded in the middle of a string.
- Relational expressions use the relational operators listed in “Operators” on page 732. For example, **\$2 > \$1** selects lines for which the second field is greater than the first. Comparisons can be either string or numeric. Thus, depending upon the types of data in **\$1** and **\$2**, **awk** will do either a numeric or a string comparison. This can change from one record to the next.
- Pattern-matching expressions use the operators ~ (match) and !~ (don’t match). See “Operators” on page 732.

- The **BEGIN** pattern lets you specify procedures that will take place *before* the first input line is processed. (Generally, you process the command line and set global variables here.)
- The **END** pattern lets you specify procedures that will take place *after* the last input record is read.
- **BEGIN** and **END** patterns may appear multiple times. The procedures are merged as if there had been one large procedure.

Except for **BEGIN** and **END**, patterns can be combined with the Boolean operators **||** (or), **&&** (and), and **!** (not). A range of lines can also be specified using comma-separated patterns:

pattern, pattern

Procedures

Procedures consist of one or more commands, function calls, or variable assignments, separated by newlines or semicolons, and are contained within curly braces. Commands fall into five groups:

- Variable or array assignments
- Input/Output commands
- Built-in functions
- Control-flow commands
- User-defined functions

Simple Pattern-Procedure Examples

Print first field of each line:

```
{ print $1 }
```

Print all lines that contain *pattern*:

```
/pattern/
```

Print first field of lines that contain *pattern*:

```
/pattern/ { print $1 }
```

Select records containing more than two fields:

```
NF > 2
```

Interpret input records as a group of lines up to a blank line. Each line is a single field:

```
BEGIN { FS = "\n"; RS = "" }
```

Print fields 2 and 3 in switched order, but only on lines whose first field matches the string **URGENT**:

```
$1 ~ /URGENT/ { print $3, $2 }
```

Count and print the number of *pattern* found:

```
/pattern/ { ++x }
END { print x }
```

Add numbers in second column and print the total:

```
{ total += $2 }
END { print "column total is", total}
```

Print lines that contain less than 20 characters:

```
length($0) < 20
```

Print each line that begins with **Name:** and that contains exactly seven fields:

```
NF = 7 && /^[^Name:]/
```

Print the fields of each record in reverse order, one per line:

```
{
    for (i = NF; i >= 1; i--)
        print $i
}
```

gawk

Built-in Variables

All **awk** variables are included in **gawk**.

Version	Variable	Description
awk	ARGC	Number of arguments on the command line.
	ARGV	An array containing the command-line arguments, indexed from 0 to ARGC - 1.
	ENVIRON	An associative array of environment variables.
	FILENAME	Current filename.
	FNR	Like NR, but relative to the current file.
	FS	Field separator (a space).
	NF	Number of fields in current record.
	NR	Number of the current record.
	OFMT	Output format for numbers ("%.6g").
	OFS	Output field separator (a space).
	ORS	Output record separator (a newline).
	RLENGTH	Length of the string matched by match() function.
	RS	Record separator (a newline).
	RSTART	First position in the string matched by match() function.
	SUBSEP	Separator character for array subscripts ("\034").
gawk	\$0	Entire input record.
	\$n	n th field in current record; fields are separated by FS.
	ARGIND	Index in ARGV of current input file.
	BINMODE	Controls binary I/O for input and output files. Use values of 1, 2, or 3 for input, output, or both kinds of files, respectively. Set it on the command line to affect standard input, standard output, and standard error.
	ERRNO	A string indicating the error when a redirection fails for getline or if close() fails.

Version	Variable	Description
	FIELDWIDTHS	A space-separated list of field widths to use for splitting up the record, instead of FS.
	IGNORECASE	When true, all regular expression matches, string comparisons, and index() ignore case.
	LINT	Dynamically controls production of "lint" warnings. With a value of "fatal", lint warnings become fatal errors.
	PROCINFO	An array containing information about the process, such as real and effective UID numbers, process ID number, and so on.
	RT	The text matched by RS, which can be a regular expression in gawk.
	TEXTDOMAIN	The text domain (application name) for internationalized messages ("messages").

Operators

The following table lists the operators, in order of increasing precedence, that are available in **awk**.

Symbol	Meaning
= += -= *= /= %= ^= **=	Assignment.
?:	C conditional expression.
	Logical OR (short-circuit).
&&	Logical AND (short-circuit).
in	Array membership.
~ !~	Match regular expression and negation.
< <= > >= != ==	Relational operators.
(blank)	Concatenation.
+ -	Addition, subtraction.
* / %	Multiplication, division, and modulus (remainder).
+ - !	Unary plus and minus, and logical negation.
^ **	Exponentiation.
++ --	Increment and decrement, either prefix or postfix.
\$	Field reference.



While ** and **= are common extensions, they are not part of POSIX awk.

Variable and Array Assignment

Variables can be assigned a value with an equals sign. For example:

```
FS = ","
```

Expressions using the operators +, -, /, and % (modulo) can be assigned to variables.

Arrays can be created with the `split()` function (described later), or they can simply be named in an assignment statement. Array elements can be subscripted with numbers (`array[1], ..., array[n]`) or with strings. Arrays subscripted by strings are called *associative arrays*.^{*} For example, to count the number of widgets you have, you could use the following script:

```
/widget/ { count["widget"]++ }          Count widgets
    ND      { print count["widget"] }     Print the count
```

You can use the special `for` loop to read all the elements of an associative array:

```
for (item in array)
    process array[item]
```

The index of the array is available as `item`, while the value of an element of the array can be referenced as `array[item]`.

You can use the operator `in` to test that an element exists by testing to see if its index exists. For example:

```
if (index in array)
    ...
```

tests that `array[index]` exists, but you cannot use it to test the value of the element referenced by `array[index]`.

You can also delete individual elements of the array using the `delete` statement. (See also the `delete` entry in “Alphabetical Summary of awk Functions and Commands” on page 738.)

Escape sequences

Within string and regular-expression constants, the following escape sequences may be used.

Sequence	Meaning	Sequence	Meaning
<code>\a</code>	Alert (bell)	<code>\v</code>	Vertical tab
<code>\b</code>	Backspace	<code>\\\</code>	Literal backslash
<code>\f</code>	Form feed	<code>\nnn</code>	Octal value <i>nnn</i>
<code>\n</code>	Newline	<code>\xnn</code>	Hexadecimal value <i>nn</i>
<code>\r</code>	Carriage return	<code>\"</code>	Literal double quote (in strings)
<code>\t</code>	Tab	<code>\ </code>	Literal slash (in regular expressions)



The `\x` escape sequence is a common extension; it is not part of POSIX awk.

* In fact, all arrays in awk are associative; numeric subscripts are converted to strings before being used as array subscripts. Associative arrays are one of awk’s most powerful features.

Octal and Hexadecimal Constants in gawk

gawk allows you to use octal and hexadecimal constants in your program source code. The form is as in C: octal constants start with a leading **0**, and hexadecimal constants with a leading **0x** or **0X**. The hexadecimal digits **a–f** may be in either uppercase or lowercase.

```
$ gawk 'BEGIN { print 042, 42, 0x42 }'  
34 42 66
```

Use the **strtonum()** function to convert octal or hexadecimal input data into numerical values.

User-Defined Functions

gawk allows you to define your own functions. This makes it easy to encapsulate sequences of steps that need to be repeated into a single place and reuse the code from anywhere in your program.

The following function capitalizes each word in a string. It has one parameter, named **input**, and five local variables, which are written as extra parameters:

```
# capitalize each word in a string  
function capitalize(input, result, words, n, i, w)  
{  
    result = ""  
    n = split(input, words, " ")  
    for (i = 1; i <= n; i++) {  
        w = words[i]  
        w = toupper(substr(w, 1, 1)) substr(w, 2)  
        if (i > 1)  
            result = result " "  
        result = result w  
    }  
    return result  
}  
  
# main program, for testing  
{ print capitalize($0) }
```

With this input data:

A test line with words and numbers like 12 on it.

This program produces:

A Test Line With Words And Numbers Like 12 On It.



For user-defined functions, no space is allowed between the function name and the left parenthesis when the function is called.

gawk-Specific Features

This section describes features unique to **gawk**.

Coprocesses and Sockets

gawk allows you to open a two-way pipe to another process, called a *coprocess*. This is done with the `|&` operator used with `getline` and `print` or `printf`.

```
print database command |& "db_server"
"db_server" |& getline response
```

If the *command* used with `|&` is a filename beginning with `/inet/`, **gawk** opens a TCP/IP connection. The filename should be of the following form:

```
/inet/protocol/lport/hostname/rport
```

The parts of the filename are:

protocol

One of **tcp**, **udp** or **raw**, for TCP, UDP, or raw IP sockets, respectively. Note: **raw** is currently reserved but unsupported.

lport

The local TCP or UDP port number to use. Use **0** to let the operating system pick a port.

hostname

The name or IP address of the remote host to connect to.

rport

The port (application) on the remote host to connect to. A service name (e.g., **tftp**) is looked up using the C `getservbyname()` function.

Profiling

When **gawk** is built and installed, a separate program named **pgawk** (*profiling gawk*) is built and installed with it. The two programs behave identically; however, **pgawk** runs more slowly because it keeps execution counts for each statement as it runs. When it is done, it automatically places an execution profile of your program in a file named *awkprof.out*. (You can change the filename with the **--profile** option.)

The execution profile is a “prettyprinted” version of your program, with execution counts listed in the left margin. For example, after running this program:

```
$ pgawk '/bash$/ { nusers++ }
> END { print nusers, "users use Bash." }' /etc/passwd
16 users use Bash.
```

the execution profile looks like this:

```
# gawk profile, created Mon Nov 1 14:34:38 2004
# Rule(s)
```

```

35 /bash$/ { # 16
16         nusers++
}
# END block(s)
END {
1       print nusers, "users use Bash."
}

```

If sent **SIGUSR1**, **pgawk** prints the profile and an **awk** function call stack trace, and then keeps going. Multiple **SIGUSR1** signals may be sent; the profile and trace will be printed each time. This facility is useful if your **awk** program appears to be looping and you want to see if something unexpected is being executed.

If sent **SIGHUP**, **pgawk** prints the profile and stack trace, and then exits.

File Inclusion

The **igawk** program provides a file-inclusion facility for **gawk**. You invoke it the same way you do **gawk**: it passes all command line arguments on to **gawk**. However, **igawk** processes source files and command-line programs for special statements of the form:

```
@include file.awk
```

Such files are searched for along the list of directories specified by the AWKPATH environment variable. When found, the **@include** line is replaced with the text of the corresponding file. Included files may themselves include other files with **@include**.

The combination of the AWKPATH environment variable and **igawk** makes it easy to have and use libraries of **awk** functions.

Internationalization

You can *internationalize* your programs if you use **gawk**. This consists of choosing a text domain for your program, marking strings that are to be translated, and, if necessary, using the **bindtextdomain()**, **dcgettext()**, and **dcngettext()** functions.

Localizing your program consists of extracting the marked strings, creating translations, and compiling and installing the translations in the proper place. Full details are given in *sed & awk* by Dale Dougherty and Arnold Robbins (O'Reilly).

The internationalization features in **gawk** use GNU **gettext**. You may need to install the GNU **gettext** tools to create translations if your system doesn't already have them. Here is a very brief outline of the steps involved:

1. Set **TEXTDOMAIN** to your text domain in a **BEGIN** block:

```
BEGIN { TEXTDOMAIN = "whizprog" }
```

2. Mark all strings to be translated by prepending a leading underscore:

```
printf(_("whizprog: can't open /dev/telepath (%s)\n",
        dcgettext(ERRNO)) > "/dev/stderr"
```

3. Extract the strings with the **--gen-po** option:

```
$ gawk --gen-po -f whizprog.awk > whizprog.po
```

4. Copy the file for translating, and make the translations:

```
$ cp whizprog.po esperanto.po
$ ed esperanto.po
```

5. Use the **msgfmt** program from GNU **gettext** to compile the translations. The binary format allows fast lookup of the translations at runtime. The default output is a file named *messages*:

```
$ msgfmt esperanto.po
$ mv messages esperanto.mo
```

6. Install the file in the standard location. This is usually done at program installation. The location can vary from system to system.

That's it! **gawk** will automatically find and use the translated messages, if they exist.

Implementation Limits

Many versions of **awk** have various implementation limits, on things such as:

- Number of fields per record
- Number of characters per input record
- Number of characters per output record
- Number of characters per field
- Number of characters per **printf** string
- Number of characters in literal string
- Number of characters in character class
- Number of files open
- Number of pipes open
- The ability to handle 8-bit characters and characters that are all zero (ASCII NUL)

gawk does not have limits on any of the above items, other than those imposed by the machine architecture and/or the operating system.

Group Listing of awk Functions and Commands

The following table classifies **awk** functions and commands.

Function type		Functions or commands			
Arithmetic	atan2 rand	cos sin	exp sqrt	int srand	log
String	asort^a length sub	asort^a match substr	gensub^a split tolower	gsub sprintf toupper	index strtonum^a
Control flow	break if/else	continue return	do/while while	exit	for

Function type			Functions or commands		
I/O	close	fflush ^b	getline	next	nextfile ^b
	print	printf			
Programming	extension ^a	delete	function	system	

^a Available in **gawk**.

^b Available in Bell Labs **awk** and **gawk**.

The following functions are specific to **gawk**.

Function type			Functions or commands		
Bit manipulation	and	compl	lshift	or	rshift
	xor				
Time Translation	mktime	strftime	systime		
	bindtextdomain	dgettext	dcgettext		

Alphabetical Summary of awk Functions and Commands

The following alphabetical list of keywords and functions includes all that are available in POSIX **awk** and **gawk**. Extensions that aren't part of POSIX **awk** but that are in both **gawk** and the Bell Laboratories **awk** are marked as {E}. Cases where **gawk** has extensions are marked as {G}. Items that aren't marked with a symbol are available in all versions.

#	#	Ignore all text that follows on the same line. # is used in awk scripts as the comment character and is not really a command.
and	and(<i>expr1, expr2</i>) {G}	Return the bitwise AND of <i>expr1</i> and <i>expr2</i> , which should be values that fit in a C unsigned long .
asort	asort(<i>src [,dest]</i>) {G}	Sort the array <i>src</i> based on the element values, destructively replacing the indices with values from one to the number of elements in the array. If <i>dest</i> is supplied, copy <i>src</i> to <i>dest</i> and sort <i>dest</i> , leaving <i>src</i> unchanged. Returns the number of elements in <i>src</i> .
asorti	asorti(<i>src [,dest]</i>) {G}	Like asort() , but the sorting is done based on the indices in the array, not based on the element values. For gawk 3.1.2 and later.

atan2	<code>atan2(y, x)</code>
	Return the arctangent of y/x in radians.
bindtextdomain	<code>bindtextdomain(dir [,domain]) {G}</code>
	Look in directory <i>dir</i> for message translation files for text domain <i>domain</i> (default: value of TEXTDOMAIN). Returns the directory where <i>domain</i> is bound.
break	<code>break</code>
	Exit from a while , for , or do loop.
close	<code>close(expr)</code> <code>close(expr, how) {G}</code>
	In most implementations of awk , you can only have up to ten files and one pipe open simultaneously. Therefore, POSIX awk provides a close() function that allows you to close a file or a pipe. It takes the same expression that opened the pipe or file as an argument. This expression must be identical, character by character, to the one that opened the file or pipe—even whitespace is significant.
	In the second form, close one end of either a TCP/IP socket or a two-way pipe to a coprocess. <i>how</i> is a string, either “from” or “to”. Case does not matter.
compl	<code>compl(expr) {G}</code>
	Return the bitwise complement of <i>expr</i> , which should be a value that fits in a C unsigned long .
continue	<code>continue</code>
	Begin next iteration of while , for , or do loop.
cos	<code>cos(x)</code>
	Return the cosine of <i>x</i> , an angle in radians.
dcgettext	<code>dcgettext(str [, dom [,cat]]) {G}</code>
	Return the translation of <i>str</i> for the text domain <i>dom</i> in message category <i>cat</i> . Default text domain is value of TEXTDOMAIN. Default category is "LC_MESSAGES".
dcngettext	<code>dcngettext(str1, str2, num [, dom [,cat]]) {G}</code>
	If <i>num</i> is one, return the translation of <i>str1</i> for the text domain <i>dom</i> in message category <i>cat</i> . Otherwise, return the translation of <i>str2</i> . Default text domain is value of TEXTDOMAIN. Default category is "LC_MESSAGES". For gawk 3.1.1 and later.

delete	<code>delete array[element]</code> <code>delete array {E}</code>	Delete <i>element</i> from <i>array</i> . The brackets are typed literally. The second form is a common extension, which deletes <i>all</i> elements of the array in one shot.
do	<code>do</code> <i>statement</i> <i>while (expr)</i>	Looping statement. Execute <i>statement</i> , then evaluate <i>expr</i> and if true, execute <i>statement</i> again. A series of statements must be put within braces.
exit	<code>exit [expr]</code>	Exit from script, reading no new input. The END procedure, if it exists, will be executed. An optional <i>expr</i> becomes awk 's return value.
exp	<code>exp(x)</code>	Return exponential of <i>x</i> (<i>ex</i>).
extension	<code>extension(lib, init) {G}</code>	Dynamically load the shared object file <i>lib</i> , calling the function <i>init</i> to initialize it. Return the value returned by the <i>init</i> function. This function allows you to add new built-in functions to gawk . See Arnold Robbins's <i>Effective awk Programming</i> (O'Reilly) for the details.
fflush	<code>fflush([output-expr]) {E}</code>	Flush any buffers associated with open output file or pipe <i>output-expr</i> . gawk extends this function. If no <i>output-expr</i> is supplied, it flushes standard output. If <i>output-expr</i> is the null string (""), it flushes all open files and pipes.
for	<code>for (init-expr; test-expr; incr-expr)</code> <i>statement</i>	C-style looping construct. <i>init-expr</i> assigns the initial value of a counter variable. <i>test-expr</i> is a relational expression that is evaluated each time before executing the <i>statement</i> . When <i>test-expr</i> is false, the loop is exited. <i>incr-expr</i> is used to increment the counter variable after each pass. All of the expressions are optional. A missing <i>test-expr</i> is considered to be true. A series of statements must be put within braces.

for	<pre>for (<i>item</i> in <i>array</i>) <i>statement</i></pre>
	Special loop designed for reading associative arrays. For each element of the array, the <i>statement</i> is executed; the element can be referenced by <i>array</i> [<i>item</i>]. A series of statements must be put within braces.
function	<pre>function <i>name</i>(<i>parameter-list</i>) { <i>statements</i> }</pre>
	Create <i>name</i> as a user-defined function consisting of awk <i>statements</i> that apply to the specified list of parameters. No space is allowed between <i>name</i> and the left parenthesis when the function is called.
gensub	<pre>gensub(<i>regex</i>, <i>str</i>, <i>how</i> [, <i>target</i>]) {G}</pre>
	General substitution function. Substitute <i>str</i> for matches of the regular expression <i>regex</i> in the string <i>target</i> . If <i>how</i> is a number, replace the <i>how</i> th match. If it is "g" or "G", substitute globally. If <i>target</i> is not supplied, \$0 is used. Return the new string value. The original <i>target</i> is <i>not</i> modified. (Compare with gsub and sub .) Use & in the replacement string to stand for the text matched by the pattern.
getline	<pre>getline getline [<i>var</i>] [< <i>file</i>] <i>command</i> getline [<i>var</i>] <i>command</i> & getline [<i>var</i>] {G}</pre>
	Read next line of input. The second form reads input from <i>file</i> , and the third form reads the output of <i>command</i> . All forms read one record at a time, and each time the statement is executed, it gets the next record of input. The record is assigned to \$0 and is parsed into fields, setting NF, NR and FNR. If <i>var</i> is specified, the result is assigned to <i>var</i> and \$0 and NF are not changed. Thus, if the result is assigned to a variable, the current record does not change. getline is actually a function, and it returns 1 if it reads a record successfully, 0 if end-of-file is encountered, and -1 if for some reason it is otherwise unsuccessful. The fourth form reads the output from coprocess <i>command</i> . See the section “Coprocesses and Sockets” on page 735 for more information.
gsub	<pre>gsub(<i>regex</i>, <i>str</i> [, <i>target</i>])</pre>
	Globally substitute <i>str</i> for each match of the regular expression <i>regex</i> in the string <i>target</i> . If <i>target</i> is not supplied, defaults to \$0. Return the number of substitutions. Use & in the replacement string to stand for the text matched by the pattern.

if	<code>if (<i>condition</i>) <i>statement1</i> [else <i>statement2</i>]</code>	If <i>condition</i> is true, do <i>statement1</i> ; otherwise do <i>statement2</i> in optional else clause. The <i>condition</i> can be an expression using any of the relational operators <, <=, ==, !=, >=, or >, as well as the array membership operator in , and the pattern-matching operators ~ and !~ (e.g., if (\$1 ~ /[Aa].*/)). A series of statements must be put within braces. Another if can directly follow an else in order to produce a chain of tests or decisions.
index	<code>index(<i>str</i>, <i>substr</i>)</code>	Return the position (starting at 1) of <i>substr</i> in <i>str</i> , or zero if <i>substr</i> is not present in <i>str</i> .
int	<code>int(<i>x</i>)</code>	Return integer value of <i>x</i> by truncating any fractional part.
length	<code>length([<i>arg</i>])</code>	Return length of <i>arg</i> , or the length of \$0 if no argument.
log	<code>log(<i>x</i>)</code>	Return the natural logarithm (base <i>e</i>) of <i>x</i> .
lshift	<code>lshift(<i>expr</i>, <i>count</i>) {G}</code>	Return the result of shifting <i>expr</i> left by <i>count</i> bits. Both <i>expr</i> and <i>count</i> should be values that fit in a C unsigned long .
match	<code>match(<i>str</i>, <i>regex</i>) match(<i>str</i>, <i>regex</i> [, <i>array</i>]) {G}</code>	Function that matches the pattern, specified by the regular expression <i>regex</i> , in the string <i>str</i> and returns either the position in <i>str</i> where the match begins, or 0 if no occurrences are found. Sets the values of RSTART and RLENGTH to the start and length of the match, respectively. If <i>array</i> is provided, gawk puts the text that matched the entire regular expression in <i>array[0]</i> , the text that matched the first parenthesized subexpression in <i>array[1]</i> , the second in <i>array[2]</i> , and so on.
mktime	<code>mktime(<i>timespec</i>) {G}</code>	Turns <i>timespec</i> (a string of the form YYYY MM DD HH MM SS[DST] representing a local time) into a time-of-day value in seconds since midnight, January 1, 1970, UTC.

next	<code>next</code> Read next input line and start new cycle through pattern/procedures statements.
nextfile	<code>nextfile {E}</code> Stop processing the current input file and start new cycle through pattern/procedures statements, beginning with the first record of the next file.
or	<code>or(expr1, expr2) {G}</code> Return the bitwise OR of <i>expr1</i> and <i>expr2</i> , which should be values that fit in a C <code>unsigned long</code> .
print	<code>print [output-expr[, ...]] [dest-expr]</code> Evaluate the <i>output-expr</i> and direct it to standard output followed by the value of ORS. Each comma-separated <i>output-expr</i> is separated in the output by the value of OFS. With no <i>output-expr</i> , print \$0. The output may be redirected to a file or pipe via the <i>dest-expr</i> , which is described in “Output Redirections” on page 746.
printf	<code>printf(format [, expr-list]) [dest-expr]</code> An alternative output statement borrowed from the C language. It has the ability to produce formatted output. It can also be used to output data without automatically producing a newline. <i>format</i> is a string of format specifications and constants. <i>expr-list</i> is a list of arguments corresponding to format specifiers. As for print , output may be redirected to a file or pipe. See “printf Formats” on page 746 for a description of allowed format specifiers. Like any string, <i>format</i> can also contain embedded escape sequences: \n (newline) or \t (tab) being the most common. Spaces and literal text can be placed in the <i>format</i> argument by quoting the entire argument. If there are multiple expressions to be printed, there should be multiple formats specified.
Examples	
Using the script:	
<code>{ printf("The sum on line %d is %.of.\n", NR, \$1+\$2) }</code>	
The following input line:	
5 5	
produces this output, followed by a newline:	
The sum on line 1 is 10.	
rand	<code>rand()</code> Generate a random number between 0 and 1. This function returns the same series of numbers each time the script is executed, unless the random number generator is seeded using <code>rand()</code> .

return	<code>return [expr]</code>
	Used within a user-defined function to exit the function, returning the value of <i>expr</i> . The return value of a function is undefined if <i>expr</i> is not provided.
rshift	<code>rshift(expr, count) {G}</code>
	Return the result of shifting <i>expr</i> right by <i>count</i> bits. Both <i>expr</i> and <i>count</i> should be values that fit in a C unsigned long .
sin	<code>sin(x)</code>
	Return the sine of <i>x</i> , an angle in radians.
split	<code>split(string, array [, sep])</code>
	Split <i>string</i> into elements of array <i>array[1],...,array[n]</i> . Return the number of array elements created. The string is split at each occurrence of separator <i>sep</i> . If <i>sep</i> is not specified, FS is used.
sprintf	<code>sprintf(format [, expressions])</code>
	Return the formatted value of one or more <i>expressions</i> , using the specified <i>format</i> . Data is formatted but not printed. See “printf Formats” on page 746 for a description of allowed format specifiers.
sqrt	<code>sqrt(arg)</code>
	Return the square root of <i>arg</i> .
 srand	<code>srand([expr])</code>
	Use optional <i>expr</i> to set a new seed for the random number generator. Default is the time of day. Return value is the old seed.
strftime	<code>strftime([format [,timestamp]]) {G}</code>
	Format <i>timestamp</i> according to <i>format</i> . Return the formatted string. The <i>timestamp</i> is a time-of-day value in seconds since midnight, January 1, 1970, UTC. The <i>format</i> string is similar to that of sprintf . If <i>timestamp</i> is omitted, it defaults to the current time. If <i>format</i> is omitted, it defaults to a value that produces output similar to that of the Unix date command. See the date entry in Chapter 3 for a list.
strtonum	<code>strtonum(expr) {G}</code>
	Return the numeric value of <i>expr</i> , which is a string representing an octal, decimal, or hexadecimal number in the usual C notations. Use this function for processing nondecimal input data.

sub `sub(regex, str [, target])`
 Substitute *str* for first match of the regular expression *regex* in the string *target*. If *target* is not supplied, defaults to **\$0**. Returns 1 if successful, 0 otherwise. Use **&** in the replacement string to stand for the text matched by the pattern.

substr `substr(string, beg [, len])`
 Return substring of *string* at beginning position *beg* (counting from 1), and the characters that follow to maximum specified length *len*. If no length is given, use the rest of the string.

system `system(command)`
 Function that executes the specified *command* and returns its exit status. The status of the executed command typically indicates success or failure. A value of 0 means that the command executed successfully. A nonzero value indicates a failure of some sort. The documentation for the command will give you the details.
awk does *not* make the output of the command available for processing within the **awk** script. Use *command* | **getline** to read the output of a command into the script.

systime `systime() {G}`
 Return a time-of-day value in seconds since midnight, January 1, 1970, UTC.

Examples

Log the start and end times of a data-processing program:

```
BEGIN {
    now = systime()
    mesg = strftime("Started at %m/%d/%Y %H:%M:%S",
now)
    print mesg
}
process data ...
END {
    now = systime()
    mesg = strftime("Ended at %m/%d/%Y %H:%M:%S",
now)
    print mesg
}
```

tolower `tolower(str)`
 Translate all uppercase characters in *str* to lowercase and return the new string.*

* Very early versions of **nawk** don't support **tolower()** and **toupper()**. However, they are now part of the POSIX specification for **awk**.

toupper	<code>toupper(str)</code>
	Translate all lowercase characters in <i>str</i> to uppercase and return the new string.
while	<code>while (condition) statement</code>
	Do <i>statement</i> while <i>condition</i> is true (see if for a description of allowable conditions). A series of statements must be put within braces.
xor	<code>xor(expr1, expr2) {G}</code>
	Return the bitwise XOR of <i>expr1</i> and <i>expr2</i> , which should be values that fit in a C unsigned long .

Output Redirections

For **print** and **printf**, *dest-expr* is an optional expression that directs the output to a file or pipe.

`> file`

Direct the output to a file, overwriting its previous contents.

`>> file`

Append the output to a file, preserving its previous contents. In both this case and the `> file` case, the file will be created if it does not already exist.

`| command`

Direct the output as the input to a system command.

`|& command`

Direct the output as the input to a coprocess. **gawk** only.

Be careful not to mix `>` and `>>` for the same file. Once a file has been opened with `>`, subsequent output statements continue to append to the file until it is closed.

Remember to call **close()** when you have finished with a file, pipe, or coprocess. If you don't, eventually you will hit the system limit on the number of simultaneously open files.

printf Formats

Format specifiers for **printf** and **sprintf** have the following form:

`%[posn$][flag][width][.precision]letter`

The control letter is required. The format-conversion control letters are given in the following table.

Character	Description
c	ASCII character.
d	Decimal integer.
i	Decimal integer (added in POSIX).
e	Floating-point format $([-]d,precisione[+-]dd)$.
E	Floating-point format $([-]d,precisionE[+-]dd)$.
f	Floating-point format $([-]ddd.precision)$.
g	e or f conversion, whichever is shortest, with trailing zeros removed.
G	E or f conversion, whichever is shortest, with trailing zeros removed.
o	Unsigned octal value.
s	String.
u	Unsigned decimal value.
x	Unsigned hexadecimal number. Uses a-f for 10 to 15.
X	Unsigned hexadecimal number. Uses A-F for 10 to 15.
%	Literal %.

gawk allows you to provide a *positional specifier* after the % (*posn\$*). A positional specifier is an integer count followed by a \$. The count indicates which argument to use at that point. Counts start at one and don't include the format string. This feature is primarily for use in producing translations of format strings. For example:

```
$ gawk 'BEGIN { printf "%2$s, %1$s\n", "world", "hello" }'
hello, world
```

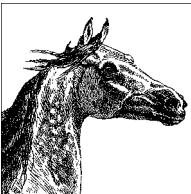
The optional *flag* is one of the following.

Character	Description
-	Left-justify the formatted value within the field.
space	Prefix positive values with a space and negative values with a minus.
+	Always prefix numeric values with a sign, even if the value is positive.
#	Use an alternate form: %o has a preceding 0; %x and %X are prefixed with 0x and 0X, respectively; %e, %E and %f always have a decimal point in the result; and %g and %G do not have trailing zeros removed.
0	Pad output with zeros, not spaces. This only happens when the field width is wider than the converted result. This flag applies to all output formats, even nonnumeric ones.
,	gawk 3.1.4 and later only. For numeric formats, in locales that support it, supply a thousands-separator character.

The optional *width* is the minimum number of characters to output. The result will be padded to this size if it is smaller. The 0 flag causes padding with zeros; otherwise, padding is with spaces.

The *precision* is optional. Its meaning varies by control letter, as shown in the following table.

Conversion	Precision means
<code>%d, %i, %o, %u, %x, %X</code>	The minimum number of digits to print.
<code>%e, %E, %f</code>	The number of digits to the right of the decimal point.
<code>%g, %G</code>	The maximum number of significant digits.
<code>%s</code>	The maximum number of characters to print.



12

Source Code Management: An Overview

Chapters 13 and 14 describe two popular source code management systems for Linux: Subversion (SVN) and Git. This chapter introduces the major concepts involved with using these systems for users who may never have used one. If you're already familiar with source code management, feel free to skip ahead to the particular software suite that interests you.

This chapter covers the following topics:

- Introduction and terminology
- Usage models
- Source code management systems
- Other source code management systems

Introduction and Terminology

Source code management systems let you store and retrieve multiple versions of a file. While originally designed for program source code, they can be used for any kind of file: source code, documentation, configuration files, and so on. Modern systems allow you to store binary files as well, such as image or audio data.

Source code management systems let you compare different versions of a file, as well as do “parallel development.” In other words, you can work on two different versions of a file at the same time, with the source code management system storing both versions. You can then merge changes from two versions into a third version. This will become more clear shortly. We’ll start by defining some terms:

Repository

A *repository* is where the source code management system stores its copy of your file. Usually one file in the source code management system is used to hold all the different versions of a source file. Each source code management system uses its own format to allow it to retrieve different versions easily and to track who made what changes, and when.

Sandbox

A *sandbox* is your personal, so-called “working copy” of the program or set of documents under development. You edit your private copy of the file in your own sandbox, returning changes to the source code management system when you’re satisfied with the new version.

Check in, check out

You “check out” files from the repository, edit them, and then “check them in” when you’re satisfied with your changes. Other developers working against the same repository will not see your changes until after you check them back in. Another term used for check-in is *commit*.

Log message

Every time you check in a file, you are prompted for a message describing the changes you made. You should do so in a concise fashion. If your software development practices include the use of a bug tracking system, you might also wish to include the bug number or problem report (PR) number that your change resolves.

Keyword substitutions

When you check out a file, the source code management system can replace special *keywords* with values representing such things as the file’s version number, the name of the user who made the most recent change, the date and time the file was last changed, the file’s name, and so on. Each of the systems described in this book uses an overlapping set of keywords. Some systems always do keyword substitution, while others require that you explicitly enable the feature for each file.

Branch

A *branch* is a separate development path. For example, once you’ve released version 1.0 of **whizprog**, you will wish to proceed with the development for version 2.0. The main line of development is often called the *trunk*.

Now consider what happens when you wish to make a bug-fix release to **whizprog** 1.0, to be named version 1.1. You create a separate branch, based on the original 1.0 code, in a new sandbox. You perform all your development *there*, without disturbing the development being done for the 2.0 release.

Tag

A *tag* is a name you give to a whole group of files at once, at whatever version each individual file may be, in order to identify those files as being part of a particular group. For example, you might create tags **WHIZPROG-1_0-ALPHA**, **WHIZPROG-1_0-BETA**, **WHIZPROG-1_0-RELEASE** and so on. This is a powerful facility that should be used well, since it allows you to retrieve a “snapshot” of your entire development tree as it existed at different points in time.

Merging

Most typically, when development along a branch is completed, it becomes necessary to *merge* the changes from that branch back into the main line of development. In our hypothetical example, all the bugs fixed in **whizprog** 1.0 to create version 1.1 should also be fixed in the ongoing 2.0 development. Source code management systems can help you automate the process of merging.

Conflict

A *conflict* occurs when two developers make inconsistent changes to the same part of a source file. Modern source code management systems detect the conflict, usually marking the conflicting parts of the file in your working copy using special markers. You first discuss the conflict with the other developer, in order to arrive at a correct resolution of the conflict. Once that's done, you then resolve the conflict manually (by making the appropriate changes) and check in the new version of the file.

Client/Server

As with other “client/server” networking models, the idea here is that the repository is stored on one machine, the *server*, and that different developers may access the repository from multiple *client* systems. This powerful feature facilitates distributed development, allowing developers to work easily on their local systems, with the repository kept in a central place where it can be easily accessed and administered.

Usage Models

Different systems have different conceptual “models” as to how they’re used.

Older systems such as SCCS and RCS use a “check out with locking” model. These systems were developed before client/server computing, when software development was done on centralized minicomputers and mainframes. In this model, the repository is a central directory on the same machine where the developers work, and each developer checks out a private copy into their own sandbox. In order to avoid two developers making conflicting changes to a file, the file must be *locked* when it’s checked out. Only one user may lock a particular version of a file at a time. When that user has checked in their changes, they *unlock* the file so that the next user can check in changes. If necessary, the second user may “break” the first user’s lock, in which case the first user is notified via electronic mail.

This model works well for small projects where developers are co-located and can communicate easily. As long as one developer locks a file when she checks it out, another developer wishing to work with the file will know that he can’t until the first one is done. The drawback is that such locking can slow down development significantly.

Newer systems, such as CVS and Subversion, use a “copy, modify, merge” model. In practice, when two developers wish to work on the same file, they usually end up changing different, unrelated parts of the file. Most of the time, each developer can make changes without adversely affecting the other. Thus, files are not locked upon checkout into a sandbox. Instead, the source code management system detects conflicts and disallows a check-in when conflicts exist.

For example, consider two developers, *dangermouse* and *penfold*, who are both working on *whizprog.c*. They each start with version 1.4 of the file. *dangermouse* commits his changes, creating version 1.5. Before *penfold* can commit his changes, the source code management system notices that the file has changed in the repository. *penfold* must first merge *dangermouse*’s changes into his working copy.

If there are no conflicts, he can then commit his changes, creating version 1.6. On the other hand, if there are conflicts, he must first resolve them (they'll be marked in the working copy), and only then may he commit his version.

The combination of the “copy, modify, merge” model with a networked client/server facility creates a powerful environment for doing distributed development. Developers no longer have to worry about file locks. Because the source code management system enforces serialization (making sure that new changes are based on the latest version in the repository), development can move more smoothly, with little danger of miscommunication or that successive changes will be lost.

Source Code Management Systems

There are several source code management systems used in the Unix community:

SCCS

The Source Code Control System. SCCS is the original Unix source code management system. It was developed in the late 1970s for the Programmer’s Workbench (PWB) Unix systems within Bell Labs. It is still in use at a few large longtime Unix sites. However, for a long time it was not available as a standard part of most commercial or BSD Unix systems, and it did not achieve the widespread popularity of other, later systems. (It is still available with Solaris.) SCCS uses a file storage format that allows it to retrieve any version of a source file in constant time.

RCS

The Revision Control System. RCS was developed in the early 1980s at Purdue University by Walter F. Tichy. It became popular in the Unix world when it was shipped with 4.2 BSD in 1983. At the time, Berkeley Unix was the most widely used Unix variant, even though to get it, a site had to have a Unix license from AT&T.

RCS is easier to use than SCCS. Although it has a number of related commands, only three or four are needed for day-to-day use, and they are quickly mastered. A central repository is easy to use: you first create a directory for the sandbox. In the sandbox, you make a symbolic link to the repository named *RCS*, and then all the developers can share the repository. RCS uses a file format that is optimized for retrieving the most recent version of a file.

CVS

The Concurrent Versions System. CVS was initially built as a series of shell scripts sitting atop RCS. Later it was rewritten in C for robustness, although still using RCS commands to manage the storage of files. However, for quite some time, CVS has had the RCS functionality built into it, and it no longer requires that RCS be available. The file format continues to be the same. CVS was the first distributed source code management system and was until recently the standard one for Unix systems—in particular for collaborative, distributed, free and open source development projects.

The repository is named when you create a sandbox and is then stored in the files in the sandbox, so that it need not be provided every time you run a CVS command. Unlike SCCS and RCS, which provide multiple commands, CVS has one main command (named `cvs`), which you use for just about every operation.

Subversion

With increasing use, it became clear that CVS lacked some fundamental capabilities. The Subversion project was started by several longtime CVS users and developers with the explicit goal to “build a better CVS,” not necessarily to explore uncharted territory in source code management systems. Subversion is thus intentionally easy to learn for CVS users. Subversion uses its own format for data storage, based on the Berkeley DB in-process data library. Distributed use was designed in from day one, providing useful facilities that leverage the capabilities of the well-known Apache HTTP server.

Git

Git is a source control system originally developed by Linus Torvalds as the source control system to manage the Linux kernel. Today, it is maintained by a community of developers and used for many diverse projects. Unlike the previous source control systems, Git is distributed. A distributed system differs from a centralized one such as CVS in several ways. The most notable is that in distributed systems, individual check outs of the source tree, called clones in distributed systems, are themselves complete and fully functioning repositories. Instead of submitting changes to a centralized server, changes are pushed and pulled among repositories.

RCS, CVS, and Subversion represent a progression, each one building on the features of its predecessors. For example, all three share a large subset of the same keyword substitutions, and command names are similar or identical in all three. They also demonstrate the progression from locking-based development to conflict-resolution-based development. Git is a radical departure from the centralized model.

Other Source Code Management Systems

Besides the source code management systems covered in this book, several other systems are worth knowing about. The following list, though, is by no means exhaustive:

Arch

GNU Arch is a distributed source code management system similar to CVS and Subversion. One of its significant strengths is that you can do offline development with it, working on multiple versions even on systems that are not connected to the Internet and that cannot communicate with the central repository. For more information, see <http://www.gnu.org/software/gnu-arch/>.

Codeville

Codeville is a distributed version-control system in the early stages of development. It is written in Python, is easy to set up and use, and shows a lot of promise. For more information, see <http://codeville.org/>.

CSSC

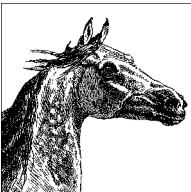
CSSC is a free clone of SCCS. It intends to provide full compatibility with SCCS, including file format, command names and options, and “bug for bug” compatible behavior. If you have an existing SCCS repository, you should be able to drop CSSC into your environment, in place of SCCS. CSSC can be used to migrate from a commercial Unix system to a freely available clone, such as GNU/Linux or a BSD system. For more information, see <http://directory.fsf.org/project/cssc>.

Monotone

The web page for **monotone** describes it well:

monotone is a free distributed version control system. It provides a simple, single-file transactional version store, with fully disconnected operation and an efficient peer-to-peer synchronization protocol. It understands history-sensitive merging, lightweight branches, integrated code review, and third party testing. It uses cryptographic version naming and client-side RSA certificates. It has good internationalization support, has no external dependencies, runs on [Linux, Solaris, Mac OS X, NetBSD, and Windows], and is licensed under the GNU GPL.

For more information, see <http://monotone.ca>.



13

The Subversion Version Control System

The Subversion version control system is a powerful, open source system for management of file and directory versions. Designed from the ground up to support distributed development, it offers many leading-edge features.

This chapter covers the following topics:

- Conceptual overview
- Using Subversion: a quick tour
- The Subversion command line client: **svn**
- Repository administration: **svnadmin**
- Examining the repository: **svnlook**
- Providing remote access: **svnserve**

Version control was introduced in Chapter 12, which contains a comparison of Subversion, GIT, and other popular systems. The material in the current chapter is adapted from *Version Control with Subversion*, Second Edition, by C. Michael Pilato et al. (O'Reilly). See that book for much more information on Subversion.

Conceptual Overview

Subversion is a version-control system. It lets you track changes to an entire project directory tree. Every change made to the tree is recorded and can be retrieved.

Basic Version-Control Operations

Project data is kept in a *repository*, a set of directories and files managed by Subversion. Users use the **svn** client program to access the repository and make changes to it.

Subversion uses the *copy-modify-merge* development model. You make a private copy of a given project in a *sandbox*. (This is often called *checking out* a copy.) This private copy is not locked in the repository. You then make all the changes you like to the copy within the sandbox, without having to worry about what other developers are doing. As you work, you can compare your changes to the version you started with, as well as the version currently in the repository. Once you're satisfied with the changes, you *commit* them, sometimes referred to as a *check-in*.

In the event that another developer has modified part of a file that you were working on and checked it in, when you commit your changes, Subversion notices and indicates that a *conflict* exists. Conflicts are marked as such in the file, and Subversion creates pristine copies of the file as it exists in the repository and of the file as you modified it, so that you can do full comparisons. Once you have resolved the conflict, you tell Subversion about it, and then commit the final version.

Subversion lets you create a development *branch*, a separate stream of development versions. You can periodically merge changes from the main development stream (the *trunk*) into your branch, and also merge changes from your branch back into the trunk.

Finally, you can *tag* a particular copy of the project. For instance, when a project is ready for a release, you can create a snapshot of the project and give it a descriptive tag that allows you to re-create the project tree exactly as it was for the release. This is particularly valuable when you need to produce a bug fix for an older version of the project, or when you have to attempt to retrofit a fix or feature from current development into an older version.

Key Features

Directory versioning

Subversion implements a virtual versioned filesystem that tracks changes to whole directory trees over time. Files *and* directories are versioned. Because it tracks the history of the directory tree rather than just the files, you can add, delete, copy, and rename both files and directories. Every newly added file begins with a fresh, clean history all its own, even if the filename was previously used.

Atomic commits

A collection of modifications either goes into the repository completely, or not at all. This allows developers to construct and commit changes as logical chunks, and prevents problems that can occur when only a portion of a set of changes is successfully sent to the repository.

Versioned metadata

Each file and directory has a set of properties—keys and their values—associated with it. You can create and store any arbitrary key/value pairs. Properties are versioned over time, just like file contents.

Choice of network layers

Subversion has an abstracted notion of repository access, making it easy for people to implement new network mechanisms. Subversion can plug into the

Apache HTTP Server as an extension module. A more lightweight, stand-alone Subversion server process is also available. This server speaks a custom protocol that can be easily tunneled over SSH.

Consistent data handling

Subversion expresses file differences using a binary differencing algorithm, which works identically on both text (human-readable) and binary (human-unreadable) files. Both types of files are stored equally compressed in the repository, and only the differences are transmitted in both directions across the network.

Efficient branching and tagging

The cost of branching and tagging need not be proportional to the project size. Subversion creates branches and tags by simply copying the project, using a mechanism similar to a hard link. Thus these operations take only a very small, constant amount of time.

Hackability

Subversion is implemented as a collection of shared C libraries with well-defined APIs. This makes Subversion extremely maintainable and usable by other applications and languages. Subversion is an open source project as well. You can contribute to its development.

Special File Properties

Subversion allows you to associate *properties* with files or directories. A property is just a keyword/value pair associated with the file. Subversion reserves property names starting with **svn:** for its own use. The special properties in Subversion 1.5.4 are:

svn:author

The username of the person who committed a particular revision.

svn:date

The server time when the transaction for a revision was created.

svn:eol-style

Different operating systems use different conventions to mark the end of lines in text files. It should be set to one of the following values:

CR Clients should always use carriage return (CR) line terminators, no matter what the native format is.

CRLF

Clients should always use carriage return and line feed (CR-LF) line terminators, no matter what the native format is.

LF Clients should always use linefeed (LF) line terminators, no matter what the native format is.

native

Clients should use the native format when checking out files.

Subversion always stores files in normalized, LF-only format in the repository.

svn:executable

Valid only for files. It indicates that the file should be made executable when it's checked out or updated from the repository. It has no effect on filesystems, such as FAT-32 or NTFS, that don't have the concept of an execute bit.

svn:externals

This property, when set on a directory under version control, allows you to specify other, external repositories to use for particular local subdirectories. You set this property with **svn propset** or **svn propedit** (see “**svn Subcommands**” on page 766). The value is a multiline table of directories and fully qualified Subversion URLs. For example:

```
$ svn propget svn:externals calc
third-party/sounds          http://sounds.red-bean.com/repos
third-party/skins            http://skins.red-bean.com/repositories/
skinproj
third-party/skins/toolkit -r21 http://svn.red-bean.com/repos/skin-maker
```

Once set, anyone else who checks out a working copy will also get the third party files checked out automatically.

svn:ignore

Used to tell Subversion to not place certain file types under version control. This property is set on a directory and should contain a list of file patterns that certain Subversion operations (like **svn status**, **svn add**, and **svn import**) will ignore.

svn:keywords

A list of keywords for which Subversion should perform *keyword expansion* when checking out the file. The valid keywords are listed below.

svn:log

The log message associated with the commit of a particular revision.

svn:mime-type

An indication of the type of data stored in the file. This prevents an attempt to perform a “merge” on data that can’t be merged. This property also influences how the Subversion Apache module sets the HTTP **Content-type:** header. In general, if it does not begin with **text/**, Subversion assumes that the file contains binary data. For updates, this causes Subversion to rename a modified working copy of the file with a **.orig** extension and replace the file with the current version from the repository.

svn:realmstring

A specialized property that describes the “authentication realm” for a file in Subversion’s cached copy of the authentication credentials. See Chapter 6 of *Version Control with Subversion* for more information.

Valid subversion keywords

Subversion defines the list of keywords available for substitution. That list contains the following five keywords, some of which have shorter aliases that you can also use:

Date

This keyword describes the last time the file was changed in the repository and looks like **\$Date: 2009-02-23 21:42:37 -0700 (Mon, 23 Feb 2009) \$**. It may also be given as **LastChangedDate**.

Revision

This keyword describes the last revision in which this file changed in the repository and looks like `$Revision$`. It may also be given as `LastChanged-Revision` or abbreviated as `Rev`.

Author

This keyword describes the last user to change this file in the repository, and looks like `$Author$`. It may be given as `LastChangedBy`.

HeadURL

This keyword describes the full URL to the latest version of the file in the repository. It looks like `$HeadURL: http://svn.collab.net/repos/trunk/README $`. It may be abbreviated as `URL`.

- Id** This keyword is a compressed combination of the other keywords. Its substitution looks like `$Id: ch14.xml,v 1.5 2005/08/12 21:21:32 sally Exp sally $`, and is interpreted to mean that the file `calc.c` was last changed in revision 148 on the evening of July 28, 2005 by the user `sally`.

Obtaining Subversion

The Subversion project website is <http://subversion.tigris.org>. It contains links to project documentation, Frequently Asked Questions (FAQs), and project source code.

Most GNU/Linux systems come with Subversion available on the installation CDs. Thus, you may be able to install a prebuilt binary for your system or use a package manager to download and install it.

Using Subversion: A Quick Tour

This section provides a very quick tour of using Subversion for version control. We start with the initial version of a project for importing into Subversion:

```
$ find /tmp/hello -print      Show directory layout
/tmp/hello
/tmp/hello/branches          Directory for branch development
/tmp/hello/tags               Directory for tagged releases
/tmp/hello/trunk
/tmp/hello/trunk/hello.c     Mainline development is done on the trunk
/tmp/hello/trunk/Makefile
/tmp/hello/trunk/README
```

The next steps are to create the repository and then to import the project into it:

```
$ svnadmin create /path/to/svnrepos
$ svn import /tmp/hello file:///path/to/svnrepos -m "initial import"
Adding      /tmp/hello/trunk
Adding      /tmp/hello/trunk/hello.c
Adding      /tmp/hello/trunk/Makefile
Adding      /tmp/hello/trunk/README
Adding      /tmp/hello/branches
Adding      /tmp/hello/tags
```

Committed revision 1.

Now that the project exists in Subversion, we check out a working copy into a sandbox underneath our home directory and start making changes:

```
$ cd                                         Move to home directory
$ svn checkout file:///path/to/svnrepos hello   Check out working copy
A  hello/trunk
A  hello/trunk/hello.c
A  hello/trunk/README
A  hello/trunk/Makefile
A  hello/branches
A  hello/tags
Checked out revision 1.

$ cd hello/trunk                                Change to sandbox
$ vi message.c hello.c Makefile                 Make changes
3 files to edit

$ cat message.c                                    Show newly created file
const char message[ ] = "hello, world!";
$ make                                         Compile program and test it
cc    -c -o hello.o hello.c
cc    -c -o message.o message.c
cc -O hello.o message.o -o hello
$ hello
hello, world!
```

One of the most common operations is to compare the changed copy with the original. The result is in *unified diff* format, the equivalent of the regular **diff -u** command:

```
$ svn diff hello.c
Index: hello.c
=====
=====
--- hello.c      (revision 1)
+++ hello.c      (working copy)
@@ -1,7 +1,9 @@
#include <stdio.h>

+extern const char message[ ];
+
int main(void)
{
-    printf("hello, world!\n");
+    printf("%s\n", message);
        return 0;
}
```

Now that we're comfortable with the changes, we schedule the new file, *message.c*, for addition to the repository, and then we actually commit our changes:

```
$ svn add message.c                           Schedule message.c for addition
A          message.c
$ svn commit                                     Commit all the changes
Sending      trunk/Makefile
Sending      trunk/hello.c
```

```
Adding          trunk/message.c
Transmitting file data ...
Committed revision 2.
```

Finally, we can view *all* our changes relative to the initial revision:

```
$ svn diff -r 1
Index: hello.c
=====
--- hello.c      (revision 1)
+++ hello.c      (working copy)
@@ -1,7 +1,9 @@
#include <stdio.h>

+extern const char message[  ];
+
int main(void)
{
-    printf("hello, world!\n");
+    printf("%s\n", message);
        return 0;
}
Index: Makefile
=====
===
--- Makefile      (revision 1)
+++ Makefile      (working copy)
@@ -1,2 +1,2 @@
-hello: hello.c
-    $(CC) -O $< -o $@
+hello: hello.o message.o
+    $(CC) -O hello.o message.o -o $@
Index: message.c
=====
===
--- message.c    (revision 0)
+++ message.c    (revision 2)
@@ -0,0 +1 @@
+const char message[  ] = "hello, world!";
```

The Subversion Command Line Client: **svn**

The syntax for the Subversion command line client, **svn**, consists of options and subcommand. **svn**'s *options* and *subcommand* may be provided in any order.

Common **svn** Options

While Subversion subcommands have different valid options, all options mean the same thing regardless of the subcommand you use it with. For example, **--verbose** (**-v**) always means verbose output, regardless of the subcommand you use it with.

- accept arg**
Specify action for automatic conflict resolution. Possible actions are postpone, base, mine-full, theirs-full, edit, and launch.
- auto-props**
Automatically set properties an newly added or imported files, overriding the **enable-auto-props** directive in the *config* file. By default this is disabled.
- change arg, -c arg**
Apply subcommand to specified change (a.k.a. revision.) This can be used as shorthand for "-r *arg-1:arg*".
- changelist name, -cl name**
Limit subcommand to files belonging to changelist *name*. You use the **changelist** subcommand to name the set of files to which you are making changes. A file can only belong to one changelist at a time. Changelists are local and are not saved in the repository. The name is usually discarded after committing. This option can be repeated to include more than one set of files.
- config-dir dir**
Read configuration information from the specified directory instead of the default location (*.subversion* in the user's home directory).
- depth arg**
Control the tree-depth to which the subcommand should be recursively applied. When used with the checkout command this will set the depth property of the checked out files as well, affecting what you receive and what future commands will affect in the repository. To change this "ambient depth" use the **--set-depth** option. This option replaces the **--recursive** and **--non-recursive** options. *arg* may be one of the following:
 - empty**
Apply to specified target only.
 - files**
Apply to the immediate file children of the target.
 - immediates**
Apply to the immediate file and directory children of target.
 - infinity**
Apply recursively to all file and directory children of target.
- diff-cmd cmd**
Use *cmd* as the external program to show differences between files instead of Subversion's internal **diff** engine. (Use **--extensions** to pass options to the external **diff** program.)
- diff3-cmd cmd**
Use *cmd* as the external program to merge files.
- dry-run**
Run a command, but make no actual changes—either on disk or in the repository.

--editor-cmd *cmd*

Use *cmd* as the program for editing a log message or a property value. If not set, Subversion checks the environment variable SVN_EDITOR, the runtime configuration (usually *~/.subversion/config*), then environment variables VISUAL and EDITOR for the name of the editor to use.

--encoding *enc*

Use *enc* as the encoding for the commit message. The default encoding is your operating system's native locale, and you should specify the encoding if your commit message is in any other encoding.

--extensions *args*, **-x** *args*

Pass *args* to an external **diff** command when providing differences between files. To pass multiple arguments, enclose all of them in quotes (for example, **svn diff --diff-cmd /usr/bin/diff -x "-b -E"**). This option can be used *only* if you also pass the **--diff-cmd** option.

--file *filename*, **-F** *filename*

Use the contents of *filename* for the specified subcommand. How it's used depends on the subcommand.

--force

Force a particular command or operation to run. There are some operations that Subversion prevents you from doing in normal usage, but you can pass this option to tell Subversion that you know what you're doing, as well as the possible repercussions of doing it, so do it anyway. Use with caution.

--force-log

Force a suspicious parameter passed to the **--message** (**-m**) or **--file** (**-F**) options to be accepted as valid. This can be used to pass a versioned file as the source for the commit log message, something Subversion would usually consider a mistake and reject.

--help, **-h**, **-?**

If used with one or more subcommands, show the built-in help text for each subcommand. If used alone, display the general client help text.

--ignore-ancestry

Ignore ancestry when calculating differences (i.e., rely on path contents alone).

--ignore-externals

Ignore external definitions and external working copies managed by them.

--incremental

Print output in a format suitable for concatenation.

--keep-changelists

Don't delete the changelist association after committing.

--keep-local

Keep the local copy of a file when using the **delete** subcommand to remove a file from the repository.

--limit *num*, **-l** *num*

Only show the first *num* log messages.

--message message, -m message

Use *message* as the commit message. For example:

```
$ svn commit -m "They don't make Sunday."
```

--native-eol format

Used with the **export** subcommand sets the end of line marker to use for all files with the **svn:eol-style** property set to **native**. You can specify LR, CR or CRLF for *format*.

--new arg

Use *arg* as the newer target when producing a diff.

--no-auth-cache

Do not cache authentication information (e.g., username and password) in the Subversion administrative directories.

--no-auto-props

Disable auto-props, overriding the **enable-auto-props** directive in the *config* file.

--no-diff-deleted

Do not print differences for deleted files. The default behavior when you remove a file is for **svn diff** to print the same differences that you would see if you had left the file but removed all the content.

--no-ignore

Show files in the status listing that would normally be omitted because they match a pattern in the **svn:ignore** property.

--no-unlock

Do not unlock files on commit.

--non-interactive

In the case of an authentication failure, or insufficient credentials, do not prompt for credentials (e.g., username or password). This is useful if you're running Subversion inside of an automated script where it's better to have Subversion fail instead of trying to prompt for more information.

--non-recursive, -N

Stop a subcommand from recursing into subdirectories. Most subcommands recurse by default, but some subcommands—usually those that have the potential to remove or undo your local modifications—do not. This command is deprecated. You should use **--depth** instead.

--notice-ancestry

Pay attention to ancestry when calculating differences.

--old arg

Use *arg* as the older target when producing a diff.

--parents

Create and add nonversioned parent directories to the working copy or the repository. Useful for creating directories on commit.

--password pass

Use *pass* as the password for authentication on the command line; otherwise, if it is needed, Subversion prompts you for it.

--quiet, -q

Print only essential information while performing an operation.

--record-only

Mark a revision as merged.

--reintegrate

Used with the **svn merge** subcommand, merges changes in a specified source URL into the working copy. This can be used to merge changes from a branch back into its original line.

--recursive, -R

Make a subcommand recurse into subdirectories. Most subcommands recurse by default. This option is deprecated. Use **--depth** instead.

--relocate *from to [path ...]*

Used with the **svn switch** subcommand to change the location of the repository that your working copy references. This is useful if the location of your repository changes and you have an existing working copy that you'd like to continue to use. See **svn switch** in “**svn Subcommands**” on page 766 for an example.

--revision rev, -r rev

Use *rev* as the revision (or range of revisions) for a particular operation. You can provide revision numbers, revision keywords, or dates (in curly braces) as arguments to the revision option. To provide a range of revisions, provide two revisions separated by a colon. For example:

```
$ svn log -r 1729
$ svn log -r 1729:HEAD
$ svn log -r 1729:1744
$ svn log -r {2001-12-04}:{2002-02-17}
$ svn log -r 1729:{2002-02-17}
```

The acceptable revision keywords for **--revision** are:

BASE

The original, unmodified version of the working copy. This keyword cannot refer to a URL.

COMMITTED

The last revision, before or at **BASE**, at which an item actually changed. This keyword cannot refer to a URL.

HEAD

The most recent revision in the repository.

PREV

The revision just before that at which an item changed. Equivalent to **COMMITTED - 1**. This keyword cannot refer to a URL.

Revision Date

A date specification enclosed in curly braces, { and }, such as {2002-02-17}, {15:30}, {"2002-02-17 15:30"}, {2002-02-17T15:30}, or {20020217T1530-0500}. See *Version Control with Subversion* for full details.

--revprop

Operate on a revision property instead of a Subversion property specific to a file or directory. This option requires that you also pass a revision with the **--revision** (-r) option.

--set-depth arg

Use *arg* as the new recursive depth for the target. This accepts the same values as the **--depth** command.

--show-revs arg

Used with **svn mergeinfo**, specifies which collection of merge information to display. *arg* may be either merged or eligible.

--show-updates, -u

Display information about which files in your working copy are out of date. This doesn't actually update any of your files; it just shows you which files will be updated if you run **svn update**.

--stop-on-copy

Cause a Subversion subcommand that is traversing the history of a versioned resource to stop harvesting that historical information when it encounters a copy—that is, a location in history where that resource was copied from another location in the repository.

--strict

Use strict semantics. See *Version Control with Subversion* for more information.

--summarize

Use with **diff** to get a summary of changes without a list of the changes themselves.

--targets filename

Retrieve the list of files to operate on from *filename* instead of listing all the files on the command line.

--username name

Use *name* as the username for authentication; otherwise, if it is needed, Subversion prompts you for it.

--verbose, -v

Print out as much information as possible while running any subcommand. This may result in Subversion printing out additional fields, detailed information about every file, or additional information regarding its actions.

--version

Print the client version info. This information not only includes the version number of the client, but also a listing of all repository access modules that the client can use to access a Subversion repository.

--with-revprop property

Set a revision property when writing to the repository (specify in NAME=VALUE format). When used with **svn log -xml**, display the value of the specified property name in the log output.

--xml

Prints output in XML format.

svn Subcommands

The **svn** command is the main user interface to Subversion. It works by accepting subcommands with arguments. Five of the previous options are global in version 1.5.

All subcommands will accept **--config-dir**, **--no-auth-cache**, **--non-interactive**, **--password**, and **--username**. Even commands for which these are meaningless will accept these options without fail. This is intended to make scripting easier. Because all subcommands accept these, we don't list them in options the below.

The general form of a subcommand is:

svn subcommand [options] arguments

add *svn add path ...*

Add files and directories to your working copy and schedule them for addition to the repository. They will be uploaded and added to the repository on your next commit. If you add something and change your mind before committing, you can unschedule the addition using **svn revert**.

Alternate names: none

Changes: working copy

Accesses repository: no

Options

- auto-props**
- depth arg**
- no-auto-props**
- no-ignore**
- no-parents**
- non-recursive (-N)**
- quiet (-q)**
- targets filename**

Examples

To add a file to your working copy:

```
$ svn add foo.c
A          foo.c
```

You can add a directory without adding its contents:

```
$ svn add --depth empty otherdir
A          otherdir
```

blame *svn blame target ...*

Show author and revision information inline for the specified files or URLs. Each line of text is annotated at the beginning with the author (username) and the revision number for the last change to that line.

Alternate names: **praise**, **annotate**, **ann**

Changes: nothing

Accesses repository: yes

Options

- extensions *args*, -x *args*
- force
- incremental
- revision *rev*, -r *rev*
- use-merge-history, -g
- verbose, -v
- xml

cat

`svn cat target ...`

Output the contents of the specified files or URLs. For listing the contents of directories, see `svn list`.

Alternate names: none

Changes: nothing

Accesses repository: yes

Options

- revision *rev*, -r *rev*

Examples

To view *readme.txt* in your repository without checking it out:

```
$ svn cat http://svn.red-bean.com/repos/test/readme.txt
This is a README file.
You should read this.
```



If your working copy is out of date (or if you have local modifications) and you want to see the **HEAD** revision of a file in your working copy, `svn cat` automatically fetches the **HEAD** revision when you give it a path.

```
$ cat foo.c
This file is in my local working copy
and has changes that I've made.
```

```
$ svn cat foo.c
Latest revision fresh from the repository!
```

changelist

`svn changelist name target...`

`svn changelist --remove name target...`

Group files for operations into named collections. This makes it easier to work on multiple groups of files.

Alternate names: cl

Changes: working copy

Accesses repository: no

Options

- changelist** *name*, **-cl** *name*
- depth** *arg*
- quiet (-q)**
- remove**
- targets** *filename*

Examples

Edit three files, add them to a changelist, then commit only files in that changelist.:

```
$ svn changelist issue1729 foo.c bar.c baz.c
Path "foo.c" is now a member of changelist 'issue1729'.
Path "bar.c" is now a member of changelist 'issue1729'.
Path "baz.c" is now a member of changelist 'issue1729'.

$ svn status
A someotherfile.c
A test/sometest.c

--Changelist 'issue1729':
A foo.c
A bar.c
A baz.c

$ svn commit --changelist issue1729 -m "Fixing Issue 1729"
Adding foo.c
Adding bar.c
Adding baz.c
Transmitting file data...
Committed revision 2

$ svn status
A someotherfile.c
A test/sometest.c
```

checkout

`svn checkout URL ... [path]`

Check out a working copy from a repository. If *path* is omitted, the basename of the URL is used as the destination. If multiple URLs are given, each one is checked out into a subdirectory of *path*, with the name of the subdirectory being the basename of the URL.

Alternate names: **co**

Changes: creates a working copy

Accesses repository: yes

Options

- depth** *arg*
- force**
- ignore-externals**
- quiet (-q)**
- revision** *rev*, **-r** *rev*

Examples

Check out a working copy into a directory called *mine*:

```
$ svn checkout file:///tmp/repos/test mine
A  mine/a
A  mine/b
Checked out revision 2.
$ ls
mine
```

If you interrupt a checkout (or something else interrupts your checkout, such as loss of connectivity, etc.), you can restart it either by issuing the identical checkout command again or by updating the incomplete working copy:

```
$ svn checkout file:///tmp/repos/test test
A  test/a
A  test/b
^C
svn: The operation was interrupted
svn: caught SIGINT

$ svn checkout file:///tmp/repos/test test
A  test/c
A  test/d
^C
svn: The operation was interrupted
svn: caught SIGINT

$ cd test
$ svn update
A  test/e
A  test/f
Updated to revision 3.
```

cleanup

`svn cleanup [path ...]`

Recursively clean up the working copy, removing locks and resuming unfinished operations. If you ever get a working-copy-locked error, run this command to remove stale locks and get your working copy into a usable state again.

If, for some reason, an `svn update` fails due to a problem running an external `diff` program (e.g., user input or network failure), pass the `--diff3-cmd` option to allow cleanup to complete any merging with your external `diff` program. You can also specify any configuration directory with the `--config-dir` option, but you should need these options extremely infrequently.

Alternate names: none

Changes: working copy

Accesses repository: no

Options:

`--diff3-cmd cmd`

commit`svn commit [path ...]`

Send changes from your working copy to the repository. If you do not supply a log message with your commit by using either the `--file` or `--message` option, `svn` starts your editor for you to compose a commit message.



If you begin a commit and Subversion starts your editor to compose the commit message, you can still abort without committing your changes. To cancel your commit, just quit your editor without saving your commit message and Subversion prompts you to abort the commit, continue with no message, or edit the message again.

Alternate names: `ci` (short for check in—not `co`, which is short for checkout)

Changes: working copy, repository

Accesses repository: yes

Options

- `--changelist name, -cl name`
- `--depth arg`
- `--editor-cmd cmd`
- `--encoding enc`
- `--file file, -F file`
- `--force-log`
- `--keep-changelists`
- `--message text, -m text`
- `--no-unlock`
- `--quiet (-q)`
- `--targets filename`
- `--with-revprop property`

Examples

Commit a simple modification to a file with the commit message on the command line and an implicit target of your current directory (.):

```
$ svn commit -m "added howto section."
Sending      a
Transmitting file data.
Committed revision 3.
```

To commit a file scheduled for deletion:

```
$ svn commit -m "removed file 'c'." 
Deleting      c
Committed revision 7.
```

copy	<code>svn copy src dst</code>
	Copy a file in a working copy or in the repository. <i>src</i> and <i>dst</i> can each be either a working-copy (WC) path or a URL:
WC→WC	Copy and schedule an item for addition (with history).
WC→URL	Immediately commit a copy of WC to URL.
URL→WC	Check out URL into WC, and schedule it for addition.
URL→URL	Complete server-side copy. This is usually used to branch and tag.



You can only copy files within a single repository. Subversion does not support cross-repository copying.

Alternate names: **cp**

Changes: repository if destination is a URL; working copy if destination is a WC path

Accesses repository: if source or destination is in the repository, or if needed to look **up** the source revision number

Options

- editor-cmd** *editor*
- encoding** *enc*
- file** *file*, **-F** *file*
- force-log**
- message** *text*, **-m** *text*
- parents**
- quiet** (**-q**)
- revision** *rev*, **-r** *rev*
- with-revprop** *property*

Examples

Copy an item within your working copy (just schedules the copy; nothing goes into the repository until you commit):

```
$ svn copy foo.txt bar.txt
A          bar.txt
$ svn status
A +    bar.txt
```

Copy an item from the repository to your working copy (just schedules the copy; nothing goes into the repository until you commit):

```
$ svn copy file:///tmp/repos/test/far-away near-here  
A           near-here
```



This is the recommended way to resurrect a dead file in your repository!

And finally, copying between two URLs:

```
$ svn copy file:///tmp/repos/test/far-away \  
> file:///tmp/repos/test/over-there -m "remote copy."  
Committed revision 9.
```



This is the easiest way to tag a revision in your repository; just `svn copy` that revision (usually `HEAD`) into your tags directory.

```
$ svn copy file:///tmp/repos/test/trunk \  
>           file:///tmp/repos/test/tags/0.6.32-prerelease \  
>           -m "tag tree"  
Committed revision 12.
```

delete

`svn delete path ...`
`svn delete URL ...`

Items specified by *path* are scheduled for deletion upon the next commit. Files (and directories that have not been committed) are *immediately* removed from the working copy. The command will not remove any unversioned or modified items; use the `--force` option to override this behavior.

Items specified by *URL* are deleted from the repository via an immediate commit. Multiple URLs are committed atomically.

Alternate names: `del`, `remove`, `rm`

Changes: working copy if operating on files; repository if operating on URLs

Accesses repository: only if operating on URLs

Options

- `--editor-cmd editor`
- `--encoding enc`
- `--file file, -F file`
- `--force`
- `--force-log`
- `--keep-local`
- `--message text, -m text`
- `--quiet (-q)`
- `--targets filename`
- `--with-revprop property`

diff	<code>svn diff [-r N[:M]] [--old old-tgt] [--new new-tgt] [path ...]</code> <code>svn diff -r N:M URL</code> <code>svn diff [-r N[:M]] URL1[@N] URL2[@M]</code>
	Display the differences between two paths. The three different ways you can use svn diff are:
	svn diff [-r N[:M]] [--old old-tgt] [--new new-tgt] [path ...] Display the differences between <i>old-tgt</i> and <i>new-tgt</i> . If <i>paths</i> are given, they are treated as relative to <i>old-tgt</i> and <i>new-tgt</i> , and the output is restricted to differences in only those paths. <i>old-tgt</i> and <i>new-tgt</i> may be working copy paths or <i>URL[@rev]</i> . <i>old-tgt</i> defaults to the current working directory, and <i>new-tgt</i> defaults to <i>old-tgt</i> . <i>N</i> defaults to BASE or, if <i>old-tgt</i> is a URL, to HEAD . <i>M</i> defaults to the current working version or, if <i>new-tgt</i> is a URL, to HEAD . svn diff -r N sets the revision of <i>old-tgt</i> to <i>N</i> , whereas svn diff -r N:M also sets the revision of <i>new-tgt</i> to <i>M</i> .
	svn diff -r N:M URL A shorthand for svn diff -r N:M --old=URL --new=URL .
	svn diff [-r N[:M]] URL1[@N] URL2[@M] A shorthand for svn diff [-r N[:M]] --old=URL1 --new=URL2 .
	If <i>target</i> is a URL, then revisions <i>N</i> and <i>M</i> can be given either via the --revision option or by using @ notation as described earlier.
	If <i>target</i> is a working copy path, then the --revision option means: --revision N:M The server compares <i>target@N</i> and <i>target@M</i> . --revision N The client compares <i>target@N</i> against the working copy. No --revision option The client compares the base and working copies of <i>target</i> . If the alternate syntax is used, the server compares <i>URL1</i> and <i>URL2</i> at revisions <i>N</i> and <i>M</i> respectively. If either <i>N</i> or <i>M</i> are omitted, a value of HEAD is assumed. By default, svn diff ignores the ancestry of files and merely compares the contents of the two files being compared. If you use --notice-ancestry , the ancestry of the paths in question is taken into consideration when comparing revisions. (That is, if you run svn diff on two files with identical contents but different ancestry you will see the entire contents of the file as having been removed and added again.) Alternate names: di Changes: nothing Accesses repository: for obtaining differences against anything but the BASE revision in your working copy.

Options

--change *args*, -c *args*
 --changelist *name*, -cl *name*
 --depth *arg*
 --diff-cmd *cmd*
 --extensions *args*, -x *args*
 --force
 --new *new-target*
 --no-diff-deleted
 --notice-ancestry
 --old *old-target*
 --revision *rev*, -r *rev*
 --summarize
 --xml

Examples

Compare BASE and your working copy:

```
$ svn diff COMMITTERS
Index: COMMITTERS
=====
=====
--- COMMITTERS (revision 4404)
+++ COMMITTERS (working copy)
...
```

See how your working copy's modifications compare against an older revision:

```
$ svn diff -r 3900 COMMITTERS
Index: COMMITTERS
=====
=====
--- COMMITTERS (revision 3900)
+++ COMMITTERS (working copy)
...
```

Use **--diff-cmd** *cmd* and **-x** to pass arguments directly to the external **diff** program:

```
$ svn diff --diff-cmd /usr/bin/diff -x "-i -b" COMMITTERS
Index: COMMITTERS
=====
=====
0a1,2
> This is a test
>
```

export

`svn export [-r rev] URL [path]
svn export path1 path2`

The first form exports a clean directory tree into *path* from the repository specified by *URL*, at revision *rev* if it is given—otherwise at **HEAD**. If *path* is omitted, the last component of the *URL* is used for the local directory name.

The second form exports a clean directory tree from the working copy specified by *path1* into *path2*. All local changes are preserved, but files not under version control are not copied.

This command will also take the unique **--native-eol** option.

Alternate names: none

Changes: local disk

Accesses repository: only if exporting from a URL

Options

- depth arg**
- force**
- ignore-externals**
- native-eol format**
- quiet (-q)**
- revision rev, -r rev**

help

`svn help [subcommand ...]`

Provide a quick usage summary. With *subcommand*, provide information about the given subcommand.

Alternate names: ?, h

Changes: nothing

Accesses repository: no

import

`svn import [path] URL`

Recursively commit a copy of *path* to *URL*. If *path* is omitted, . is assumed. Parent directories are created in the repository as necessary.

Alternate names: none

Changes: repository

Accesses repository: yes

Options

- auto-props**
- depth arg**
- editor-cmd editor**
- encoding enc**
- file file, -F file**
- force**
- force-log**
- message text, -m text**
- no-auto-props**
- no-ignore**

Examples

Import the local directory *myproj* into the root of your repository:

```
$ svn import -m "New import" myproj \
> http://svn.red-bean.com/repos/test
```

```
Adding myproj/sample.txt
...
Transmitting file data .....
Committed revision 16.
```

Import the local directory *myproj* into *trunk/vendors* in your repository. The directory *trunk/vendors* need not exist before you import into it; **svn import** will recursively create directories for you:

```
$ svn import -m "New import" myproj \
> http://svn.red-bean.com/repos/test/trunk/vendors/myproj
Adding myproj/sample.txt
...
Transmitting file data .....
Committed revision 16.
```

After importing data, note that the original tree is *not* under version control. To start working, you still need to **svn checkout** a fresh working copy of the tree.

info

`svn info [path ...]`
`svn info URL`

Print information about paths in your working copy or specified URLs, including:

- Path
- Name
- URL
- Repository root
- Repository UUID
- Revision
- Node kind
- Last changed author
- Last changed revision
- Last changed date
- Last token
- Lock owner
- Lock created (date)
- Lock expires (date)
- Schedule
- Copied from URL
- Copied from rev
- Text last updated
- Properties last updated
- Checksum

Alternate names: none

Changes: nothing

Accesses repository: no

Options

```
--changelist name, -cl name
--depth arg
--incremental
--revision rev, -r rev
--targets filename
--xml
```

list

```
svn list [target ...]
```

List each *target* file and the contents of each *target* directory as they exist in the repository. If *target* is a working copy path, the corresponding repository URL is used. The default *target* is ., meaning the repository URL of the current working-copy directory.

With **--verbose**, the following fields show the status of the item:

- Revision number of the last commit
- Author of the last commit
- Size (in bytes)
- Date and time of the last commit

Alternate names: ls

Changes: nothing

Accesses repository: yes

Options

```
--depth arg
--incremental
--revision rev, -r rev
--verbose (-v)
--xml
```

Examples

To see what files a repository has without downloading a working copy:

```
$ svn list http://svn.red-bean.com/repos/test/support
README.txt
INSTALL
examples/
...
```

Pass the **--verbose** option for additional information:

```
$ svn list --verbose file:///tmp/repos
16 sally          28361 Jan 16 23:18 README.txt
27 sally          0 Jan 18 15:27 INSTALL
24 harry          Jan 18 11:27 examples/
```

lock

`svn lock path ...`
`svn lock URL`

Set a lock token on a specified file to prevent other users or even the same user on another system from updating the file. Only the system with the lock token may commit changes. Locks are useful when working on binary files that cannot be merged. By default, Subversion's locks are not strict, however. Locks can be broken or taken over by other users by using the `--force` option.

Alternate names: none

Changes: working copy; repository

Accesses repository: yes

Options

- `--encoding enc`
- `--file file, -F file`
- `--force`
- `--force-log`
- `--message text, -m text`
- `--targets filename`

log

`svn log [path]`
`svn log URL [path ...]`

The default target is the path of your current directory. If no arguments are supplied, `svn log` shows the log messages for all files and directories inside of (and including) the current working directory of your working copy. You can refine the results by specifying a path, one or more revisions, or any combination of the two. The default revision range for a local path is **BASE:1**.

If you specify a URL alone, the command prints log messages for everything that the URL contains. If you add paths past the URL, only messages for those paths under that URL are printed. The default revision range for a URL is **HEAD:1**.

With `--verbose`, `svn log` also prints all affected paths with each log message. With `--quiet`, `svn log` does not print the log message body itself (this is compatible with `--verbose`).

Each log message is printed just once, even if more than one of the affected paths for that revision were explicitly requested. Logs follow copy history by default. Use `--stop-on-copy` to disable this behavior, which can be useful for determining branch points.

Alternate names: none

Changes: nothing

Accesses repository: yes

Options

```
--change arg, -c arg
--incremental
--limit num, -l num
--quiet (-q)
--revision rev, -r rev
--stop-on-copy
--targets filename
--use-merge-history, -g
--verbose (-v)
--with-all-revprops
--with-revprop property
--xml
```

Examples

To see the log messages for all the paths that changed in your working copy, run `svn log` from the top:

```
$ svn log
-----
-----
r20 | harry | 2003-01-17 22:56:19 -0600 (Fri, 17 Jan 2003)
| 1 line

Tweak.

-----
-----
r17 | sally | 2003-01-16 23:21:19 -0600 (Thu, 16 Jan 2003)
| 2 lines
...
...
```

If you don't have a working copy handy, you can log a URL:

```
$ svn log http://svn.red-bean.com/repos/test/foo.c
-----
-----
r32 | sally | 2003-01-13 00:43:13 -0600 (Mon, 13 Jan 2003)
| 1 line

Added defines.

-----
-----
r28 | sally | 2003-01-07 21:48:33 -0600 (Tue, 07 Jan 2003)
| 3 lines
...
...
```

If you run `svn log` on a specific path and provide a specific revision and get no output at all:

```
$ svn log -r 20 http://svn.red-bean.com/untouched.txt
-----
-----
```

That just means that the path was not modified in that revision. If you log from the top of the repository, or know the file that changed in that revision, you can specify it explicitly:

```
$ svn log -r 20 touched.txt
-----
-----r20 | sally | 2003-01-17 22:56:19 -0600 (Fri, 17 Jan 2003)
| 1 line
Made a change.
-----
```

merge

`svn merge sourceURL1[@N] sourceURL2[@M] [wcpath]`
`svn merge sourceWCPATH1@N sourceWCPATH2@M [wcpath]`
`svn merge -r N:M source [path]`

In the first form, the source URLs are specified at revisions *N* and *M*. These are the two sources to be compared. The revisions default to **HEAD** if omitted.

In the second form, the URLs corresponding to the source working copy paths define the sources to be compared. The revisions must be specified.

In the third form, *source* can be a URL or working-copy item, in which case the corresponding URL is used. This URL, at revisions *N* and *M*, defines the two sources to be compared.

wcpath is the working-copy path that will receive the changes. If *wcpath* is omitted, a default value of `".` is assumed, unless the sources have identical basenames that match a file within `".`, in which case, the differences are applied to that file.

Unlike **svn diff**, this command takes the ancestry of a file into consideration when performing a merge operation. This is very important when you're merging changes from one branch into another and you've renamed a file on one branch but not the other.

Alternate names: none

Changes: working copy

Accesses repository: only if working with URLs

Options

- `--accept arg`
- `--change arg, -c arg`
- `--depth arg`
- `--diff3-cmd cmd`
- `--dry-run`
- `--extensions args, -x args`
- `--force`
- `--ignore-ancestry`
- `--quiet (-q)`
- `--record-only`
- `--reintegrate`
- `--revision rev, -r rev`

Examples

Merge a branch back into the trunk (assuming that you have a working copy of the trunk and that the branch was created in revision 250):

```
$ svn merge -r 250:HEAD \
> http://svn.red-bean.com/repos/branches/my-branch
U  myproj/tiny.txt
U  myproj/thhgttg.txt
U  myproj/win.txt
U  myproj/flo.txt
```

If you branched at revision 23, and you want to merge changes from the trunk into your branch, you could do this from inside the working copy of your branch:

```
$ svn merge -r 23:30 file:///tmp/repos/trunk/vendors
U  myproj/thhgttg.txt
...
...
```

To merge changes to a single file:

```
$ cd myproj
$ svn merge -r 30:31 thhgttg.txt
U  thhgttg.txt
```

mergeinfo

`svn mergeinfo sourceURL[@rev] [target ...]`

Query information about merges or potential merges between *sourceURL* and *target*. By default it shows merged information. The option `--show-revs` can be used to get information about eligible merges.

Alternate names: none

Changes: nothing

Accesses repository: yes

Options

```
--revision rev, -r rev
--show-revs arg
```

Examples

Find out which changesets your trunk directory has already received as well as what changesets it's still eligible to receive:

```
$ svn mergeinfo branches/test
Path: branches/test
Source path: /trunk
Merged ranges: r2:13
Eligible ranges: r13:15
```

mkdir

`svn mkdir path ...`
`svn mkdir URL ...`

Create a directory with a name given by the final component of the *path* or URL. A directory specified by a working copy *path* is scheduled for addition in the working copy. A directory specified by a

URL is created in the repository via an immediate commit. Multiple directory URLs are committed atomically. In both cases, all the intermediate directories must already exist.

Alternate names: none

Changes: working copy; repository if operating on a URL

Accesses repository: only if operating on a URL

Options

- `--editor-cmd editor`
- `--encoding enc`
- `--file file, -F file`
- `--force-log`
- `--message text, -m text`
- `--parents`
- `--quiet (-q)`
- `--with-revprop property`

move

`svn move src dst`

This command moves (renames) a file or directory in your working copy or in the repository.



This command is equivalent to an `svn copy` followed by `svn delete`.

`WC→WC`

Move and schedule a file or directory for addition (with history).

`URL→URL`

Complete server-side rename.



Subversion does not support moving between working copies and URLs. In addition, you can move files only within a single repository; Subversion does not support cross-repository moving.

Alternate names: `mv`, `rename`, `ren`

Changes: working copy; repository if operating on a URL

Accesses repository: only if operating on a URL

Options

- `--editor-cmd editor`
- `--encoding enc`
- `--file file, -F file`
- `--force`
- `--force-log`
- `--message text, -m text`
- `--revision rev, -r rev`
- `--revprop`
- `--with-revprop property`

propdel	<pre>svn propdel <i>propname</i> [<i>path</i> ...] svn propdel <i>propname</i> --revprop -r <i>rev</i> [<i>URL</i>]</pre> <p>This removes properties from files, directories, or revisions. The first form removes versioned properties in your working copy, while the second removes unversioned remote properties on a repository revision.</p> <p>Alternate names: pdel, pd</p> <p>Changes: working copy; repository only if operating on a URL</p> <p>Accesses repository: only if operating on a URL</p>
	<p>Options</p> <ul style="list-style-type: none">--changelist <i>name</i>, -cl <i>name</i>--depth <i>arg</i>--quiet (-q)--revision <i>rev</i>, -r <i>rev</i>--revprop

propedit	<p>Delete a property from a file in your working copy:</p> <pre>\$ svn propdel svn:mime-type some-script property 'svn:mime-type' deleted from 'some-script'.</pre> <p>Delete a revision property:</p> <pre>\$ svn propdel --revprop -r 26 release-date property 'release-date' deleted from repository revision '26'</pre>
-----------------	---

	<p>svn propedit <i>propname</i> <i>path</i> ... svn propedit <i>propname</i> --revprop -r <i>rev</i> [<i>URL</i>]</p> <p>Edit one or more properties using your favorite editor. The first form edits versioned properties in your working copy, while the second edits unversioned remote properties on a repository revision.</p> <p>Alternate names: pedit, pe</p> <p>Changes: working copy; repository only if operating on a URL</p> <p>Accesses repository: only if operating on a URL</p>
	<p>Options</p> <ul style="list-style-type: none">--editor-cmd <i>editor</i>--encoding <i>enc</i>--force--force-log--password <i>pass</i>--revision <i>rev</i>, -r <i>rev</i>--revprop--with-revprop <i>property</i>

propget	<code>svn propget <i>propname</i> [<i>path</i> ...]</code> <code>svn propget <i>propname</i> --revprop -r <i>rev</i> [<i>URL</i>]</code>
----------------	---

Print the value of a property on files, directories, or revisions. The first form prints the versioned property of an item or items in your working copy, while the second prints the unversioned remote property on a repository revision.

Alternate names: **pget**, **pg**

Changes: working copy; repository only if operating on a URL

Accesses repository: only if operating on a URL

Options

- `--changelist name, -cl name`
- `--depth arg`
- `--revision rev, -r rev`
- `--revprop`
- `--strict`
- `--xml`

proplist	<code>svn proplist [<i>path</i> ...]</code> <code>svn proplist --revprop -r <i>rev</i> [<i>URL</i>]</code>
-----------------	---

List all properties on files, directories, or revisions. The first form lists versioned properties in your working copy, while the second lists unversioned remote properties on a repository revision.

Alternate names: **plist**, **pl**

Changes: working copy; repository only if operating on a URL

Accesses repository: only if operating on a URL

Options

- `--changelist name, -cl name`
- `--depth arg`
- `--quiet (-q)`
- `--revision rev, -r rev`
- `--revprop`
- `--verbose (-v)`
- `--xml`

Examples

You can use **svn proplist** to see the properties on an item in your working copy:

```
$ svn proplist foo.c
Properties on 'foo.c':
  svn:mime-type
  svn:keywords
  owner
```

But with the `--verbose` flag, `svn proplist` is extremely handy, as it also shows you the values for the properties:

```
$ svn proplist --verbose foo.c
Properties on 'foo.c':
  svn:mime-type : text/plain
  svn:keywords : Author Date Rev
  owner : sally
```

propset	<code>svn propset <i>propname</i> [<i>propval</i>] <i>path</i> ...</code> <code>svn propset <i>propname</i> --revprop -r <i>rev</i> [<i>propval</i>] [<i>URL</i>]</code> Set <i>propname</i> to <i>propval</i> on files, directories, or revisions. The first example creates a versioned, local property change in the working copy, and the second creates an unversioned, remote property change on a repository revision. The new property value, <i>propval</i> , may be provided literally, or using the <code>-F <i>valfile</i></code> option. Alternate names: pset, ps Changes: working copy; repository only if operating on a URL Accesses repository: only if operating on a URL
Options	
<code>--changelist <i>name</i>, -cl <i>name</i></code> <code>--depth <i>arg</i></code> <code>--encoding <i>enc</i></code> <code>--file <i>file</i>, -F <i>file</i></code> <code>--force</code> <code>--quiet (-q)</code> <code>--revision <i>rev</i>, -r <i>rev</i></code> <code>--revprop</code> <code>--targets <i>filename</i></code>	

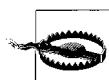
Examples

Set the mimetype on a file:

```
$ svn propset svn:mime-type image/jpeg foo.jpg
property 'svn:mime-type' set on 'foo.jpg'
```

On a Unix system, if you want a file to have the executable permission set:

```
$ svn propset svn:executable ON somescript
property 'svn:executable' set on 'somescript'
```



By default, you cannot modify revision properties in a Subversion repository. Your repository administrator must explicitly enable revision property modifications by creating a hook named `pre-revprop-change`.

resolve**svn resolve** *path* ...

Remove the conflicted state on working-copy files or directories. This command does not semantically resolve conflict markers; instead it replaces *path* and then removes conflict-related artifact files. Use the **--accept** argument to specify what version to use when replacing *path*. This command allows *path* to be committed again by telling Subversion that the conflicts have been resolved. Use it after you have resolved the conflict in the file. You can pass the following arguments to the **--accept** option:

Alternate names: none

Changes: working copy

Accesses repository: no

Options

- accept** *arg*
- depth** *arg*
- quiet** (-q)
- recursive**, -R
- targets** *filename*

Example

If you get a conflict on an update, your working copy will contain three additional files:

```
$ svn update
Conflict discovered in 'foo.c'.
Select: (p) postpone, (df) diff-full, (e) edit,
(h) help for more options: p
C   foo.c
Updated to revision 31
$ svn resolve --accept mine-full foo.c
Resolved conflicted state of 'foo.c'
```



You *can* just remove the conflict files and commit, but **svn resolve** fixes up some bookkeeping data in the working-copy administrative area in addition to removing the conflict files, so you should use this command.

resolved**svn resolved** *path* ...

Deprecated. Remove the conflicted state on working-copy files or directories. Syntactically it is the same as '**svn resolve --accept working path**', which you should now use instead.

Alternate names: none

Changes: working copy

Accesses repository: no

Options

- depth *arg*
- quiet (-q)
- recursive (-R)
- targets *filename*

revert

`svn revert path ...`

Revert any local changes to a file or directory, and resolve any conflicted states. **svn revert** reverts not only the contents of an item in your working copy, but also any property changes. Finally, you can use it to undo any scheduling operations that you may have done (e.g., files scheduled for addition or deletion can be unscheduled).

Alternate names: none

Changes: working copy

Accesses repository: no

Options

- changelist *name*, -cl *name*
- depth *arg*
- quiet (-q)
- recursive (-R)
- targets *filename*

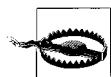
Examples

Discard changes to a file:

```
$ svn revert foo.c  
Reverted foo.c
```

If you want to revert a whole directory of files, use the --depth=infinity flag:

```
$ svn revert --depth=infinity .  
Reverted newdir/afile  
Reverted foo.c  
Reverted bar.txt
```



If you provide no targets to **svn revert**, it does nothing; to protect you from accidentally losing changes in your working copy, **svn revert** requires you to provide at least one target.

status

`svn status [path ...]`

Print the status of working-copy files and directories. With no arguments, it prints only locally modified items (no repository access). With **--show-updates**, add working revision and server out-of-date information. With **--verbose**, print full revision information on every item.

The first five columns in the output are each one character wide, and each column gives you information about different aspects of each working-copy item.

The first column indicates that an item was added, deleted, or otherwise changed:

space

No modifications.

- A Item is scheduled for addition.
- D Item is scheduled for deletion.
- M Item has been modified.
- R Item has been replaced in your working copy. This means the file was scheduled for deletion, and then a new file with the same name was scheduled for addition in its place.
- C The contents of the item conflicts with updates received from the repository.
- X Item is related to an externals definition.
- I Item is being ignored (e.g., with the **svn:ignore** property).
- ? Item is not under version control.
- ! Item is missing (e.g., you moved or deleted it without using **svn**). This also indicates that a directory is incomplete (a checkout or update was interrupted).
- ~ Item is versioned as a directory but has been replaced by a file, or vice versa.

The second column tells the status of a file's or directory's properties:

space

No modifications.

- M Properties for this item have been modified.
- C Properties for this item are in conflict with property updates received from the repository.

The third column is populated only if the working copy directory is locked:

space

Item is not locked.

- L Item is locked.

The fourth column is populated only if the item is scheduled for addition-with-history:

space

No history scheduled with commit.

- + History scheduled with commit.

The fifth column is populated only if the item is switched relative to its parent:

space

Item is a child of its parent directory.

- S Item is switched.

The sixth column is populated with lock information:

space

When **--show-updates** is used, the file is not locked. Otherwise, it merely means that the file is not locked in this working copy.

- K** File is locked in this working copy.
- O** File is locked by another user or in another working copy. This will only appear when **--show-updates** is used.
- T** File is locked in this working copy, but the lock has been stolen and is invalid. The file is locked in the repository. This will only appear when **--show-updates** is used.
- B** File is locked in this working copy, but the lock has been broken and is invalid. The file is no longer locked. This will only appear when **--show-updates** is used.

If you pass the **--show-updates** option, then out-of-date information appears in the seventh column:

space

The item in your working copy is up to date.

- * A newer revision of the item exists on the server.

The remaining fields are variable width and delimited by spaces. The working revision is the next field if the **--show-updates** or **--verbose** options are passed.

If the **--verbose** option is passed, the last committed revision and last committed author are displayed next.

The working-copy path is always the final field, so it can include spaces.

Alternate names: **stat**, **st**

Changes: nothing

Accesses repository: only if using **--show-updates**

Options

- changelist** *name*, **-cl** *name*
- depth** *arg*
- ignore-externals**
- incremental**
- no-ignore**
- non-recursive (-N)**
- quiet (-q)**
- show-updates (-u)**
- no-ignore**
- xml**

Examples

To find out what changes you have made to your working copy:

```
$ svn status wc
M      wc/bar.c
A +    wc/qax.c
```

To find out what files in your working copy are out of date, pass the **--show-updates** option (this does *not* make any changes to your working copy). Here you can see that *wc/foo.c* has changed in the repository since we last updated our working copy:

```
$ svn status --show-updates wc
M          965    wc/bar.c
*          965    wc/foo.c
A +        965    wc/qax.c
Status against revision: 981
```



--show-updates places an asterisk *only* next to items that are out of date (that is, items that will be updated from the repository if you run **svn update**). **--show-updates** does *not* cause the status listing to reflect the repository's version of the item.

And finally, the most information you can get out of the status subcommand:

```
$ svn status --show-updates --verbose wc
M          965    938 sally      wc/bar.c
*          965    922 harry     wc/foo.c
A +        965    687 harry     wc/qax.c
                           965    687 harry     wc/zig.c
Head revision: 981
```

switch

`svn switch URL [path]`

This subcommand updates your working copy to mirror a new URL—usually a URL that shares a common ancestor with your working copy, although not necessarily. This is the Subversion way to move a working copy to a new branch.

As with most subcommands, you can limit the scope of the switch operation to a particular tree depth using the **--depth** option. Alternatively, you can use the **--set-depth** option to set a new “sticky” working copy depth on the switch target. Currently, the depth of a working copy directory can only be increased (telescoped more deeply); you cannot make a directory more shallow.

Alternate names: **sw**

Changes: working copy

Accesses repository: yes

Options

- accept arg**
- depth arg**
- diff3-cmd cmd**
- force**
- ignore-externals**
- quiet (-q)**
- relocate**
- revision rev, -r rev**
- set-depth arg**

Examples

If you're currently inside the directory `vendors`, which was branched to `vendors-with-fix`, and you'd like to switch your working copy to that branch:

```
$ svn switch http://svn.red-bean.com/repos/branches/ \
> vendors-with-fix .
U  myproj/foo.txt
U  myproj/bar.txt
U  myproj/baz.c
U  myproj/qux.c
Updated to revision 31.
```

And to switch back, just provide the URL to the location in the repository from which you originally checked out your working copy:

```
$ svn switch http://svn.red-bean.com/repos/trunk/vendors .
U  myproj/foo.txt
U  myproj/bar.txt
U  myproj/baz.c
U  myproj/qux.c
Updated to revision 31.
```



You can just switch part of your working copy to a branch if you don't want to switch your entire working copy.

Sometimes an administrator might change the “base location” of your repository; in other words, the contents of the repository don't change, but the main URL used to reach the root of the repository does. For example, the hostname may change, or the URL schema, or perhaps just the path that leads to the repository. Rather than checking out a new working copy, you can have the `svn switch` command “rewrite” the beginnings of all the URLs in your working copy. Use the `--relocate` command to do the substitution. No file contents are changed, nor is the repository contacted. It's similar to running a `sed` script over your working copy `.svn/` directories, which runs `s/OldRoot/NewRoot/`:

```
$ cd /tmp
$ svn checkout file:///tmp/repos test
A  test/a
A  test/b
...
$ mv repos newlocation
$ cd test/
$ svn update
svn: Unable to open an ra_local session to URL
svn: Unable to open repository 'file:///tmp/repos'
```

```
$ svn switch --relocate file:///tmp/repos file:///tmp/
newlocation .
$ svn update
At revision 3.
```

unlock

`svn unlock path ...`
`svn unlock URL`

Remove the lock token from the specified files. This command will print a warning if the target is locked by another user or no lock token exists, but will continue to unlock files it can unlock. Use the **--force** option to break a lock belonging to another user or working copy.

Alternate names: none

Changes: working copy; repository

Accesses repository: yes

Options

- force**
- targets filename**

update

`svn update [PATH ...]`

svn update brings changes from the repository into your working copy. If no revision is given, it brings your working copy up to date with the **HEAD** revision. Otherwise, it synchronizes the working copy to the revision given by the **--revision** option.

For each updated item Subversion prints a line starting with a specific character reporting the action taken. These characters have the following meaning:

- A Added
- B Broken lock (third column only)
- D Deleted
- U Updated
- C Conflicted
- G Merged
- E Existed

A character in the first column signifies an update to the actual file, while updates to the file's properties are shown in the second column. Lock information is printed in the third column.



If you want to examine an older revision of a single file, you may want to use **svn cat**.

Alternate names: **up**

Changes: working copy

Accesses repository: yes

Options

```
--accept arg  
--changelist name, -cl name  
--depth arg  
--diff3-cmd cmd  
--editor-cmd editor  
--force  
--ignore-externals  
--quiet (-q)  
--revision rev, -r rev  
--set-depth arg
```

Repository Administration: svnadmin

svnadmin is the administrative tool for monitoring and repairing your Subversion repository.

Common svnadmin Options

--bdb-log-keep

(Berkeley DB specific) Disable automatic log removal of database logfiles.

--bdb-txn-nosync

(Berkeley DB specific) Disable use of `fsync()` when committing database transactions.

--bypass-hooks

Bypass the repository hook system.

--clean-logs

Remove unused Berkeley DB logs.

--force-uuid

By default, when loading data into a repository that already contains revisions, **svnadmin** ignores the UUID from the dump stream. This option causes the repository's UUID to be set to the UUID from the stream.

--ignore-uuid

By default, when loading an empty repository, **svnadmin** uses the UUID from the dump stream. This option causes that UUID to be ignored.

--incremental

Dump a revision only as a diff against the previous revision, instead of the usual full text.

--parent-dir dir

When loading a dumpfile, root paths at *dir* instead of */*.

--quiet

Do not show normal progress; show only errors.

--revision rev, -r rev

Specify a particular revision to operate on.

Common svnadmin Subcommands

The `svnadmin` command creates and administers the repository. As such, it always operates on local paths, not on URLs.

create `svnadmin create repos_path`

Create a new, empty repository at the path provided. If the provided directory does not exist, it is created for you.

Options

- `--bdb-log-keep`
- `--bdb-txn-nosync`

Example

Creating a new repository is just this easy:

```
$ svnadmin create /usr/local/svn/repos
```

deltify `svnadmin deltify [-rlower[:upper]]repos_path`

`svnadmin deltify` only exists in 1.0.x due to historical reasons. This command is deprecated and no longer needed.

It dates from a time when Subversion offered administrators greater control over compression strategies in the repository. This turned out to be a lot of complexity for *very* little gain, and the feature was deprecated.

Options

- `--quiet`
- `--revision rev, -r rev`

dump `svnadmin dump repos_path [-r lower[:upper]] [--incremental]`

Dump the contents of filesystem to standard output in a dumpfile portable format, sending feedback to standard error. Dump revisions *lower* rev through *upper* rev. If no revisions are given, dump all revision trees. If only *lower* is given, dump that one revision tree.

Options

- `--incremental`
- `--quiet`
- `--revision rev, -r rev`

Examples

Dump your whole repository:

```
$ svnadmin dump /usr/local/svn/repos
SVN-fs-dump-format-version: 1
Revision-number: 0
* Dumped revision 0.
Prop-content-length: 56
Content-length: 56
...
```

Incrementally dump a single transaction from your repository:

```
$ svnadmin dump /usr/local/svn/repos -r 21 --incremental
* Dumped revision 21.
SVN-fs-dump-format-version: 1
Revision-number: 21
Prop-content-length: 101
Content-length: 101
...
```

help

`svnadmin help [subcommand ...]`

Provide a quick usage summary. With *subcommand*, provide information about the given subcommand.

Alternate names: ?, h

hotcopy

`svnadmin hotcopy old_repos_path new_repos_path`

This subcommand makes a full hot backup of your repository, including all hooks, configuration files, and, of course, database files. If you pass the `--clean-logs` option, `svnadmin` performs a hotcopy of your repository, and then removes unused Berkeley DB logs from the original repository. You can run this command at any time and make a safe copy of the repository, regardless of whether other processes are using the repository.

Option

`--clean-logs`

list-dblogs

`svnadmin list-dblogs repos_path`

List Berkeley DB logfiles. Berkeley DB creates logs of all changes to the repository, which allow it to recover in the face of catastrophe. Unless you enable `DB_LOGS_AUTOREMOVE`, the logfiles accumulate, although most are no longer used and can be deleted to reclaim disk space.

list-unused-dblogs

`svnadmin list-unused-dblogs repos_path`

List unused Berkeley DB logfiles (see `svnlook list-dblogs`).

Example

Remove all unused logfiles from a repository:

```
$ svnadmin list-unused-dblogs /path/to/repos | xargs rm
## disk space reclaimed!
```

load

`svnadmin load repos_path`

Read a dumpfile-formatted stream from standard input, committing new revisions into the repository's filesystem. Send progress feedback to standard output.

Options

- force-uuid
- ignore-uuid
- parent-dir
- quiet (-q)

Examples

This shows the beginning of loading a repository from a backup file (made, of course, with `svn dump`):

```
$ svnadmin load /usr/local/svn/restored < repos-backup
<<< Started new txn, based on original revision 1
    * adding path : test ... done.
    * adding path : test/a ... done.
...
...
```

Or, to load into a subdirectory:

```
$ svnadmin load --parent-dir new/subdir/for/project \
>   /usr/local/svn/restored < repos-backup
<<< Started new txn, based on original revision 1
    * adding path : test ... done.
    * adding path : test/a ... done.
...
...
```

lslocks

`svnadmin lslocks repos_path [path]`

Print descriptions of all locks in repository `repos_path` underneath `path`. If `path` isn't given it defaults to the root directory of the repository.

lstxns

`svnadmin lstxns repos_path`

Print the names of all uncommitted transactions.

recover

`svnadmin recover repos_path`

Run this command if you get an error indicating that your repository needs to be recovered. This command requires a database lock. Normally failure to obtain a lock will cause an error. Use the `--wait` option to cause the command to wait indefinitely for a database lock.

Options

- wait

rmlocks

`svnadmin rmlocks repos_path locked_path...`

Unconditionally remove locks from each locked path.

rmtxns

`svnadmin rmtxns repos_path txn_name ...`

Delete outstanding transactions from a repository.

Options

--quiet (-q)

Examples

Remove all uncommitted transactions from your repository, using **svn lstxns** to provide the list of transactions to remove:

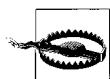
```
$ svnadmin rmtxns /usr/local/svn/repos/ \
>`svnadmin lstxns /usr/local/svn/repos/`
```

setlog

svnadmin setlog repos_path -r revision file

Set the log message on revision *revision* to the contents of *file*.

This is similar to using **svn propset --revprop** to set the **svn:log** property on a revision, except that you can also use the option **--bypass-hooks** to avoid running any pre- or post-commit hooks, which is useful if the modification of revision properties has not been enabled in the pre-revprop-change hook.



Revision properties are not under version control, so this command permanently overwrites the previous log message.

Options

--bypass-hooks
--revision rev, -r rev

Example

Set the log message for revision 19 to the contents of the file *msg*:

```
$ svnadmin setlog /usr/local/svn/repos/ -r 19 msg
```

setrevprop

svnadmin setrevprop repos_path -r revision name file

Set the property *name* on revision *revision* to the contents of *file*.

Options

--revision rev, -r rev

Example

Set the revision property *repository-photo* to the contents of the file *repo.png*:

```
$ svnadmin setrevprop /var/svn/repos/ -r 0 repository-
photo repo.png
```

setuuid

svnadmin setuuid repos_path [new_uuid]

Reset the repository UUID to *new_uuid*. If no new uuid is given generate a new one.

verify `svnadmin verify repos_path`

Run this command to verify the integrity of your repository. This iterates through all revisions in the repository by internally dumping all revisions and discarding the output.

Examining the Repository: `svnlook`

`svnlook` is a command-line utility for examining different aspects of a Subversion repository. It does not make any changes to the repository. `svnlook` is typically used by the repository hooks, but a repository administrator might find it useful for diagnostic purposes.

Since `svnlook` works via direct repository access (and thus can only be used on the machine that holds the repository), it refers to the repository with a path, not a URL.

If no revision or transaction is specified, `svnlook` defaults to the youngest (most recent) revision of the repository.

`svnlook Options`

Options in `svnlook` are global, just as in `svn` and `svnadmin`; however, most options apply to only one subcommand because the functionality of `svnlook` is (intentionally) limited in scope.

`--copy-info`

Used with the `changed` command to show detailed copy source information.

`--no-diff-deleted`

Do not print differences for deleted files. The default behavior when a file is deleted in a transaction/revision is to print the same differences that you would see if you had left the file but removed all the content.

`--no-diff-added`

Do not print differences for added files. The default behavior is to print the same differences that you would see if you added the entire contents of an existing but empty file.

`--revision rev, -r rev`

Examine revision number `rev`.

`--revprop`

Operate on a revision property rather than the property of a file or directory.
You must also specify a revision using `--revision` when using this option.

`--show-ids`

Show the filesystem node revision IDs for each path in the filesystem tree.

`--transaction tid, -t tid`

Examine transaction ID `tid`.

svnlook Subcommands

author `svnlook author repos_path`
Print the author of a revision or transaction in the repository.

Options

--revision rev, -r rev
--transaction tid, -t tid

cat `svnlook cat repos_path path_in_repos`
Print the contents of a file.

Options

--revision rev, -r rev
--transaction tid, -t tid

changed `svnlook changed repos_path`
Print the paths that were changed in a particular revision or transaction, as well as an **svn update**-style status letter in the first column: A for added, D for deleted, and U for updated (modified).

Options

--copy-info
--revision rev, -r rev
--transaction tid, -t tid

Example

Show a list of all the changed files in revision 39 of a test repository:

```
$ svnlook changed -r 39 /usr/local/svn/repos
A   trunk/vendors/deli/
A   trunk/vendors/deli/chips.txt
A   trunk/vendors/deli/sandwich.txt
A   trunk/vendors/deli/pickle.txt
```

date `svnlook date repos_path`
Print the timestamp of a revision or transaction in a repository.

Options

--revision rev, -r rev
--transaction tid, -t tid

diff `svnlook diff repos_path`
Print GNU-style differences of changed files and properties in a repository. If a file has a nontextual **svn:mime-type** property, the differences are explicitly not shown.

Options

- no-diff-added**
- no-diff-deleted**
- revision rev, -r rev**
- transaction tid, -t tid**

dirs-changed

`svnlook dirs-changed repos_path`

Print the directories that were themselves changed (property edits) or whose file children were changed.

Options

- revision rev, -r rev**
- transaction tid, -t tid**

help

`svnlook help [subcommand]`
`svnlook -h [subcommand]`
`svnlook -? [subcommand]`

Provide a quick usage summary. With *subcommand*, provide information about the given subcommand.

Alternate names: ?, h

history

`svnlook history repos_path [path_in_repos]`

Print information about the history of a path in the repository (or the root directory if no path is supplied).

Options

- limit num, -l num**
- revision rev, -r rev**
- show-ids**

Example

This shows the history output for the path /tags/1.0, as of revision 20 in our sample repository.

```
$ svnlook history -r 20 /usr/local/svn/repos /tags/1.0 \
> --show-ids
REVISION PATH <ID>
-----
19   /tags/1.0 <1.2.12>
17   /branches/1.0-rc2 <1.1.10>
16   /branches/1.0-rc2 <1.1.x>
14   /trunk <1.0.q>
...
...
```

info

`svnlook info repos_path`

Print the author, datestamp, log message size, and log message.

Options

- revision rev, -r rev**
- transaction tid, -t tid**

lock	<code>svnlook lock repos_path path_in_repos</code> Print available information about existing lock for path_in_repos . If no lock exists, print nothing.
log	<code>svnlook log repos_path</code> Print the log message.
	Options <code>--revision rev, -r rev</code> <code>--transaction tid, -t tid</code>
propget	<code>svnlook propget repos_path propname path_in_repos</code> List the value of a property on a path in the repository. Alternate names: pg , pget
	Options <code>--revision rev, -r rev</code> <code>--transaction tid, -t tid</code>
	Example Show the value of the <code>seasonings</code> property on the file <code>/trunk/sandwich</code> in the HEAD revision: <pre>\$ svnlook pg /usr/local/svn/repos sandwich /trunk/ mustard</pre>
proplist	<code>svnlook proplist repos_path path_in_repos</code> List the properties of a path in the repository. With <code>--verbose</code> , show the property values too. Alternate names: pl , plist
	Options <code>--revision rev, -r rev</code> <code>--revprop</code> <code>--transaction tid, -t tid</code> <code>--verbose (-v)</code>
	Examples Show the names of properties set on the file <code>/trunk/README</code> in the HEAD revision: <pre>\$ svnlook proplist /usr/local/svn/repos /trunk/README original-author svn:mime-type</pre> This is the same command as in the previous example, but this time it shows the property values as well:

```
$ svnlook proplist --verbose /usr/local/svn/repos \
> /trunk/README
original-author : fitz
svn:mime-type : text/plain
```

tree `svnlook tree repos_path [path_in_repos]`

Print the tree, starting at *path_in_repos* (if supplied; at the root of the tree otherwise), optionally showing node revision IDs.

Options

- `--full-paths`
- `--non-recursive, -N`
- `--revision rev, -r rev`
- `--show-ids`
- `--transaction tid, -t tid`

Example

This shows the tree output (with node IDs) for revision 40 in our sample repository:

```
$ svnlook tree -r 40 /usr/local/svn/repos --show-ids
/ <0.0.2j>
  trunk/ <p.0.2j>
    vendors/ <q.0.2j>
      deli/ <1g.0.2j>
        egg.txt <1i.e.2j>
        soda.txt <1k.0.2j>
        sandwich.txt <1j.0.2j>
```

uuid `svnlook uuid repos_path`

Print the UUID for the repository. The UUID is the repository's Universal Unique IDentifier. The Subversion client uses this identifier to differentiate between one repository and another.

youngest `svnlook youngest repos_path`

Print the youngest revision number of a repository.

Providing Remote Access: svnservice

svnservice allows access to Subversion repositories using the **svn** network protocol. You can run **svnservice** either as a standalone server process, or by having another process—such as **inetd**, **xinetd**, or **sshd**—start it for you.

Once the client has selected a repository by transmitting its URL, **svnservice** reads a file named *conf/svnservice.conf* in the repository directory to determine repository-specific settings, such as what authentication database to use and what authorization policies to apply. The details are provided in *Version Control with Subversion*.

svnserve Options

Unlike the previous commands we've described, **svnserve** has no subcommands; **svnserve** is controlled exclusively by options.

--daemon, -d

Run in daemon mode. **svnserve** backgrounds itself, and accepts and serves TCP/IP connections on the **svn** port (3690, by default).

--foreground

When used together with **-d**, this option causes **svnserve** to stay in the foreground. This option is mainly useful for debugging.

--help, -h

Display a usage summary and exit.

--inetd, -i

Use the standard input/standard output file descriptors, as appropriate for a server running out of **inetd**.

--listen-host=host

Listen on the interface specified by *host*, which may be either a hostname or an IP address.

--listen-once, -X

Accept one connection on the **svn** port, serve it, and exit. This option is mainly useful for debugging.

--listen-port=port

Listen on *port* when run in daemon mode.

--pid-file filename

Write process ID to *filename*.

--root=root, -r=root

Set the virtual root for repositories served by **svnserve** to *root*. The pathname in URLs provided by the client are interpreted relative to this root and are not allowed to escape this root.

--threads, -T

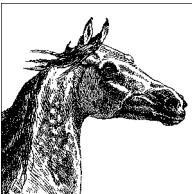
When running in daemon mode, spawn a thread instead of a process for each connection. The **svnserve** process still backgrounds itself at startup time.

--tunnel, -t

Run in tunnel mode, which is just like the **inetd** mode of operation (serve one connection over standard input/standard output), except that the connection is considered to be pre-authenticated with the username of the current UID. This flag is selected by the client when running over a tunnelling agent such as **ssh**.

--tunnel-user username

Used with **--tunnel** to specify an alternate username for pre-authentication.



14

The Git Version Control System

Git is a powerful, open source, distributed version control system (DVCS). It is not considered as user-friendly as Subversion but has many advanced features, including elegant merges of contributions from multiple independent sources and the ability to access the entire project history, even when not connected to a central server.

This chapter covers the following topics:

- Conceptual overview
- Using Git: a quick tour
- The Git command line client: `git`

Version control was introduced in Chapter 12, which contains a comparison of Subversion, Git, and other popular systems. A more thorough discussion on Git can be found in Jon Loeliger's *Version Control with Git* (O'Reilly).

Conceptual Overview

Git was originally created in 2005 by Linus Torvalds as a system for managing changes to the Linux kernel. Although it is used for many of the same tasks as other version control systems like Subversion, Git's internal workings are very different. It's important to understand some of these concepts in order to use Git successfully.

Git maintains a *repository*; a directory structure that tracks the historical contents of a set of files. Generally, the repository is stored in a `.git` directory (or another directory named by `$GIT_DIR`) along with the files themselves.

Unlike other version control systems, Git does not enforce the concept of a “central repository.” Instead, every set of files tracked by Git has its own `.git` repository, and revisions can be easily *pushed* and *pulled* from one repository to another.

The set of files currently being tracked by Git is called the *working tree*. The working tree is where most of your daily work takes place. By default, the working tree starts in the parent directory of the `.git` directory. (You can override this by setting `$GIT_WORK_TREE`.) The operation of creating a working tree from a repository is called *checking out* the files, which is done by `git checkout`. After modifying the files in the working tree, you save the changes back to the repository by *committing* them with `git commit` (this is also called *checking in*).

Unlike other systems, Git introduces an extra, intermediate state between the repository and the working tree called the *index*. (You might see the index called the *cache* in a few places, but this terminology is obsolete and should be avoided.) Initially, the index tracks the set of files as they were at checkout time. As you become confident that changes you made to the working tree are correct, you add them to the index using `git add`. Then, when the set of changes forms a coherent batch, you commit the index into the repository (along with a log message) using `git commit`. The main advantage of this two-step process is it's easy to limit a commit to only some of the changes you've been working on, which leads to more coherent individual commits. Git's working style—including the index—encourages many small commits instead of large batches.

Git Repository Format

The set of commits in a Git repository is stored in the form of a *directed acyclic graph*, or *DAG*. This simply means that each commit can reference one or more earlier “parent” commits, and more than one commit can refer to the same parents. The word “acyclic” refers to the fact that the structure is not allowed to contain loops; no parent commit can refer back to a commit that lists it as a parent.

The structure of the DAG defines the repository’s *history*. Normally, each commit has exactly one *parent*, which describes the repository exactly as it was before the new commit was made. By comparing a commit to its parent, you can produce a *diff*, which is a precise set of changes that were applied to the parent in order to produce the new version.

Some commits have more than one parent. These commits are called *merge commits* because they express a merging of two separate branches of history. If two people have a copy of a particular repository and start making commits, those two histories will start to diverge, which is called *forking*. Eventually, someone will need to rejoin the two histories into one, which is called *merging*. (As with other version control systems, you can also create additional named *branches* in each repository if you want. For example, you might create a maintenance branch for each major release of your software.)

Referring to Commits

Each commit in Git can be uniquely identified by its *commitid*, a SHA-1 hash code made up of 40 hexadecimal digits. Unlike revisions in centralized systems like Subversion, Git revision numbers cannot be sequential, since there’s no central server to assign the sequential IDs. Because it’s impossible for a human to remember strings of 40 digits, Git provides several more convenient ways to refer to commits.

Any Git command that can accept a revision can accept any of the following forms:

Full 40-digit hash

You can always simply supply the full 40-digit hash code, such as `da87b5990c03a799ae7a581c2edb1287dba08a43`.

Abbreviated hash

Since the 40 digits of a hash code are effectively random, it's very unlikely (though not impossible) that there will be more than one commit with the same hash. Thus, you can refer to any commit by the first few digits of its name, as long as only one commit starts with those digits. People often choose seven digits as a reasonably safe length. For example, `da87b59`.

Tags

Using the `git tag` command, you can create user-friendly names for individual commits. For example, if you released version v1.1 of your software after making commit `da87b59`, you could run `git tag v1.1 da87b59` so that in the future, you can always refer to a commit named `v1.1`. Tag names can be shared between repositories, but you have to do it explicitly using `git push` and `git pull`.

Local branches

Branches are similar to tags, in that they name a particular commit. However, branches are special because if you check out a branch and make a new commit, the branch advances automatically to point at the new commit.

Unlike with other version control systems, branch names in Git are maintained locally for every copy of a repository. That means if you clone someone else's repository, you take a copy of the branch called `master` (the default Git branch name). They may continue committing to their `master` branch, and you might commit to yours, and then there will exist two branches called `master`, each with different contents. You can resync them using `git push` and `git pull`.

git

Remote tracking branches

When you clone a central repository, you will frequently want to keep track of the branches as they exist in that repository, even as you make changes to them in your own repository. Git helps do this by naming repositories using the `git remote` command. After that, a particular branch on a particular remote would be named `remotename/branchname`. (Git automatically creates a remote named `origin` to identify the repository you originally cloned from, and the default branch in a repository is usually `master`. So you might have a local branch called `master` corresponding to a remote tracking branch called `origin/master`.)

HEAD

`HEAD` is a special name that always refers to the currently checked-out commit.

FETCH_HEAD

`FETCH_HEAD` is a special name that refers to the most recent commit retrieved by `git fetch`.

commitⁿ notation

For any commit, you can refer to its n -th direct parent by giving its name (using any acceptable form of a commit name, such as a branch name, tag, or hash), followed by \wedge , followed by the parent number you want. Most commits (other than merge commits) have only one parent, so if n is omitted, the immediate parent is returned. For example, `da87b591` is the first parent of `da87b59`; `HEAD21` is the first parent of the second parent of `HEAD`; `origin/master41` is four parents up from `origin/master`, taking the first parent at each step.

commit $\sim n$ notation

You can refer to a parent n steps up the tree using **commit $\sim n$** notation. For example, `HEAD ~ 4` is four parents up from `HEAD`, and is equivalent to `HEAD41`. Note that you can combine \sim and \wedge notation, so `HEAD2 ~ 4` is four steps up the tree from the second direct parent of `HEAD`.

branch@{n} notation

Because branch names can be retargeted at any time to refer to a different commit, you run the risk of accidentally losing a lot of work if you change or delete a branch name incorrectly. `branch@{n}` notation is designed to fix this; it refers to the commit `branch` referred to n commits ago. For example, if you use `git reset` or `git merge` and then change your mind, you can refer to the commit before the most recent change using `HEAD@{1}`.

Unlike \wedge and \sim notation, the name here must really be a branch name, not an arbitrary commit, since it makes no sense to refer to the old meaning of a particular commit. However, you can use \wedge and \sim on the output of `commit@{n}`, however. For example, `HEAD@{1}2 ~ 4` .

You can see the history of a branch name in a format suitable for use with `commit@{n}`, using `git reflog`.

Using Git: A Quick Tour

Git permits a staggering number of different workflow styles, from “almost centralized” (everyone frequently pushes and pulls to the same central server) to entirely email-based (people exchange patches using a mailing list). It would be impossible to explain all the different Git workflows here, so we’ll focus on just one: a simple push/pull workflow with a single shared repository.

Before You Start

Before you start using Git for the first time, you need to set two global configuration values, `user.email` and `user.name`, or else you won’t be able to make new commits. Here’s how:

```
$ git config --global user.name 'John Smith'  
$ git config --global user.email 'jsmith@example.com'
```

These settings will remain in place across all your repositories, so you don’t have to reset them every time. You can also override them on a per-repository basis if you want, by running the commands in a particular repository and omitting the `--global` option.

Example: The Linux Kernel Repository

As our first example, let's clone a copy of the Linux kernel into `/tmp/linux-2.6`:

```
$ cd /tmp  
$ git clone git://git.kernel.org/pub/scm/linux/kernel/git/torvalds/  
linux-2.6.git linux-2.6  
Initialized empty Git repository in /tmp/linux-2.6/.git/  
remote: Counting objects: 1177432, done.  
remote: Compressing objects: 100% (189064/189064), done.  
remote: Total 1177432 (delta 982454), reused 1176803 (delta 981879)  
Receiving objects: 100% (1177432/1177432), 288.00 MiB | 406 KiB/s, done.  
Resolving deltas: 100% (982454/982454), done.  
Checking out files: 100% (27842/27842), done.
```

Now you have a copy of the entire Linux kernel and its development history, including many different tags and branches. You've started out on branch `master`, which is the latest version. Check out the `v2.6.20` tag to get an older version:

```
$ cd linux-2.6  
$ git checkout v2.6.20  
Checking out files: 100% (33554/33554), done.  
Note: moving to 'v2.6.20' which isn't a local branch  
If you want to create a new branch from this checkout, you may do so  
(now or later) by using -b with the checkout command again. Example:  
  git checkout -b <new_branch_name>  
HEAD is now at 62d0cfc... Linux 2.6.20
```

You can't make changes to tags, and you haven't created a local branch for your work, so Git has given you what it calls a *disconnected HEAD*. You can make commits, but they won't be attached to any branch at all. That's a bit dangerous, since it's easy to lose track of your work that way. Let's name our work so it doesn't get lost:

```
$ git checkout -b my-test-branch  
Switched to a new branch "my-test-branch"
```

And look at the list of local branches:

```
$ git branch  
  master  
* my-test-branch
```

You can make a test commit:

```
$ echo 'hello world' >hello.txt  
$ git add hello.txt  
$ git commit -m 'my first hello commit'  
Created commit 22b0a19: my first hello commit  
1 files changed, 1 insertions(+), 0 deletions(-)  
create mode 100644 hello.txt
```

Remember, this new commit hasn't been shared with anyone; you've only committed it to your local repository.

Try pulling in the changes from *v2.6.20* to *v2.6.21*:

```
$ git pull origin v2.6.21
Merge made by recursive.
...
$ ls hello.txt
hello.txt
```

All the changes from *v2.6.21* have been merged into your branch, but your new *hello.txt* is still there too. Success! If you had permission to push your new branch back to the central kernel repository, which you probably don't, you could do it now:

```
$ git push origin my-test-branch
Counting objects: 228, done.
Delta compression using 2 threads.
Compressing objects: 100% (165/165), done.
Writing objects: 100% (228/228), 38.66 KiB, done.
Total 228 (delta 142), reused 77 (delta 60)
To git://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux-2.6.git
  linux-2.6
 * [new branch]      my-test-branch -> my-test-branch
```

Someone could then pull your branch into theirs and receive your *hello.txt* changes.

Creating and Sharing a New Repository

If you have a project that isn't already in Git, you will need to first create a local repository for it. Let's say you have a directory called *my-project* in which you already have a set of files. You want to keep it under version control. Here's what you do:

```
$ cd my-project
$ git init
Initialized empty Git repository in .git/
$ git add .
$ git commit -m 'Initial commit'
```

(The *git add* command above won't work unless there is at least one file to add, and *git commit* won't work until you've *git added* at least one file.)

For personal projects, that might be all you need; you can now create commits, branches, and tags, compare differences, and so on.

If you want to share your project with someone else, however, you will need to create a bare repository (i.e., one with no work tree of its own) and give other people access to it. In this example, we'll create a new bare repository in */home/git/my-project.git*, copied from */tmp/my-project*, and give access to everyone in the *mygroup* Unix group:

```
$ git clone --bare /tmp/my-project /home/git/my-project.git
Initialized empty Git repository in .git/
$ cd /home/git/my-project.git
$ git config core.sharedRepository group
$ chgrp -R mygroup .
$ chmod -R g+rwx .
```

By convention, bare repositories have pathnames that end in a `.git` suffix. This is because they don't contain a `.git` subdirectory—the `.git` subdirectory contents are in the root of the bare repository, since there's no work tree.

People in `mygroup` can now `git clone /home/git/my-project.git` and push and pull from it.

If you want to host an open source project that everyone on the Internet can share, you have several options:

git daemon

You can run the Git daemon on your server. The `git daemon` command is beyond the scope of this book, but you can find instructions in `git help daemon`.

Gitosis

Gitosis is a separate tool that lets you create a single SSH account and give multiple people access to a Git repository through that.

git shell

The `git shell` command (see `git help shell`) is included with Git and does a similar job to Gitosis. Most people prefer Gitosis.

<http://github.com>

A commercial service, `github.com`, offers free hosting services for open source projects and reasonably priced hosting of proprietary projects.

Gitorious

Gitorious is an open source tool similar to `github.com`. You can use it to create your own `github.com`-like hosting service.

<http://repo.or.cz>

`repo.or.cz` was the original public Git hosting service. It provides basic push/pull access, although it isn't as user friendly as the other alternatives.

Because Git repositories are fully distributed, you can choose to host your project in more than one place at a time for added reliability, in case a server is unavailable or loses its data.

The Git Command Line Client: git

The git Command

Everything you do with Git is accomplished using the `git` command. The `git` main program doesn't actually do anything itself; instead, it runs subcommands based on the first word on the command line, often executing a subprogram based on that name. For example, if you run `git add`, Git might end up executing a program called `git-add`. In fact, if you create additional `git-*` commands in your `$PATH`, Git will add them to its repertoire automatically.

Git contains a daunting number of subcommands—well over 100 of them. Most of these commands are meant to be used internally to create other subcommands. We'll cover only the most important commands in this book.



Accessing Git's Online Help

You can ask the `git` command for help using the `git help` command. By default, it just gives a list of the most common commands:

```
$ git help
The most commonly used git commands are:
  add           Add file contents to the index
  apply         Apply a patch on a git index file and a working tree
  archive      Create an archive of files from a named tree
  ...
  ...
```

You can also get a complete list of all the commands:

```
$ git help -a
git commands available in '/usr/bin'
-----
  add           gui          reflog
  add--interactive  hash-object  relink
  am            http-fetch   remote
  ...
  ...
```

Finally, you can get the Unix manpage for a particular command (every Git command, even the obscure ones, has a manpage). For example, to get help for the `git add` command:

```
$ git help add
```

git Subcommands

add `git add filename ...`

Add or update a file in the index so its changes will be committed upon the next call to `git commit`. This is called *staging* the change. Unlike other systems, Git commits only the exact file contents that you have added, so if you make further changes to a file, you will need to add it again before running `git commit`.

With the `-p` option, Git lets you stage individual changes (“ hunks ”) interactively, even inside a particular file. You can use this to help break your commits into smaller, more understandable pieces.

Options

- n (no-op)
- v (verbose)
- f (force)
- p (partial)

Examples

To stage a new or updated file in the index:

```
$ git add myfile.c
```

To stage all the files in the current directory, except the ones ignored by `.gitignore`:

```
$ git add.
```

To add a file even though it is ignored by `.gitignore`:

```
$ git add myfile.o
The following paths are ignored by one of your .gitignore
files:
myfile.o
Use -f if you really want to add them.
fatal: no files added
$ git add -f myfile.o
```

To add only part of a file:

```
$ git add -p README
diff --git a/README b/README
index 42644cd..e2ad5c3 100644
--- a/README
+++ b/README
@@ -1 +1,2 @@
...
Stage this hunk [y,n,q,a,d/,e,?]? y
```

archive `git archive [--remote=repository-path] [--format=zip] revision
-- [path ...] >outputfile`

Create a tar or zip format archive from the current repository or from the repository at repository-path if `--remote` is specified. The default output file format is (uncompressed) tar, but `git archive` can also produce a zip file instead if you specify `--format=zip`.

`git archive` always writes the new archive to standard output, so you should redirect or pipe its output somewhere.

You must specify the revision from which to take the files. If you want to use the current revision, use `HEAD`.

If you specify one or more paths, only those paths are included in the produced archive.

Options

- `--list`
- `--remote`
- `--format=tar|zip`

Examples

To list the available options for `--format`:

```
$ git archive --list
tar
zip
```

To create a `tar.gz` file of the currently checked-out revision:

```
$ git archive HEAD | gzip >my-release.tar.gz
```

To create a zip file of tag `v1.2` in a repository from another computer named `myserver` (where you have Unix shell access via ssh):

```
$ git archive --format=zip --remote=myserver:src/myapp.git
v1.2 >myapp-1.2.zip
```

A small black square containing the white text "git".

bisect

```
git bisect start [bad-commit [good-commit ...]] -- [path ...]
git bisect bad [commit ...]
git bisect good [commit ...]
git bisect skip [commit ...]
git bisect reset
git bisect view
git bisect run cmd [args ...]
```

Go back in history to find the first commit that introduced a problem. `git bisect` uses a binary search algorithm to narrow down which commit caused a problem using as few steps as possible. If there are n commits to consider, `git bisect` can find the exact culprit with approximately $\log_2 n$ attempts. For example, with 100 commits, it will take about 7 tries; with 1,000 commits, it will take about 10 tries.

git bisect start [bad-commit [good-commit ...]] -- [path ...]

Use this command to start the bisection. You can optionally specify one known *bad-commit* (which exhibits the problem) and one or more known *good-commits* (which do not exhibit the problem). If you know the bug is in a particular set of files or directory, specify them as *paths* to further narrow the set of commits to consider.

git bisect bad [commit ...]

Mark the given *commit(s)* as bad and check out the next candidate. If no *commits* are provided, the default is the currently checked-out **HEAD**.

git bisect good [commit ...]

Mark the given *commit(s)* as good and check out the next candidate. If no *commits* are provided, the default is the currently checked-out **HEAD**.

git bisect skip [commit ...]

Mark the given *commit(s)* as untestable and check out the next candidate. You need this if some other bug exists in the current commit that prevents you from testing the bug you're looking for. If no *commits* are provided, the default is the currently checked-out **HEAD**.

git bisect reset

Abort the bisection and check out the commit you were using when you ran `git bisect start`.

git bisect view

Show the current bisection status using `gitk`.

git bisect run cmd [args ...]

Continue the bisection automatically by running the given `cmd` on each candidate. If the command returns an error, the commit is considered bad; otherwise, it is considered good.

Examples

To look for a problem when you know revisions *v2.0* and *v1.1* are good, but the current version is bad, and the bug is probably in the *mymodule* subdirectory:

```
$ git bisect start HEAD v2.0 v1.1 -- mymodule
Bisecting: 100 revisions left to test after this (roughly
7 steps)
[fa16ab36ad014bcc03acc4313bb0918fb241b54d] Fix the widget
$ make test
...
$ git bisect bad
Bisecting: 50 revisions left to test after this (roughly
6 steps)
[49cf82288aac5f0dc152e2d75cd340e48d9e760] Change some
bits
$ make test
...
$ git bisect good
fa16ab36ad014bcc03acc4313bb0918fb241b54d is first bad
commit
```

To find a bug that was introduced somewhere between *v1.1* and *v2.0*, and that could be in any file:

```
$ git bisect start
$ git bisect good v1.1
$ git bisect bad v2.0
Bisecting: 97 revisions left to test after this (roughly
7 steps)
[fa16ab36ad014bcc03acc4313bb0918fb241b54d] Fix the widget
$ git bisect run make test
...
1f73862f3b63bbc9f0a8a8a12dd58e1a39a3355f is first bad commit
git bisect run success
```

branch

```
git branch [-r] [-a] [--contains commit]
git branch [-f] [--track] branchname [commit]
git branch -m oldname newname
git branch -d|-D [-r] branchname
```

List, create, rename, and delete branches.

git branch [-r] [-a] [--contains commit]

List the existing branches. By default, lists only local branches (*.git/refs/heads/**). With **-r**, list remote tracking branches instead (*.git/refs/remotes/*/**). With **-a**, list all local and remote tracking branches. With **--contains**, list only the branches that contain the given *commit*.

git branch [-f] [--track] [--no-track] *branchname* [*commit*]

Create a new branch, *branchname*, that points at the given *commit*. The default for *commit* is **HEAD**. If *branchname* already exists, the operation will fail unless **-f** is specified. If you specify **--track** (the default in newer versions of Git) and *commit* is a remote tracking branch, you will be able to type simply **git merge** or **git pull** in the future to merge remote changes into your branch.



This command does not check out your new branch. If you want to create a new branch and check it out in one step, use **git checkout -b**.

git branch -m|-M *oldname* *newname*

Rename a branch from *oldname* to *newname*. If *newname* already exists, **-m** will fail while **-M** will overwrite it.

git branch -d|-D [-r] *branchname*

Delete *branchname*. If **-r** is specified, *branchname* is a remote tracking branch (`.git/refs/remotes/*/*`); otherwise it is a local branch (`.git/refs/heads/*`). If you use **-d**, the branch will only be deleted if your current **HEAD** already includes everything in *branchname*; otherwise, it will fail. If you use **-D**, this safety check doesn't occur.

Examples

To get the list of all branches:

```
$ git branch -a
```

To create a new branch that starts off pointing at your current **HEAD**:

```
$ git branch savepoint1
```

To later delete the *savepoint1* branch:

```
$ git branch -d savepoint1
```

To create and check out a new branch *test* based on remote tracking branch *origin/master*:

```
$ git branch --track test origin/master
```

```
$ git checkout test
```

Switched to branch 'test'

checkout

git checkout [*revision*] *path* ...

Copy files from the repository into your working tree and possibly switch branches. **git checkout** does one of four different things, depending on which options are provided:

revision but no *path*

Switch branches to the one named by *revision*. If files are modified in the working tree and also changed in the new branch, the operation will fail. The next commit will be in the new branch.

If *revision* is a valid commit but not the name of a local branch, Git still switches **HEAD** to the given commit but does not give the new branch a name. This is called a *detached HEAD*. You can later name the new branch with **git checkout -b**.

In either case, if the **-b** option is given, a new branch is created from the given *revision*, and Git switches to that branch.

path(s) but no *revision*

Destroys all working tree changes in the named files or directories, replacing them with their contents from the index.

revision and *path(s)*

Destroys all working tree changes in the named files or directories, replacing them with their contents from the given *revision*. This does *not* switch branches; instead, it replaces contents of the given files in the index. The next **git commit**, will affect the same branch as before, but the named paths will be considered modified.

no *revision* and no *path(s)*

No changes. Prints a list of files that have been modified in the working tree.

Options

- q (quiet)
- f (force)
- b *new_branch_name*
- track (-t)
- no-track
- m (merge)

Examples

To revert all the files in directory *docs* to their version from the index:

```
$ git checkout docs
```

To switch to the branch called *new-feature*, which must already exist:

```
$ git checkout new-feature
Switched to branch "new-feature"
```

To switch to the remote branch *origin/master*, creating a new branch called *new-feature*:

```
$ git checkout -b new-feature origin/master
Branch new-feature set up to track remote branch refs/
remotes/origin/master.
Switched to a new branch "new-feature"
```

cherry-pick

git cherry-pick *commit*

Take the changes from an individual *commit* (usually one from another branch) and apply it to the current branch. Note that unlike **git merge**, **git cherry-pick** applies only a single change, not all the history leading up to that change.

The newly created commit is completely independent of the original, although it has the same commit message by default. `git cherry-pick commit` is the rough equivalent of `git show commit | patch -p1`

Options

- e, --edit
- x (extend commit message)
- m *parent-number*
- n, --no-commit
- s, --signoff

Examples

To cherry-pick commit `1a48191` onto the current branch and edit the commit message:

```
$ git cherry-pick -e 1a48191
Finished one cherry-pick.
```

clean

`git clean [-n|-f [-d] [-x]]`

Removes files from the work tree that are not in the Git index. By default, it only removes files that are not listed in `.gitignore` and does not remove directories. (To undo work tree changes to files that are in the index, use `git checkout`.)

You must provide one of `-n` or `-f`.

Options

- n (print only, do not remove)
- f (force removal of files)
- d (also remove extra directories)
- x (also remove files skipped by `.gitignore`)
- X (only remove files skipped by `.gitignore`)
- q (quiet)

Examples

To remove all extra files in the work tree so the only files remaining are ones that would be there after a fresh `git clone`:

```
$ git clean -f -d -x
Removing foo
```

To see what would happen if you ran the above command:

```
$ git clean -n -d -x
Would remove foo
```

clone

`git clone repository [local-directory]`

Make a copy of repository (which can be a local or remote Git repository) in local-directory. By default, if repository is on the local filesystem, Git will use hardlinks to copy the `.git/objects` folder to minimize disk space waste.

Available forms for *repository* are:

```
/path/repo.git  
rsync://hostname/path/repo.git  
http://hostname/path/repo.git  
https://hostname/path/repo.git  
git://host/path/repo.git  
ssh://host/path/repo.git  
ssh://user@host:port/path/repo.git  
host:path/repo.git  
user@host:path/repo.git
```

Options

- s, --shared
- no-hardlinks
- reference *other-parent-repository*
- q, --quiet
- n, --no-checkout
- bare
- o *origin-name*, --origin *origin-name*
- depth *shallow-clone-depth*

Examples

To clone the Linux kernel repository:

```
$ git clone git://git.kernel.org/pub/scm/linux/kernel/git/  
torvalds/linux-2.6.git linux-2.6
```

commit

git commit [-- *file* ...]

Commit the changes that have already been staged in the index, updating **HEAD**.

If **-a** is given, all changed files in the work tree—not only staged changes—will be committed. (**-a** is the common behavior in systems like Subversion.)

If files are specified, exactly those files are committed, regardless of the state of the index.

You cannot commit a file for the first time, even with **-a**, unless it has first been added with **git add**.

Options

- a (all files, not just added ones)
- amend
- m *commit-message*, --message=*commit-message*
- F *commit-message-file*
- q, --quiet
- s, --signoff

The Git logo, which consists of the word "git" in a white sans-serif font inside a dark blue rounded rectangle.

Examples

To commit changes only to files in the *mylib* directory:

```
$ git commit -- mylib  
Created commit 90716b6: my new commit  
 1 files changed, 13 insertions(+), 0 deletions(-)  
 create mode 100644 test-file
```

config

```
git config [--global] --list  
git config [--global] name  
git config [--global] name value
```

List, get, or set configuration values. If **--global** is given, uses your account-wide default settings (in *~/.gitconfig*); otherwise uses the settings for the current repository (in *.git/config*). Instead of using **git config**, you can also just view or edit these files directly.

The most important settings are:

core.autocrlf

If true, converts LF line endings to CRLF on files when checking out and converts the line endings back to LF when committing.

core.bare

Makes this repository a bare repository, which means it has no work tree. (Public shared repositories are usually bare.)

core.sharedRepository

Set to either **all**, **group**, or **false**. If **group**, the files in *.git* will be group readable and writable. If all, the files will also be readable (but not writable) by everyone.

core.editor

The path of your favourite text editor, for editing commit messages. The **\$EDITOR** environment variable overrides this.

gc.autopacklimit

Controls how frequently Git will automatically run **git gc** after common operations. To disable it completely, use **0**.

rerere.enabled

When committing a conflicted merge, Git will remember how you resolved the conflict and attempt to reuse the recorded resolution if it encounters it again in the future.

user.email

Your email address. This will be attached to commit messages automatically. The **\$EMAIL** environment variable overrides this.

user.name

You full name. This will be attached to commit messages automatically. The **\$GIT_COMMITTER_NAME** and **\$GIT_AUTHOR_NAME** environment variables override this.

Options

--unset

Examples

To set your email address and name for future commits:

```
$ git config --global user.name 'John Smith'  
$ git config --global user.email 'jsmith@example.com'
```

To list all your global settings:

```
$ git config --global --list  
user.name=John Smith  
user.email=jsmith@example.com  
gc.autopacklimit=0
```

diff

`git diff first-commit [second-commit] [-- path ...]`
`git diff --cached [first-commit] [-- path ...]`

Show the differences between two revisions. If paths are specified, restricts the comparison to only the given files or directories.

The default *first-commit* is HEAD.

The default *second-commit* is normally the work tree. With --cached, the default *second-commit* is the index.

Options

- `--name-only`
- `--name-status`
- `--stat` (show diffstat instead of patch)
- `-a, --text`
- `-M` (detect renames)
- `-C` (detect copies and renames)
- `--find-copies-harder`
- `-R` (reverse patch)
- `-w, --ignore-all-space`
- `--exit-code`
- `-Un, --unified=n`

Examples

To see which changes are already staged in the index:

```
$ git diff --cached
```

To see which changes are in the work tree but not yet staged:

```
$ git diff
```

To see a summary of changes between two tags (*v1.1* and *v2.0*):

```
$ git diff --stat v1.1 v2.0
```

To see the changes between *v1.1* and the current work tree:

```
$ git diff v1.1
```

To see the changes between *v1.1* and HEAD:

```
$ git diff v1.1 HEAD
```

fetch	<code>git fetch [remote-name]</code> <code>git fetch repository [remoteref[:localref]]</code>
	Fetch commits from a remote Git repository and adds it to the local one.
	In the first form, fetch all branches and tags from the given <i>remote-name</i> (a remote repository set up using <code>git remote</code>). The default remote name is origin .
	In the second form, fetch a particular <i>remoteref</i> from repository and store it as local branch <i>localref</i> . The default <i>localref</i> is FETCH_HEAD . The default <i>remoteref</i> is HEAD .

Options

- f, --force
- n, --no-tags
- t, --tags

Examples

To fetch the latest Linux kernel release and compare it against your current work tree:

```
$ git fetch git://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux-2.6.git linux-2.6 master
From git://git.kernel.org/...
 * branch           master    -> FETCH_HEAD
$ git diff FETCH_HEAD
```

To update all the remote tracking branches attached to the remote named *origin*:

```
$ git fetch origin
remote: Counting objects: 15, done.
remote: Compressing objects: 100% (10/10), done.
remote: Total 10 (delta 8), reused 0 (delta 0)
Unpacking objects: 100% (10/10), done.
From git://git.kernel.org/...
   6544ab2..e3498f3  master    -> origin/master
```

To retrieve the branch *test1* from someone's repository and save it as the local branch *mytest1*:

```
$ git fetch git://git.kernel.org/... test1:mytest1
```

gc	<code>git gc [--prune] [--aggressive]</code>
	Pack the <i>.git/objects</i> directory to save disk space and increase speed.

`git gc` is run automatically from time to time, so it is rarely needed unless you want to use the **--prune** or **--aggressive** options or want to force packing to happen at a particular time (such as before making a backup).

Options

- aggressive** (take extra time to save even more space)
- prune** (delete unused objects)

Examples

To aggressively repack the current repository and save as much space as possible:

```
$ git gc --aggressive --prune
Generating pack...
Done counting 3299 objects.
Deltifying 3299 objects...
 100% (3299/3299) done
Writing 3299 objects...
 100% (3299/3299) done
Total 3299 (delta 2225), reused 0 (delta 0)
Pack pack-4eb8f89a145f826ef93923fe97c4ab23bd8abb62
created.
Removing unused objects 100%...
Done.
```

gitk

`gitk [git-log options...]`

Display a graphical browser showing the Git history. `gitk` takes all the same options as `git log`.

Examples

To show the history of all changes to the *mylib* directory on branches *test1* and *test2*, but leave out all changes that are included in *v1.1*:

```
$ gitk test1 test2 ^v1.1 -- mylib
```

grep

`git grep [-e] pattern [--cached|commit ...] [-- path ...]`

Search the given commits for the given regular expression pattern(s).

If no *commits* are specified, normally searches the current work tree. With `--cached`, searches the index instead. If *paths* are specified, restricts the search to the given files or directories.

The main advantage of `git grep` over plain `grep` is that it ignores files (such as compiler output and editor backups) that have not been added with `git add`.

When multiple patterns are specified (using `-e`), the default combination is `--or`, unless `--and` or `--not` is specified.

Options

- `-e pattern`
- `--and`
- `--or`
- `--not`
- `(`
- `)`
- `-E` (act like egrep)
- `-F` (act like fgrep)
- `-i, --ignore-case`

A small black square containing the white text "git".

-v, --invert-match
-w, --word-regexp
-l, --files-with-matches
-L, --files-without-matches

Examples

To search the work tree for lines containing *chicken* without considering case:

```
$ git grep -i chicken
```

To search the top of branches *test1* and *test2* for lines containing *alpha* and either *beta* or *gamma*:

```
$ git grep -e alpha --and \(\ -e beta -e gamma \) test1  
test2
```

To search only the *mylib* directory for files starting with the word *chicken*:

```
$ git grep '^chicken' -- mylib
```

init

`git init [-q] [--bare] [--shared=false|group|all]`

Create a new Git repository in the current directory.

Options

-q, --quiet
--bare
--shared=false|group|all

Examples

To create a new repository with default settings:

```
$ git init
```

To create a new bare repository that will be shared with everyone in your Unix group:

```
$ git init --bare --shared=group
```

log

`git [revision ...] [-- path ...]`

Show the commit history. If *paths* are specified, show only the history related to the given files or directories. If *revisions* are specified, show the history starting from the given *revisions*. The default *revision* is **HEAD**. With **--all**, the default is to show all revisions in all local branches.

Each *revision* can be specified in one of the following formats:

commit

Include the history for the given commit.

^commit

Exclude the history starting at the given commit.

a..b

Includes all commits in *b* that are not in *a*. Equivalent to *b ^a*.

a...b

Includes all commits in *a* and *b* that are not in both.

Options

- all
- p (show patch)
- Un, --unified=n
- a, --text
- raw
- stat
- M (detect renames)
- C (detect copies)
- find-copies-harder
- R (reverse patch)
- w, --ignore-all-space
- n, --max-count=n (show up to only n commits)
- pretty=(oneline|short|medium|full|fuller|email|raw
format....)
- abbrev-commit
- full-history
- no-merges
- first-parent
- parents
- left-right
- graph
- author=regex
- committer=regex
- grep=regex
- reverse

Examples

To show the commits on this branch, starting with the most recent:

```
$ git log
commit 3ad3a1c1866bef36461d549d87fe39babe412d61
Author: John Smith <jsmith@example.com>
Date:   Sun Jan 18 18:41:32 2009 -0500
```

Make some changes.

```
commit 529fc80df85a5ec7c88552bcb27bc0770a84e336
Author: John Smith <jsmith@example.com>
Date:   Sun Jan 17 12:13:06 2009 -0500
```

Do the first thing.

...

To show what was changed by the most recent commit:

```
$ git log -1 -p
diff --git a/server/hello.html b/server/hello.html
index 63ede62..e26d280 100644
--- a/server/hello.html
+++ b/server/hello.html
@@ -1,17 +1 @@
...
...
```

To show all commits in all local branches where the commit message contains the word *hello*:

```
$ git log --grep=hello --all
commit 3ad3a1c1866bef36461d549d87fe39babe412d61
Author: John Smith <jsmith@example.com>
Date:   Sun Jan 18 18:41:32 2009 -0500
```

Undo changes to hello.html.

...

merge

git merge commit ...

Merge one or more other branches into the current **HEAD**. For each specified *commit*, Git calculates the set of changes on that branch that are not currently in **HEAD** and attempts to apply those changes to the current **HEAD**. If the changes cannot be applied successfully, Git will leave conflicts in the index, which you will need to resolve by hand (using an editor, **git add**, and **git commit**).

Almost always, you will supply exactly one *commit*. In some situations, you may wish to merge more than one branch at the same time; this is called an *octopus* merge and is allowed only if none of the merges cause any conflicts. It is equivalent to merging the given *commits* one by one, except that the merged result produces only one new commit.

You can override the merge strategy to be used, but this is almost never necessary. When more than one commit is specified, the only allowed strategy is *octopus*.

Normally, Git combines the branch histories so **git log** will show all the commits from all branches that have been merged. With **--squash**, it eliminates the history of the branches other than **HEAD**. This simplifies the **git log** output, but prevents successful merges from that branch in the future, so it is usually a bad idea.

Options

- no-commit**
- squash**
- log**
- no-ff**
- s (resolve|recursive|octopus|ours|subtree)**
- m msg**

Examples

To merge feature branches *feature1* and *feature2* into your current branch using an *octopus* merge:

```
$ git merge feature1 feature2
Already up-to-date with
aa871d4ef9657e03b2ef7053dc13a16777955499
Trying simple merge with
bdd84225b4f7c282731aed540171e7cbe392c00d
Merge made by octopus.
 58 files changed, 1371 insertions(+), 195 deletions(-)
```

To merge the branch `production` into the current `HEAD`, resulting in a conflict:

```
$ git merge production
Auto-merging testfile.c
Auto-merging server/hello.html
CONFLICT (content): Merge conflict in server/hello.html
Auto-merging test2.c
Recorded preimage for 'server/hello.html'
Automatic merge failed; fix conflicts and then commit the
result.
```

To resolve the conflict above and commit the completed merge:

```
(edit server/hello.html)
$ git add server/hello.html
$ git commit
```

mv

```
git mv oldfile newfile
git mv oldfiles ... newdir
```

Rename *oldfile* to *newfile* or move a series of *oldfiles* into the directory *newdir*, which must already exist. `git mv` updates both the work tree and the index.



There is no need for a `git cp` (copy) command. Just copy the file using Unix `cp` and `git add`. Git is still able to track the history of files moved and copied in this way.

git

Options

- f (overwrite even if newfile exists)
- n, --dry-run

Examples

To rename *file1.c* to *file1b.c*:

```
$ git mv file1.c file1b.c
```

To move *file1.c* and *file2.c* into the *mylib* directory:

```
$ git mv file1.c file2.c mylib/
```

pull

```
git pull
git pull repository branch
```

Fetch the given branch then merge it into `HEAD`. `git pull` is just a short form for `git fetch` followed by `git merge`.

If no *repository* or *branch* is specified, `git pull` attempts to pull from the remote tracking branch associated with your current branch, if any. (Associations are set up using the `--track` option when creating a branch with `git branch`. You can also add them later with `git config`.)

The *repository* can be a local repository path, a repository URL (see `git clone`), or a remote name (see `git remote`). The *branch* must be a valid branch or tag name in the remote repository.

For more information about how pull works, see `git fetch` and `git merge`.

Options

- no-commit
- squash
- log
- no-ff
- s (resolve|recursive|octopus|ours|subtree)
- tags
- no-tags

Examples

To pull the latest Linux kernel changes into your current branch:

```
$ git pull git://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux-2.6.git master
From git://git.kernel.org/...
 * branch            master      -> FETCH_HEAD
Merge made by recursive.
 21 files changed, 932 insertions(+), 66 deletions(-)
```

push

git push [--all] [--tags] [repository [localref][:remoteref]]

Push changes from one or more branches or tags into the specified remote repository.

Repository can be a local repository path, a repository URL (see `git clone`), or a remote name (see `git remote`). If repository is omitted, it defaults to `origin`.

If *localref* is omitted, it defaults to pushing all the refs that currently exist on the remote. For example, if you have local branches named A, B, C, and D, and the remote has branches named A, C, and E, then Git will push branches A and C unless you specify a specific *localref*.

As an alternative to providing *localref*, you can provide `--all` or `--tags`. With `--all`, Git pushes all the local branches. With `--tags`, Git pushes all the local tags.

If *remoteref* is omitted, it defaults to the same name as *localref*. If you supply *remoteref*, it needs to be the full refname, such as `refs/heads/master`, not just the base branch name.

If you supply a *remoteref* without a *localref*, Git deletes *remoteref* in the remote repository. This is useful if the remote end refuses to update a ref because it is not a fast forward; you can delete the remote ref, and then re-create it.

Options

- all
- tags
- dry-run
- thin

Examples

To push the current branch (master) to the remote named `origin`:

```
$ git push origin master
```

To push all local branches to the default remote (`origin`), as long as they already exist there:

```
$ git push
```

To push all the local branches to a particular remote repository:

```
$ git push --all server:/git/myrepo.git
```

To delete the branch `test1` on a remote named `myremote`:

```
$ git push myremote :test1
```

To push the current `HEAD` to a branch named `test1` on `myremote`:

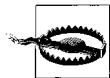
```
$ git push myremote HEAD:refs/heads/test1
```

rebase

`git rebase [-i] [--onto onto-commit] base-commit [switch-branch]`
`git rebase --continue | --skip | --abort`

Automatically cherry-pick a series of commits between `base-commit` and `switch-branch` onto `onto-commit`, leaving the result in `switch-branch`.

If `switch-branch` is omitted, the default is `HEAD`. If `onto-commit` is omitted, the default is `base-commit`.



Using `git rebase` can completely disrupt future merges to and from the rebased branch. Use `git rebase` only for changing commits that have never been shared with others, such as before a new branch has been pushed to a shared repository.

`git rebase` is a powerful alternative to `git merge`. You can use it to simplify and rewrite the history of your changes in order to make them easier to review, audit, and share. However, because `git rebase` rewrites the history of your repository, misuse can cause hard-to-resolve errors, including duplicate commits and merge conflicts.

You can imagine `git rebase` as a series of commands that looks something like this:

```
git checkout switch-branch  
git reset --hard onto-commit  
for commit in base-commit..switch-branch  
    git cherry-pick commit  
done
```

The end result is a new branch that contains `onto-branch` plus a linear series of commits (i.e., with no merges). The new commits consist of all the commits that were previously part of `switch-branch` but not `base-commit`.

With `-i`, `git rebase` opens an interactive editor before the rebase operation starts. The editor contains a list of all the commits in

base-commit..switch-branch. You can rearrange the list, add/delete individual entries, or merge (squash) multiple commits into a single one, thus allowing you to produce a set of commits that is easier for others to review.

During a rebase operation, cherry-picking a particular commit may result in a conflict. In that case, you need to do one of the following:

- Resolve the conflict, commit it with **git commit**, and then run **git rebase --continue**.
- Run **git rebase --skip** to skip this commit entirely.
- Run **git rebase --abort** to cancel the rebase operation and change the Git history back to the way it started.

Options

-i, --interactive
--onto *onto-commit*
-p, --preserve-merges

Examples

To take all the patches in **HEAD** that are not in *origin/master*, and apply them on top of the current *origin/master*:

```
$ git rebase origin/master
```

To do the same as the above, but interactively reorder, edit, and squash the patches:

```
$ git rebase -i origin/master
```

To take all the patches in the *feature1* branch that are not in *origin/master*, and add them on top of *feature2*, resulting in a new *feature1*:

```
$ git rebase --onto feature2 origin/master feature1
```

To do the same as the above in two steps for clarity:

```
$ git checkout feature1  
$ git rebase --onto feature2 origin/master
```

To resolve a conflict that arose during the rebase operation:

```
(edit test1.c)  
$ git add test1.c  
$ git commit  
$ git rebase --continue
```

reflog

git reflog show [branch]

Show entries from the *reflog*, which tracks changes to local refs (branches). If *branch* is omitted, shows the reflog for **HEAD**.

The reflog tracks the “history of history.” Although commands like **git reset** can be used to undo a commit, the old commit still stays around in the reflog until it eventually expires. This allows you to undo many Git operations you might have performed by accident.

You can refer to entries in the reflog using `branch@{n}` notation. For example, `HEAD@{5}` means to use `HEAD` as it existed five changes ago.

Examples

To list all the recent changes to `HEAD`:

```
$ git reflog
```

To undo the most recent merge operation:

```
$ git reset HEAD@{1}
```

remote

```
git remote add [-f] name repository-url  
git remote rm name  
git remote show name  
git remote prune [--dry-run] name  
git remote update [name ...]
```

Manipulate remotes, which act as short forms for tracking repository URLs and branches. This command takes one of several forms:

git remote add name repository-url

Register a new remote `name` at `repository-url`. With `-f`, also `git fetch` the set of remote branches from the new remote.

git remote rm name

Unregister the remote `name`.

git remote show name

Show information about the given remote `name`.

git remote prune name

Delete remote tracking branches that no longer exist on the remote `name`. To prevent accidental data loss, remote tracking branches in the local repository are never deleted unless you run this command.

git remote update [name ...]

Equivalent to running `git fetch name` for each of the names individually. If no `names` are provided, fetches all the registered remotes.

Examples

To show information about the remote named `origin` (which is created automatically by `git clone`):

```
$ git remote show origin  
* remote origin  
  URL: git://git.kernel.org/pub/scm/linux/kernel/git/  
    torvalds/linux-2.6.git  
  HEAD branch: master  
  Remote branches:  
    master      tracked  
    production  tracked  
  Local branches configured for 'git pull':  
    master      merges with remote master  
    production merges with remote production
```



```
Local refs configured for 'git push':  
  master      pushes to master      (up to date)  
  production  pushes to production (local out of date)
```

To replace *origin* with a pointer to a new server and fetch the branches on the new server:

```
$ git remote rm origin  
$ git remote add -f origin myserver:/git/myproj.git  
Updating origin  
From myserver:/git/myproj.git  
 * [new branch]    master      -> origin/master  
 * [new branch]    production -> origin/production
```

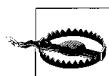
To update all registered remotes:

```
$ git remote update  
Updating origin
```

reset

```
git reset [--soft|--hard] [commit] [-- path ...]
```

If no *paths* are given, update **HEAD** to point at the given *commit*. This can be used to undo one or more *commits* as if they had never happened. If *paths* are given, **HEAD** is not changed; only the listed files and directories are modified.



git reset is extremely dangerous and can cause you to lose track of portions of your repository's history. Use it only on branches that have never been shared with anyone else. If you make a mistake with **git reset**, you can usually undo it using **git reflog**.

With **--soft**, neither the index nor the work tree are modified, and only the **HEAD** pointer is changed. Thus, the difference between **HEAD** and the index is the set of changes needed to convert the new **HEAD** into the original **HEAD**. If you then run **git commit**, a new commit will be created that changes the files back to the way they were before you ran **git reset**.

With **--hard**, both the index and the work tree are modified to match the given *commit*; any changes from the given *commit* to **HEAD** are lost.

If neither **--soft** nor **--hard** is specified, then the index is updated, but not the work tree.

Examples

To undo the most recent commit without losing the changes to the files themselves (so they can be committed again):

```
$ git reset HEAD^
```

To change the current branch to point at the same place as *origin/master*, destroying anything else that might have happened in the current branch previously:

```
$ git reset --hard origin/master
```

To undo the above `git reset` operation by recovering the previous `HEAD` from the reflog:

```
$ git reset --hard HEAD@{1}
```

To undo the most recent five commits and recommit their changes as a single commit:

```
$ git reset --soft HEAD~5  
$ git commit
```

revert

`git revert [-n] commit`

Create a new commit that reverses the effect of *commit*.

Options

- n, --no-commit
- s, --signoff

Examples

To undo the most recent commit:

```
$ git revert HEAD  
[master ae3f932] Revert "Say hello."  
 1 files changed, 0 insertions(+), 50 deletions(-)
```

To undo commit *ae3f932*, even if it is not the most recent commit:

```
$ git revert ae3f932  
[master ae3f932] Revert "Say hello."  
 1 files changed, 0 insertions(+), 50 deletions(-)
```

rm

`git rm [-f] [-r] -- path ...`

Remove files from the work tree and index. They will be permanently removed when you run `git commit`.

With `-f`, forces removal of a file even if it doesn't match `HEAD`.

With `-r`, removes entire directories and all their contained files.



Options

- f (force)
- r (recursive)
- cached (ignore work tree)
- n, --dry-run

Examples

To remove the *mylib* directory:

```
$ git rm -r mylib  
rm 'mylib/test1.c'  
rm 'mylib/test2.c'
```

stash	<code>git stash</code> <code>git stash list</code> <code>git stash show [stashid]</code> <code>git stash apply [stashid]</code> Save, list, or reapply the set of uncommitted changes from a work tree and index.
	This command takes one of four forms:
git stash	Save the current set of uncommitted changes and undo them. The index and work tree are reset to match HEAD .
git stash list	Show the list of all stashes that have previously been saved.
git stash show [stashid]	Show the exact set of changes that are saved as <i>stashid</i> . If <i>stashid</i> is omitted, uses the most recently saved changes.
git stash apply [stashid]	Brings back the changes from the given <i>stashid</i> and applies them to the current index and work tree. If <i>stashid</i> is omitted, uses the most recently saved changes.

Examples

To save the current set of uncommitted changes, switch branches, and apply those changes to the new branch:

```
$ git stash  
Saved working directory and index state "WIP on master:  
44951b7... Say hello"  
(To restore them type "git stash apply")  
HEAD is now at 44951b7 Say hello
```

```
$ git checkout feature1  
Switched to a new branch "feature1"
```

```
$ git stash apply  
Removed test1.c
```

status	<code>git status [path ...]</code>
	Check what would happen if you ran git commit . If <i>paths</i> are specified, check what would happen if you ran git commit paths .

Examples

To check the status of the current branch:

```
$ git status  
# On branch master  
# Changed but not updated:  
#   (use "git add <file>..." to update what will be  
#    committed)  
#  
#       modified:   test1.c
```

```
#      modified: test2.c
#
no changes added to commit (use "git add" and/or "git
commit -a")
```

To see what would happen if you ran **git commit test1.c**:

```
$ git status test1.c
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#      modified: test1.c
#
# Changed but not updated:
#   (use "git add <file>..." to update what will be
committed)
#
#      modified: test2.c
#
```

tag

```
git tag [-a|-s|-u gpg-key-id] [-f] [-m msg | -F msg-file]
tagname commit
git tag -d tagname
git tag -l [glob-pattern]
git tag -v tagname ...
```

Manipulate tags. A tag is simply a user-friendly name for a particular commit. The command takes one of four forms:

git tag [-a|-s|-u gpg-key-id] [-m msg | -F msg-file] tagname commit
Create a new tag named *tagname* based on the given *commit*. With **-a**, the tag is annotated (i.e., it has a commit message attached) but not signed. With **-s** or **-u**, the tag is annotated and signed with the default or specified gpg private key, respectively. With **-m** or **-F**, use the given commit message or filename containing the commit message, respectively. If none of **-a**, **-s**, or **-u** is specified, the tag has no annotation. (Tags you shared with other people should always have an annotation and should usually be gpg signed.)

git tag -d tagname

Delete the given *tagname* from the local repository. Note that if the tag has already been pushed to a remote repository, there is no way to make sure everyone erases it.

git tag -l [glob-pattern]

List all the tags matching the *glob-pattern*. If *glob-pattern* is omitted, lists all the tags.

git tag -v tagname ...

Verifies the gpg signature of each requested tag.

A small black square containing the white text "git".

Examples

To mark the current version with a *v1.1* tag and sign it with your gpg key:

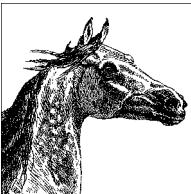
```
$ git tag -s v1.1 HEAD
```

To delete the *v1.1* tag (only useful if it has never been pushed):

```
$ git tag -d v1.1
```

To list all tags for version *1.x*:

```
$ git tag -l 'v1.*'
```



15

Virtualization Command-Line Tools

This chapter covers common tools used with Linux to run many virtual servers on a single physical server. We cover the two dominant hypervisors used with Linux: Xen and KVM. We also cover libvirt-based tools used to manage both Xen and KVM. Although it isn't strictly a Linux solution, VMware uses a Linux operating system for the Service Console of their ESX server and recent vSphere products, so we have included basic console commands they provide.

This chapter covers the following topics:

- Conceptual overview
- Basic virtualization operations
- The Xen Hypervisor
- The KVM Hypervisor
- The libvirt virtualization API
- VMware command line interface

We do not provide a tutorial or in-depth information on virtualization here. This is meant as a quick reference to virtualization concepts, commands, and utilities. Refer to the following locations for more in-depth coverage on each of these technologies.

The Xen hypervisor <http://www.xen.org/>

The KVM hypervisor <http://www.linux-kvm.org/>

libvirt virtualization API <http://libvirt.org/>

VMware Documentation <http://www.vmware.com/support/pubs/>

Conceptual Overview

All of the virtualization programs we cover in this chapter provide full system virtualization. They use a software layer to present a virtualized hardware system. This layer, called the virtual machine monitor (VMM) or hypervisor, makes it possible to run multiple operating systems simultaneously. The operating system at the base—running directly on the hardware and interacting with it directly—is called the *host*, and the software instances that run on top of it—interacting only indirectly with hardware—are called the *guests*.

Virtualization isn't a new topic. As a means to divide a computer's resources into multiple running environments, virtualization has been around since the 60s. Back then, IBM used it to partition mainframe computers into separate virtual machines, allowing it to run multiple applications and processes simultaneously. Although low-cost PC hardware generally replaced mainframe hardware virtualization in the 80s and 90s, virtual machines were still commonly used to provide a virtualized environment for a process: e.g., creating a security sandbox or providing greater portability. The Java Virtual Machine (JVM) is a good example of process virtualization. This kind of virtualization also includes tools like `chroot` and `virtfs`.

Another popular type of virtualization creates a new instance of the same operating system on top of the host. This technology is often called *containers*. It typically includes such projects as Solaris Zones and User Mode Linux (UML). We don't cover containers in this chapter.

Here, we cover the tools of industrial-strength server virtualization. The solutions in this chapter offer environments that look like complete, independent hardware computers even though they are just software instances running on a common hardware base. While hardware costs for PCs remain fairly low, systems with multiple CPUs or CPU cores can easily handle the task of running dozens of highly available virtualized servers simultaneously. Businesses use virtualization to reduce deployment and administration costs. Many businesses also use virtualized-desktop systems to replace physical desktops, simplifying management tasks throughout the company.

Configuring virtualization requires administrative access to systems, CPUs with built-in virtualization technology, or even modified operating systems. The tools covered here are intended for large-scale server deployments. Although these tools might allow you to run an alternative operating system on your desktop or notebook, if you are just looking to run a single guest you will find that easier to do with another kind of tool. VMware Player or VirtualBox, for example, both run in user space, require no kernel modifications or special CPU, and are available for free.

System Requirements

To run the tools covered in this chapter you should have at minimum:

- A processor that supports virtualization technology (Intel VT or AMD-V). Support should be enabled in the system BIOS.
- At least 2 GB of memory.
- At least 6 GB of disk space.

Xen can run paravirtualized guests without virtualization technology in the CPU, but it requires it for full virtualization. You can check whether your existing CPU supports virtualization by grepping `/proc/cpuinfo` as follows:

```
$ egrep '(vmx|svm)' /proc/cpuinfo
```

If nothing shows up, your CPU does not support virtualization. If something does appear, you may still need to enable it in BIOS to use it.

Each solution has additional requirements or limitations. Before making a hardware purchase you should check the requirements and supported hardware for the solution you plan to use.

If you want to support migration (moving a running guest from one host to another) you will also want some kind of shared-storage solution. That might be an iSCSI or SAN volume to which all servers have access. You don't need that to get started though.

Virtualization Technology

In full virtualization, the guest operating system doesn't need to be aware that it is running in a virtual environment. It interacts with the virtual hardware in the same way it would physical hardware. Some tasks in a fully virtualized environment, however, are computationally expensive. One way to handle the expensive tasks is to use a modified processor that performs these tasks for the hypervisor.

Paravirtualization provides another way to handle these expensive tasks. In paravirtualization the guest operating system is modified to hand these expensive tasks to the hypervisor, which can handle the task without having to emulate hardware.

Network Concepts

For your virtual server to connect to the Internet it must have a way to use the physical connection of the host. Xen, KVM, and VMware will all create a default bridge interface to handle mapping the virtual systems to the physical interface. A virtual machine could also be assigned directly to a physical device, but for most situations a bridge is what you want. More complex network configurations can be made by defining a kind of virtual switch or router. We will cover some commands that can do this here.

libvirt Tools and Terminology

Xen, KVM, and other Linux-based hypervisors support a single generic API called **libvirt**. For most tasks, it doesn't matter which Linux hypervisor you are using. You can use the same basic tools. libvirt has its own terminology which will be helpful to know when reading the libvirt section.

- A **node** is a single physical machine.
- A **hypervisor** is a layer of software that virtualizes a node, providing a set of virtual machines that may differ from the node itself.
- A **domain** is an instance of an operating system running on a virtualized machine provided by the hypervisor.

Throughout this chapter, though, we'll stick to more common industry terminology, referring to a **domain** as a **guest** and the **node** running the guest as a **host**.

Basic Virtualization Operations

The complexity of virtualization prevents us from covering everything in a small chapter within this book. Fortunately, developers have written tools that handle much of the complexity for you. They have worked hard to make the defaults do the right thing for your system and have written scripts to simplify the tasks. Here we cover the higher-order tools that will help you create and manage virtual systems and networks.

Creating Virtual Systems

For VMWare you will use the VI Client GUI interface to create your virtual systems. We won't cover that task here. With Xen and KVM you can use virt-manager's **virt-install** to install a new guest. At minimum you must tell **virt-install**:

- the name of the new guest domain
- whether the guest will be paravirtualized or fully virtualized (**-p** or **-v**)
- the amount of RAM to give to the new system (**-r**)
- what to use as a system disk and if creating a disk, how large it should be (**-f** and **-s**)
- whether to support graphics or not (**--vnc** or **--nographics**)
- where to find the installation files (**-c**, **-l**, or **--pxe**)

virt-install will prompt for any values you do not provide to it on the command line. The name should be a short unique identifier; whatever helps you distinguish it from other virtual machines controlled by the same hypervisor. Paravirtualization or full virtualization depends on what your OS supports. If you do not have an installation already prepared for virtualization, you need full virtualization.

The system disk can be stored anywhere in your filesystem. Use a full path to specify where, or just a filename to have it stored in the default location for your hypervisor.

You will need a way to connect to the new system. See the section “Graphic and Console Interfaces” on page 842. If you are installing a fully virtualized system, use the **--vnc** option. If you aren't installing on a system running X Window System, you may need to connect to your guest remotely from a system that does. If you are installing a paravirtualized system and the guest system supports it, you can use either **--vnc** or **--nographics**. If using **--nographics** fails, you may need to use **virsh** to **destroy** the domain and start over as you will have no way to connect to it.

What you use for installation files will depend on what kind of virtualization you are using. For a fully virtualized system you may want to install from a CDROM drive or an installation disk ISO. You can use the **-c** option to point to a physical device on the host (*/dev/cdrom*) or to an ISO file.

To install a paravirtualized Linux or UNIX system, you will instead use the **-l** option with **virt-install** to point to the location of the initial boot image. For some distributions, you can point this to an ISO image, but the **-l** option does not configure a virtual CDROM, so after the boot you will need to tell the install program where to find a mirror of the installation media. You can use HTTP, NFS, or FTP to connect to that media. This means you will have already had to have set up a server to provide these files. You can quickly prepare an installation ISO for this by mounting it via the loop back device and either exporting the directory via NFS or serving the files through HTTP. We'll provide examples.

virt-install has many other options that can help you customize your virtual server. If you are using KVM, you might want to use the **--accelerate** option to better take advantage of the hypervisor. You might want to use **--os-type** option to optimize guest configuration. You can provide more detailed information on how to configure the network interface (without any options, it just connects the guest to the default bridge interface). See **virt-install**'s entry in this chapter for further information.

Some distributions come with wrapper-scripts that call **virt-install** with default options to install a basic Linux operating system. Ubuntu, for example, provides a **vm-builder** script that will install a stripped-down version of Ubuntu, ready to run.

Examples

Mounting a Red Hat ISO for export via NFS:

```
# mkdir /mount/rhel5.3  
# mount -o ro,loop RHEL-5.3.iso /mount/jeos-8  
# exportfs *:/mount/rhel5.3/
```

Mounting a Red Hat ISO for HTTP access:

```
# mkdir /var/www/html/rhel5.3  
# mount -o ro,loop RHEL-5.3.iso /var/www/html/rhel5.3
```

Full virtualization install from an installation iso:

```
# virt-install -name rhel-fv -v -r 256 -f /images/rhel-fv.img -s 8 -vnc \  
-c RHEL-5.3.iso
```

Paravirtual installation via HTTP using a serial console:

```
# virt-install -name rhel-pv -p -r 256 -f /images/rhel-pv.img -s 6 \  
--graphics -l http://example.com/rhel-5.3/
```

Managing Virtual Systems

While hypervisors can control the starting and stopping of virtual machines, they can't always shut them down gracefully. You may want to power off your systems by connecting to each system and initiating a shutdown in the normal way. Sometimes you can't do that, though, and you need to do the virtual equivalent of pulling the plug on a system that isn't responding. The commands listed here cover these basic operations.

Xen and KVM virtual guests can be managed with `virsh` commands. Use `virsh list --all` to get a list of all managed domains. The `start`, `shutdown`, `reboot`, and `destroy` commands are your virtual power and reset buttons. From the service console in Vmware's ESX server you can do similar things with `vmware-cmd`.

Graphic and Console Interfaces

You need some way to interact with the virtual systems. Generally, you will use a VNC or SDL graphic interface set up by your guest installation tool. VMware provides its own client software to handle this. For Xen and KVM you will likely use the VNC client `virt-viewer` or the GUI management tool `virt-manager` to connect to a virtual system. As an alternative, you could interact with a virtual system through a text-based console connected to a virtual system's serial device. The `virt-install` command has a `--nographics` option that will attempt to set this console up for you, but it works only with paravirtualized guests. If your guest system has a properly functioning network connection, you may also be able to use `ssh` to connect to it.

Configuring Networks

Your basic network options are to use bridging or Network Address Translation (NAT). In bridging, a special bridge interface transparently connects your virtual interfaces to your host's physical interface(s). Your virtual servers will share the physical interface.

Using NAT, your virtual servers are assigned private network IP addresses, and another interface then provides the NAT for your system. This NAT interface is essentially a bridge as well. A bridge is really any interface used to hook your virtual interfaces up to your physical interfaces. Still, these two approaches are commonly referred to as bridging and NAT.

VMware uses a third option. It creates a virtual switch to which you connect your guests' interfaces. The default switch, however, simply bridges the switch to the server's network interface card. Neither Xen or KVM come with a ready-made virtual switch solution, although some vendors may offer virtual switch solutions you could add to your system.

Upon installation of the hypervisor and libvirt, most distributions will automatically configure basic networking support. They will likely configure both a NAT and bridge interface. Xen's tools will use the bridge interface by default, but `virt-manager`'s tools will likely use NAT.

The approach you want to use will depend on the kind of system on which you have installed the host operating system. If you are installing on a system that might frequently change networks (for example, a notebook you move between home and work) or want to keep your virtual systems separate from your physical network, you will probably want NAT. If you are using a wireless card, you must use NAT. Bridging will not work with wireless interfaces because wireless chipsets reject foreign MAC addresses. If you are installing on a server and want your virtual systems to interact with your network the way any physical system would, you want to use bridging.

While we don't cover it in this chapter, the Net:Bridge or (bridge-utils) package contains the **brctl** command. This command can help you build and manage more complex bridges. You can find more information on this package at:

<http://www.linuxfoundation.org/en/Net:Bridge>

We highly recommend you read that page and explore the tool if you want to dig any deeper into configuring your bridges.

The basic concept to keep in mind for virtual networking is that your host systems will have interfaces for physical devices, virtual devices, and bridge devices. What they are named by default depends on your distribution and hypervisor. The following are the interfaces you might find running if you used **ifconfig** on a CentOS system running Xen.

vifX.Y

A vif interface on the host connects to the virtual interface on a guest domain. Consider it the server's representation of that virtual Network Interface Card (NIC). vif0.0 is Domain-0's first interface. vif0.1 would be its second. vif1.0 is the first interface on the first running guest (Domain 1).

eth0

The virtual interface for Domain-0, which is connected to vif0.0.

peth0

This device performs packet distribution for guest systems, including Domain-0. It is bound to the bridge device, the physical network card. It has no IP address of its own, but instead acts as a simple switch. This is how packets get to the appropriate vif device.

xenbr0

Xen's default bridge device that connects vif interfaces to the physical interface (peth0).

virbr0

An interface providing NAT. This is the default bridge device for virt-manager tools. You may need to enable packet forwarding in Domain-0 to make this device work.

You will likely find these devices on an Ubuntu system running KVM:

eth0

The physical interface for the host. Note that it doesn't have an IP address. The host's interface is bridged to this device.

vnet0

An interface providing NAT. The default bridge device for virt-manager.

vnetX

Interfaces connected to a guest's virtual interface, where X is numbered 1 or higher. These are bridged through vnet0 by default.

br0

The host system's bridge device. This one has the host's IP address and connects to the physical interface, eth0.

You can set the interface to be used by a virtual domain when using **virt-install** with the **--network** option.

MAC Addresses

You may want to set a MAC address on a virtual interface manually. This is a special hexadecimal number that uniquely identifies your interface card. If not set, your configuration tool will generate a random number for it, but knowing what it will be ahead of time can help you configure DHCP or a PXE boot. With **virt-install** you can set it with the **--mac** option. VMware's VI Client provides an option as well.

There are some restrictions. The first three sets of numbers in a MAC address identify the vendor. When setting them, use the following vendor addresses:

Xen	00:16:3e
KVM/Qemu	54:52:00
VMware	12:50:56 AM

Making Changes to Virtual Machines

VMware's VI Client can be used to edit virtual machine configurations. Making changes to a virtual machine under Xen or KVM can be a little trickier. You can make some changes (such as memory settings) using **xm** or **virsh**. For more complex changes with Xen, you could change the configuration files located under **/etc/xen/**. These are in a text format, and most of the settings you will want to change will be easy to figure out.

virsh provides one common way to make complex changes:

1. Run **virsh dumpxml** to get an XML configuration file for the domain you wish to change.
2. Use a text editor to edit the XML file.
3. Shut down the virtual system if it is still running.
4. Run **virsh define** to remove the old configuration from the hypervisor and replace it with the configuration in your edited XML file.

You can now start the domain with your new configuration.

Creating and Manipulating Disk Image Files

Tools such as **virt-install** will create disks as needed when installing a new virtual server, but you might want to add a disk to an existing server. One common form of disk image for virtual machines is a sparse file. The simplest tool for creating this is the **dd** command documented in Chapter 3. For instance, to create a 10,000 MB sparse file suitable for use as a disk image you can use the following command:

```
# dd if=/dev/zero of=newdisk.img bs=1M seek=10000 count=1
```

Change the system's configuration file to add the new drive. You can use the guest's system tools (such as **fdisk**) to configure the new disk.

You can easily increase the size of a sparse file by using **dd** to create a new sparse file then append the existing file with the new file. Don't try this with an image file

currently in use by a running guest, though: shut down the guest system first. Here we use **dd** and **cat** to add a 5000 MB extension to the original image.

```
# dd if=/dev/zero of=diskextension.img bs=1M seek=5000 count=1  
# cat diskextension.img >> newdisk.img
```

You will need to use a tool appropriate to the guest operating system to take advantage of the new space. For example, **resize2fs** makes the space available to an ext3 filesystem.

For more advanced features on a disk image, you can use **qemu-img**, a tool we don't document here. It can create and convert disk images to a several different formats, including to VMware's **.vmdk** format. If you are using KVM you should already have it installed on your system. If you do not, you can install it via the **qemu** package.

Xen

Xen is the most common hypervisor used on Red Hat Advanced Server 5, Fedora, and distributions based on these, such as CentOS 5. Assuming you are using one of these, to use Xen you must install the **xen** and **kernel-xen** packages from your package repository:

```
# yum install xen kernel-xen
```

Other operating systems will probably have similar packages. Some offer a **groupinstall** that will include all the necessary packages:

```
# yum groupinstall Virtualization
```

Once installed, reboot your system using the Xen kernel. You may want to modify your **/boot/grub/grub.conf** file to boot the Xen kernel by default. Usually this means changing the **default=** entry to **default=0**, where 0 refers to the first boot configuration described by the file. See Chapter 4 for details on configuring **grub** and modifying the **grub.conf** file.

The installation of the previously mentioned packages should also configure two services to run: the Xen daemon (**xend**) and the **xendomains** script that the system uses to automatically start and stop your guest domains.

Once the new kernel and **xend** service is running, you should be able to run the command

```
# xm list
```

to list your current nodes. At first you will have one domain running, Domain-0.

To understand Domain-0, it helps to know that your system is not really running Linux directly. It's running Xen's own kernel, a virtual machine monitor based on the Nemesis microkernel. Xen's kernel doesn't provide an administrative interface; it depends on a modified operating system running as a guest to do this. This privileged guest also provides guest domains access to hardware devices. In our case, Domain-0 is running a modified version of Linux. Your OS is actually the controlling domain in a cluster of guest domains.

In Xen, these other guest domains are referred to as unprivileged domains, or DomU (the U stands for unprivileged). Domain-0 can also delegate control of hardware devices to other privileged domains, but that's a more advanced topic we won't cover here.

Paravirtualization and Architecture

Xen supports paravirtualized guests that run on the same architecture as Domain-0. Thus, if Domain-0 is running X86_64, paravirtualized guests must be X86_64 as well. To run a 32-bit guest on a 64-bit system, you must run it fully virtualized.

Like Domain-0, paravirtualized Linux guests run the Xen kernel instead of the regular kernel.

Xen Networking

Mixing libvirt and Xm tools can lead to some confusion when it comes to networking. Guest domains installed using Xen's tools bridge to device **xenbr0** by default. Those installed by libvirt's **virt-install** bridge to device **virbr0**. To cut down on confusion and take full advantage of libvirt, you may want to disable Xen's bridge networking altogether. Details on how to configure networking for Xen using libvirt can be found on the libvirt wiki: <http://wiki.libvirt.org/page/Networking>. Alternatively you can tell **virt-install** to bridge to device **xenbr0** using the **-w** option (**-w bridge:xenbr0**).

Xen Commands

Xen has its own special commands for installing and managing domains, but for most things you can use the **libvirt** tools documented in this chapter. Here, we will cover the most commonly used **xm** subcommands and the **xentop** command.

xm	<code>[options] command</code>
	Xen hypervisor management interface. Generally you will use virsh to control VMs, but there are a few useful functions you can get from xm but not virsh . We cover the most common commands here.
	xm can be used to configure virtual machine access rights and security policies. Policies are given labels and the labels applied to domains. This is an advanced topic we do not cover in this book.

Commands

console domain

Connect to serial console on domain, if available. Use **ctrl-**] to exit a console.

create [-c] configfile [settings]

Start the guest domain described in configfile. This command is generally used to start a previously installed domain that is not currently running. By default, a domain will have a configuration file of the same name in **/etc/xen/**. You can specify the configfile with a full path, or a path relative to **/etc/xen/**. The configuration file, a Python executable file, largely consists

of *name=value* pairs. These can also be given in a space-separated list on the command line. The manpage for **xmdomain.cfg** contains details on configuration file format and valid entries. The **-c** option causes **xm** to connect to the console as soon as the machine starts.

destroy *domain*

Immediately kill any running instance of the domain and free its resources to the hypervisor. (You usually want to use **shutdown** instead.)

dmesg [**-c**]

Print the Xen message buffer. This buffer contains informational, warning, and error messages. The **-c** option clears the message buffer.

help [**--long**]

Print a list of **xm** commands. If **--long** is given, print all commands grouped by function.

info

Print information about host system.

list [*options*] [*domains*]

List information about one or more *domains* (a space-separated list of domain names, ids or UUIDs). This is a more verbose listing than that provided by **libvirt**. It includes resource allocation and running time information.

Options

-l, --long

Print detailed information about domains in a format that can easily be consumed by external programs.

-label

Include security label information.

log

Print the contents of the **xend** log. This log is also found at */var/log/xen/xend.log*.

top

Run the **xentop** command (described next).

list [*options*] [*domains*]

List information about one or more *domains* (a space-separated list of domain names, ids or UUIDs). This is a more verbose listing than that provided by **libvirt**. It includes resource allocation and running time information as well as state. There are six possible states: running, blocked, paused, shutdown, crashed, and dying.

Options

-l, --long

Print detailed information about domains in a format that can easily be consumed by external programs.

-label

Include security label information.

pause *domain*

Cease hypervisor scheduling for *domain*.

shutdown [*options*] *domain*

Begin a graceful shutdown on *domain*. For guests that support this, it is the same as running **shutdown** on the guest. It may not succeed. By default, the command does not wait for the shutdown to complete. **xm list** will reveal the guest's current state.

Options

-a Print detailed information about domains in a format that can easily be consumed by external programs.

-w Wait for domain to complete shutdown before exiting.

unpause *domain*

Resume hypervisor scheduling for *domain*.

xentop**xentop** [*options*]

Provide frequently updated information about current domains. This program performs a function similar to the Linux **top** command, only with information about the hypervisor and domains instead of processes. Its various options change what information is displayed. Like **top**, **xentop** also has some interactive commands to change what information **xentop** displays as it runs.

Options**-b, --batch**

Run in batch mode; don't accept command-line input. Useful for sending output to another command or to a file.

-d *seconds*, **--delay**=*seconds*

Set the delay between refreshes. Default is 3.

-i *num*, **--iterations**=*num*

Update display *num* times, then exit.

-n, --networks

Show network information.

-r, --repeat-header

Repeat the table header before each domain.

-v, --vcpus

Show virtual CPU information.

-x, --vbds

Show virtual block device information.

Interactive commands

All interactive commands are case-insensitive.

B Toggle display of virtual block device information.

D Prompt for a new delay setting.

N Toggle display of network information.

Q, Esc

Quit.

R Toggle display of table header before each domain.

S Change sort order by cycling through the available columns.

V Toggle display of virtual CPU information.

Up, Down

Scroll through displayed domains.

KVM

The Kernel-based Virtual Machine (KVM) is a full virtualization hypervisor for Linux. The work of the KVM hypervisor is handled by the Linux kernel. Each guest in KVM runs as a process and can be managed by Linux tools such as **top** and **kill**.

KVM isn't a complete virtualization solution. It depends on both the libvirt tools for management and the open source processor emulator QEMU for hardware emulation. Therefore, you will need those installed as well.

KVM is the most popular hypervisor on Ubuntu distributions. Assuming you are using Ubuntu, you can use **apt-get** to install the necessary packages:

```
$ sudo apt-get install kvm libvirt-bin ubuntu-vm-builder qemu bridge-utils
```

You may also want the **virt-viewer** and **virt-manager** packages, though these will require the X Window System as well. As of Ubuntu version 8.10 (Intrepid) you can install the meta package **ubuntu-virt-server** for the basic tools mentioned previously and **ubuntu-virt-mgmt** for the GUI management tools.

You will likely want to provide your controlling user account with access to network devices created by libvirt. Use **usermod** to add your account to the **libvirdt** user group.

Restart your system. If all goes well, you should be able to run the following command without error:

```
$ virsh -c qemu:///system list
```

Also check to make sure KVM is working:

```
sudo kvm
```

This will confirm that you have enabled virtualization support in BIOS properly.

QEMU

QEMU is really its own virtualization solution that works in user space. Combining it with KVM speeds it up considerably, providing a robust hypervisor running at the kernel level. QEMU comes with some of its own commands for the installation and launching of virtual systems. Some older tutorials show how to use them. We won't cover those here; instead use libvirt tools to handle these tasks.

Ubuntu Builder Scripts

Ubuntu comes with tools to help you quickly build guest systems. In older systems these tools are in the **ubuntu-vm-builder** package; in newer systems, they are in the **python-vm-builder** package. These tools handle the defaults for installing a simple Ubuntu guest. In most cases all you need to provide is a name for your new domain.

libvirt and Red Hat Virtual Machine Manager

The libvirt virtualization API provides an open source programming interface to hypervisors like Xen and KVM. It comes with **virsh**, a management interface for controlling hypervisors. A closely related project, the Red Hat Virtual Machine Manager application, is a collection of tools built using libvirt. This includes a few command-line tools as well as the GUI **virt-manager** application.

Whether you are using Xen or KVM, these **libvirt** based tools will handle most of the tasks of creating and managing your guests. Except for some minor differences in what each hypervisor supports and the connection string used to get to the hypervisor, these commands work the same regardless of what hypervisor you choose. You'll want to know these tools well.

XML Configuration Files

libvirt uses XML files to define or describe the capabilities and configuration of domains, networks, and hardware devices. Many of the command tools expect to receive information from an XML file. You don't have to know the XML tags to get started, as the tools will build the required files for you, but you may want to edit an XML file directly to change the configuration of a guest.

The XML formats, as well as more in-depth information on **virsh** and the API, can be found on the **libvirt** website: <http://libvirt.org>.

Connection URIs

To control a hypervisor, **libvirt** tools must first connect to it. You will use a URI to specify the location of the hypervisor. For KVM use a **qemu://** URI and for Xen use **xen://**. For backwards compatibility **virsh** will also accept just plain **xen**, which it will treat as **xen://**. If no connection option is given, **virsh** will try to connect to the URI given in the environment variable **LIBVIRT_DEFAULT_URI**. If none exists, it will probe the defaults in whatever hypervisor drivers it has available.

You can also make a connection to a remote server. For security reasons you will usually use SSH to do this by using an **xen+ssh://** or **qemu+ssh://** URI.

Connection URI Examples

Connect to the local Xen hypervisor:

```
xen:///
```

Use SSH to connect to the Xen hypervisor running on *xenhost.yoyodyne.com*:

```
xen+ssh://xenhost.yoyodyne.com/
```

Connect as root to the local KVM hypervisor:

```
qemu:///system
```

Connect as an unprivileged user to the local KVM hypervisor:

```
qemu:///session
```

Use SSH to connect as user john to the KVM hypervisor on *kvmhost.yoyodyne.com*.

```
ssh+qemu://john@kvmhost.yoyodyne.com/system
```

Remote GUI control

Installing the X Window System on your server nodes will consume resources you might rather give to a virtual machine, as well as increase the security risk to your servers. Instead, consider creating a server to control your nodes. That system can be a desktop system with a full X Window installation. Install the libvirt tools and **virt-manager** to that system and use a URI connection string to connect to your client. Using SSH will keep the connection secure. Use the **VIRSH_DEFAULT_CONNECT_URI** environment variable to hold the connection string. **virsh** will look there before trying the default connection URIs.

You may have problems connecting with **virt-manager** over SSH. If you launch **virt-manager** as an unprivileged user, it will usually try to switch you to the root user. This will clear environment settings like your **ssh-agent** settings. If prompted for the root password, select the “run unprivileged” option. This will preserve your current SSH settings. You don’t need to run as root because you aren’t controlling a hypervisor on the local system. For privileged access you will connect to a privileged user on the remote node.

At this time, **virt-manager** does not support creating new guests on remote systems. But you can manage your system just fine.

IP Forwarding and libvirt Networking

libvirt’s default network configures an isolated bridge device to be used by guest domains. This default bridge creates a private network for the virtual machines, but does not connect that private network to your physical network. The simplest way to complete that connection is to enable IP Forwarding in the kernel. You can quickly enable IP Forwarding using **sysctl** like so:

```
# sysctl -w net.ipv4.ip_forward=1
```

To set your system to enable IP forwarding upon bootup, edit the **net.ipv4.ip_forward** setting in */etc/sysctl.conf*.

While IP forwarding can get you running quickly, for production environments you probably want to configure a more robust bridge network. Information on creating bridge networks for libvirt can be found on the libvirt wiki at:

<http://wiki.libvirt.org/page/Networking>

libvirt and Virtual Machine Manager Commands

virsh

`virsh [options] [command [command-options]]`

libvirt management interface. **virsh** uses the **libvirt** API to connect to a hypervisor and manipulate the configuration and state of virtual machines controlled by that hypervisor. If invoked with a command, it will execute the command and then exit. If invoked without a command, **virsh** enters a shell mode from which you can execute commands.

This manager will work with any hypervisor that has libvirt support. However, not all **virsh** commands are supported by all hypervisors. Nor will they work on all guests. Still, if a hypervisor can do something, you can generally use **virsh** to do it. Some hypervisors, including Xen, come with their own tools for managing virtual machines, but we still recommend you use **virsh** instead.

Most **virsh** commands expect a *domain* option. A domain refers to a virtual machine. You can use the virtual machine's name or UUID. If the machine is running, it should also have an ID number within the hypervisor, and you can use this for the *domain* option as well. Use the **list** command to see names and IDs of domains managed by the hypervisor.

Options

-c uri, --connect=uri

Connect to a hypervisor specified by *uri*.

-d level, --debug=level

Set the level of debugging information to be printed to standard output. Accepted values are 0–5. 0 disables debugging messages and is the default. 5 prints all debugging messages.

-h, --help

Print a brief description of options and commands.

-l file, --log=file

Log debugging information and errors to the specified *file*.

-q, --quiet

Quiet mode. Do not print informational messages to standard out.

-r, --readonly

Connect to hypervisor in read-only mode.

-t, --timing

Print timing information.

-v, --version

Print version information, and exit.

Commands

attach-device domain xmlfile

Attach device defined in *xmlfile* to *domain*.

attach-disk *domain source target [options]*

Attach disk device to *domain*. *source* and *target* are paths for the files and devices.

Options**--driver** *name*

Set the driver *name* attribute. Valid values include **file**, **tap** (network tap), or **phy** (physical).

--subdriver *type*

Set the driver *type* attribute. The valid values depend on the driver. It's common to use **aio** with the **tap** driver.

--mode *mode*

Set the *mode* of device. Valid values include **readonly** and **shareable**.

--type *type*

Set the *type* of device. Valid values include **cdrom** and **floppy**.

attach-interface *domain type source [options]*

Attach interface device to *domain*. Usually *type* will be either **network** or **bridge**. The source is the host's network or bridge device to which the virtual network should connect, or default, to use the default gateway.

Options**--mac** *macaddress*

Assign an address to the virtual interface.

--target *name*

Assign the target interface *name* to use in the guest.

--script *path*

Provide the *path* to a script to handle the bridge.

autostart [**--disable**] *domain*

Automatically start *domain* at system boot.

capabilities

Print hypervisor capabilities in XML format.

connect [*uri*] [**--readonly**]

Connect or reconnect to a hypervisor.

console *domain*

If a serial console is available on *domain*, connect to it. Use **ctrl-]** to exit a console.

create *xmlfile*

Start a domain described in the specified *xmlfile*.

start *domain*

Start an inactive domain.

destroy *domain*

Immediately kill any running instance of the domain and free its resources to the hypervisor. (You usually want to use **shutdown** instead.)

define *xmlfile*
Add persistent domain information described in the specified XML file. The domain is added but not started.

detach-device *domain file*
Detach device defined in *file*.

detach-disk *domain target*
Detach disk device *target* from *domain*.

detach-interface *domain type* [–*mac macaddress*]
Detach network interface from *domain*. Use *macaddress* to distinguish between multiple interfaces of the specified *type*.

domblkstat *domain device*
Print basic stats for block *device* on *domain*.

domid *domain*
Given the domain name, print the *domain*'s ID.

domifstat *domain interface*
Print basic stats for network *interface* device on *domain*.

dominfo *domain*
Print basic information on *domain*.

domuuid *domain*
Print the *domain*'s UUID.

domname *domain*
Given the domain ID, print the *domain*'s name.

domstate *domain*
Print the *domain*'s state.

dump *domain file*
Dump *domain*'s core to *file*.

dumpxml *domain*
Print the *domain*'s information in XML. Output is suitable for use with the **create** and **define** commands.

freecell [*number*]
Print available memory on the machine, or if specified, available memory in Non-Uniform Memory Access (NUMA) cell *number*.

help [*command*]
Print a list of available commands or information on a specified *command*.

hostname
Print the hypervisor's hostname.

list [*option*]
Print a list of active domains and their current state. Use option **--inactive** to list inactive domains, or **--all** to list both active and inactive domains. There are six possible states: running, blocked, paused, shutdown, crashed, and dying.

migrate [–*live*] *domain destination* [*transport*]
Migrate a *domain* to a new *destination* host. The *destination* should be given as a connection URI. If the transport method differs it can be given as a separate URI, though this isn't usually necessary. Use **--live** to migrate an active domain without interruption.

net-autostart [--disable] *network*
 Automatically start *network* at system boot.

net-create *xmlfile*
 Create and activate a network described in the specified *xmlfile*.

net-define *xmlfile*
 Define a network described in the specified *xmlfile* but don't start it.

net-destroy *network*
 Immediately stop an active *network*.

net-dumpxml *network*
 Print the *network*'s information in XML.

net-list [*option*]
 Print a list of active networks. Use option **--inactive** to list inactive networks, or **--all** to list both active and inactive networks.

net-name *uuid*
 Print the name of the network specified by *uuid*.

net-start *network*
 Start a previously defined but inactive network.

net-undefine *network*
 Remove *network*'s definition.

net-uuid *name*
 Print the UUID of the network specified by *name*.

nodeinfo
 Print basic information about the physical system.

quit
 Exit virsh's interactive terminal.

reboot *domain*
 Reboot the *domain* as if a reboot command was run from the console.

restore *file*
 Restore a domain from a domain state saved in *file* by the **save** command.

resume *domain*
 Resume execution of a suspended *domain*.

save *domain file*
 Save a *domain*'s state to *file*. This will suspend the domain, similar to hibernating a system.

schedinfo [*options*] *domain*
 Set or show CPU scheduler settings.

Options

--weight *n*
 Set scheduler weight. Valid values of *n* are between 1 and 65535.

--cap *percent*
 Set the maximum *percent* that any one physical CPU *domain* can consume.

setmem *domain size*
Change *domain*'s current memory allocation to the *size* given in kilobytes.

setmaxmem *domain size*
Change *domain*'s maximum memory allocation to the *size* given in kilobytes. The new size can't be larger than was specified when the virtual machine was created.

setvcpus *domain n*
Change the number of active virtual CPUs. *n* cannot exceed the number of CPUs specified in *domain*'s configuration file at boot time.

shutdown *domain*
Shut down the *domain* as if the shutdown command was run from the console.

suspend *domain*
Pause execution of *domain*.

ttyconsole *domain*
Print *domain*'s tty console device if it has one.

undefine *domain*
Remove *domain*'s definition.

uri
Print hypervisor's URI.

vcpuinfo *domain*
Print information about *domain*'s virtual CPU.

vcpupin *domain vcpu cpulist*
Control domain virtual CPU affinity. Assign the given *vcpu* number to the physical *cpulist* given as a comma-separated list of numbers or number ranges: e.g., 0,2 or 0–2,5,6.

version
Print library API version number and the hypervisor URI.

vncdisplay *domain*
Print the IP address and port number of *domain*'s display, if available.

virt-clone	virt-clone [<i>options</i>] Clone a virtual machine. virt-clone copies hard drives and duplicates the virtual hardware configuration of an existing virtual machine. Elements that must be unique, such as the MAC address of a virtualized network interfaces, are updated so as to avoid conflicts with the old system. This system to be cloned must be shut off, as an active system cannot be safely cloned. If no options are given, virt-clone will query for the needed information. virt-clone writes debugging information to \$HOME/.virtinst/virt-clone.log .
-------------------	---

Options**--connect=uri**The hypervisor connection *uri*.**-d, --debug**

Print debugging information to standard output.

-f path, --file=path

The path to the file, disk partition, or logical volume to use to store the new guest's virtual disk. If the original system has multiple disks, this option should be run once per disk.

--force

Run without prompting. Assume a yes response to any yes/no prompt. For all other prompts, exit.

-n name, --name=nameThe new domain *name* for the copied system.**-o domain, --original=domain**The name or UUID of the original *domain* to copy.**--preserve-data**

Preserves any data that exists on the target of an -f option.

-m address, --mac=addressThe new hardware MAC *address* for the network interface on the copied system.**-u uuid, --uuid=uuid**The new *uuid* for the copied system. This is a 32-digit hexadecimal number. If not specified, a random UUID is used.**`virt-image`**`virt-image [options] xmlfile`

Create a guest machine from the image descriptor in *xmlfile*. Attributes for the new guest are taken from the descriptor file, although some missing information can be provided on the command line. Command-line options supersede information in the image descriptor file.

Options**--boot=n**

If image has multiple boot descriptors, use descriptor *n*. The first descriptor is descriptor **0**. When omitted, the one that best fits the current hypervisor is chosen.

--check-cpu

Warn if the number of virtual CPUs for the guest exceeds the number of physical CPUs in the host system.

--connect=uriThe hypervisor connection *uri*.**-d, --debug**

Print debugging information to standard output.

-h, --help

Print information on the command and options, then exit.

-k *map*, **--keymap**=*map*
Set the *map* to use for the graphical console. By default, an English keymap is used.

-m *address*, **--mac**=*address*
The hardware MAC *address* for the network interface on the copied system. If none is specified or the address is given as **RANDOM**, assign a random address.

-n *name*, **--name**=*name*
The domain *name* for the new guest.

--noacpi
Disable Advanced Configuration and Power Interface (ACPI).

--noapic
Disable Advanced Programmable Interrupt Controller (APIC).

--nographics
Do not set up graphic support.

-r *size*, **--ram**=*size*
Allocate *size* in megabytes of memory for guest.

--sdl
Use Simple DirectMedia Layer for graphic access.

-u *uuid*, **--uuid**=*uuid*
The *uuid* for the new guest system. This is a 32-digit hexadecimal number. If not specified, a random UUID is used.

--vnc
Use permanent port *n* for VNC instead of a random port.
Note that specifying a port may cause clashes with other guests.

--vncport=*n*
Use port *n* for VNC. Note that specifying a port may cause clashes with other guests.

--vcpus *n*
Configure the new system with *n* virtual CPUs.

-w *type[:name]*, **--network**=*type[:name]*
Configure the guest's network connection. The type of network may be **bridge**, **network**, or **user**. For **bridge** and **network** you should also provide the network name as configured in the hypervisor. (Use **virsh net-list** to see configured network names.) **user** is supported by **qemu**'s unprivileged mode. It provides limited NAT by way of SliRP, a SLIP/PPP emulator.

virt-install

virt-install [*options*]

Create a guest machine completely from the command line. **Virt-install** does not require an XML image descriptor file. **virt-install** will prompt for any required information not provided on the command line. Following is the required information.

Options**--accelerate**

Use KVM or KQEMU kernel acceleration if available. Recommended when the guest OS is compatible.

--boot=n

If an image has multiple boot descriptors, use descriptor *n*. The first descriptor is descriptor 0. When omitted, the one that best fits the current hypervisor is chosen.

-c path, --cdrom=path

File to use for virtual CDROM device. The file may be an ISO file, CDROM device, or a URL that refers to an ISO image to fetch.

--check-cpu

Warn if the number of virtual CPUs for the guest exceeds the number of physical CPUs in the host system.

--connect=uri

The hypervisor connection *uri*.

--cpuset=set

Allow guest to use only the given *set* of CPUs. A set is a comma-separated list of numbers or number ranges: e.g., 0,2 or 0-2,5,6.

-d, --debug

Print debugging information to standard output.

-f path, --file=path

Required. The path to the file, disk partition, or logical volume to use to store the new guest's virtual disk. If the original system has multiple disks, this option should be given once per disk.

-h, --help

Print information on the command and options, then exit.

-k map, --keymap=map

Set the keymap to use for the virtual console. By default, an English keymap is used.

-l path, --location=path

Specify the location of the installation source. *path* may be given as a URI. Use this instead of -c when installing from a kernel/initrd pair instead of a CDROM. Required for installing paravirtualized guests.

--livecd

The guest will always use a live CDROM image. Configure guest to boot from CDROM drive by default. Commonly used with --nodisks option.

-m address, --mac=address

The hardware MAC *address* for the network interface on the copied system. If none is specified or address is given as RANDOM, assign a random address.

-n name, --name=name

Required. The domain *name* for the new guest.

--noacpi
Disable Advanced Configuration and Power Interface (ACPI).

--noapic
Disable Advanced Programmable Interrupt Controller (APIC).

--noautoconsole
Don't automatically launch a console for the installation. By default, **virt-install** launches a VNC client or **virsh console** to connect to the guest.

--nodisks
Guest is being created without disks. Do not prompt for disk setup.

--nographics
Don't prompt for virtual console setup if neither **--sdl** nor **--vnc** is specified. Set up a text based console on the first serial port (or equivalent console device) instead.

--nonsparse
Fully allocate space for the virtual disk. This takes longer to initialize, but results in a faster disk and ensures you don't run out of space. Recommended.

--os-type=*type*
Optimize guest for the specified guest OS *type*. Valid values for *type* include **linux**, **windows**, **unix**, and **other**.

-p, --paravirt
Guest will be a paravirtualized system.

--pxe
Use PXE boot protocol to load initial kernel.

-r *size*, --ram=*size*
Required. Allocate *size* in megabytes of memory for guest.

-s *size*, --file-size=*size*
Required when creating a new file. Create a file of *size* in gigabytes. Size may be given as a decimal, for example: **2.5**. By default, **virt-install** creates a sparse file. The full size will be allocated, though, if used with the **--nonsparse** option.

--sdl
Set up a virtual console and export it using Simple DirectMedia Layer (SDL).

-u *uuid*, --uuid=*uuid*
The *uuid* for the new guest system. This is a 32-digit hexadeciml number. If not specified, a random UUID is used.

-v, --hvm
Configure guest with full virtualization. Don't use paravirtualization if available.

--vnc
Set up a virtual console and export it using VNC.

--vncport=*n*
Use permanent port *n* for VNC instead of a random port. Note that specifying a port may cause clashes with other guests.

--vcpus *n*

Configure new system with *n* virtual CPUs.

-w *type[:name]*, --network=*type[:name]*

Configure guest's network connection. The type of network may be **bridge**, **network**, or **user**. For **bridge** and **network** you should also provide the network name as configured in the hypervisor. (Use `virsh net-list` to see configured network names.) **user** is supported by `qemu`'s unprivileged mode. It provides limited NAT by way of SliRP, a SLIP/PPP emulator. If this option is omitted, a single NIC is created to a bridge device or to the default network. Specify the option multiple times to set up multiple network interfaces.

-x *kernelargs*, --extra-args=*kernelargs*

Pass additional kernel arguments to a Linux kernel. Used with the **-l** option. Commonly used to pass a Kickstart file to the kernel.

`virt-viewer`

`virt-viewer [options] domain`

Use VNC to open the console of a virtual machine. The domain may be given by name, ID, or UUID.

Options**--c *uri*, --connect=*uri***

Connect to a hypervisor specified by *uri*.

--d, --direct

Don't tunnel the VNC connection over SSH, even if the connection URI uses SSH.

--v, --verbose

Print information about the connection to standard output.

--w, --wait

Wait for the domain to start up before connecting to the console.

`virt-manager`

`virt-manager [options]`

Launch GUI virtual machine manager. This manager visually displays domain and hypervisor stats. Like `virsh`, it can manage domains, and when run on the same system as the hypervisor, it can even be used to configure new systems. It has a couple of useful options for controlling how it starts.

Options**--c *uri*, --connect=*uri***

Connect to a hypervisor specified by *uri*.

--no-fork

Run without forking a new process. This can be useful for debugging connection problems.

VMware ESX 3.5

While VMware isn't exactly a Linux product and isn't free, it does provide access to a Service Console, which is something like Xen's Domain-0. The Service Console runs a modified version of Red Hat Enterprise Server 3.0. So even though it is a commercial product, we want to include some information on it in this chapter.

Here, we will cover useful Service Console commands for administrators using the commercial ESX 3.5 product. VMware has also provided a free version of ESX, named ESXi. ESXi does not technically have a Service Console, though you can launch an unsupported Linux interface similar to the Service Console that has many of the same commands. Many of the commands listed in this section will work in the unsupported console, although in keeping with the unsupported nature of this console, nothing is guaranteed. VMware also provides a remote command line interface (rCLI), which is a collection of Perl utilities that mostly mimic the Service Console commands.

As we are writing this, VMware is preparing to release vSphere, the new version of ESX. It too, has a Service Console and remote CLI, the vCLI. VMware has deprecated some commands in vSphere moving functionality to the vCLI. still, they continue to provide the Service Console for troubleshooting and technical support sessions. Most of what we document here still works in vSphere.

You will need to install ESX or ESXi on a bare system. Its installation is nearly identical to an older Red Hat installation. Given your network information, it should be able to configure a default network switch that will provide your guest systems with network access.

ESX Management Client

While we cover some of the common commands below, you won't use these for most of your management tasks. Once installed and booted, the initial screen will give you a URL you can connect to in order to perform some basic configuration. That URL will also direct you to where you can find a Virtual Infrastructure Client (VI Client) to manage your ESX server. It's that client you will use to manage your servers. VI Client runs on Microsoft Windows.

Virtual Center

If you are using VMware servers, you are probably going to use multiple servers to take advantage of the High Availability features VMware offers, such as Vmotion. To coordinate those servers you will need to run a Microsoft Windows Server system with VMware's virtual center installed. It's rumored that vSphere will have a Linux-based Virtual Center, but this has not been announced for the initial release. It still requires Microsoft Windows.

VMware is GUI driven and VMware expects you will primarily use VI Client to interact with your systems. You will need to run at least one windows system to manage VMware ESX servers. The Service Console won't let you get around this requirement. However, when your network stops working right or you need to take a closer look at a system's performance, you will be glad to have the service console available.

VMware Networking

When configuring a virtual network switch, you will configure three different network connection types: Virtual Machine, VMkernel, and Service Console. VMware refers to these as different kinds of *port groups*. A port group specifies port configuration options, such as VLAN tagging or bandwidth limitations for devices assigned to that group. The switch handles the mapping of virtual devices to physical devices. In a default installation of ESX you should have a Service Console and Virtual Machine port group.

The **Service Console** port group provides network access to the service console and is used by the virtual center or VI Client to control the server. The **Virtual Machine** port group is a default port group that will bridge your virtual machines to your physical network devices. Each network card on a virtual machine must be assigned to a virtual machine port group. For a high availability cluster of ESX servers, you will need to add a **VMkernel** port group as well. This is the port group the hypervisors will use to speak with each other and for moving virtual machines between servers.

Physical interfaces on a server are given a name beginning with **vmnic**. The service console's interfaces are given names beginning with **vswif**. Both of these will show up if you run **ifconfig** from the service console.

We won't cover virtual switch configuration further, but these terms will help you better understand the network and switch-related commands.

Shared Disks

To run multiple ESX servers, you need a shared data store. It can be SAN-based block storage (e.g., Fibre Channel or iSCSI) or a NAS connection (NFS), but you need some place the servers can all access the same data. For block storage, VMware uses a shared filesystem, **vmfs**, which is designed to handle multiple systems reading and writing to the same drive.

Snapshots

ESX can take snapshots of a running system. This process essentially freezes machine state and begins tracking changes to the system in a separate file. Later you can apply those changes or discard them to revert your system to the state it had when you took the snapshot. This is commonly used to test a configuration change or system update.

Because it preserves a system's state at a specific time, you want your virtual machine doing as little as possible when taking the snapshot. Activity such as network file transfers and database transactions will fail if you decide to revert to the snapshot. Depending on virtual machine configuration, some disk activity may not be included in the snapshot.

The file tracking changes can grow quickly, so you will need sufficient space to hold the file, and you will probably not want to keep snapshots for very long. Having a snapshot may also limit your ability to migrate a system between servers.

Restoring the snapshot on a new system may not work. VMware will generate warnings when you try to migrate a system with a snapshot. Generally you want to remove all snapshots before migration.

VMware Tools

Paravirtualization on ESX depends on VMware tools. This is a collection of drivers and daemons that allow the hypervisor to communicate with the guest system. VMware tools can synchronize the guests clock with the ESX server's clock, initiate graceful shutdowns when needed, provide better access to an X Window System running on a guest, and more. You almost always want to install these.

ESX Server Commands

Most commands for ESX server begin with either **vm** or **esx**. Commands that begin with **vm** all have something to do with manipulating virtual machines. Commands that begin with **esx** manipulate the server and its configuration. The commands may have more options than those we document here. Here, we focus on the most useful commands and command options.

esxcfg-firewall esxcfg-firewall [*options*]

Configure a firewall for the service console. ESX uses **iptables** to provide the firewall. Don't use the **iptables** command to configure it, though; use this command instead. In its default configuration, it denies all incoming or outgoing connections except for the ports required for the server to function:

--allowIncoming

Allow incoming connections by default.

--allowOutgoing

Allow outgoing connections by default.

--blockIncoming

Block incoming connections by default.

--blockOutgoing

Block outgoing connections by default.

-c *port type direction*, --close *port,type,direction*

Close a *port*. The *type* may be **tcp** or **udp**. The *direction* may be **in** or **out**.

-d [*service*], --disableService [*service*]

Close the ports required by the specified *service*.

-e [*service*], --enableService [*service*]

Open the ports required by the specified *service*.

-h, --help

Print a usage message. This is the default if no options are given.

-l, --load

Load current firewall settings into **iptables**.

- q [service], --query [service]
Print all current settings or the state of a *service* specified by name. Instead of a service you can query the default settings for **incoming** or **outgoing** packets.
- o *port type direction name*, --open *port,type,direction,name*
Open a *port*. The type may be **tcp** or **udp**. The *direction* may be **in** or **out**. You must give the service you have opened a *name*.
- r, --resetDefaults
Reset all parameters to the installed defaults.
- s, --services
List known service names.

Example

Allow vmware-cmd to connect to remote hosts by opening outgoing connections to port 443:

```
# esxcfg-firewall -o 443,tcp,out,vmware-cmd
```

esxcfg-vswif

esxcfg-vswif *options* [*interface*]

Configure the service console network interface. Don't use **ifconfig** or manipulate **/etc/sysconfig/network-script** files directly; use this command instead. Service console interface names start with **vswif** followed by a number. For example: **vswif0** or **vswif1**.

Options

- a, --add
Add a service console interface. Requires the **--ip** and **--port-group** options.
- b *address*, --broadcast *address*
Set the broadcast address.
- d, --del
Remove a service console interface.
- D, --disable-all
Disable all service console interfaces.
- e, --enable
Enable a disabled interface.
- E, --enable-all
Enable all service console interfaces.
- i *address*, --ip *address*
Set the IP address for the interface. You can also specify **DHCP** for *address*.
- l, --list
List service console interfaces along with current configuration and state.
- n *mask*, --netmask *mask*
Set the netmask.

- p *name*, --portgroup *name***
Set the interface's portgroup name.
- s, --disable**
Disable an interface.

esxcfg-vswitch *esxcfg-vswitch options [switch]*
Configure a virtual switch. The virtual switch handles port groups and VLAN tagging, and sets the uplink for virtual port groups. By default, there is a single virtual switch: vSwitch0. You can add additional switches and assign them to different physical network interfaces.

Options

- a, --add**
Add a new virtual switch. You must provide a switch name.
- A *name*, --add-pg *name***
Add port group *name* to the switch.
- b, --get-cdp**
Print the current Cisco Discovery Protocol (CDP) for the switch.
- B *status*, --set-cdp *status***
Set the CDP *status* for the switch. This may be **down**, **listen**, **advertise**, or **both**.
- d, --delete**
Delete a virtual switch. If the switch has any ports in use, this will fail.
- D *name*, --del-pg *name***
Remove port group *name* from the switch.
- l, --list**
List switches and their port groups.
- L *interface*, --link *interface***
Attach an uplink interface to a virtual switch. This should be an unused physical interface or **vmnic**.
- m *size*, --mtu *size***
Set the size of the Maximum Transfer Unit (MTU) for the switch. This will affect all interfaces attached to the virtual switch.
- M *interface*, --add-pg-uplink *interface***
Add an uplink *interface* to a port group.
- N *interface*, --del-pg-uplink *interface***
Remove an uplink *interface* from a port group.
- p *name*, --pg *name***
Specify name of the port group to use with the **--vlan** option.
Specify ALL to apply to all port groups on the switch.
- U *interface*, --unlink *interface***
Remove an uplink *interface* from a virtual switch.
- v *id*, --vlan *id***
Set the VLAN *id* for a port group. Use with the **--pg** option.
Setting *id* to 0 disables VLAN tagging for the port group.

esxcfg-nics	<code>esxcfg-nics options [interface]</code> Display information about physical network interfaces and manage some interface settings.
--------------------	---

Options

- a, --auto** Set the interface to auto-negotiate speed and duplex settings.
 - d mode, --duplex mode** Set duplex *mode* for the interface. Value may be **full** or **half**.
 - l, --list** Print current configuration of all network interfaces on the system.
 - s n, --speed n** Set the speed of the interface to *n*. Valid values are **10**, **100**, **1000**, or **10000** and are measured in megabits per second.
-

esxtop	<code>esxtop [options]</code> Provide frequently updated information about resource usage on the ESX server. This program performs a function similar to the Linux top command. Where top shows information on processes and process IDs, esxtop shows information on hypervisor resource pools, and resource pool groups. A resource pool is any item the hypervisor must schedule. For example, each virtual CPU, has a resource pool ID, and each virtual processor on a single virtual machine belongs to the same resource pool group.
---------------	---

esxtop's various options change what information is displayed. Like **top**, **esxtop** also offers interactive commands to change what information it displays as it runs. **esxtop** has more interactive commands than we'll show here. We are just covering the basic screens and common commands. The **h** command, however, will always show you the currently valid interactive commands.

Options

- a** Override configuration file settings and show all statistics.
- b** Run in batch mode; don't accept command-line input. Useful for sending output to another command or to a file.
- c file** Use alternative configuration *file*. This also becomes the default file for the interactive command **W**.
- d seconds** Set the delay between refreshes. Default is 3.
- n num** Update display *num* times, then exit.
- s** Secure mode. Disable the interactive command **s**.

Interactive commands

- # Prompt for number of rows to show.
- c Switch to CPU resource screen.
- d Switch to storage device screen.
- f, F Add or remove fields.
- h Show information on currently available commands.
- m Switch to memory resource screen.
- o, O Change field order.
- Q, Esc Quit.
- s Prompt for a new delay setting.
- n Switch to network resource usage screen.
- v Switch to virtual machine disk usage screen.
- W Write a configuration file matching current setup to `~/esxtop3rc`.

vmware-cmd

`vmware-cmd [options][vmxpath command]`

Console-based virtual machine manager. This command can work on the local system, or connect to a remote system using the Simple Object Access Protocol (SOAP). To connect to a remote system you may need to alter your firewall settings on the system issuing the commands so that it will allow outgoing connections to port 443. See the example in `esxcfg-firewall`.

The guest you want to work on is identified by the full path to its `.vmx` configuration file (`vmxpath`). The path must include the UUID of the vmfs volume and so can be quite long. Use the `-I` option to get the correct path for all registered virtual machines. While `vmware-cmd` has options for getting and setting virtual machine configuration values, we don't cover those here.

Options

- h Print basic usage and commands.
- H *host* Connect to remote *host*. This can be given as a domain name or IP address.
- s [*un*]register *configfile* Add or remove the guest described in `.vmx` file found at *path* from server control.
- l List virtual machines registered with server.
- O *port* Connect to alternative *port*, specified by port number. The default is port 443.
- P *password* Provide *password* for the user specified in the -U option.

- q Quiet mode. Minimize output.
- U *username* Connect with the given *username*.
- v Verbose mode.

Command

Return values for the command generally show the command function and whether they succeed (value 1) or fail (value 0.) Power-related commands generally take an option power operation *mode* of **soft**, **hard**, or **trysoft**. **soft** power operations use **vmware-tools** to generally do the right thing to the guest, such as gracefully shutting it down before pulling the virtual plug, or running special scripts to prepare the system for suspension. **hard** operations just perform the requested action immediately. **trysoft** operations will attempt to use **vmware-tools** for its operations, but if that fails, will force the operation without using tools.

connectdevice *name*

Connect device *name* to system. Valid device names can be found in the system's *.vmx* configuration file.

createsnapshot *name* *description* *quiesce* *memory*

Take a snapshot of the system, setting the new snapshot's *name* and *description*. If these values contain spaces, enclose them in quotes. You can quiesce drive activity and save memory state while taking the snapshot. *quiesce* and *memory* should be 0 for false or 1 for true. You usually will want both to be 1.

getstate

Report whether the virtual machine is off, on, suspended, or stuck.

getuptime

Report how long, in minutes, a machine has been running.

hassnapshot

Report whether virtual machine has a snapshot.

reset [*mode*]

Restart virtual machine.

removesnapshots

Apply saved changes and delete all snapshots.

revertsnapshot

Discard changes and restore virtual machine to most recent snapshot. Begin tracking changes again from the snapshot state.

start [*mode*]

Power up virtual machine.

stop [*mode*]

Power off virtual machine.

suspend

Save virtual machine's state and suspend its operation.

Examples

List registered virtual machines:

```
vmware-cmd -l
```

Register the yoyodyne system with the hypervisor:

```
vmware-cmd -s register /vmfs/volumes/uuid/yoodyne/  
yoodyne.vmx
```

Start a registered virtual machine:

```
vmware-cmd /vmfs/volumes/uuid/yoodyne/yoodyne.vmx start
```

Create a snapshot of the yoyodyne system:

```
vmware-cmd /vmfs/volumes/uuid/yoodyne/yoodyne.vmx \  
createsnapshot beforeupdate "state before system update" 1 1
```

Connect a virtual IDE CDROM to yoyodyne system:

```
vmware-cmd /vmfs/volumes/uuid/yoodyne/yoodyne.vmx \  
connectdevice ide0:0
```

vmkfstools

vmkfstools options [target]

Manipulate disk images and vmfs filesystems. As with other commands, you generally use VI Client to perform these tasks. But you must use the command line when importing a disk from another VMware product, such as VMware Workstation. Use this command to create or extend a VMFS system or to extend, rename, or remove a disk image. It can also set or query disk image properties. Use **vmkfstools** instead of **mv** and **cp** to move and copy virtual disks.

You generally do not want to perform virtual disk operations on disks currently in use by a virtual machine. Shut down virtual machines first. Changes in a virtual disk will often require changes in a virtual machine's configuration or changes in the virtual machine's operating system. One example is changing virtual machine partition information to take advantage of an extended disk.

Targets

The *target* will be a device, partition, or file, depending on what you intend to do.

device

A device *target* will begin with */vmfs/devices/*, the mount point of the device filesystem. This will include SAN-based disks, logical volumes, and SCSI devices attached to the ESX server.

partition

A partition *target* begins with **vmhba** followed by numbers that signify *adapter:target:LUN:partition[:filepath]*. For example, **vmhba0:1:14:2** would be the 2nd partition of LUN 14 on target 1 HBA 0. You can also target a file on a partition by appending the *filepath*.

path

A path *target* is simply a path to a *.vmdk* virtual disk file. This path can be given relative to the */vmfs* directory. The path to the file target isn't required if it is in the current working directory.

Options

We have grouped options that are commonly used together, generally following the order used in the manpages.

VMFS options:

-b size, --blocksize size

Set new VMFS volume block size. Default is 1m for 1 mega-byte. Other valid options are 2m, 4m and 8m. Used with the -C option.

-C type, --createfs type

Create a VMFS filesystem on a partition *target*. In ESX 3.5, the only valid type is **vmfs3**.

-S label, -setfsname label

Set the label for a new VMFS filesystem. This is used only with the -C option.

-Z partition, --extendfs partition

Extend existing vmfs3 filesystem by adding the specified *partition* to it. vmfs3 filesystems can have at most 32 partitions.

-P, --queryfs

List attributes of a VMFS filesystem. The target of this command may be the path to any directory or file on a VMFS filesystem.

-h, --human-readable

Used in conjunction with the -P option. Print sizes in a more easily read form.

Virtual disk options:

-c size, --createvirtualdisk size

Create a new virtual disk of the specified *size* in bytes (by default). You can also append g, m, or k to give numbers in gigabytes, megabytes, or kilobytes, respectively. For example: 8g for 8 gigabytes.

-a [type], --adAPTERtype [type]

Specify adapter *type* to use with the virtual disk. The only valid types are **buslogic** or **lsilogic**. Default is **buslogic**, but for Linux you probably want **lsilogic**.

-d format, --diskformat format

Set virtual disk **format**. This option is used when creating and cloning virtual disks (-c and -i options). See later options for valid virtual disk formats. When cloning, this option can be used to specify the format of the target disk; for example, when cloning a VMFS virtual disk file to a 2 GB sparse file useable by other VMware products.

-U, --deletevirtualdisk

Delete a virtual disk and any associated files.

-E source, --renamevirtualdisk source

Rename *source* virtual disk and associated files to *target*.

-i source, --clonevirtualdisk source

Clone *source* virtual disk and associated files to *target*. This is commonly used to import disks from other VMware products. You can use **-d** to specify the format of the source.

-X size, --extendvirtualdisk size

Extend an existing disk to the specified *size* in bytes (by default). You can also append g, m, or k to give numbers in gigabytes, megabytes, or kilobytes respectively. This action will break snapshots and the guest will need some way to recognize and take advantage of the new disk size (such as by resizing a partition with **resize2fs**).

-r device, --createrdm device

Create a virtual raw device mapping. (See the following section, “Virtual disk formats.”) Give *device* as the full path to a device under */vmfs/devices*.

-q, --queryrdm

Print attributes of an existing raw disk mapping.

-z device, --creatermpassthrough device

Create a physical raw device mapping. (See the following section, “Virtual disk formats.”) Give *device* as the full path to a device under */vmfs/devices*.

-g, --geometry

Print geometry information for a virtual disk.

-w, --writezeroes

Initialize a virtual disk with zeroes. This will wipe any existing data from the disk.

-j, --inflatedisk

Convert a thin virtual disk to a preallocated virtual disk, preserving existing data and zeroing out unallocated blocks.

Virtual disk formats

This section lists the valid target disk formats to use with the **-d** option when creating or cloning a virtual disk. Created disks may be in any of the ESX vmdk formats. When cloning a disk, use **-d** if you need to export to an alternate format or target a raw device.

ESX vmdk formats:

zeroedthick

Allocate all space and wipe any previous contents of that space during virtual machine read and write operations. This is the default format.

eagerzeroedthick

Like **zeroedthick**, only the allocated space is wiped at creation time.

thick

Allocate all space, but don’t bother cleaning any existing content on the disk.

thin

Supply and zero out space for the filesystem as it is used by the virtual machine.

VMware compatible formats:

2gbsparse

Sparse disk with 2 GB maximum extent size (the Virtual disk will be split into 2 GB sections). This is a useful format when writing to disks that don't support files larger than 2 GB.

monosparse

One monolithic disk file.

monoflat

One monolithic flat file.

Raw device mappings:

rdm:device

Virtual raw device mapping. These are treated by a guest as any other virtual disk.

rdmp:device

Physical raw device mapping. A virtual machine is given direct access to a physical raw device and interacts with it as it would any SCSI device. LUN listing requests are virtualized, however, so only the system that owns the virtual device can access it.

raw:device

A raw SCSI device, like a tape archive or an unmapped LUN.

Examples

Export a VMFS virtual disk to 2 GB sparse file:

```
# vmkfstools -i original.vmdk -d 2gbsparse new.vmdk
```

Index

Symbols

- & (ampersand)
 - background execution, 601, 602
 - command redirection, 603
 - repeat substitute command in ex, 710
 - reuse text matched by pattern, 657
- && (ampersand, double)
 - AND operator for commands, 602
 - AND operator, in expressions, 620, 732
- < or > (angle brackets)
 - command redirection, 601, 602–604
 - shift commands in ex, 710
- * (asterisk)
 - match multiple instances of a character, 655
 - in filename, 599
 - multiplication operator, 732
- ** (asterisk, double) exponentiation operator, 732
- *= (asterisk, equals sign) assignment operator, 732
- *() (asterisk, parentheses) in filename, 600
- **= (asterisks, equals sign) assignment operator, 732
- @ (at sign) execute register command in ex, 710
- @() (at sign, parentheses) in filename, 600
- ` (backtick) command substitution, 601, 602
- \ (backslash)
 - escape character, Bash shell, 601
 - escape character, pattern matching, 657
 - escape character, prompt strings, 612
- \< or \> (backslash, angle brackets)
 - match beginning or end of word, 656
- \' (backslash, apostrophe) escape character, pattern matching, 656
- \` (backslash, backtick) escape character, pattern matching, 656
- \{\...\} (backslash, braces) enclosing range of characters to match, 655
- \(...\) (backslash parentheses) save enclosed subpattern, 656
- \...; (backslash, semicolon) enclosing pattern address in sed, 714
- \{\...\} (braces)
 - enclosing command grouping, 602
 - enclosing nested pattern address in sed, 714
- enclosing procedures in gawk, 729
- enclosing range of characters to match, 655

We'd like to hear your suggestions for improving our indexes. Send email to index@oreilly.com.

[...] (brackets)
 enclosing array elements, 611
 enclosing characters to match, 655
 enclosing expressions, 647–650
 in filename, 599

[[]] (brackets, double) enclosing
 expressions, 620

^ (caret)
 match at beginning of line or
 string, 655
 exponentiation operator, 732

^= (caret, equals sign) assignment
 operator, 732

:

: (colon)
 null command, 620
 preceding ex commands, 696
 preceding label, sed, 717

\$ (dollar sign)
 field reference, 732
 last input line as pattern address in
 sed, 714
 match at end of line or string, 655
 preceding built-in variables, 607
 preceding macros in ftp, 149
 variable substitution, 601

\${} (dollar sign, braces)
 enclosing arrays, 611
 enclosing variable substitution, 606

\$() (dollar sign, double parentheses)
 arithmetic substitution, 602,
 613

\$() (dollar sign, parentheses) command
 substitution, 602

. (dot)
 end message, mailx, 267
 match single character, 655
 read and execute file command, 620

"..." (double quotes) enclosing special
 characters, 601

= (equals sign)
 assignment operator, 732
 print line number command in
 ex, 710
 command in sed, 716, 717

! (exclamation point)
 execute command in ex, 709
 execute command, mailx, 269
 history expansion character, 614
 logical negation operator, 732
 negating pipeline, 619

NOT operator for commands, 602
preceding commands in ftp, 149

!= (exclamation point, equal) not equal
 operator, 732

!() (exclamation point, parentheses) in
 filename, 600

!~ (exclamation point, tilde) match
 negation of regular
 expression, 732

(hash sign)
preceding comments in gawk, 738
preceding comments in LILO
 configuration file, 507

preceding comments in sed
 scripts, 717

preceding comments in shell
 scripts, 619

- (hyphen)
 previous command, mailx, 269
 subtraction operator, 732
 unary minus operator, 732

-- (hyphen, double) decrement
 operator, 732

-= (hyphen, equals sign) assignment
 operator, 732

- or -- (hyphen(s)) preceding command
 options, 33

< (left angle bracket) less than
 operator, 620, 732

<= (left angle bracket, equals sign) less
 than or equal operator, 732

(...) (parentheses)
 enclosing group of commands, 601,
 602

enclosing group of regular
 expressions, 656

% (percent sign)
preceding job ID in commands, 617
reuse previous replacement
 pattern, 657

modulus operator, 732

%= (percent sign, equals sign)
 assignment operator, 732

+ (plus sign)
 addition operator, 732
 match one or more instances, 656
 unary plus operator, 732

++ (plus sign, double) increment
 operator, 732

+= (plus sign, equals sign) assignment operator, 732
+() (plus sign, parentheses) in filename, 600
? (question mark)
 in filename, 599
 help command, mailx, 269
 match zero or one instance, 656
?: (question mark, colon) conditional expression, 732
?) (question mark, parentheses) in filename, 600
> (right angle bracket)
 greater than operator, 620, 732
 output redirection, 746
>> (right angle bracket, double) append output, 746
>= (right angle bracket, equals sign)
 greater than or equal operator, 732
> or >> (right angle brackets) in cat command, 68
; (semicolon) command separator, 601, 602
'...' (single quotes) enclosing special characters, 601
/ (slash) division operator, 732
/= (slash, equals sign) assignment operator, 732
/.../ (slashes) enclosing pattern address in sed, 714
~ (tilde)
 in filename, 599
 match regular expression, 732
 tilde escape commands, mailx, 268
 replace regular expression command in ex, 710
 reuse previous replacement pattern, 657
~- (tilde, hyphen) in filename, 599
~+ (tilde, plus sign) in filename, 599
| (vertical bar)
 match either one or the other, 656
 output directed to next command, 746
 pipe character, 601, 602
|& (vertical bar, ampersand) output directed to coprocess, 735, 746
|| (vertical bar, double)
 OR operator for commands, 602
 OR operator, in expressions, 620, 732

A

\a alert escape sequence, 601, 715, 733, 612
a command, sed, 716, 718
\A time escape sequence, 612
abbreviate command, ex, 697
accept command, 34
access command, 34
access mode for files, checking, 34
access permissions, changing, 80
account command, ftp, 149
account command, mailx, 269
aclocal command, 34
aconnect command, 35
ACPI (Advanced Configuration and Power Interface)
 displaying information about, 35
 events from, informing user programs of, 36
acpi command, 35
acpid daemon, 36
add command, git, 812
add command, svn, 767
addition operator (+), 732
addr add command, ip, 212
addr del command, ip, 212
addr show command, ip, 212
addr2line command, 37
Address Resolution Protocol (ARP), 22, 49
addresses of lines, ex, 696, 710
addresses, program, translating to filenames and line numbers, 37
Advanced Configuration and Power Interface (see ACPI)
Advanced Linux Sound Architecture (see ALSA)
Advanced Package Tool (see APT)
Advanced Power Management (APM)
 hardware, handling events from, 45
agetty command, 38
alias command, Bash, 620
alias command, lftp, 245
alias command, mailx, 269
aliases
 assigning, 620
 removing, 653
aliases database, rebuilding, 308
alnum character class, 600, 656

alpha character class, 600, 656
ALSA (Advanced Linux Sound Architecture)
 MIDI files
 playing, 44
 recording, 48
 mixer
 connecting to one or more sound cards, 40
 controlling, 41
 ports, reading and writing MIDI files to, 40
 sound cards, advanced configuration settings for, 39
sound files
 playing, 43
 recording, 48
alsactl command, 39
alsamixer command, 40
alternates (alt) command, mailx, 269
amidi command, 40
amixer command, 41
ampersand (&)
 background execution, 601, 602
 command redirection, 603
 repeat substitute command in ex, 710
 reuse text matched by pattern, 657
ampersand, double (&&)
 AND operator, in expression, 620
 AND operator, in expressions, 732
 for commands, 602
anacron command, 43
and() function, gawk, 738
AND operator (&&)
 for commands, 602
 in expression, 620
angle brackets (< or >)
 command redirection, 601, 602–604
 shift commands in ex, 710
 (see also left angle bracket; right angle bracket)
anon command, lftp, 245
ANSI/VT100 terminal
 emulation, 379–384
Apache web server
 benefits of using with Linux, 3
 user authentication files for, creating or updating, 199
aplay command, 43
aplaymidi command, 44
APM (Advanced Power Management)
 hardware, handling events from, 45
apmd command, 45
append command, ex, 698
append command, ftp, 149
apropos command, 46
APT (Advanced Package Tool), 46, 544
 (see also aptitude command, Debian; synaptic command, Debian)
apt command, 46
apt-cache command, Debian, 569–571
apt-cdrom command, Debian, 571
apt-config command, Debian, 572
apt-extracttemplates command, Debian, 573
apt-file script, Debian, 568
apt-ftparchive command, Debian, 574
apt-get command, Debian, 575–579
aptitude command, Debian, 544, 579–583
apt-rdepends script, Debian, 568
apt-sortpkgs command, Debian, 579
ar command, 47
arch command, 48
Arch system, 753
architecture for package, 543
architecture type, printing, 48
archive command, git, 813
archives
 copying files to or from tape or disk, 89–92
 copying files to or restoring from, 431–437
 of Debian packages, 566
 index for, generating, 347
 maintaining, 47
 of RPM packages, 552
 (see also packages)
arecord command, 48
arecordmidi command, 48
args command, ex, 698
arguments on command-line
 checking, 633
 printing, 494
 shifting, 643
arithmetic expressions, 602, 613, 636
arithmetic for loop, 632
ARP (Address Resolution Protocol), 22, 49
arp command, 49

arrays
 Bash, 611, 637
 gawk, 733
AS (autonomous system), 25
as command, 50
ASCII codes of keys, printing, 399
ascii command, ftp, 149
ASCII format, displaying files in, 196
asort() function, gawk, 738
asorti() function, gawk, 738
assembly source files, generating object files from, 50
assignment operators, gawk, 732
associative arrays
 Bash, 611
 gawk, 733
asterisk (*)
 in filename, 599
 match multiple instances of a character, 655
 multiplication operator, 732
asterisk, double (**) exponentiation operator, 732
asterisk, equals sign (=) assignment operator, 732
asterisk, parentheses (*)() in filename, 600
asterisks, equals sign (==) assignment operator, 732
at command, 51
at command, lftp, 245
at sign (@) execute register command in ex, 710
at sign, parentheses (@()) in filename, 600
atan2() function, gawk, 739
atd command, 53
atq command, 53
atrm command, 53
audio files
 playing, 43
 recording, 48
 recording to CDs, 69–71, 75, 484–488
 recording to DVDs, 125, 484–488
authentication
 adding RSA or DSA identities to agent, 409
 gathering keys from multiple hosts, 411
hold private keys for public key authentication, 410
keys, managing, 410
author command, svnlook, 800
autoconf command, 34, 53
autoheader command, 54
automake command, 55
autonomous system (AS), 25
autoreconf command, 56
auto_resume built-in variable, 611
autoscan command, 57
autoupdate command, 58
awk, GNU (see gawk language)
AWKPATH environment variable, gawk, 736

B

\b backspace escape sequence, 601, 733
b command, sed, 717, 718
\B escape sequence, 656, 715
\b escape sequence, 656, 715
background execution of commands, 601, 602, 621, 653
backing up files, 122–123
backslash ()
 escape character, Bash shell, 601
 escape character, pattern matching, 657
 escape character, prompt strings, 612
backslash, angle brackets (< or >)
 match beginning or end of word, 656
backslash, apostrophe (\') escape character, pattern matching, 656
backslash, backtick (`) escape character, pattern matching, 656
backslash, braces (\{\}) enclosing range of characters to match, 655
backslash parentheses (\(...\)) save enclosed subpattern, 656
backslash, semicolon (;;) enclosing pattern address in sed, 714
backtick (`) command substitution, 601, 602
backup files, restoring, 357–359

badblocks command, 58
base64 command, 59
basename command, 59
BASH built-in variable, 608
Bash shell, 60, 596–653
 arguments for, 599
 arithmetic expressions, 602, 613, 636
 arrays, 611
 background execution, 601, 621
 built-in commands, executing
 bypassing functions, 622
 built-in variables, 607–611
 character classes, 600
 command execution, 618
 command forms, 602
 command history, 614–617, 631, 634
 command redirection, 601, 602–604
 command substitution, 601, 602
 command-line options for, 598
 commands, enabling or
 disabling, 629
 commands, list of, 619–653
 coprocesses, 604
 debugger, 608
 escape character, 601
 filenames, metacharacters for, 599
 functions, 605
 help on commands for, 634
 home directories for, 599
 invoking, 597–599
 job control, 617
 line-edit mode, 614
 options for, setting or
 unsetting, 643–647
 POSIX notation in, 600
 programmable
 completion, 615–617, 624–626
 prompt strings, 612
 quoting special characters in, 600
 restricted shells, 619
 startup files for, 599
 suspending, 647
 variable substitution, 601, 606
 version of, covered in this book, 596
 websites about, 597

BASH_ARGC built-in variable, 608
BASH_ARGV built-in variable, 608
BASH_COMMAND built-in
 variable, 608

BASH_EXECUTION_STRING built-in
 variable, 608

BASH_LINENO built-in variable, 608

BASH_REMATCH built-in
 variable, 608

BASH_SOURCE built-in variable, 608

BASH_SUBSHELL built-in variable, 608

BASH_VERSINFO built-in
 variable, 608

BASH_VERSION built-in variable, 608

batch command, 60

Bautts, Tony (*Linux Network Administrator’s Guide*), xvii, 23, 29

bc compiler, 60–64

bdelete command, ex, 698

Beebe, Nelson H.F. (*Classic Shell Scripting*), 597

bell command, ftp, 149

Berkeley C shell, 596

Berkeley Internet Name Domain (BIND), 26

Berkeley Software Distribution (BSD), 4, 7

bg command, Bash, 621

BGP (Border Gateway Protocol), 25

binary command, ftp, 149

BIND (Berkeley Internet Name Domain), 26

bind command, Bash, 621

BIND DNS server, sending commands to, 363

binder process, NIS, 494

bindtextdomain() function, gawk, 739

biod daemon, 30

bisect command, git, 814–815

bison command, 64

blame command, svn, 767

blank character class, 600, 656

block files, creating, 292

blocklist command, GRUB, 530

blocks group, printing information about, 124

bookmark command, lftp, 245

books and publications

- Building Internet Firewalls (Zwicky; Cooper; Chapman), 29
- Classic Shell Scripting (Robbins; Beebe), 597
- GNU Emacs Pocket Reference (Cameron), 654

Internetworking with TCP/IP
(Comer), 23

Learning GNU Emacs (Cameron et al.), 654, 662

Learning Red Hat Enterprise Linux and Fedora (McCarty), 5

Learning the bash Shell (Newham), 597

Learning the Unix Operating System (Peek; Todino-Gonguet; Strang), xvii

Linux iptables Pocket Reference (Purdy), 29

Linux Journal, xviii

Linux Magazine, xviii

Linux Network Administrator's Guide (Bautts; Dawson; Purdy), xvii, 23, 29

Managing Projects with GNU Make (Mecklenburg), 275

Mastering Regular Expressions (Friedl), 654

Perl 5 Pocket Reference (Vromans), 654

Perl in a Nutshell (Patwardhan et al.), 654

Regular Expression Pocket Reference (Stubblebine), 654

Running Linux (Dalheimer; Welsh), xvii, 4

TCP/IP Network Administration (Hunt), 23

Version Control with Subversion (Pilato et al.), 755

boot command, GRUB, 530

boot loader, 505

boot sector, 505

booting, 504–541

- dual booting, 504, 505, 536–539
- GRUB, 504, 516–536
- kernel options during, 539–541
- LILO, 504, 506–516
- virtualization for, 504

bootp command, GRUB, 526

Border Gateway Protocol (BGP), 25

Bourne shell, 596

Bourne-Again SHell (see Bash shell)

braces ({}), 602

- enclosing command grouping, 602
- enclosing nested pattern address in sed, 714

enclosing procedures in gawk, 729

enclosing range of characters to match, 655

brackets ([...])

- enclosing array elements, 611
- enclosing characters to match, 655
- enclosing expressions, 647–650
- in filename, 599

brackets, double ([[...]]) enclosing expressions, 620

branch command, git, 815

branches, for source code management system, 750, 756

branches, in Git, 806, 815

- merging, 826, 827
- pushing, 828
- tracking changes to, 830

brctl command, 843

break command, Bash, 622

break command, gawk, 739

break command, gdb, 164

bridge-utils package, 843

bridging, 842

BSD (Berkeley Software Distribution), 4, 7

bt command, gdb, 164

buffer command, ex, 698

buffer-manipulation commands, Emacs, 666

buffers command, ex, 698

Building Internet Firewalls (Zwicky; Cooper; Chapman), 29

builtin command, Bash, 622

built-in commands (see commands, Bash)

built-in variables, 607–611

built-in variables, gawk, 731

bunzip2 command, 65

bye command, ftp, 149

bzcat command, 65

bzcmp command, 65

bzdiff command, 65

bzgrep command, 65

bzip2 command, 65

bzip2 files, 65–67

- (see also compressed files)

bzip2recover command, 65

bzless command, 67

bzmore command, 67

C

- c command, gdb, 164
- c command, sed, 716, 718
- c++ compiler (see g++ compiler)
- \c control character escape sequence, 601
- C language
 - generating from RPC code, 366
 - preprocessor for, 92–98
- cache command, lftp, 245
- cache command, mailx, 269
- cal command, 68
- calendar, printing, 68
- caller command, Bash, 622
- cameras, reading images from, 376
- Cameron, Debra
 - GNU Emacs Pocket Reference, 654
 - Learning GNU Emacs, 654, 662
- capitalization commands, Emacs, 666
- caret (^)
 - exponentiation operator, 732
 - match at beginning of line or string, 655
- caret, equals sign (^=) assignment operator, 732
- case command, Bash, 622
- case command, ftp, 149
- cat command, 68
- cat command, GRUB, 531
- cat command, svn, 768
- cat command, svnlook, 800
- cc compiler (see gcc compiler)
- cd command, Bash, 623
- cd command, ex, 698
- cd command, ftp, 149
- cd command, gdb, 164
- CDDA (Compact Disc Digital Audio)
 - converting to WAV, 69
 - converting to WAV format, 200–203
- cdda2wav command, 69
- cdparanoia command, 69–71
- CDPATH built-in variable, 609
- cdrdao command, 71–75
- cdrecord command, 75
- CD-Rs, writing in DAO mode, 71–75
- CDs
 - commands for, list of, 9
 - converting from CDDA to WAV format, 200–203
 - ejecting, 128
 - reading and writing, 350, 352
- recording audio files to, 69–71, 75
- recording data files to, 75
- recording data or audio to, 484–488
- cdup command, ftp, 149
- center command, ex, 699
- centering commands, Emacs, 668
- cfdisk command, 75
- c++filt command, 67
- C-h command (help), Emacs, 663
- chage command, 76
- chainloader, 517
- chainloader command, GRUB, 531
- chains, 29
- change command, ex, 699
- changed command, svnlook, 800
- changelist command, svn, 768
- Chapman, D. Brent (*Building Internet Firewalls*), 29
- character classes
 - Bash, 600
 - pattern matching, 656
- character files, creating, 292
- chattr command, 77
- chdir (c) command, mailx, 269
- checkout command, git, 816
- checkout command, svn, 769
- checksums, calculating and printing, 427
- check-update command, yum, 547
- cherry-pick command, git, 817
- chfn command, 78
- chgrp command, 78
- chkconfig command, 79
- chmod command, 80
- chmod command, ftp, 149
- chown command, 82
- chpasswd command, 83
- chroot command, 83
- chrt command, 83
- chsh command, 84
- chvt command, 84
- CIDR (Classless Inter-Domain Routing), 24
- cksum command, 84
- Classic Shell Scripting (Robbins; Beebe), 597
- classify command, mailx, 270
- Classless Inter-Domain Routing (CIDR), 24
- clean command, git, 818
- clean command, yum, 548

cleanup command, svn, 770
clear command, 84
clear command, gdb, 164
clearing screen, 356
client, NIS, 32
client/server model, for source code management system, 751
clocks
 commands for, list of, 14
 reading or setting, 200
clone command, git, 818
close command, ex, 699
close command, ftp, 150
close command, lftp, 246
close() function, gawk, 739
cmp command, 85
cmp command, GRUB, 531
cntrl character class, 600, 656
code examples in this book, using, xix
Codeville system, 753
col command, 85
colcrt command, 86
colon (:)
 null command, 620
 preceding ex commands, 696
 preceding label, sed, 717
color command, GRUB, 526
colrm command, 86
column command, 86
columns
 formatting input into, 86
 removing from file, 86, 102
COLUMNS built-in variable, 609
Comer, Douglas E. (*Internetworking with TCP/IP*), 23
comm command, 86
command command, Bash, 623
command command, lftp, 246
command mode, vi, 678, 681
command-line arguments
 checking, 633
 printing, 494
 shifting, 643
commands, 3, 5
 Bash, 619–653
 aliases for, assigning, 620
 aliases for, removing, 653
 conditional execution of, 622, 635
 coprocessing, 604
 enabling or disabling, 629
executing, 618
executing based on signals, 650
executing, bypassing aliases or functions, 622, 623
executing in current process, 630
forms of, 602
hash table for, 633
history of, 614–617, 631, 634
programmable completion
 for, 624–626
redirection of, 601, 602–604
substitution of, 601, 602
 type of, determining, 651
for clocks, 14
for communications, 7
for comparisons, 8
for daemons, 14
deleting queued jobs of, 53
displaying manual pages for, 283
Emacs, 663, 669–676
 buffer-manipulation, 666
 capitalization, 666
 centering, 668
 completions for, 662
 deletion, 664
 file-handling, 663
 help, 663, 668
 indentation, 667
 jobs, 667
 listing, 663
 macro, 668
 paragraph, 665
 region, 665
 search, 665
 shell, 667
 stopping, 665
 transposition, 665
 undoing, 665
 windows, 666
 word-abbreviation, 666
ex, 697–710
 line addresses in, 696, 710
 options for, 697
 syntax for, 696
executing and timing the execution, 445
executing as superuser, 426
executing at scheduled times, 98
executing at specific time and date, 51
executing even after logging out, 311

commands (*continued*)
executing in a new session, 395
executing in batch from standard input, 60
executing on remote host, 367
executing periodically, 43
executing queued jobs of, 53
executing remotely, 359
executing repeatedly, 472
executing with arguments from standard input, 488–489
executing with lower priority, 309
executing with specific options, 119
executing within ftp, 149
for file management, 8
for filesystems, 17
full pathnames for execution of, 481
GRUB, 525–536
for hardware, 15
help on, 33
for host information, 15
for installation, 16
for kernel, 17
list of, by category, 7–13
listing queued jobs of, 53
locating files for, 480
for mail, 16
for media, 9
for networking, 18
for NFS administration, 22
for NIS administration, 22
for NIS maps, 31
options for, entering, 33
output from, sending to both standard output and files, 440
for printing, 9, 18
for process management, 19
for program maintenance, 10
for programming, 9
scheduling, 20
for searching, 10
for security, 18
sending to BIND DNS server, 363
for shell programming, 11
for storage, 11
for system administration, 14–20
for system integrity, 18
for system logging, 20
for system status, 11
for TCP/IP administration, 21
for TCP/IP troubleshooting, 28
for text processing, 12
tracing system calls and signals for, 416–418
for user management, 20
vi, 683–692
copy and move, 688
edit, 687
insert, 686
macros, 691
movement, 683–686
multiple files, accessing, 689
saving files, 688
:set command, 692–696
system access, 690
window, 690
waiting before execution of, 403
(see also specific commands)
commands command, gdb, 164
comments
in bc, 63
in C, nested, warnings for, 96, 161
in database map input file, 282
in DNS update requests, 311
in file containing format strings, 197
in gawk, 738
in GRUB configuration file, 521
in history expansion, 611
in LILO configuration files, 507
in MAC address input file, 305
in makefiles, 276
in RPM package manifest file, 557
in sed scripts, 717
in shell scripts, 619
in xinetd input, 491
commit command, git, 819
commit command, svn, 771
communications, commands for, 7
Compact Disc Digital Audio (see CDDA)
comparison operators, 620
comparisons, commands for, 8
COMP_CWORD built-in variable, 608
compgen command, Bash, 624
compl() function, gawk, 739
complete command, Bash, 615, 624–626
completion, programmable, 615–617, 624–626
COMP_LINE built-in variable, 608
COMP_POINT built-in variable, 608
COMPREPLY built-in variable, 609

compress command, logrotate, 253
compresscmd command, logrotate, 253
compressed files
 bz2 files, 65–67
 comparing, 501
 compressing executable files, 190
 compressing files with Lempel-Ziv
 coding, 191
 reading and then writing to standard
 output, 501
 renaming, 501
 uncompressing and searching, 502
 uncompressing and then paging
 through, 502
 uncompressing and then
 recompressing in .gz
 format, 503
 uncompressing files, 190
compressext command, logrotate, 253
compressoptions command,
 logrotate, 253
COMP_WORDBREAKS built-in
 variable, 608
COMP_WORDS built-in variable, 608
concatenation operator, 732
Concurrent Versions System (CVS), 752
conditional execution of commands
 case command, 622
 if command, 635
 short-circuit operations for, 602
config command, git, 820
configfile command, GRUB, 531
configuration of vi editor, 692–696
configuration script
 generating from m4 macros, 53
 updating, 56
conflict, in source code management
 system, 751
console command, xm, 846
contact information for this book, xxi
containers technology, 838
context-free grammar, converting to
 tables, 493
continue command, Bash, 626
continue command, gawk, 739
conventions used in this book, xx
Cooper, Simon (*Building Internet
Firewalls*), 29
coproc reserved word, 604
coprocesses, Bash, 604
coprocesses, gawk, 735
copy and move commands, vi, 688
copy (co) command, mailx, 270
copy command, ex, 699
copy command, logrotate, 254
copy command, svn, 772
copyleft (see GPL)
copy-modify-merge model,
 Subversion, 756
copytruncate command, logrotate, 254
cos() function, gawk, 739
cp command, 87–89
cpio command, 89–92
cpp command, 92–98
cr command, ftp, 150
CRC (cyclic redundancy check)
 checksum, computing, 84
create command, logrotate, 254
create command, svnadmin, 795
create command, xm, 846
cron command, 545
crond command, 98
crontab command, 98
csh shell, 596
csplit command, 99
CSSC system, 754
ctags command, 100
Ctrl key (C-), Emacs, 662
CUPS printer queues, configuring, 256
cupsd command, 101
current working directory
 changing, 623
 printing, 638
cursor (point) location, Emacs, 662
cursor-movement commands,
 Emacs, 664
cut and paste (kill and yank),
 Emacs, 662
cut command, 102
cut-and-paste utility for Linux
 console, 178–180
CVS (Concurrent Versions System), 752
cyclic redundancy check (CRC)
 checksum, computing, 84

D

D command, sed, 717, 719
d command, sed, 716, 719
\D date escape sequence, 612
\d date escape sequence, 612
\d decimal escape sequence, 715

daemons, 21
 commands for, list of, 14
 nfsd daemons, 30
 routing daemons, 25
daily command, logrotate, 254
Dalheimer, Matthias Kalle (Running Linux), xvii, 4
DAO (disk-at-once) mode, 71
database maps for sendmail,
 creating, 282
database of files, searching, 250
databases, searching for a specific
 key, 173
date and time
 current date and time,
 printing, 103–105
 retrieving from specific host(s) on
 network, 348
date command, 103–105
date command, svnlook, 800
Dawson, Terry (Linux Network Administrator’s Guide), xvii, 23, 29
dbm database files, 31
dbm file, NIS, creating or dumping, 281
dcgettext() function, gawk, 739
dcngettext() function, gawk, 739
dd command, 105, 844
dd command, GRUB, 519
deallocv command, 107
Debian Package Manager, 544, 565–569
 commands for, 569–595
 files used by, 565
 interactive frontend for (see `dselect`
 command, Debian)
 lower-level packaging tool for (see
 `dpkg-deb` command, Debian)
 naming conventions for, 543, 565
 package flags, 567
 package priorities, 566
 package states, 567
 scripts for, 568
debug command, ftp, 150
debug command, GRUB, 531
debugfs command, 107–110
debugger
 Bash, 608
 gdb, 163–165
debuginfo-install plugin, yum, 551
decimal format, displaying files in, 196
declare command, Bash, 626
decoding, base64, 59
decrement operator (--), 732
decryption (see encryption and
 decryption)
default command, GRUB, 522
#define statements, generating template
 of, 54
delaycompress command,
 logrotate, 254
delete command, ex, 699
delete command, ftp, 150
delete command, gawk, 740
delete command, gdb, 164
delete command, svn, 773
delete (d) command, mailx, 270
deletion commands, Emacs, 664
deltify command, svnadmin, 795
dependencies for packages, 542, 543
dependency file, creating, 110
deplist command, yum, 548
depmod command, 110
desktop environments, 4
desktop system, Linux as, 3
destroy command, xm, 847
devdump command, 111
device command, GRUB, 526
device map, GRUB, 518
devices
 assigning interfaces to, 305
 displaying contents of, 111
 formatting, 287
 formatting as ext2
 filesystem, 288–290
 paging, turning off, 427
 paging, turning on, 427
 swapping, turning off, 427
 swapping, turning on, 427
 volume name for, determining, 471
df command, 112
dhcp command, GRUB, 527
diff command, 113–115
diff command, git, 821
diff command, svn, 774–775
diff command, svnlook, 800
diff3 command, 115–116
dig command, 26, 116
digit character class, 600, 656
dir command, 118
dir command, ftp, 150

dircolors command, 118
directories
 adding to stack, 638
 contents of, listing verbosely, 468
 creating, 287
 deleting, 362
 listing contents of, 118, 260–263
 lost+found directory, creating, 292
 moving or renaming, 302–303
 printing stack for, 627
 printing top directory from
 stack, 637
 removing from path, 59

Dired (Directory Editor) mode,
 Emacs, 662

dirname command, 119

dirs command, Bash, 627

dirs-changed command, synlook, 801

DIRSTACK built-in variable, 609

disaster recovery data, storing, 126

disconnect command, ftp, 150

disk image files, for virtualization, 844

disk usage
 auditing, 345
 available space, determining, 112
 printing, 120, 344
 quota statistics for, printing, 347
 reporting on, 355

disk-at-once (DAO) mode, 71

disown command, Bash, 628

display command, gdb, 164

displayapm command, GRUB, 531

displaymem command, GRUB, 531

distributions of Linux, 3

division operator (/), 732

dmesg command, 119

dmesg command, xm, 847

DNS (Domain Name System), 26

DNS servers, querying, 116

do command, gawk, 740

do reserved word, Bash, 628

document formatting (see *groff*
 command)

document processing, *gs* command
 for, 189

doexec command, 119

dollar sign (\$)
 field reference, 732
 last input line as pattern address in
 sed, 714
 match at end of line or string, 655

preceding built-in variables, 607

preceding macros in ftp, 149

variable substitution, 601

dollar sign, braces (\${...})
 enclosing arrays, 611
 enclosing variable substitution, 606

dollar sign, double parentheses (\$())
 arithmetic substitution, 602,
 613

dollar sign, parentheses (\$()) command
 substitution, 602

Domain Name System (DNS), 26

domain names
 contact for, finding, 483
 definition of, 26

domain nameserver
 querying, 311
 updating, 311

domainname command, 120

domains
 definition of, 26
 NIS, 31

done reserved word, Bash, 628

dosfsck command, 120

dot (.)
 end message, mailx, 267
 match single character, 655
 read and execute file command, 620

double quotes ("...") enclosing special
 characters, 601

down command, gdb, 164

downloading files from
 Internet, 474–480

dp (dt) command, mailx, 270

dpkg command, Debian, 583–588
 (see also Debian Package Manager)

dpkg-architecture script, Debian, 568

dpkg-buildpackage script, Debian, 568

dpkg-checkbuilddeps script,
 Debian, 568

dpkg-distaddfile script, Debian, 568

dpkg-divert script, Debian, 568

dpkg-genchanges script, Debian, 568

dpkg-gencontrol script, Debian, 568

dpkg-name script, Debian, 568

dpkg-parsechangelog script,
 Debian, 568

dpkg-preconfigure script, Debian, 568

dpkg-query command, Debian, 590

dpkg-reconfigure script, Debian, 568
dpkg-scanpackages script, Debian, 568
dpkg-shlibdeps script, Debian, 568
dpkg-source script, Debian, 569
dpkg-split command, Debian, 591
dpkg-statoverride script, Debian, 569
DSA identities, adding to authentication agent, 409
dselect command, Debian, 544, 592–594
du command, 120
dual booting, 504, 505, 536–539
dump command, 122–123
dump command, GRUB, 531
dump command, svnadmin, 795
dumpfs command, 124
dumpiso command, 124
dumpkeys command, 124
dvorecord command, 125
DVDs
 commands for, list of, 9
 ejecting, 128
 recording, 125, 484–488

E

e command, sed, 716, 719
\E escape character, pattern matching, 657
\E escape sequence, 601
\e escape character, pattern matching, 657
\e escape sequence, 601, 612
e2fsck command, 125
e2image command, 126
e2label command, 127
EasyBCD software, 536
echo command, 127
echo command, Bash, 628
ed editor, metacharacters supported by, 657
edit command, ex, 699
edit commands, vi, 687
edit (e) command, mailx, 270
EDITOR built-in variable, 609
editors (see ed editor; Emacs editor; ex editor; sed editor; vi editor; vim editor)
edquota command, 127
EGP (Exterior Gateway Protocol), 25
egrep command, 128, 657

eject command, 128
elevator algorithm, setting latency for, 130
ELF (Executable and Linking Format) object files, 351
elvis editor, 677
elvtune command, 130
EMACS built-in variable, 609
emacs command, 663
Emacs editor, 130, 661–676
 commands
 buffer-manipulation, 666
 capitalization, 666
 centering, 668
 completions for, 662
 cursor-movement, 664
 deletion, 664
 file-handling, 663
 help, 663, 668, 669
 indentation, 667
 jobs, 667
 list of, alphabetic by keystroke, 669–672
 list of, alphabetic by name, 672–676
 list of, most important, 663
 listing, 663
 macro, 668
 paragraph, 665
 region, 665
 search, 665
 shell, 667
 stopping, 665
 transposition, 665
 undoing, 665
 windows, 666
 word-abbreviation, 666
Ctrl key notation (C-), 662
file buffers in, 662
invoking, 663
kill and yank (cut and paste) in, 662
mark location in, 662
Meta key notation (M-), 662
modes in, 661
point (cursor) location in, 662
windows in, 662
embed command, GRUB, 532
embedded versions of Linux, 3
enable command, 130
enable command, Bash, 629
encoding, base64, 59

encryption and decryption with GNU Privacy Guard, 173–177
(see also security)

endscript command, logrotate, 254

Enter key, print next line command in ex, 710

ENV built-in variable, 609

env command, 130

environment, displaying, 130

environment variables

- printing, 334
- setting, 130
- substituting in shell string or script, 131

envsubst command, 131

equals sign (=)

- assignment operator, 732
- command in sed, 716, 717
- print line number command in ex, 710

esac reserved word, Bash, 629

escape sequences, 601

- gawk, 733
- pattern matching, 656

ESX management client, for VMware, 862

ESX server commands, for VMware, 864–873

esxcfg-firewall command, VMware, 864

esxcfg-nics command, VMware, 867

esxcfg-vswif command, VMware, 865

esxcfg-vswitch command, VMware, 866

esxtop command, VMware, 867

etags command, 131

EUID built-in variable, 609

eval command, Bash, 629

ex editor, 132, 677, 678–680, 696–710

- command options, 697
- command syntax, 696
- command-line options, 678–680
- commands, 697–710
- exiting, 681
- invoking, 680, 696
- line addresses in, 696, 710
- pattern matching metacharacters supported by, 657

examples in this book, using, xix

exclamation point (!)

- execute command in ex, 709
- execute command, mailx, 269
- history expansion character, 614

logical negation operator, 732

negating pipeline, 619

NOT operator for commands, 602

preceding commands in ftp, 149

exclamation point, equal (!=) not equal operator, 732

exclamation point, parentheses (!()) in filename, 600

exclamation point, tilde (!~) match negation of regular expression, 732

exec command, Bash, 630

Executable and Linking Format (ELF) object files, 351

executable files, compressing, 190

exit command, Bash, 630

exit command, gawk, 740

exit (ex, x) command, mailx, 270

exp() function, gawk, 740

expand command, 132

exponentiation operator (^ or **), 732

export command, Bash, 630

export command, svn, 775

expr command, 133–135

expressions, evaluating, 133–135, 443, 620, 647–650

ext2 filesystem

- checking and repairing, 125
- debugging, 107–110
- formatting devices as, 288–290
- label for, displaying, 127
- printing block and superblock information, 124
- resizing, 356
- storing disaster recovery data for, 126
- tuning parameters of, 457–460

ext3 filesystem, 16

- debugging, 107–110
- formatting devices as, 291
- printing block and superblock information, 124

extended Internet services

- daemon, 490–493

extended regular expressions, searching with, 128

Extensible Filesystem (see XFS)

extension command, logrotate, 254

extension() function, gawk, 740

Exterior Gateway Protocol (EGP), 25

eXternal Data Representation (XDR), 32

F

\f form feed escape sequence, 715, 733, 601
factor command, 135
fallback command, GRUB, 522
false command, 135
false command, Bash, 631
Fast Lexical Analyzer Generator, 143
FAT filesystem, 16
fc command, Bash, 614, 631
fc-cache command, 135
FCEDIT built-in variable, 609
fc-list command, 136
fdisk command, 136
fetch command, git, 822
fflush() function, gawk, 740
fg command, Bash, 631
fgconsole command, 137
fgrep command, 137
fi reserved word, Bash, 632
field reference (\$), gawk, 732
FIFO pipes, creating, 292
IGNORE built-in variable, 609
file command, 137
file command, ex, 700
file (fi) command, mailx, 270
file inclusion, gawk, 736
filenames
 checking validity of, 326
 database of, searching, 250
 metacharacters for, 599
 temporary, 293
 validity and portability of, 326
files
 access mode for, checking, 34
 access permissions for, changing, 80
 attributes of, changing, 77
 attributes of, listing, 263
 backing up, 122–123
 base64 encoding or decoding, 59
 byte count for, 473
 classifying, 137
 columns in, removing, 86
 comparing, 85, 113–116
 comparing and finding differences between, 384
 comparing lines common to, 86
 compressing, 191
 compressing and decompressing, 65
 computing or checking MD5 checksums for, 285
concatenating, 68
conditions of, testing for, 647
converting into tables for parsing, 64
converting to alternate character encoding, 203
converting to tables, 493
copying, 87–89
copying and converting format of, 105
copying or restoring from archives, 431–437
copying securely, 378
copying to or from remote systems, 347
creation mask for, displaying or setting, 652
deleting, 361
deleting files older than a specific access time, 447
displaying in hex, octal, decimal, or ASCII, 196
displaying one screen at a time, 235–242, 295
downloading from Internet, 474–480
dumping, in octal format by default, 319–322
executable, compressing, 190
formatting into columns, 86
group for, changing, 78
line count for, 473
linking, 247
locating, 480
maintaining in archives, 47
management of, commands for, 8, 663
merging, 286
merging corresponding lines of, 323
moving or renaming, 302–303
overwriting, to make contents unrecoverable, 400
ownership of, changing, 82
paginating, suitable for printing, 332–334
paging through, 235–242, 295
paging, turning off, 427
printing, 258
printing first few lines of, 196
printing last ten lines of, 430
printing lines in reverse order, 429
processes using, identifying, 154

recompiling and relinking based on dependencies, 275–281
reformatting to a specific width, 144, 145
remote file distribution, 348–350
removing by unlinking, 464
removing columns or fields from, 102
removing duplicate adjacent lines in, 463
renaming, 354
restoring from backup, 357–359
reversing characters in each line of, 359
searching, 137
searching for files meeting specific conditions, 139–143
searching for printable strings in, 418
searching for strings at beginning of lines, 256
searching with extended regular expressions, 128
separating into context-based sections, 99
sorted, joining, 226
sorting lines of, 404
sorting strings in, 457
splitting into equal-sized segments, 405
swapping, turning off, 427
transferring to and from remote sites, 148–153, 244–247, 398
unchanged, verifying, 285
uncompressing, 190
updating access and modification times to current time, 452
updating across a network, 368–374
word count for, 473
(see also directories; paths)
filesystem checker, for MS-DOS filesystems, 120
filesystem types, 16
filesystems, 16
 checking and repairing, 125, 147
 commands for, list of, 17
 creating on a device, 291
 debugging, 107–110
 displaying contents of, 111
 ISO9660/Joliet/HFS,
 generating, 165–173, 292
mounting, 297–299, 300
quotas, editing, 127
quotas for, enforcing, 346
quotas for, turning off enforcement of, 347
unmounting, 461
writing buffers to disk, 428
(see also NFS)
find command, 139–143
find command, GRUB, 532
firewalls, 28
 book about, 29
 rules, printing, 223
 rules, restoring, 222
firstaction command, logrotate, 254
fixmbr command, 506
flex command, 143
floppy disks, ejecting, 128
fmt command, 144
fold command, 145
fold command, ex, 700
foldclose command, ex, 700
folder (fold) command, mailx, 270
folders command, mailx, 270
foldopen command, ex, 700
font conventions used in this book, xx
font information caches, creating, 135
fonts, listing, 136
for command, Bash, 632
for command, gawk, 740, 741
foreground, moving job to, 631
form command, ftp, 150
formail command, 145–147
forward (fwd) command, mailx, 270
frame command, gdb, 164
free command, 147
free disk space, determining, 112
Free Software Foundation (see FSF)
Friedl, Jeffrey E.F. (*Mastering Regular Expressions*), 654
from (f) command, mailx, 270
fsck command, 147
fsck.ext2 command, 125
FSF (Free Software Foundation), 6
fstest command, GRUB, 532
ftp command, 148–153
 (see also sftp command)
ftpd command, 153
FUNCNAME built-in variable, 609
function command, Bash, 633
function command, gawk, 741

functions, 605
 declaring, 633
 defining, 637
 deleting, 653
 exiting and returning status
 from, 640
 listing for source file, 131
 overloading, decoding symbols
 for, 67
 source and line number for, 622
 user-defined, gawk, 734
fuser command, 154

G

G command, sed, 716, 719
g command, sed, 716, 719
g++ compiler, 155
gateways, 25
gawk command, 155
gawk language, 726–747
 array assignment, 733
 built-in variables, 731
 command-line options for, 728–729
 command-line syntax for, 727
 commands for, 737–746
 coprocesses, 735
 escape sequences, 733
 file inclusion for, 736
 functions for, 737–746
 hexadecimal constants in, 734
 implementation limits of, 737
 internationalization for, 736
 invoking, 727
 octal constants in, 734
 operators, 732
 output redirection, 746
 pattern matching metacharacters
 supported by, 657
 patterns in, 729, 730
 procedures in, 729, 730
 profiling, 735
 sockets, 735
 user-defined functions, 734
 variable assignment, 732
gc command, git, 822
gcc compiler, 156–163
gdb debugger, 163–165
General Public License (see GPL)
genisoimage command, 165–173
gensub() function, gawk, 741

geometry command, GRUB, 532
get command, ftp, 150
getent command, 173
getkeycodes command, 173
getline command, gawk, 741
getopts command, Bash, 633
getpwnam() function, 599
getpwuid() function, 599
gettext, GNU, 736
GhostScript, 189
git command, 811–836
 add subcommand, 812
 archive subcommand, 813
 bisect subcommand, 814–815
 branch subcommand, 815
 checkout subcommand, 816
 cherry-pick subcommand, 817
 clean subcommand, 818
 clone subcommand, 818
 commit subcommand, 819
 config subcommand, 820
 diff subcommand, 821
 fetch subcommand, 822
 gc subcommand, 822
 gitk subcommand, 823
 grep subcommand, 823
 help subcommand, 812
 init subcommand, 824
 log subcommand, 824
 merge subcommand, 826
 mv subcommand, 827
 pull subcommand, 827
 push subcommand, 828
 rebase subcommand, 829
 reflog subcommand, 830
 remote subcommand, 831
 reset subcommand, 832
 revert subcommand, 833
 rm subcommand, 833
 stash subcommand, 834
 status subcommand, 834
 tag subcommand, 835
git daemon, 811
git shell, 811
Git version control system, 753, 805–836
 branches in, 806, 815
 merging, 826, 827
 pushing, 828
 tracking changes to, 830

checking in (committing) files, 806, 809, 819
cherry-picking commits, 817, 829
finding problem in, 814–815
referring to commits, 806–808
remotely, 822
reversing, 833
searching commits, 823
testing results of, 834
undoing, 832
checking out files, 806, 809, 816
command line for, 811
configuring, 820
differences, generating, 821
help for, 812
history, displaying, 823, 824
hosting projects, 811
index for, 806
 adding or updating files in, 812
 removing files from, 833
remotes for, 831
repositories for, 805, 806
 cloning, 818
 creating, 810, 824
 creating archive from, 813
 packing, 822
subcommands for, 812–836
tags for, 807, 835
uncommitted changes, working with, 834
user name and email for, 808
working tree for, 806
 cleaning, 818
 removing files from, 833
github.com service, 811
gitk command, git, 823
Gitorious tool, 811
Gitosis tool, 811
glob command, ftp, 150
global command, ex, 700
GLOBIGNORE built-in variable, 609
GNOME desktop, 7
GNU Arch system, 753
GNU awk (see gawk language)
GNU Emacs Pocket Reference (Cameron), 654
GNU Emacs (see Emacs editor)
GNU gettext, 736
GNU makefiles, creating from template files, 55
GNU Privacy Guard application, 173–177
GNU project, 3
GNU sed editor (see sed editor)
GNU utilities documentation, website for, xvii
GNU/Linux, 1
 (see also Linux)
good (go) command, mailx, 270
gpasswd command, 173
gpg command, 173–177
gpgsplit command, 177
gpgv command, 178
GPL (General Public License), 6
gpm command, 178–180
gprof command, 181–182
GRand Unified Bootloader (see GRUB)
graph character class, 600, 656
graphical desktop environments for Linux, 4
graphical tools for Linux, 4
greater than (>) operator, 620
grep command, 182–185, 657
grep command, git, 823
groff command, 185–187
group ID, displaying information about, 204
groupadd command, 187
groupdel command, 188
groupinfo command, yum, 548
groupinstall command, yum, 548
grouplist command, yum, 548
groupmod command, 188
groupremove command, yum, 548
groups
 changing for files, 78
 changing for user, 308
 creating, 187
 creating shadowed groups from, 189
 listing for users, 188
 managing, 173
 modifying, 188
 removing, 188
 removing corrupt and duplicate entries in, 188
 space allowed on disk for, 344
GROUPS built-in variable, 609
groups command, 188
grpck command, 188
grpconv command, 189
grub command, GRUB, 519, 523

- GRUB (GRand Unified Bootloader), 504, 516–536
boot CD for, 519
boot floppy for, 519
commands for, 525–536
configuration file, 521–523, 539, 541
device map, 518
installation stages, 517
installing, 518–521
menu interface, 523
naming conventions, 517
shell, 523–525
grub-install shell script, 518, 520
grub-md5-crypt command, GRUB, 522
gs command, 189
gsub() function, gawk, 741
gunzip command, 190
gxpath command, 190
gzip command, 191
- ## H
- H command, sed, 716, 720
h command, sed, 716, 720
H hostname escape sequence, 612
\h hostname escape sequence, 612
halt command, 192
halt command, GRUB, 532
hard drive
 parameters for, reading or setting, 192–195
 partitioning, 75, 136
hard links, 248
hardware
 commands for, list of, 15
 performance improvements on, 3
 supported by Linux, 3
hardware clock, reading or setting, 200
hash command, Bash, 633
hash command, ftp, 150
hash sign (#)
 preceding comments, 619
 preceding comments in gawk, 738
 preceding comments in LILO configuration file, 507
 preceding comments in sed scripts, 717
hdparm command, 192–195
head command, 196
header for packages, 552
headers (h) command, mailx, 270
headers+ (h+) command, mailx, 270
headers- (h-) command, mailx, 270
Hello Protocol, 25
help command, Bash, 634
help command (C-h), Emacs, 663
help command, ftp, 150
help command, git, 812
help command, GRUB, 532
help command, ip, 212
help command, mailx, 270
help command, svn, 776
help command, svnadmin, 796
help command, svnlook, 801
help command, xm, 847
help command, yum, 549
help commands, Emacs, 663, 668, 669
hexadecimal constants, gawk, 734
hexadecimal format
 displaying files in, 196
 printing host ID in, 198
hexdump command, 196
HFS filesystem, generating, 165–173, 292
hiddenmenu command, GRUB, 522
hide command, ex, 700
hide command, GRUB, 527
histchars built-in variable, 611
HISTCMD built-in variable, 607, 609
HISTCONTROL built-in variable, 610
HISTFILE built-in variable, 610
HISTFILESIZE built-in variable, 610
HISTIGNORE built-in variable, 610
history command, Bash, 634
history command, svnlook, 801
history expansion character (!), 614
history of commands, 614–617, 631, 634
HISTSIZE built-in variable, 610
HISTTIMEFORMAT built-in variable, 610
hoclock command, 200
hold (ho) command, mailx, 270
hold space, sed, 712
HOME built-in variable, 610
home directories, determining, 599
host command, 26, 197
host ID, printing in hexadecimal format, 198
HOSTFILE built-in variable, 610
hostid command, 198
HOSTNAME built-in variable, 609
hostname command, 198

hosts
 information about, commands
 for, 15
 name of, displaying or setting, 198
 printing information about, 197

HOSTTYPE built-in variable, 609

hotcopy command, svnadmin, 796

htdigest command, 199

hunspell command, 199

Hunt, Craig (TCP/IP Network Administration), 23

hypervisor, 838, 839
 (see also virtualization tools)

hyphen (-)
 previous command, mailx, 269
 subtraction operator, 732
 unary minus operator, 732

hyphen, double (--) decrement operator, 732

hyphen, equals sign (=) assignment operator, 732

hyphen(s) (- or --) preceding command options, 33

|

i command, sed, 716, 720

IANA (Internet Assigned Numbers Authority), 25

IBM Journaled Filesystem (see JFS)

ICANN (Internet Corporation for Assigned Names and Numbers), 27

icedax command, 200–203

ICMP (Internet Control Message Protocol), 23

iconv command, 203

id command, 204

idle command, ftp, 150

IEEE 1394 packets, dumping, 124

if command, Bash, 635

if command, gawk, 742

ifconfig command, 27, 204–206

ifconfig command, GRUB, 527

isempty command, logrotate, 254

IFS built-in variable, 610

igawk (inclusion gawk), 736

ignore command, mailx, 271

IGNOREEOF built-in variable, 610

image command, ftp, 149

imap command, mailx, 271

IMAP (Interactive Mail Access Protocol)
 server daemon, 207

imapd command, 207

import command, svn, 776

imsprobe command, GRUB, 532

in operator, for array membership, 732

include command, logrotate, 254

@include directive, gawk, 736

inclusion gawk (igawk), 736

InCrEdible Digital Audio eXtractor (see icedax command)

increment operator (++), 732

indentation commands, Emacs, 667

index, for Git, 806
 adding or updating files in, 812
 removing files from, 833

index() function, gawk, 742

inetd command, 207

info command, 207

info command, gdb, 164

info command, svn, 777

info command, svnlk, 801

info command, xm, 847

info command, yum, 549

init command, git, 824

init daemon, Upstart, 208

init program, 19

initctl command, 208

initrd command, GRUB, 522, 533, 541

inode, printing contents of, 413–416

input command, ftp, 151

INPUTRC built-in variable, 610

insert command, ex, 701

insert commands, vi, 686

insert mode, vi, 678, 681

insmod command, 209

install command, 210

install command, GRUB, 533

install command, yum, 549

installation, commands for, 16

int() function, gawk, 742

integer comparisons, testing for, 649

Intel Multiprocessor Specification, 532

Interactive Mail Access Protocol (IMAP)
 server daemon, 207

interfaces, assigning to a device, 305

internationalization, gawk, 736

Internet Assigned Numbers Authority (IANA), 25

Internet Control Message Protocol (ICMP), 23

Internet Corporation for Assigned Names and Numbers (ICANN), 27
Internet protocols, 21
Internet services daemon, 207
Internetworking with TCP/IP (Comer), 23
interprocess communication (see IPC)
I/O latency for elevator algorithm, 130
I/O scheduling class, setting, 211
ionice command, 211
ioprobe command, GRUB, 534
IP address
 contact for, finding, 483
 definition of, 23–25
ip command, 27, 211–215
IP ports, mapping to RPC program numbers, 329
IPC (interprocess communication)
 removing message queues, 215
 removing semaphore arrays, 215
 removing shared memory segments, 215
 reporting on, 216
ipcrm command, 215
ipcs command, 216
iptables command, 29, 216–222
iptables (see netfilter)
iptables-restore command, 29, 222
iptables-save command, 29, 223
IPv4 addresses, 23
IPv6 addresses, 24
ISO9660 images
 displaying content of, 223
 displaying information about, 223
 displaying length of, 224
 verifying integrity of, 224
ISO9660/Joliet/HFS filesystem, generating, 165–173, 292
isodump command, 223
isoinfo command, 223
isosize command, 224
isofvfy command, 224
ispell command, 224

J

\j jobs escape sequence, 612
JFS (Journalized Filesystem), 16
job control, 617
 Emacs commands for, 667
 listing jobs, 635

moving job to foreground, 631
removing job from job list, 628
terminating jobs, 636
waiting for background jobs to finish, 653
jobs command, Bash, 635
join command, 226
join command, ex, 701
Journalized Filesystem (see JFS)
jump command, gdb, 164
jumps command, ex, 701
junk (j) command, mailx, 271

K

k command, ex, 701
kbd_mode command, 227
kbdrate command, 227
kernel, 1
 commands for, list of, 17
 loading modules into, 209
 messages, displaying, prioritizing, and logging, 229
 parameters, displaying or modifying, 428
kernel command, GRUB, 522, 523, 534
Kernel-based Virtual Machine (see KVM hypervisor)
keyboard
 assigning keycode events to scancodes, 393
 driver translation tables, printing information about, 124
 mode for, printing or setting, 227
 printing keycodes, scancodes, or ASCII codes of keys pressed, 399
 rate, setting, 227
 Unicode mode for, 463
keycodes
 mapping table for, 173
 printing, 399
keymaps, loading, 248
keys, authentication
 gathering public and private keys from multiple, 411
 generating, managing, and converting, 410
 holding private keys for public authentication, 410
keyword substitutions, in source code management system, 750

- kill and yank (cut and paste),
 Emacs, 662
- kill command, 227
- kill command, Bash, 636
- kill command, gdb, 164
- killall command, 228
- klogd command, 229
- Korn shell, 596
- ksh shell, 596
- KVM hypervisor, 849
 building guest systems, 850
 changing virtual machine
 configurations, 844
- device interfaces for, 843
- graphic interface for, 842
- installing, 849
- libvirt used with (see libvirt
 virtualization API)
- managing virtual systems, 842
- QEMU used with, 849
- website for, 837
-
- ## L
- \l basename escape sequence, 612
- l command, sed, 716, 721
- \L escape character, pattern
 matching, 657
- \l escape character, pattern
 matching, 657
- LANG built-in variable, 610
- last command, 230
- lastaction command, logrotate, 254
- lastb command, 230
- lastlog command, 231
- latency in elevator algorithm,
 setting, 130
- LC_ALL built-in variable, 610
- LC_COLLATE built-in variable, 610
- LC_CTYPE built-in variable, 610
- lcd command, ftp, 150
- LC_MESSAGES built-in variable, 610
- LC_NUMERIC built-in variable, 610
- ld command, 231–234
- ldconfig command, 234
- ldd command, 235
- Learning GNU Emacs (Cameron
 et al.), 654, 662
- Learning Red Hat Enterprise Linux and
 Fedora (McCarty), 5
- Learning the bash Shell (Newham), 597
- Learning the Unix Operating System
 (Peek; Todino-Gonguet;
 Strang), xvii
- LED flag settings, displaying or
 changing, 393
- left angle bracket (<) less than
 operator, 620, 732
- left angle bracket, equals sign (=<) less
 than or equal operator, 732
- left command, ex, 701
- Lempel-Ziv coding (LZ77), compressing
 files using, 191
- length() function, gawk, 742
- less command, 235–244
- less than operator (<), 620
- lesskey command, 242–244
- let command, Bash, 613, 636
- lexical analysis program,
 generating, 143
- lftp command, 244–247
- lftpget command, 247
- libraries
 examining and updating, 234
 shared, displaying program
 requirements for, 235
- libvirt virtualization API, 839, 850–861
 connection URIs for hypervisor, 850
 networking for, 851
 remote GUI control for, 851
 virsh commands, 852–856
 virt-clone command, 856
 virt-image command, 857
 virt-install command, 858–861
 virt-manager command, 861
 virt-viewer command, 861
 website for, 837, 850
 XML configuration files for, 850
- licenses for Linux, 6
 (see also GPL)
- lilo command, 506, 514–516
- LILO (Linux Loader), 504, 506–516
 boot errors, 516
 configuration file, 506, 507–514,
 539, 541
 global options, 508–511
 image options, 512
 kernel options, 513
 rescue CD for, making, 506
- line-edit mode, Bash, 614
- LINENO built-in variable, 607
- LINES built-in variable, 610

link command, 247
link set command, ip, 212
link show command, ip, 212
Linux
 benefits of, 2
 commands, 3, 5
 compared to Unix, 1
 as desktop system, 3
 distributions of, 3
 embedded versions of, 3
 hardware supported by, 3
 history of, 1
 installing, book about, xvii
 licenses for, 6
 performance improvements by, 3
 source code for, 6
 starting, 19
 stopping, 19
 support for, xviii, 3
 (see also shells)

Linux Documentation Project
 website, xvii

Linux Gazette website, xvii

Linux Insider website, xvii

Linux iptables Pocket Reference
 (Purdy), 29

Linux Journal, xviii

Linux Loader (see LILO)

Linux Magazine, xviii

Linux Network Administrator's Guide
 (Bautts; Dawson; Purdy), xvii,
 23, 29

Linux Second Extended Filesystem (see
 ext2 filesystem)

Linux Security website, xvii

Linux Third Extended Filesystem (see
 ext3 filesystem)

Linux Today website, xvii

Linux Weekly News website, xvii

list command, ex, 701
list command, gdb, 164
list command, mailx, 271
list command, svn, 778
list command, xm, 847
list command, yum, 549
list-dblogs command, svnadmin, 796
list-unused-dblogs command,
 svnadmin, 796

ln command, 247

load command, svnadmin, 796
loaded modules, listing, 264

loadkeys command, 248
local command, Bash, 636
local variables, Bash, 636
locale command, 249
locale settings, printing, 249
localinstall command, yum, 549
localizing gawk programs, 736
localupdate command, yum, 550
locate command, 250
lock command, GRUB, 522, 534
lock command, svn, 779
lock command, svnlook, 802
lockfile command, 251
log command, git, 824
log command, svn, 779
log command, svnlook, 802
log command, xm, 847
log() function, gawk, 742
log message, for source code
 management system, 750

logger command, 252

logging in
 to remote host, 360
 to remote system, securely, 406–409
 to tty port, 38

logging (see system logging)

logical AND operator (`&&`), 620

logical negation operator (`!`), 732

logical operators, 620

logical OR operator (`||`), 620

login shell
 changing, 84
 exiting, 636

logins
 bad attempts at, displaying, 230
 for system accounts, displaying times
 of, 231
 most recent, printing, 230
 printing list of, 467
 printing summary of, 471

logout command, Bash, 636

logrotate command, 253–256

look command, 256

loop devices, setting up and
 controlling, 256

loops
 exiting from, 622
 for command, 632
 skipping remaining commands in, 626
 until command, 653
 while command, 653

losetup command, 256
lost+found directory, creating, 292
lower character class, 600, 656
lpadmin command, 256
lpinfo command, 258
lpmove command, 258
lpq command, 258
lpr command, 258
lprm command, 259
lpstat command, 260
ls command, 118, 260–263
ls command, ftp, 150
lsattr command, 263
lshift() function, gawk, 742
lslocks command, svnadmin, 797
lsmod command, 264
lspci command, 263
lstxns command, synadmin, 797
lsusb command, 264
LZ77 (Lempel-Ziv coding), compressing files using, 191

M

m4 command, 264
m4 macros
 combining into one file, 34
 generating configuration script from, 53
 generating template of #define statements from, 54
MAC address, for virtualization, 844
macdef command, ftp, 150
MACHTYPE built-in variable, 609
macro commands, Emacs, 668
macro commands, vi, 691
macro processor, 264
macros
 executing in ftp, 149
 in package spec file, 553
 listing for source file, 131
magnetic tape drive, controlling, 300
mail
 aliases database, rebuilding, 308
 aliases, printing, 334
 commands for, list of, 16
 reading or sending, 265, 266–275
 remote mail, receiving, 362
 sendmail queue, listing messages in, 265
 sendmail statistics, reporting, 266

MAIL built-in variable, 610
mail command, 265
mail command, logrotate, 254
mail (m) command, mailx, 271
mail transfer agent (MTA), 385–392
mailbox format, filtering input into, 145–147
MAILCHECK built-in variable, 610
mailfirst command, logrotate, 254
maillast command, logrotate, 254
MAILPATH built-in variable, 610
mailq command, 265
mailstats command, 266
mailx command, 266–275
make command, 275–281
makeactive command, GRUB, 534
makecache command, yum, 550
makedbm command, 281
makefiles
 creating from template files, 55
 executing, 275–281
makemap command, 282
man command, 283
Managing Projects with GNU Make (Mecklenburg), 275
manpath command, 285
manual pages
 displaying, 283
 path to, determining, 285
 searching for keyword, 480
 searching short descriptions for strings, 46
map command, ex, 701
map command, GRUB, 534
mapfile command, Bash, 637
maps, NIS, 31
mark command, ex, 702
mark location, Emacs, 662
marks command, ex, 702
masquerading, 29
master boot record (see MBR)
Mastering Regular Expressions (Friedl), 654
match() function, gawk, 742
mbox command, mailx, 271
MBR (master boot record), 505
 installing GRUB boot loader on, 518, 520, 521
 installing LILO boot loader on, 506
 restoring, 506

McCarty, Bill (*Learning Red Hat Enterprise Linux and Fedora*), 5
MD5 checksums for files, computing or checking, 285
md5crypt command, GRUB, 522, 534
md5sum command, 285
mdelete command, ftp, 151
mdir command, ftp, 151
Mecklenburg, Robert (*Managing Projects with GNU Make*), 275
media (see CDs; DVDs)
memory maps, displaying for a process, 329
memory usage, statistics about, 147
menu items, providing to user, 640
merge command, 286
merge command, git, 826
merge command, svn, 781
mergeinfo command, svn, 782
mesg command, 286
message queues, IPC, removing, 215
Meta key handling, displaying or setting, 394
Meta key (M-), Emacs, 662
metacharacters in pattern matching, 654–658
mget command, ftp, 151
MIDI files playing, 44
reading and writing to ALSA ports, 40
recording, 48
MIDI ports, connecting to route events, 35
mirror command, lftp, 246
missingok command, logrotate, 254
mixer, ALSA connecting to ALSA sound card(s), 40 controlling, 41
mkdir command, 287
mkdir command, ftp, 151
mkdir command, svn, 782
mkdosfs command, 287
mke2fs command, 288–290
mkexec command, ex, 702
mkfifo command, 291
mkfs command, 291
mkfs.ext3 command, 291
mkisofs command, 292
mkisofs command, GRUB, 519
mklost+found command, 292
mknod command, 292
mkrescue command, 506
mkswap command, 293
mktemp command, 293
mktime() function, gawk, 742
mls command, ftp, 151
mode command, ftp, 151
modinfo command, 294
modprobe command, 294
modtime command, ftp, 151
module command, GRUB, 534
modulenounzip command, GRUB, 534
modules loading, 294
loading into kernel, 209
printing information about, 294
unloading, 362
modulus operator (%), 732
monotone system, 754
monthly command, logrotate, 254
more command, 295
mount command, 30, 297–299
mountd command, 300
mountd daemon, 30
mouse server for Linux console, 178–180
move command, ex, 702
move command, svn, 783
move (mv) command, mailx, 271
movement commands, vi, 683–686
mt command, 300
MTA (mail transfer agent), 385–392
Multiboot Specification, 516
multiplication operator (*), 732
mv command, 302–303
mv command, git, 827
M-x Tab (command names list) command, Emacs, 663

N

N command, sed, 717, 721
n command, sed, 716, 721
\n escape character, pattern matching, 656, 657
\n newline escape sequence, 601, 715, 733, 612
name command, Bash, 637

name service, 26
named daemon, 26
named pipes (FIFO), making, 291
namei command, 304
nameif command, 305
nameserver, 26, 311
naming conventions
 GRUB, 517
 packages, 543
NAT (Network Address Translation), 842
nc command, 305–306
neigh add command, ip, 212
neigh chg command, ip, 213
neigh del command, ip, 213
neigh flush command, ip, 213
neigh list command, ip, 213
neigh repl command, ip, 213
Net:Bridge package, 843
netcat command (see nc command)
netfilter (iptables), 29
 books about, 29
 filtering rules, configuring, 216–222
netstat command, 306
network, 21–22
 administration, book about, xvii
 commands for, list of, 18
 configuring for virtualization, 842
 confirming response of remote hosts
 on, 327
 devices, manipulating, 211–215
 dumping headers and packets of
 network traffic, 437–439
 interface, configuring, 27, 204–206
 packet capture files for, reading and
 manipulating, 439
 reading and writing data
 across, 305–306
 serial lines, attaching as network
 interfaces, 403
 status of, displaying, 306
TCP/IP administration, 21, 22–28
updating files across, 368–374
Windows-based, Linux server
 supporting, 3
(see also NFS; NIS)
Network Address Translation (NAT), 842
Network File System (see NFS)

Network Information System (see NIS)
new command, ex, 702
newaliases command, 308
newer command, ftp, 151
newgrp command, 308
Newham, Cameron (*Learning the bash Shell*), 597
newline character, as word separator, 601
newsgroups, xviii
newusers command, 308
next command, ex, 702
next command, gawk, 743
next command, gdb, 164
next (n) command, mailx, 271
nextfile command, gawk, 743
NFS (Network File System), 29
 administering, 30
 commands for, list of, 22
 daemons, 30
 exporting filesystems, 30
 module, launching threads for, 308
 mounting filesystems, 30
 server, displaying information
 about, 400
 statistics on, printing, 309
nfsd command, 308
nfsd daemons, 30
nfstat command, 309
NIC names, finding contact for, 483
nice command, 309
NIS+, 31
NIS binder process, 494
NIS database
 creating, 495
 printing specific values in, 495
NIS dbm file, creating or dumping, 281
NIS map
 forcing propagation of, 497
 printing values of keys, 496
 transferring from server to local
 host, 500, 501
 version of, determining, 497
NIS (Network Information System), 31
 administering, 32
 client, setting up, 32
 commands for, list of, 22, 31
 domain, setting or displaying name
 of, 120

NIS (*continued*)
domains, 31
login password for, changing, 496
maps, 31
password file, modifying, 496
server, setting up, 32
servers, 31
services, checking configuration
for, 499
user accounts, 32
NIS server, 498
hostname of, determining, 500
setting for `ypbind` command, 499
`nlist` command, ftp, 151
`nm` command, 310
`nmap` command, ftp, 151
`nocompress` command, logrotate, 255
`nocopy` command, logrotate, 255
`nocopytruncate` command,
logrotate, 255
`nocreate` command, logrotate, 255
`nodelaycompress` command,
logrotate, 255
`nohlsearch` command, ex, 703
`nohup` command, 311
`nomail` command, logrotate, 255
`nomissingok` command, logrotate, 255
`noolddir` command, logrotate, 255
`nosharedscripts` command,
logrotate, 255
`NOT` operator (!) for commands, 602
`notifempty` command, logrotate, 255
`nroff` command, formatting output
from, 85, 86
`nslookup` command, 311
`nsupdate` command, 311
`ntrans` command, ftp, 151
`null` command (:), 620
`number` command, ex, 703
numbers, printing in sequence, 393
`nvi` editor, 677

object files
combining into executable object
module, 231–234
copying, 313–317
displaying information
about, 317–319
generating from assembly source
files, 50
removing symbols from, 419
size of, printing, 402
octal constants, gawk, 734
octal format, displaying files in, 196
`od` command, 319–322
`olddir` command, logrotate, 255
`OLDPWD` built-in variable, 607
online documentation, displaying, 207
`only` command, ex, 703
`open` command, ex, 703
`open` command, ftp, 152
OpenPGP-format messages, splitting
into packets, 177
OpenPGP-signed files, checking
signature of, 178
`openvt` command, 322
operating system, printing information
about, 461
operators
arithmetic, 613
comparison, 620
gawk, 732
logical, 620
`OPTARG` built-in variable, 607
`OPTERR` built-in variable, 610
`OPTIND` built-in variable, 607
`or()` function, gawk, 743
OR operator (||)
for commands, 602
for expressions, 620, 732
`OSTYPE` built-in variable, 609
output redirection, gawk, 746
ownership of files, changing, 82

0

\o octal escape sequence, sed, 715
`objcopy` command, 313–317
`objdump` command, 317–319

P

`P` command, sed, 716, 717, 722
`p` command, sed, 722
package managers, 542–545
(see also APT; Debian Package
Manager; RPM; yum program)

package-cleanup plugin, yum, 551
packages, 542–595
 Debian, 565–595
 archive of, 566
 cache for, managing, 569–571
 files used by, 565
 flags for, 567
 generating, 574
 managing, 575–590, 592–594
 naming conventions for, 565
 priorities of, 566
 querying, 590
 sorting records in, 579
 sources for, adding, 571
 splitting, 591
 states of, 567
 dependencies for, 543
 naming conventions for, 543
RPM, 545–565
 archive of, 552
 building, 564–565
 header for, 552
 managing with rpm
 command, 553–563
 managing with yum, 545–552
 repository of, 545
 signature verification for, 552
 spec file for, 552
 signature verification for, 543
 user privileges required for, 543
packets
 IEEE394, dumping, 124
 tracing path to host taken by, 454, 455
pager command, GRUB, 527
paragraph commands, Emacs, 665
paravirtualization, 839, 846, 864
parentheses (...)
 enclosing group of commands, 601, 602
 enclosing group of regular expressions, 656
partnew command, GRUB, 527
parttype command, GRUB, 528
passwd command, 322
password command, GRUB, 522, 528
passwords
 changing, 83
 created shadowed entries for, 344
creating or changing, 322
 expirations for, changing, 76
 information stored with,
 changing, 78
removing corrupt or duplicate entries, 343
paste command, 323
paste (yank), Emacs, 662
patch command, 324–326
patches, applying, 324–326
PATH built-in variable, 610
pathchk command, 326
paths
 checking validity of, 326
 following to terminal point, 304
 for current working directory,
 printing, 344
 printing only leading directory components from, 119
 removing leading directory components from, 59
pattern addressing, sed, 714–715
pattern matching, 155, 654–660
 character classes in, 656
 escape sequences in, 656
 metacharacters in, 654–658
 replacement patterns, 656
 search patterns, 655
 searching and replacing, examples of, 659
 searching, examples of, 658
 searching files with extended regular expressions, 128
 searching files with regular expressions, 182–185
pattern space, sed, 712
patterns, gawk, 729, 730
Patwardhan, Nathan (Perl in a Nutshell), 654
pause command, GRUB, 535
pause command, xm, 848
pccardctl command, 327
PCI (Peripheral Component Interconnect) devices, listing, 263
PCMCIA sockets, controlling, 327
PDF (Portable Document Format)
 language, interpreter for, 189

Peek, Jerry (*Learning the Unix Operating System*), xvii
percent sign (%)
 modulus operator, 732
 preceding job ID in commands, 617
 reuse previous replacement
 pattern, 657
percent sign, equals sign (=%)
 assignment operator, 732
performance, Linux improving, 3
Peripheral Component Interconnect (PCI) devices, listing, 263
Perl 5 Pocket Reference (Vromans), 654
Perl in a Nutshell (Patwardhan et al.), 654
permuted indexes, 342
pgawk (profiling gawk), 735
pidof command, 327
Pilato, C. Michael (*Version Control with Subversion*), 755
ping command, 327
pipe character (|), 601, 602
pipe (pi) command, mailx, 271
pipes
 named pipes (FIFO), making, 291
 negating result of, 619
PIPESTATUS built-in variable, 609
plugins for yum program, 551
plus, equals sign (+=) assignment operator, 732
plus sign (+)
 addition operator, 732
 match one or more instances, 656
 unary plus operator, 732
plus sign, double (++) increment operator, 732
plus sign, parentheses (+()) in filename, 600
pmap command, 329
PNM (Portable aNyMap) format, writing images to, 376
point (cursor) location, Emacs, 662
Point-to-Point Protocol (PPP), 23, 28, 330–332
popd command, Bash, 637
Portable aNyMap (PNM) format, writing images to, 376
Portable Document Format (PDF) language, interpreter for, 189
portmap command, 329
portmap daemon, 30
POSIX notation, in Bash, 600, 602
POSIXLY_CORRECT built-in variable, 610
postrotate command, logrotate, 255
PostScript language, interpreter for, 189
poweroff command, 330
PPID built-in variable, 607
PPP (Point-to-Point Protocol), 23, 28, 330–332
pppd command, 330–332
pppd daemon, 28
pr command, 332–334
#pragma implementation directive, 163
#pragma interface directive, 162
praliases command, 334
prerotate command, logrotate, 255
preserve command, ex, 703
preserve (pre) command, mailx, 271
previous command, ex, 703
prime factors, calculating, 135
print character class, 600, 656
print command, ex, 703
print command, gawk, 743, 746
print command, gdb, 164
print jobs
 cancelling, 259
 moving to a new destination, 258
Print (P) command, mailx, 271
print (p) command, mailx, 271
print queues
 accepting, 34
 CUPS, configuring, 256
 displaying status of, 260
 rejecting, 354
print scheduler, starting, 101
print spool queue, displaying information about, 258
printenv command, 334
printer drivers, printing information about, 258
printers
 enabling, 130
 printing information about, 258
 tuning device parameters for, 460
printf command, 334
printf command, Bash, 638
printf() function, gawk, 743, 746
printing, commands for, 9, 18, 258
priority of processes, controlling, 355
privileges required for packages, 543
procedures, gawk, 729, 730

process IDs
determining for specific files or filesystems, 154
displaying for programs, 327

processes
active, reporting on, 335–341
managing, commands for, 19
memory maps for, displaying, 329
most CPU-intensive, displaying information about, 448–452
priority of, controlling, 355
real-time scheduling for, changing, 83
terminating, 227, 636
terminating all processes, 228

processor affinity mask, retrieving or setting, 437

profile data for an object, displaying, 181–182

profiling, gawk, 735

program addresses, translating to filenames and line numbers, 37

programmable completion, 615–617, 624–626

programming
commands for, list of, 9
maintenance of, commands for, 10
(see also gawk language; source code management systems)

prompt command, ftp, 152

prompt strings, 612

PROMPT_COMMAND built-in variable, 610

propdel command, svn, 784–786

propedit command, svn, 784

properties, Subversion, 757

propget command, svn, 785

propget command, svnlook, 802

proplist command, svn, 785

proplist command, svnlook, 802

propset command, svn, 786

protocols, Internet, 21

provides command, yum, 550

proxy command, ftp, 152

ps command, 335–341

PS1 built-in variable, 610, 612

PS2 build-in variable, 612

PS2 built-in variable, 610

PS3 built-in variable, 610

PS4 built-in variable, 611, 612

pseudonyms (links) for files, 247

ptx command, 342

ptype command, gdb, 164

pull command, git, 827

punct character class, 600, 656

Purdy, Gregor
Linux iptables Pocket Reference, 29
Linux Network Administrator's Guide, xvii, 23, 29

push command, git, 828

pushd command, Bash, 638

put command, ex, 703

put command, ftp, 152

pwck command, 343

pwconv command, 344

PWD built-in variable, 607

pwd command, 344

pwd command, Bash, 638

pwd command, ftp, 152

pwd command, gdb, 164

python-vm-builder package, 850

Q

Q command, sed, 716, 722

q command, sed, 716, 722

qall command, ex, 703

QEMU, combining with KVM hypervisor, 849

qemu-img tool, 845

question mark (?)
in filename, 599
help command, mailx, 269
match zero or one instance, 656

question mark, colon (:) conditional expression, 732

question mark, parentheses (?) in filename, 600

quit command, ex, 704

quit command, ftp, 149

quit command, gdb, 165

quit command, GRUB, 535

quit (q) command, mailx, 271

quota command, 344

quotacheck command, 345

quotaoff command, 347

quotaon command, 346

quotas
exceeding limit, sending messages to users regarding, 472
setting, 394

quotastats command, 347
quote command, ftp, 152
quoting special characters in Bash, 600

R

\r carriage return escape sequence, 612, 715, 733
R command, sed, 716, 723
r command, sed, 716, 723
rand() function, gawk, 743
RANDOM built-in variable, 608
ranlib command, 347
rarp command, GRUB, 528
RARP (Reverse Address Resolution Protocol), 23
rcp command, 347
RCS (Revision Control System), 752
rdate command, 348
rdist command, 348–350
rdistd command, 350
read command, Bash, 639
read command, ex, 704
read command, GRUB, 535
readcd command, 350
readelf command, 351
readline library, managing, 621
readlink command, 352
readom command, 352
readonly command, Bash, 640
real-time scheduling for processes, changing, 83
rebase command, git, 829
reboot command, 354
reboot command, GRUB, 535
recover command, ex, 704
recover command, svnadmin, 797
Red Hat Package Manager (see RPM)
Red Hat Virtual Machine
 Manager, 850–851
redirection of commands, 602–604
redo command, ex, 704
reflog command, git, 830
reget command, ftp, 152
region commands, Emacs, 665
Regular Expression Pocket Reference
 (Stubblebine), 654
regular expressions, 654
 (see also pattern matching)
reinstall command, yum, 550
reject command, 354
relational operators, gawk, 732
release of package, 543
remainder operator (%), 732
remote command, git, 831
remote file distribution, 348–350
remote host, logging in to, 360
remote hosts, confirming responses of, 327
Remote Procedural Call (RPC) language, generating C code from, 366
Remote Procedure Call (RPC), 32
remote procedure call (RPC) activity, statistics on, 309
remote shell server, 367
remote systems, accessing, 441
remotehelp command, ftp, 152
remotestatus command, ftp, 152
remove command, yum, 550
remove (rem) command, mailx, 271
rename command, 354
rename command, ftp, 152
rename (ren) command, mailx, 271
renice command, 355
replacement patterns, metacharacters for, 656
replacing, with pattern matching, 659
REPLY built-in variable, 608
Reply (R) command, mailx, 271
reply (r) command, mailx, 271
replyall command, mailx, 271
repoclosure plugin, yum, 551
repodiff plugin, yum, 551
repo-graph plugin, yum, 551
repolist command, yum, 550
repomanage plugin, yum, 551
repo.or.cz hosting service, 811
repoquery plugin, yum, 551
repo-rss plugin, yum, 551
repositories, for Git, 805, 806
 cloning, 818
 creating, 810, 824
 creating archive from, 813
 packing, 822
repository, for source code management system, 749, 755
repository, of RPM packages, 545
reposync plugin, yum, 552
repotrack plugin, yum, 552
repquota command, 355
reset command, 356
reset command, ftp, 152
reset command, git, 832

resize command, ex, 704
resize2fs command, 356
resolve command, svn, 787
resolved command, svn, 787
resolvedep command, yum, 550
resolver, 26
resource limits, printing, 652
resources (see books and publications; website resources)
respond command, mailx, 271
restart command, ftp, 152
restore command, 357–359
restricted shells, 619
retain command, mailx, 271
return command, Bash, 640
return command, gawk, 744
rev command, 359
Reverse Address Resolution Protocol (RARP), 23
reverse-search command, gdb, 165
revert command, git, 833
revert command, svn, 788
Revision Control System (RCS), 752
revision of package, 543
rewind command, ex, 704
rexec command, 359
rexecd command, 360
right angle bracket (>)
 greater than operator, 620, 732
 output redirection, 746
right angle bracket, double (>>) append output, 746
right angle bracket, equals sign (>=)
 greater than or equal operator, 732
right angle brackets (> or >>) in cat command, 68
right command, ex, 704
RIP (Routing Information Protocol), 25
rlogin command, 360
rlogind command, 360
rm command, 361
rm command, git, 833
rmail command, 362
rmdir command, 362
rmdir command, ftp, 152
rmlocks command, svnadmin, 797
rmmod command, 362
rmtxns command, svnadmin, 797
rndc command, 363
Robbins, Arnold (Classic Shell Scripting), 597
root command, GRUB, 535
root directory, changing, 83
rootnoverify command, GRUB, 535
rotate command, logrotate, 255
route add command, ip, 213
route chg command, ip, 213
route command, 364–365
route del command, ip, 213
route events, connecting MIDI ports to, 35
route list command, ip, 213
route repl command, ip, 213
routed daemon, 26
routing, controlling, 211–215
routing daemons, 25
Routing Information Protocol (RIP), 25
routing tables, 26, 364–365
RPC (Remote Procedure Call), 32
 activity of, statistics on, 309
 generating C code from, 366
 program numbers
 mapping to IP ports, 329
 mapping to universal addresses, 365
 reporting information for programs, 366
rpcbind command, 365
rpcbind daemon, 30
rpcgen command, 366
rpcinfo command, 366
rpm command, 367, 552–563
RPM (Red Hat Package Manager), 544, 552–565
 database rebuild options for, 562
 examples of, 563
 freshen options for, 555
 install options for, 554
 naming conventions for, 543
 options for, 554–563
 package-query options for, 559
 package-selection options for, 557–559
 query options for, 557
 rpm command syntax for, 553
 rpmbuild command syntax and options for, 564–565
 signature-check options for, 562
 uninstall options for, 560
 upgrade options for, 554
 verify options for, 560
rpmbuild command, RPM, 552, 564–565

RSA identities, adding to authentication agent, 409
rsh command, 367
rshd command, 367
rshift() function, gawk, 744
rsync command, 368–374
rsyslogd command, 375
run command, gdb, 165
runique command, ftp, 152
runlevel command, 376
runlevels, 19
 changing, 440
 specifying services to be run at, 79
Running Linux (Dalheimer; Welsh), xvii, 4

S

s command, sed, 716, 723
\\$ shell escape sequence, 612
sandbox, for source code management system, 750, 756
SANE (Scanner Access Now Easy), 376
sane-find-scanner command, 376
save (s) command, mailx, 272
savedefault command, GRUB, 535
saveignore command, mailx, 272
saveretain command, mailx, 272
saving files, vi, 688
sbnext command, ex, 705
sbuffer command, ex, 705
scancodes, printing, 399
scancode-to-keycode mapping table, 173
scanimage command, 376
Scanner Access Now Easy (SANE)
 detecting scanners, 376
scanners
 detecting by SANE, 376
 reading images from, 376
SCCS (Source Code Control System), 752
scheduling commands, 20
scp command, 378
screen command, 379–384
script command, 384
scripts in package spec file, 553
SCSI scanners, detecting by SANE, 376
sdiff command, 384
search command, gdb, 165
search command, yum, 550
search commands, Emacs, 665

search patterns, metacharacters for, 655
searching
 commands for, list of, 10
 with pattern matching, 658, 659
Second Extended Filesystem (see ext2 filesystem)
SECONDS built-in variable, 608
secure copying of files, 378
secure login to remote system, 406–409
security
 commands for, list of, 18
 encryption with GNU Privacy Guard, 173–177
website about, xvii
(see also authentication; signature verification)
sed command, 385
sed editor, 711–725
 branching commands, 717
 command syntax for, 713
 editing commands, 716
 hold space for, 712
 input for, 711, 712, 716, 717
 input/output commands, 716
 invoking, 712
 line information commands, 716
 multiline input commands, 717
 options for, 712
 pattern addressing, 714–715
 pattern matching metacharacters
 supported by, 657
 pattern space for, 712
 yanking and putting commands, 716
seen command, mailx, 272
select command, Bash, 640
semaphore arrays, IPC, removing, 215
semaphore files, creating, 251
semicolon (;) command separator, 601, 602
sendmail command, 385–392
 database maps for, creating, 282
 mail aliases for, printing, 334
sendport command, ftp, 152
sensor chips, displaying readings and configuring, 392
sensors command, 392
seq command, 393
serial command, GRUB, 528
Serial Line IP (SLIP), 28
serial lines, attaching as network interfaces, 403

servers, NIS, 31, 32
services, managing runlevels for, 79
set command, Bash, 641–643
set command, ex, 705
set command, lftp, 247
:set command, vi, 692–696
set (se) command, mailx, 272
set variable command, gdb, 165
setkey command, GRUB, 529
setkeycodes command, 393
setleds command, 393
setlog command, svnadmin, 798
setmetamode command, 394
setquota command, 394
setrevprop command, svnadmin, 798
setsid command, 395
setterm command, 395–397
setup command, GRUB, 521, 535
setuid command, svnadmin, 798
sftp command, 398
SGI Extensible Filesystem (see XFS)
sh shell, 399, 596, 597
SHA1 checksums, computing or
 checking, 399
sha1sum command, 399
shadow files
 creating or updating from
 passwords, 344
 removing corrupt and duplicate
 entries in, 188, 343
shadowed groups, creating, 189
shared memory segments, IPC,
 removing, 215
sharedscripts command, logrotate, 255
SHELL built-in variable, 611
shell command, ex, 705
shell command, yum, 550
shell commands, Emacs, 667
shell conditions, testing for, 649
#!shell directive, 620
shell scripts
 comments in, 619
 exiting, 630
shell (sh) command, mailx, 272
SHLOOPTS built-in variable, 609
shells, 5, 596
 changing, 84
 creating with effective user ID, 425
 invoking in a script, 620
 programming, commands for, 11
 starting on remote host, 367
 (see also Bash shell)
shift command, Bash, 643
SHLVL built-in variable, 609
shotp command, Bash, 643–647
show (Sh) command, mailx, 272
showkey command, 399
showmount command, 30, 400
shred command, 400
shutdown command, 401
shutdown command, xm, 848
shutting down the system, 192, 330,
 354, 401
signal command, gdb, 165
signature verification
 with GNU Privacy Guard, 173–177
 of OpenPGP-signed files, 178
 of packages, 543, 552
Simple Mail Transport Protocol
 (SMTP), 23
Simple Network Management Protocol
 (SNMP), 23
sin() function, gawk, 744
single quotes ('...') enclosing special
 characters, 601
site command, ftp, 153
size command, 402
size command, ftp, 153
size command, logrotate, 255
size command, mailx, 272
slab cache information, displaying in
 real time, 402
slabtop command, 402
slash (/) division operator, 732
slash, equals sign (/=) assignment
 operator, 732
Slashdot website, xvii
slashes (/.../) enclosing pattern address
 in sed, 714
slattach command, 403
sleep command, 403
sleep for a specified time, 467
SLIP (Serial Line IP), 28
SMTP (Simple Mail Transport
 Protocol), 23
snexec command, ex, 705
SNMP (Simple Network Management
 Protocol), 23

sockets, gawk language, 735
sort command, 404
sorted files, joining, 226
sorting strings in a file, 457
sound cards, ALSA
 advanced configuration settings
 for, 39
 connecting to ALSA mixer, 40
sound files
 playing, 43
 recording, 48
 recording to CDs, 69–71, 75,
 484–488
 recording to DVDs, 125, 484–488
Source Code Control System
 (SCCS), 752
source code for Linux, 6
source code management
 systems, 749–754
 checking files in or out, 750
 client/server model for, 751
 Codeville, 753
 copy, modify, merge model for, 751
 CSSC, 754
 CVS, 752
 Git (see Git Version Control System)
 GNU Arch, 753
 locking model for, 751
 merging changes in, 750
 monotone system, 754
 RCS, 752
 SCCS, 752
 Subversion (see Subversion Version
 Control System)
source command, Bash, 647
source command, ex, 705
source command, mailx, 272
source files, listing function and macro
 names in, 100, 131
space character, as word separator, 601
space character class, 600, 656
spec file for package, 552
spell checkers, 199, 224
splashimage command, GRUB, 529
split command, 405
split command, ex, 705
split() function, gawk, 733, 744
sprevious command, ex, 706
sprintf() function, gawk, 744, 746
sqrt() function, gawk, 744
rand() function, gawk, 744

ssh command, 406–409
ssh-add command, 409
ssh-agent command, 410
sshd command, 412
ssh-keygen command, 410
ssh-keyscan command, 411
Stallman, Richard (founder of FSF), 7
standard input
 determining terminal connected
 to, 457
 reading a line from, 639
start command, logrotate, 255
starting the system, 19
startup files for Bash, 599
stash command, git, 834
stat command, 413–416
stated command, 416
status command, ftp, 153
status command, git, 834
status command, svn, 788–791
status line, vi, 683
step command, gdb, 165
stop command, ex, 706
stopping commands, Emacs, 665
stopping the system, 19
storage, commands for, 11
strace command, 416–418
Strang, John (*Learning the Unix
Operating System*), xvii
stream editor (see sed editor)
strftime() function, gawk, 744
string conditions, testing for, 648
strings
 completions for, generating, 624
 converting initial whitespace to
 tabs, 462
 printing with specific format, 334
 sorting, 457
 translating characters in, 453–454
 writing to standard output, 628
strings command, 418
strip command, 419
strtonum() function, gawk, 734, 744
struct command, ftp, 153
stty command, 420–425
Stubblebine, Tony (*Regular Expression
Pocket Reference*), 654
su command, 425
sub() function, gawk, 745
subdomains, 26
substitute command, ex, 706

substr() function, gawk, 745
subtraction operator (-), 732
Subversion version control system, 753, 755–804
 atomic commits, 756
 authors of files, showing, 767
 book about, 755
 branching in, 756, 757
 checking in (committing)
 changes, 756, 760, 771
 checking out files, 760, 769
 conflicts, handling of, 756, 787, 788
 copying files, 772
 copy-modify-merge model used
 by, 756
 differencing files, 757, 761, 774–775
 directories, creating, 782
 directory versioning, 756
 displaying file contents, 768
 help for, 776
 keywords for, 758
 locking files, 779, 793, 797
 logging for, 779
 merging files, 781, 782
 moving (renaming) files, 783
 networks used with, 756
 obtaining, 759
 projects
 adding files and directories
 to, 767
 creating, 759
 deleting files and directories
 from, 773
 grouping files into
 collections, 768
 properties (metadata) for, 756, 757, 784–786
 remote access for, 803
 repository for, 755, 794–803
 exporting from, 775
 importing to, 776
 listing contents of, 778
 sandbox for, 756
 svn command line, 761–766
 svn subcommands, 766–794
 svnadmin command, 794
 svnadmin subcommands, 795–799
 svnlook command, 799
 svnlook subcommands, 800–803
 svnserv command, 803
tags in, 756, 757
working copy
 cleaning, 770
 printing information about, 777
 status of, 788–791
 updating, 791–792, 793
sudo command, 426
sum command, 427
Sun Yellow Pages (YP), 31
sunique command, ftp, 153
superblock, printing information
 about, 124
supernetting, 24
superuser, executing commands as, 426
support for Linux, xviii, 3
suspend command, Bash, 647
suspend command, ex, 706
sview command, ex, 706
svn command, Subversion, 761–766
svn subcommands,
 Subversion, 766–794
 svnadmin command, Subversion, 794
 svnadmin subcommands,
 Subversion, 795–799
 svnlook command, Subversion, 799
 svnlook subcommands,
 Subversion, 800–803
 svnserv command, Subversion, 803
swapoff command, 427
swapon command, 427
swapspace, creating, 293
switch command, svn, 791–792
symbol table, printing in alphabetical
 order, 310
symbolic links, 248, 352
synaptic command, Debian, 544, 594
sync command, 428
sysctl command, 428
sysklogd command, 428
syslogd command, 429
system access commands, vi, 690
system administration, commands
 for, 14–20
system command, ftp, 153
system control messages,
 displaying, 119
system() function, gawk, 745
system integrity, commands for, 18
system load
 averages of, graphing, 447
 averages of, printing, 464

system logging
 adding entries to log, 252
 commands for, 20
 manipulating logfiles, 253–256
 system messages, 375, 428

system status
 commands for, list of, 11
 time running, printing, 464
 usage, printing summary of, 471

System V init (SysVinit), 19

sysime() function, gawk, 745

SysVinit (System V init), 19

T

t command, ex, 706

T command, sed, 717, 725

t command, sed, 717, 724

\t tab escape sequence, 601, 733

\T time escape sequence, 612

\t time escape sequence, 612

tabooext command, logrotate, 255

tabs
 as word separator, 601
 converting initial whitespace to, 462
 converting to spaces, 132

tac command, 429

tag command, ex, 707

tag command, git, 835

tags, 750
 Git, 807, 835
 Subversion, 756

tags command, ex, 707

tail command, 430

tailf command, 431

talk command, 431

tape drive, controlling, 300

tapes, ejecting, 128

tar command, 431–437

taskset command, 437

tbl command, formatting output
 from, 85, 86

tcpdump command, 437–439

TCP/IP, 22–28
 books about, 23
 commands for, list of, 21, 28
 configuring, 27
 gateways, 25
 IP addresses, 23–25
 name service, 26
 routing daemons, 25
 troubleshooting, 28

TCP/IP Network Administration (Hunt), 23

tcpsplice command, 439

tcsh shell, 596

tee command, 440

telinit command, 440

telnet command, 441

telnetd command, 442–443

temporary filename, generating, 293

tenex command, ftp, 153

TERM built-in variable, 611

terminal
 as standard input, determining, 457
 attributes, setting, 395–397
 initializing, 456
 I/O options, setting, 420–425
 resetting, 356

terminal command, GRUB, 529

terminal emulation, 379–384

terminal session, typescript of, 384

test command, 443

test command, Bash, 647–650

testload command, GRUB, 536

testvbe command, GRUB, 536

text patterns (see pattern matching)

text processing, commands for, 12

TEXTDOMAIN environment variable, gawk, 736

tftpserver command, GRUB, 530

Third Extended Filesystem (see ext3 filesystem)

thread (th) command, mailx, 272

threads, launching for NFS module, 308

tilde (~)
 in filename, 599
 match regular expression, 732
 replace regular expression command
 in ex, 710
 reuse previous replacement
 pattern, 657

tilde escape commands, mailx, 268

tilde, hyphen (~-) in filename, 599

tilde, plus sign (~+) in filename, 599

time command, 445

time command, Bash, 650

time (see date and time)

time system has been running,
 printing, 464

TIMEFORMAT built-in variable, 611

timeout command, GRUB, 522

times command, Bash, 650

title command, GRUB, 522
tload command, 447
TMOUT built-in variable, 611
tmpwatch command, 447
Todino-Gouquet, Grace (*Learning the Unix Operating System*), xvii
tolower() function, gawk, 745
top command, 448–452
top command, mailx, 272
top command, xm, 847
Torvalds, Linus (developer of Linux), 1
touch command, 452
touch command, mailx, 272
toupper() function, gawk, 746
tr command, 453–454
trace command, ftp, 153
tracepath command, 454
traceroute command, 455
translating characters in a string, 453–454
translation tables of keyboard driver, 124
transposition commands, Emacs, 665
trap command, Bash, 650
tree command, svnlook, 803
trigger scriptlets in package spec file, 553
troff command (see groff command)
true command, 456
true command, Bash, 651
tset command, 456
tsort command, 457
tty command, 457
tty port, logging in to, 38
tune2fs command, 457–460
tunelp command, 460
type command, Bash, 651
type command, ftp, 153
Type (T) command, mailx, 272
type (t) command, mailx, 272
typescript of terminal session, 384

U

\U escape character, pattern matching, 657
\u escape character, pattern matching, 657
\u username escape sequence, 612
ubuntu-virt-mgmt package, 849
ubuntu-virt-server package, 849
ubuntu-vm-builder package, 850

UDP (User Datagram Protocol), 23
UID built-in variable, 609
ul command, 460
ulimit command, Bash, 652
umask command, Bash, 652
umask command, ftp, 153
umount command, 30, 461
unabbreviate command, ex, 707
unalias command, Bash, 653
unalias command, mailx, 272
uname command, 461
unanswered command, mailx, 272
unary minus operator (-), 732
unary plus operator (+), 732
uncollapse (unc) command, mailx, 272
uncompresscmd command, logrotate, 256
undelete (u) command, mailx, 272
underscores, translating to underlining, 460
undisplay command, gdb, 165
undo command, ex, 707
undoing commands, Emacs, 665
unexpand command, 462
ungood command, mailx, 272
unhide command, ex, 707
unhide command, GRUB, 530
Unicode mode for keyboard, 463
unicode_start command, 463
unicode_stop command, 463
uniq command, 463
universal addresses, mapping RPC program numbers to, 365
Universal Serial Bus (USB) devices, listing, 264
Universal Unique Identifier (see UUID)
Unix, compared to Linux, 1
unjunk command, mailx, 272
unlink command, 464
unlock command, svn, 793
unmap command, ex, 707
unread (U) command, mailx, 272
unset command, Bash, 653
unset (uns) command, mailx, 273
until command, Bash, 653
until command, gdb, 165
up command, gdb, 165
up2date program, 544
update command, svn, 793
update command, yum, 551
upgrade command, yum, 551

upper character class, 600, 656
uppermem command, GRUB, 536
Upstart, 19
Upstart init daemon, 208
uptime command, 464
URLs, fetching, 247
USB scanners, detecting by SANE, 376
USB (Universal Serial Bus) devices, listing, 264
Usenet newsgroups, xviii
user accounts, NIS, 32
user authentication files, creating or updating, 199
user command, ftp, 153
User Datagram Protocol (UDP), 23
user groups, xix
user ID, displaying information about, 204
useradd command, 464–466
user-defined functions, gawk, 734
userdel command, 466
usermod command, 466
users
 adding or updating, 464–466
 changing groups for, 308
 creating or updating, 308
 current, printing ID of, 483
 deleting, 466
 interactive conversations with, 488
 logged in, listing, 482
 logged in, printing summary of, 471
 mailing regarding exceeded quotas, 472
 modifying account information for, 466
 number of logged in, printing, 464
 space allowed on disk for, 344
 talking to another user, 431
 writing a message to all users, 471
 (see also groups)
users command, 467
usleep command, 467
uuid command, svnlk, 803
UUID (Universal Unique Identifier)
 creating, 468
uuidgen command, 468

V

v command, ex, 707
v command, sed, 716, 725
\V release escape sequence, 612

\v version escape sequence, 612
\v vertical tab escape sequence, 601, 715, 733
variable substitution, 601, 606
variables
 built-in, 607–611
 declaring, 626
 exporting to make global, 630
 forcing expansion of, 629
 gawk, 731, 732
 local, 636
 reading an input line and assigning to, 639
 readonly, 640
 setting or printing, 641–643
 unsetting, 653
vbprobe command, GRUB, 536
vdir command, 468
verbose command, ftp, 153
verify command, svnadmin, 799
verify (verif) command, mailx, 273
version command, ex, 707
version control system (see source code management system)
Version Control with Subversion (Pilato et al.), 755
version of package, 543
vertical bar (|)
 match either one or the other, 656
 output directed to next command, 746
 pipe character, 601, 602
vertical bar, ampersand (|&) output directed to coprocess, 735, 746
vertical bar, double (||)
 OR operator for commands, 602
 OR operator, in expressions, 620, 732
VFAT filesystem, 16
vi command, 468, 678
vi editor, 677–696
 command-line options, 678–680
 commands, 683–692
 copy and move, 688
 edit, 687
 insert, 686
 line numbering, 685
 macros, 691
 marks, 685
 movement, 683–686

multiple files, accessing, 689
on status line, 683
saving files, 688
:set command, 692–696
syntax for, 681
system access, 690
window, 690
configuration of, 692–696
exiting, 680, 688
invoking, 678
modes, 678, 681
pattern matching metacharacters
 supported by, 657
video mode, setting, 468
vidmode command, 468
view command, ex, 707
vile editor, 677
vim command, 469
vim editor, 677, 678
 command-line options, 678–680
 visual mode, 682
virsh commands, 842, 844, 850,
 852–856
virt-clone command, 856
virt-image command, 857
virt-install command, 840, 842, 844,
 846, 858–861
virt-manager command, 861
virt-manager package, 842, 849, 850,
 851
virtual center, for VMware, 862
virtual console
 deallocating, 107
 determining number of, 137
virtual machine monitor (VMM), 838
virtual memory statistics,
 printing, 469–471
Virtual PC, 504
virtual terminal
 executing commands on, 322
 Meta key handling for, 394
 switching to, 84
virtualization method for booting, 504
virtualization tools, 838–873
 connection to Internet, 839
 console interface for, 842
 creating virtual systems, 840
 disk image files for, 844
 graphic interface for, 842
 KVM (see KVM hypervisor)
 libvirt tools, 839
libvirt (see libvirt virtualization API)
MAC address for, 844
managing virtual systems, 841
migration support for, 839
network configuration for, 842
paravirtualization, 839
for a single guest, 838
system requirements for, 838
VMware (see VMware)
Xen (see Xen hypervisor)
virt-viewer command, 861
virt-viewer package, 842, 849
VISUAL built-in variable, 611
visual command, ex, 708
visual mode, vim, 682
visual (v) command, mailx, 273
vmkfstools command,
 VMware, 870–873
VMM (virtual machine monitor), 838
vmstat command, 469–471
VMware, 862–873
 changing virtual machine
 configurations, 844
 ESX management client for, 862
 ESX server commands, 864–873
 graphic interface for, 842
 managing virtual systems, 842
 networking for, 863
 paravirtualization with, 864
 shared disks for, 863
 snapshots, 863
 virtual center for, 862
 virtual switch for networking, 842
 website for, 837
VMware server, 504
vmware-cmd command, VMware, 842,
 868–870
volname command, 471
volume name for device,
 determining, 471
Vromans, Johan (Perl 5 Pocket
 Reference), 654
vSphere, 862
vsplit command, ex, 708

W

w command, 471
W command, sed, 716, 725
w command, sed, 716, 725
\W current directory escape
 sequence, 612

\w current directory escape sequence, 612
\W escape character, pattern matching, 656
\W escape sequence, 715
\w escape character, pattern matching, 656
\w escape sequence, 715
wait command, Bash, 653
wait command, lftp, 247
wall command, 471
wall command, ex, 708
warnquota command, 472
watch command, 472
watch command, gdb, 165
WAV format, converting to CDDA, 69, 200–203
wc command, 473
website resources, xvii–xix
 Arch system, 753
 Bash debugger, 608
 Bash shell, 597
 Codeville system, 753
 CSSC system, 754
 for this book, xxi
 GNU utilities documentation, xvii
 libvirt virtualization API, 850
 Linux Documentation Project, xvii
 Linux Gazette, xvii
 Linux Insider, xvii
 Linux Security, xvii
 Linux Today, xvii
 Linux Weekly News, xvii
 monotone system, 754
 netfilter, 29
 online support, xviii
 Slashdot, xvii
 Subversion, 759
 Usenet newsgroups, xviii
 user groups, xix
 vim editor, 678
 virtualization tools, 837
weekly command, logrotate, 256
Welsh, Matt (Running Linux), xvii, 4
wget command, 474–480
whatis command, 480
whatis command, gdb, 165
whereis command, 480
which command, 481
while command, Bash, 653
while command, gawk, 746
whitespace, converting to tabs, 462
who command, 482
whoami command, 483
whois command, 483
window commands, Emacs, 666
window commands, vi, 690
Windows
 accessing files from Linux, 16
 as host OS with Linux as guest, 504
 booting with GRUB, 517
 booting with LILO, 506
 dual booting Linux with, 505, 536–539
 Linux as challenge to, 2
 networking on, Linux providing, 3
 restoring original boot loader, 506
wnext command, ex, 708
wodim command, 484–488
word character class, 600
word-abbreviation commands, Emacs, 666
working tree, for Git, 806
 cleaning, 818
 removing files from, 833
wq command, ex, 708
wqall command, ex, 709
write a message to all users, 471
write command, 488
write command, ex, 708
write messages, controlling receipt of, 286

X

X command, ex, 709
x command, mailx, 270
x command, sed, 716, 725
\x hexadecimal escape sequence, 715, 733, 601
X Window System, 4
xargs command, 488–489
xdigit character class, 600, 656
XDR (eXternal Data Representation), 32
Xen hypervisor, 845–849
 changing virtual machine configurations, 844
 device interfaces for, 843
 graphic interface for, 842
 installing, 845

libvirt used with (see libvirt
virtualization API)
managing virtual systems, 842
networking for, 846
paravirtualization with, 846
website for, 837
xentop command, 848
xm command, 846–848
xentop command, 848
XFS (Extensible Filesystem), 16
xinetd command, 490–493
xit command, ex, 709
xit (x) command, mailx, 273
xm command, Xen, 846–848
xor() function, gawk, 746

Y

y command, sed, 716, 725
yacc command, 493
yank command, ex, 709
yank (paste), Emacs, 662
yes command, 494
youngest command, svnlook, 803
YP (Yellow Pages), 31
ypbind command, 494
ypcat command, 495
ypinit command, 495
ypmatch command, 496
yppasswd, 496
yppasswdd command, 496
yppoll command, 497
yppush command, 497
yps server, 31
ypserv command, 498

ypserv server, 31
ypset command, 499
ypitest command, 499
ypwhich command, 500
ypxfr command, 500
ypxfrd command, 501
yum program, 544, 545–552
 command syntax and options
 for, 545–547
 commands used in, 547–551
 configuration for, 545
 plugins for, 551
 yum-builddep plugin, yum, 552
 yum-complete-transaction plugin,
 yum, 552
 yumdownloader plugin, yum, 552
 yup program, 545

Z

z command, ex, 709
z command, mailx, 273
Z Shell, 596
zcat command, 501
zcmp command, 501
zdiff command, 501
zforce command, 501
zgrep command, 502
zless command, 502
zmore command, 502
znew command, 503
zones, printing information about, 197
zsh shell, 596
Zwickly, Elizabeth D. (Building Internet
Firewalls), 29

About the Authors

Ellen Siever is a writer and editor specializing in Linux and other open source topics. In addition to *Linux in a Nutshell*, she coauthored O'Reilly's *Perl in a Nutshell*. She is a longtime Linux and Unix user and was a programmer for many years until she decided that writing about computers was more fun.

Stephen Figgins honed many of his computer skills while working as O'Reilly's book answer guy. A lifelong learner with many interests, Stephen draws on many resources to make difficult topics understandable and accessible.

Now living in Lawrence, Kansas, he administers Linux servers for Sunflower Broadband, a cable company. When not found working with computers, writing, or spending time with his family, you will likely find him outdoors. Stephen teaches wilderness awareness and living skills.

Robert Love has been a Linux user and hacker since the early days. He is active in—and passionate about—the Linux kernel and GNOME desktop communities. His recent contributions to the Linux kernel include work on the kernel event layer and inotify. GNOME-related contributions include Beagle, GNOME Volume Manager, NetworkManager, and Project Utopia. Currently, Robert works in the Open Source Program Office at Google.

Robert is the author of *Linux Kernel Development* (SAMS, 2005) and *Linux System Programming* (O'Reilly, 2007). He is also a contributing editor at *Linux Journal*. He is currently working on a new work for O'Reilly that will be the greatest book ever written, give or take. Robert holds a BA in mathematics and a BS in computer science from the University of Florida. A proud Gator, Robert was born in South Florida but currently calls Cambridge, Massachusetts home.

Arnold Robbins, an Atlanta native, is a professional programmer and technical author. He has worked with Unix systems since 1980, when he was introduced to a PDP-11 running a version of Sixth Edition Unix. He has been a heavy AWK user since 1987, when he became involved with gawk, the GNU project's version of AWK. As a member of the POSIX 1003.2 balloting group, he helped shape the POSIX standard for AWK. He is currently the maintainer of gawk and its documentation. He is also coauthor of O'Reilly's *Learning the vi and Vim Editors*. Since late 1997, he and his family have been living happily in Israel.

Colophon

The animal featured on the cover of *Linux in a Nutshell*, Sixth Edition, is an Arabian horse. Known for its grace and intelligence, the Arabian is one of the oldest breeds of horse, with evidence of its existence dating back 5,000 years. The Arabian was instrumental as an ancestor to other popular breeds, most notably the Thoroughbred in the 17th and 18th centuries. One of the more distinctive horse breeds, the typical Arabian has large, expressive eyes and nostrils; small ears; and a short, sturdy back. Its stamina suits it particularly well for endurance

riding, a sport dominated by the Arabian breed. Its wonderful temperament makes the Arabian an all-around favorite riding horse in North America, although it also can be found in more specialized competitions such as dressage, jumping, and reining.

The cover image is from the Dover Pictorial Archive. The cover font is Adobe ITC Garamond. The text font is Linotype Birkhäuser; the heading font is Adobe Myriad Condensed; and the code font is LucasFont's TheSansMonoCondensed.