Guide_on_Unix

book@wikibooks.org

Contents

Articles

References

Article Licenses

Guide to Unix/Commands

The Unix command line is often considered difficult to learn. This book aims to help beginners by introducing various commands in lucid and simple language. Unlike most command references, this book is designed to be a self-study guide.

Contents

Guide to Unix/Commands/Summary

This is my own summary of useful Linux abbreviations, directories, files, and commands. I use my own annotations to recall useful options and arguments that are not necessarily documented in easy-to-find places. I quite often call up this file when I can't remember the syntax of a command that I use often (but not often enough to remember the syntax!). I also editorialize on the relative usefulness of different types of programs.

This document is work in progress. Send suggested changes and corrections to `ambler.steven@uqam.ca` [1]

O'Reilly has just published online an alphabetical list of commands from *Linux in a Nutshell*. It is available `here` [2]. It contains more detailed explanations of many of the commands listed here.

Shorthand at the Command Prompt

Some of these are specific to the bash shell. I have not experimented enough with other shells to know which are common to all shells. See also the ``Bash Reference Card'', SSC (2000), available online.

- / - root directory
- ./ - current directory
- ./command_name - run a command in the current directory when the current directory is not on the path
- ../ - parent directory
- ~ - home directory
- $ - typical prompt when logged in as ordinary user
- # - typical prompt when logged in as root or superuser
- ![string] - repeat command starting with [string]
- !?[string] - repeat command ending with [string]
- !?[string] - repeat command containing [string]
- ![number] - repeat specified command x
- !-[number] - repeat specified command x commands ago
- !! - repeat previous command
- ^[string]^[string2]^ - repeat previous command with [string2] substituted for [string] (equiv. to !!:s/string1/string2/)
- & - run a program in background mode
- `[Tab][Tab]` - prints a list of all available commands. This is just an example of autocomplete with no restriction on the first letter.
- x[Tab][Tab] - prints a list of all available completions for a command, where the beginning is ``x''
- `[Alt][Ctrl][F1]` - switch to the first virtual text console from X.
- `[Alt][Ctrl][Fn]` - switch to the n^{th} virtual text console from X.
- `[Alt][Ctrl][F7]` - switch to the first GUI console, if there is one running, assuming that there are 6 virtual text consoles. If the graphical console freezes, one can switch to a nongraphical console, kill the process that is giving problems, and switch back to the graphical console using this shortcut.
- `[ArrowUp]` - scroll through the command history (in bash)
- `[Shift][PageUp]` - scroll terminal output up. This also works at the login prompt, so you can scroll through your boot messages.
- `[Shift][PageDown]` - scroll terminal output down
- `[Ctrl][Alt][+]` - switch to next X server resolution (if the server is set up for more than one resolution)
- `[Ctrl][Alt][-]` - change to previous X server resolution
- `[Ctrl][Alt][BkSpc]` - kill the current X server. Used when normal exit is not possible.
- `[Ctrl][Alt][Del]` - shut down the system and reboot
- `[Ctrl]a` - move cursor to beginning of line

- `[Ctrl]c` - kill the current process
- `[Ctrl]d` - logout from the current terminal
- `[Ctrl]e` - move cursor to end of line
- `[Ctrl]l` - clear screen
- `[Ctrl]n` - next in history (same as down arrow)
- `[Ctrl]p` - previous in history (same as up arrow)
- `[Ctrl]q` - resume transfer to current terminal. This should be tried if the terminal stops responding.
- `[Ctrl]s` - stop transfer to current terminal
- `[Ctrl]u` - clear from cursor to beginning of line
- `[Ctrl]y` - send current process to the background when it asks for input. (delayed background)
- `[Ctrl]z` - send current process to the background
- reset - restore a terminal to its default settings
- `[Leftmousebutton]` - Hold down left mouse button and drag to highlight text. Releasing the button copies the region to the text buffer under X and (if gpm is installed) in console mode.
- `[Middlemousebutton]` - Copies text from the text buffer and inserts it at the cursor location. With a two-button mouse, click on both buttons simultaneously. It is necessary for three-button emulation to be enabled, either under gpm or in XF86Config.

Important Daemons and Startup Services

These are programs or processes which are run at boot time. Some remain in memory to execute various tasks when required (daemons). Most are started and stopped with scripts in the /etc/rc.d/init.d directory (see above). The exact contents of this directory will depend on which packages from a particular distribution are installed. For example, installing the Apache package will cause an httpd script to be placed in /etc/rc.d/init.d.

There are man pages on most of these. The Red Hat program tksysv (ntsysv is the non graphical version) allows root to automatically configure which of these are started automatically at boot time. The linuxconf program does the same thing, although I haven't tried it. The utility chkconfig is also designed to query and configure runtime services for different runlevels. The `www.mandrakeuser.org` site has a good page on common services/daemons, especially those included in recent versions of the Mandrake distribution.

A good source of information on daemons and services is the ``Linux Devices, Daemons, Services'' chapter of the CTDP (2000a) document.

- amd - runs the automount daemon for remote filesystem mounting such as nfs
- anacron - checks delayed `cron' tasks (see below) at boot time and executes them. Useful if you have cron jobs scheduled but don't run your machine all the time.
- apmd - Advanced Power Management BIOS daemon. For use on machines, especially laptops, that support apm. Monitors battery status and can shut down the system if power is too low.
- arpwatch - keeps watch for ethernet IP address pairings that are resolved using the ARP protocol.
- atd - runs jobs queued by `at'
- autofs - control the operation of automount daemons, used to mount and unmount devices on demand
- bootparamd - allows computers to boot from a Linux machine using the BOOTP network protocol. A server process that provides information to diskless clients necessary for booting
- crond - automatic task scheduler. Manages the execution of tasks that are executed at regular but infrequent intervals, such as rotating log files, cleaning up /tmp directories, etc.
- cups - daemon for print services under the Common Unix Printer System, a replacement for lpd
- dhcpd - implements the Dynamic Host Configuration Protocol (DHCP) and the Internet Bootstrap Protocol (BOOTP). Used to lease out IP addresses to remote machines.
- drakfont - font server in Mandrake

- fetchmail - daemon to fetch mail at regular intervals from mail servers
- ftpd - ftp server daemon
- gated - routing daemon that handles multiple routing protocols and replaces routed and egpup
- gpm - useful mouse server for applications running on the Linux console.
- httpd - the Apache webserver hypertext transfer protocol daemon
- identd - The identd server provides a means to determine the identity of a user of a particular TCP connection. Given a TCP port number pair, it returns a character string which identifies the owner of that connection on the server's system.
- inetd - listens for service requests on network connections, particularly *dial-in* services. This daemon can automatically load and unload other daemons (ftpd, telnetd, etc.), thereby economizing on system resources. In the latest version of Red Hat (7.0 at the time of writing), it has been replaced by xinetd. A partial list of services controlled by inetd is listed below. Under many distributions, inetd will execute scripts in the file /etc/inetd.conf.
- innd - Usenet news server daemon
- ipchains - daemon for packet forwarding. Used for configuring a gateway/firewall.
- isdn provides ISDN network interfacing services
- isdn4linux - for users of ISDN cards
- kerneld - automatically loads and unloads kernel modules
- keytable - loads the appropriate keyboard map from /etc/sysconfig/ keyboard
- kheader -
- kudzu - detects and configures new or changed hardware during boot
- linuxconf - ``startup hook'' needed for the linuxconf system configuration tool
- lpd - line printer and print spooler daemon
- mcserv - server program for the Midnight Commander networking file system. It provides access to the host file system to clients running the Midnight file system (currently, only the Midnight Commander file manager). If the program is run as root the program will try to get a reserved port otherwise it will use 9876 as the port. If the system has a portmapper running, then the port will be registered with the portmapper and thus clients will automatically connect to the right port. If the system does not have a portmapper, then a port should be manually specified with the -p option (see below).
- mysql - database server daemon
- named - provides DNS services
- netfs - network filesystem mounter. Used for mounting nfs, smb and ncp shares on boot.
- network -activates all network interfaces at boot time by calling scripts in /etc/sysconfig/network-scripts
- nfsd - used for exporting nfs shares when requested by remote systems
- nfslock - starts and stops nfs file locking service
- numlock - locks numlock key at init runlevel change
- pcmcia - generic services for pcmcia cards in laptops
- portmap - needed for Remote Procedure Calls
- postfix - mail transport agent which is a replacement for sendmail. Now the default on desktop installations of Mandrake.
- postgresql - database server daemon
- random - random number generating daemon, related to security and encryption
- routed - manages routing tables
- rstatd - kernel statistics server. Allows users on a network to get performance statistics for any connected machine.
- rusersd - provides services that allow users to find one another over the network
- rwalld - allows users to use rwall to write messages on remote terminals

- rwhod - server which maintains the database used by the rwho(1) and ruptime(1) programs. Its operation is predicated on the ability to broadcast messages on a network.
- sendmail - mail transfer agent. This is the agent that comes with Red Hat. Others, such as smtpd, are not included.
- smb - needed for running SAMBA
- snmpd - provides Simple Network Management Protocol support
- sound - daemon for managing sound
- squid - web page proxy server daemon
- syslogd - manages system log files
- smtpd - Simple Mail Transfer Protocol, designed for the exchange of electronic mail messages. Several daemons that support SMTP are available, including sendmail, smtpd, rsmtpd, qmail, zmail, etc.
- tcpd - from the tcp_wrappers package. Intercepts requests normally handled by inetd and filters them through the files hosts.allow and hosts.deny files, which can restrict access to services based on type of service, origin of request, destination, etc. Requests are intercepted because calls to particular services are replaced with calls to tcpd in /etc/inetd.conf.
- telnetd - telnet server daemon
- usb - daemon for devices on Universal Serial Bus
- xfs - X font server
- xinetd - more modern replacement for inetd. It apparently allows for similar kinds of access filters to the ones used by tcpd in conjunction with inetd. xinetd replaces inetd as the default network services daemon in Red Hat 7.0.
- xntpd - Network Time Protocol daemon. Provides a means to syncronize time over the network.
- webmin - daemon for webmin web-based system administration program
- ypbind - NIS binder. Needed if computer is part of Network Information Service domain.

Notes on Applications

Mail Transfer Agents (MTAs)

The Linux distributions I know come with sendmail, except for Mandrake, which as of version 7.1 uses Postfix as its default MTA. There are several competing programs available. Even the simplest don't seem to be that easy to configure.

- Exim -
- Fetchmail - seemingly one of the few ways (Pine is able to do this as well) to download mail automatically from a POP or IMAP server and pass it to local mail handling agents. Use the following line in /.fetchmailrc:
  - poll pop.uqam.ca proto pop3 user USERNAME pass PASSWORD

Use the following to have fetchmail loaded as a daemon that will download mail at regular intervals:

- fetchmail -d 6000

The interval is specified in seconds. Fetchmail will poll all of the pop servers listed in /.fetchmailrc.

- Getmail - Designed as a replacement for Fetchmail.
- MMDF -
- Postfix - a mail transport agent and potential replacement for sendmail. Mandrake 7.1 and up uses this as its default MTA.
- Qmail - a ``modern'' replacement for sendmail. It is reputed to be more secure than sendmail. Since it doesn't have a GPL license, it is not the default MTA of any Linux distributions that I know of.
- Sendmail - this one gets my vote for the most complicated and obscure configuration file, /etc/sendmail.cf. Most individual Linux users will be using machines connected to the Internet via an ISP or on networks (such as university networks) with centralized mail processing and access to the net. I have to change the following lines

in sendmail.cf to be able to send mail with emacs.

- DMuqam.ca # masquerade the domain name
- DNambler.steven # masquerade username
- DSnobel.si.uqam.ca # relay all mail through nobel server

The problem comes from the fact that, as a user on a local network, I don't have my own domain name. I want return mail to be routed to UQAM's mail server and I want the server to handle all my mail for me, even mail to other UQAM users. If I send mail to UQAM users using their normalized usernames, the net does not know who or where they are. I have managed to get a configuration that works by writing a sendmail.mc file and processing it with the m4 macro interpreter, following the Address-Rewriting mini-HOWTO. I now have something that works, but which mysteriously complains about ``dangerous write permissions'' every time the system boots up.

- Smail - seems to be a popular choice on smaller systems. It would appear that at one point in its history, Red Hat shipped with smail, but this has been replaced with sendmail.
- Zmailer - apparently designed for mail servers with a large number of users.

Mail User Agents (MUAs)

- Acmemail - Web-based mail agent. Allows you to access your mail with any browser. Involves setting up a Perl CGI script on the server side.
- Archimedes - A successor program to XFmail (see below)
- Arrow -
- Balsa - the default Gnome mail program
- Blitzmail -
- Elm -
- Emumail - Web-based mail agent. Allows you to use any browser to check your POP mail account. The Web site of the company that makes this one can be used to check your mail on a Unix system without setting up any CGI script on the server side.
- Evolution - mail reader and contact manager/calendar designed for use under Gnome
- Exmh - graphical front end for Mh
- Kmail - mail reading program included with KDE
- M - for ``Mahogany''. Seems similar to XFmail (see below). I haven't been able to figure out from the description whether it runs independently of or in conjunction with sendmail and procmail.
- Mh -
- Mumail -
- Mutt - text based mail program, which is highly configurable.
- Nmh - mail handling system. This system includes a *large* number of binary commands that are kept in /usr/bin. See the man page for nmh for details. Red Hat 5.1 and 5.2 come with exmh and xmh, which are graphical front ends for nmh. The exmh front end is a separate package, while xmh is owned by XFree86.
- Pine - text based mail and news utility. Features now include:
  - MIME support
  - ability to read and post network news
  - maintenance of an address book of mail recipients
  - spell checking during message composition
  - mouse support when using xterm on an X Window system
  - a highly configurable environment

Pine can be used to download mail from one or more POP3 mail servers. See Tip of the Week (http://tipoftheweek.darkelf.net [3]) for the fourth week of February 1999. First, set up multiple configuration files (pine -p localmail, pine -p popserver 1, pine -p popserver 2, etc.). Then, to configure Pine to use a POP3 server, use the Setup Config

command. Set something like this in the inbox-path:

- {pop.server.com/pop3/user=myid}INBOX

When Pine is restarted, it should ask for your password, connect to the remote server, and use it is if it were accessing local mail. The article is unclear on whether there is the option of leaving copies of the downloaded mail on the server.

- Sylpheed -
- XCmail -
- XFmail - This one seems very promising. It's a GUI-based mail tool that seems to offer most of the features of Netscape's mail module. It runs without using sendmail and procmail, which is a major advantage.

Editors

- cooledit - a pretty powerful GUI text editor
- emacs - powerful text editor that includes modules for reading and sending mail and postings to newsgroups, and a browser module. For editing T_EX and L^AT_EX files, the $AucT_EX$ addon package is invaluable, and makes emacs pretty hard to beat as an editor with L^AT_EX.
- jed - has pretty good emacs emulation (it can even read mail like emacs!). It does simple syntax highlighting for TeX files, including giving positioning of parentheses. It would seem to be pretty configurable and takes up much less disk space than emacs, although more than joe and muemacs. It works well in console mode, and still manages to use colors for menu bars and syntax highlighting. The program xjed which comes with some versions starts up its own X terminal when invoked.
- joe - "Joe's Own Editor", a fairly powerful editor with a compact binary and an ability to emulate Wordstar, Emacs, Pico, and a few other editors.
- jove - "Joe's Own Version of Emacs". I tried this out a couple of times and managed to crash it when making some minor errors in command syntax.
- microemacs (JASSPA) - spinoff of muemacs. Pretty powerful and configurable, while not taking up too much disk space or memory.
- muemacs - a fairly powerful emacs clone whose binary is actually smaller than that of the Joe editor.
- nedit - an X Window based text editor. Of all text editors for Linux that I've seen, it has commands which are closest to Windows text editors, for cursor movement, highlighting, marking text, etc. It has very good syntax highlighting for both L^AT_EX and HTML.
- pico - simple text editor. It often comes packaged with the Pine mail user agent.
- vi - included with most Linux distributions. If you're not used to the syntax, it can be pretty hard to understand.
- vim - improved version of vi
- xedit - simple text editor included with many Linux distributions

Other

- dfm - Desktop File Manager. Allows the user to place program icons on the desktop.
- gmc - Gnome Midnight Commander. Gnome version of Midnight Commander. Includes a graphical interface and allows the user to place icons on the desktop.
- mc - Midnight Commander file manager. Runs in console mode and in an xterm.
- scilab - a free matrix programming language. May be a good substitute for GAUSS and/or MATLAB.

Also See

- → Guide to Unix/Files
- → Guide to Unix/Environment Variables
- Guide to X11/Window Managers
- Linux Guide/Linux commands
- Use the Source/Counter-Culture
- Guide to Unix
- Guide to X11
- Use the Source

External Links

- Rosetta Stone For *Nix [4] - configurable list of equivalent programs for *nix and MAC systems.

References

Computer Technology Documentation Project (CTDP) (2000a), ``How Linux Works'', http://www.comptechdoc.org/os/linux/howlinuxworks/ [5]

Computer Technology Documentation Project (CTDP) (2000b), ``Linux Files and Command Reference'', http://www.comptechdoc.org/os/linux/commands/ [6]

Klimas, Piotr et. al. (1999), ``Linux Newbie Administrator Guide'', http://linux-newbie.sunsite.dk/ [7]

Sievers, Ellen, Stephen Spainhour, Jessica P. Hekman, and Stephen Figgins (2000), *Linux in a Nutshell*. third edition, O'Reilly

Sobell, Mark G. (1998), *A Practical Guide to Linux*. Addison-Wesley

Welsh, Matt, Matthias Kalle Dalheimer and Lar Kaufman (1999), *Running Linux*. third edition, O'Reilly and Associates

References

[1] mailto:ambler.steven@uqam.ca
[2] http://www.onlamp.com/linux/cmd/
[3] http://tipoftheweek.darkelf.net
[4] http://bhami.com/rosetta.html
[5] http://www.comptechdoc.org/os/linux/howlinuxworks/
[6] http://www.comptechdoc.org/os/linux/commands/
[7] http://sunsite.dk/linux-newbie/

Guide to Unix/Commands/Getting Help

man

man displays the manual page for the specified command

A useful option is:

```
$ man -k TEXT
```

This searches manual page titles and synopsis lines for TEXT

Examples:

To display the manual page for the chown command:

```
$ man chown
```

man has different sections.

- section 1 is user commands
- section 2 is system calls (used by programs to communicate with the kernel)
- section 3 is library reference (for programming in C)
- section 4 is device drivers
- section 5 is configuration files and other file formats
- section 6 is games
- section 7 is miscellaneous (for example, "ascii" map and C "operator" precedence)
- section 8 is system commands (like user commands, but mostly for root)

A section number can be specified before the page name. For example, **`man chmod`** normally shows the user command "chmod". To see the system call "chmod":

```
$ man 2 chmod
```

To search the man pages for "newsgroups",

```
$ man -k newsgroups
actsync               (8)  - synchronize newsgroupsoups
newsgroups            (1)  - a program to list unsubscribed newsgroups
```

- If this does not work you may have to run the makewhatis command.

info

info is an advanced man command that is sometimes available. It displays the improved manual pages in Info format for specified command.

Examples:

To display the manual page for the grep command:

```
$ info grep
```

To find occurrence of 'grep' in all info manual pages:

```
$ info --apropos grep
"(autoconf-2.13)Examining Declarations" -- EGREP_CPP
"(autoconf-2.13)Examining Declarations" -- EGREP_HEADER
"(autoconf-2.13)Old Macro Names" -- HEADER_EGREP
```

```
...
```

To see the physical location of 'grep' info manual page:

```
$ info -w grep
/usr/share/info/grep.info.gz
```

To view a file a info page:

```
$ info -f ./some_cmd.info.gz
```

apropos

apropos searches the manual page short descriptions for a specified keyword

On many systems this is exactly the same as the -k option of the **man** command.

Examples:

```
$ apropos newsgroups
active               (5)  - list of active Usenet newsgroups
newsgroups           (1)  - a program to list unsubscribed newsgroups
```

whatis

whatis displays short man page descriptions. Unlike

Examples:

```
$ whatis info
info (1)             - read Info documents
info (5)             - readable online documentation
```

```
$ whatis chmod
chmod (1) - change file modes
chmod, fchmod (2) - change mode of file
```

makewhatis

makewhatis creates the database for the whatis, appropos, and man -k commands. This is commonly run automatically by your system however sometimes you need to run this manually.

Examples:

```
# makewhatis
```

$ **sudo makewhatis**

Guide to Unix/Commands/File System Utilities

ls

ls

LiSt contents of directory

- =1 FreeBSD manual page [1]
- NetBSD manual page [2]
- =1 OpenBSD manual page [3]
- ls [4] at the GNU core-utils manual [5]
- ls at the Linux questions wiki

ls is a utility for listing the files in a directory.

Most used options are:

-a

all files (include files with . prefix)

-l

long detail (provide file statistics)

-t

order by creation time

-u

sort by access time (or show when last accessed together with -l)

-S

order by size

-r

reverse order

-F

mark directories with /, executables with *, symbolic links with @, local sockets with =, named pipes (FIFOs) with |

Other options include:

-Fx --color

color-coded listing

-s

show filesizes

-h

"human readble"; show filesizes in kilobytes and megabytes (-h can be used together with -l or -s)

```
ls *st* : list files that contain st in name
ls > list : output list to file named "list"
ls | more : fit listing to screen
```

Examples

```
$ ls
fish  hello.txt
```

```
$ ls -l
-rw-r--r--    1 username    groupname        0 Apr 11 00:09 fish
-rw-r--r--    1 username    groupname       11 Apr 11 00:10 hello.txt
```

Several systems have an alias **ll** which does the same as **ls -l**:

```
$ ll
-rw-r--r--    1 username    groupname        0 Apr 11 00:09 fish
-rw-r--r--    1 username    groupname       11 Apr 11 00:10 hello.txt
```

Be careful with the -F option. Here is one example:

```
$ ls -F /usr/X11R6/bin/X*
/usr/X11R6/bin/X@          /usr/X11R6/bin/Xnest*
/usr/X11R6/bin/Xprt*
/usr/X11R6/bin/Xmark*      /usr/X11R6/bin/Xorg*
/usr/X11R6/bin/Xvfb*
```

We do not know yet if there is a symbolic link "X" and an executable "Xmark" or if "X@" and "Xmark*" are just the names of normal files. (Though "@" and "*" are not much found in filenames, they are possible.) So we check by dropping the -F:

```
$ ls /usr/X11R6/bin/X*
/usr/X11R6/bin/X           /usr/X11R6/bin/Xnest
/usr/X11R6/bin/Xprt
/usr/X11R6/bin/Xmark       /usr/X11R6/bin/Xorg
/usr/X11R6/bin/Xvfb
```

mkdir

mkdir is a utility for creating a directory.

Examples

```
$ mkdir newdirectoryname
```

The *-p* option also makes parent-directories as needed. Instead of:

```
$ mkdir foo
$ cd foo
$ mkdir bar
```

you can just do:

```
$ mkdir -p foo/bar
```

cd

cd

change Current Directory

- cd &sektion=0 FreeBSD manual page [6]
- cd +0 NetBSD manual page [7]
- cd &sektion=0 OpenBSD manual page [8]
- cd [9] at the GNU core-utils manual [5]
- cd at the Linux questions wiki

cd changes the **c**urrent **d**irectory of the shell. This current directory will be used by other programs launched from the shell.

Because "cd" changes the state of the shell, it is a *shell built-in command*. It contrast, most commands are separate programs which the shell starts.

Examples

Change to 'foobar' directory:

```
$ cd foobar
```

Change to your home directory, cd command used without an option will drop you back into your home directory.

```
$ cd
```

~ (tilde) stores the path to your home directory, this command has same effect as the previous one.

```
$ cd ~
```

Change to parent directory:

```
$ cd ..
```

Change to the previous directory:

```
$ cd -
```

Tips:

By setting "CDPATH" environment variable in your shell you can take advantage of shell command completion facility.

```
$ echo $CDPATH
.:/usr/local:/usr/share/doc
```

If you have the $CDPATH set, then you press 'TAB' key and get possible path completions

```
$ cd bas [TAB]
base-config/  base-files/   base-passwd/  bash/         bastille/
```

pwd

pwd

Print Working Directory

- =1 FreeBSD manual page [10]
- NetBSD manual page [11]
- =1 OpenBSD manual page [12]
- pwd [13] at the GNU core-utils manual [5]
- pwd at the Linux questions wiki

pwd (for Print Working Directory) shows the current directory that you are in.

Though "pwd" is often available as an external program (like `/bin/pwd`), many shells offer an equivalent version as a shell builtin command. Like any external command, "pwd" would inherit the current directory from the shell or other program that starts it.

Examples

```
$ pwd
/home/username
```

You can change the directory, you can also

```
$ cd /usr
$ pwd
/usr
```

You can also use "pwd" in scripts. If you have enough experience with scripting, then you would know that the next line complains if the current directory is `/home/username`.

```
$ test "x$(pwd)" = x/home/username && echo wrong directory
```

chroot

chroot

CHange ROOT directory

- =8 FreeBSD manual page [14]
- NetBSD manual page [15]
- =8 OpenBSD manual page [16]
- chroot [17] at the GNU core-utils manual [5]
- chroot at the Linux questions wiki

chroot changes the root filesystem. The "chroot" page at the Linux questions wiki explains why you might want to do this.

Examples

To change the root filesystem so `/mnt/usbdrive/` becomes `/` and files outside of it cannot be seen:

```
# chroot /mnt/usbdrive/
```

You must be root user to "chroot". Other users would be able to use "chroot" to gain root (superuser) priveleges, so their use of "chroot" is disallowed.

```
$ chroot /mnt/usbdrive/
chroot: /mnt/usbdrive/: Operation not permitted
```

cp

cp copies a file

Most used options are:

-r
: copies directories (recursively)

-p
: preserves permissions, ownership, and timestamps

-i
: prompt before overwrite

-v
: verbose, show filenames as they are being copied

Examples

```
Makes a copy of file 'debian' and call it 'Debian' (assuming 'Debian'
is not already a directory)
$ cp -i debian Debian
```

```
Makes a copy of file 'debian' and put it at /tmp/debian
$ cp -i debian /tmp/debian
```

```
Same as the previous command (the filename defaults to be the same).
$ cp -i debian /tmp
```

```
Makes a copy of directory 'mydir' (and all its contents) and put it at
/tmp/debian
$ cp -ir mydir/ /tmp
```

```
Copy multiple files to directory /tmp
$ cp -i foo bar baz /tmp
```

mv

mv move and/or rename files

Examples

```
Rename file 'unix' to 'Unix' (assuming "Unix" is not a directory)
$ mv -i unix Unix
```

```
Move file Unix from your home directory to /tmp.
$ mv -i ~/Unix /tmp/Unix
```

```
Same as the previous command (the filename defaults to be the same).
$ mv -i ~/Unix /tmp
```

```
Move file Unix from your home directory to /tmp, and rename it to
'unix'.
$ mv -i ~/Unix /tmp/unix
```

```
Move multiple files to directory /tmp
$ mv -i foo bar baz /tmp
```

rm

rm

ReMove and delete files

- =1 FreeBSD manual page [18]
- NetBSD manual page [19]
- =1 OpenBSD manual page [20]
- rm [21] at the GNU core-utils manual [5]
- rm at the Linux questions wiki

rm deletes a file from the filesystem, like the "del" command in DOS.

The GNU long options (like `--directory`) are available on Linux, but not most other systems.

Some useful options are:

`-d, --directory`

unlink FILE, even if it is an empty directory (some systems let superuser unlink non-empty directories too)

`-f, --force`

ignore nonexistent files, never prompt

`-i, --interactive`

prompt before any removal

`-P`

(*BSD only) overwrite file before deletion

`-r, -R, --recursive`

remove the contents of directories recursively (the force option must often be used to successully run rm recursively)

`-v, --verbose`

(GNU only) explain what is being done

`--help`

(GNU only) display help and exit

`--version`

(GNU only) output version information and exit

Examples:

The usage of "rm" is considered potentially more dangerous than equivalents in other operating systems because of the way the shell parses wildcards and names of special directories and in its non-verbose actions.

Here is a classic example. Instead of deleting files that end with `.o` ("*.o") it deletes all files in the directory ("*") and also a file called `.o`. There is an unwanted space between the asterisk and the period.

```
$ rm * .o
rm: cannot remove `.o': No such file or directory
```

To remove a file whose name starts with a `-', for example `-foo', use one of these commands:

```
$ rm -- -foo
```

```
$ rm ./-foo
```

It might be useful to create an alias such as "remove" which moves the files to a local "trash" file so you can go there and recover files you accidentally "remove"d.

Secure deletion of files:

Note that if you use rm to remove a file, it is usually possible to recover the contents of that file since `rm` does not remove it from the hard disk. It simply removes the filesystems link to it.

On *BSD systems, the -P option overwrites the data with the file before removing it.

```
$ rm -P secretfile
```

However, as the NetBSD manual page [19] explains it:

> *Recent research indicates that as many as 35 overwrite passes with carefully chosen data patterns may be necessary to actually prevent recovery of data from a magnetic disk. Thus the -P option is likely both insufficient for its design purpose and far too costly for default operation.*

So while examining the data (using **fsdb** or making a disk image) will not reveal the secret data, other methods (such as laboratory examination of the disk) will reveal the data. In short, **rm -P** does not delete data securely. A program that attempts to delete data securely is GNU **shred**, available on Linux. But "shred" is not always successful in secure deletion; read its entry below.

rmdir

rmdir is a utility for deleting empty directories.

Examples

```
$ rmdir directoryname
```

If the directory is not empty, the correct way to remove the directory and all its contents recursively is to use

```
$ rm -r directoryname
```

shred

shred

Attempt to securely delete files

- =1 FreeBSD manual page [22]
- NetBSD manual page [23]
- =1 OpenBSD manual page [24]
- shred [25] at the GNU core-utils manual [5]
- shred at the Linux questions wiki

shred overwrites a file multiple times with special data patterns to make the old contents of the file unrecoverable from a disk, especially a hard disk. This command is part of GNU coreutils, so it is often only available on Linux systems.

Note that this **actually is ineffective** on most filesystems because they can keep old copies of data. Most popular Linux filesystems (including ext3) keep such copies through journaling. However, "shred" is very useful for destroying the data on entire partitions or disks.

Some useful options are:

-u, --remove

unlink the file after removing it

-NUMBER*, -n *NUMBER*, --iterations=*NUMBER

the number of iterations of overwriting the file; default is 25 iterations

Examples: Remove and completely destroy `secretfile` from a filesystem that overwrites data in place and does not use journaling (for example, the UFS filesystem of *BSD). For the last step, after the data is destroyed, the "-u" option unlinks the file from the filesystem.

```
$ shred -u secretfile
```

Note that if `secretfile` has multiple hard links (with **ln** for example), it will continue to exist with those other names, but will contain only random data.

touch

touch lets you change the date on a file. Can also be used to create a blank file.

Examples

This will change the access date and time of *filename* to the current time. If *filename* doesn't exist it will create a blank file.

```
$ touch filename
```

df

df reports the amount of free disk space available on each partition.

```
$ df
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/md0               5763508    207380   5263352   4% /
/dev/md1              78819376  13722288  61093296  19% /home
/dev/md4              23070564   4309572  17589056  20% /usr
/dev/md2               5763508   1757404   3713328  33% /var
/dev/md3               2877756    334740   2396832  13% /tmp
```

Reports disk usage in human readable format with block-sizes in Kilo,Mega,Gigabytes.

```
$ df -h
Filesystem            Size  Used Avail Use% Mounted on
/dev/hda1             2.3G  2.1G  133M  95% /
tmpfs                  61M  8.0K   61M   1% /dev/shm
/dev/hda2             2.0G  1.8G  113M  94% /usr
```

ln

ln creates links between files.

Symbolic links are special files that contain the absolute or relative path to a separate file. References to the symbolic link are "forwarded" to the file it (the symbolic link) points to. The file to which the reference is forwarded is not "aware" of the link; if the file pointed to is moved or deleted, the link will just point to nothing (no warnings are given, nor are any attempts made to "refresh" the link to the file's new location).

In Unix all information about files - owner, group, permissions, size, number of links, location on disk - are stored in the inode (with the notable exception of *the filename*). The filename is stored in directories, where inode-numbers (the way the OS refers to files) are paired with filenames (text-strings; the way *the user* refers to files). The "number

of links" entry in the inode, keeps track of how many times the inode-number has thus been paired with a name in some directory.

When creating a (hard) link with `ln`, the "source" file is only used to determent the inode-number. A new inode-number-to-filename entry is then made in some directory using that inode-number, and the "number of links" counter is incremented. It's important that this (unlike a *symbolc* link) is *the same file* in every way... it's just that the file can be accessed by different names and/or from different locations. Moving or deleting the "original" source-file has no effect on the other links. Deleting one of the file's links, only removes it's entry from *that* directory... it's first when the *last* link in *any* directory is removed, that the file is actually deleted from the disk.

Hard links can not (unlike symbolic links) be used to refer to a file on another filesystem, nor can hard links usually be used to link to a directory.

Examples:

To make a soft (symbolic) link "hello" to the file "/home/alice/code/bin/world":

```
$ ln -s /home/alice/code/bin/world hello
```

To make a (hard) link from the file "foo" to the file "bar":

```
$ ls -l
total 1
-rw-r--r--  1  rtm  users  50  Aug 15   foo
$ ln foo bar
$ ls -l
total 2
-rw-r--r--  2  rtm  users  50  Aug 15   foo
-rw-r--r--  2  rtm  users  50  Aug 15   bar
```

The 1st number of `ls -l`'s listing, shows the number of (hard) links a file have - in other words, the number of times it's been entered into some directory.

chown

chown

CHanges OWNer of file

- =8 FreeBSD manual page [26]
- NetBSD manual page [27]
- =8 OpenBSD manual page [28]
- chown [29] at the GNU core-utils manual [5]
- chown at the Linux questions wiki

chown changes the owner and group of files. Normally, only root is allowed to do this, but if a user owns a file, then that user can change the group, but only to groups containing that user. On old systems, the ability of users to give files to other users caused abuses, so most systems prohibit non-root users from changing the owner of files.

Some useful options are:

-R

recursively change owner/group on an entire directory tree

-h

do not follow symbolic links i.e. changes owner of the link, not the target file

-f

indicate no errors if change failed

Examples:

Root changes the ownership of "/etc/passwd" and "/etc/shadow" to user root, group wheel:

```
# chown root:wheel /etc/passwd /etc/shadow
```

The same, but only changing the owner:

```
# chown root: /etc/{passwd,shadow}
```

The same, but only changing the group:

```
# chown :wheel /etc/{passwd,shadow}
```

Root gives every file in "/etc/ssh", including files in subdirectories, to user root, group wheel:

```
# chown -R root:wheel /etc/ssh
```

The same, but excluding files in directories (and the invisible files "/etc/ssh/.*" will also be missed):

```
# chown root:wheel /etc/ssh/*
```

User "tux" changes the directory "/usr/local/src/xc" from group "tux" to group "wheel". Tux is a member of both groups.

```
$ ls -ld /usr/local/src/xc
drwxr-xr-x  11 tux  tux  512 Sep 30 16:19 /usr/local/src/xc
$ chown tux:wheel /usr/local/src/xc
$ ls -ld /usr/local/src/xc
drwxr-xr-x  11 tux  wheel  512 Sep 30 16:19 /usr/local/src/xc
```

chmod

chmod

CHanges file MODe

- =1 FreeBSD manual page [30]
- NetBSD manual page [31]
- =1 OpenBSD manual page [32]
- chmod [33] at the GNU core-utils manual [5]
- chmod at the Linux questions wiki

chmod changes permissions of files. One must be familiar with Unix file permissions to understand this command. There are three permissions: read ("r"), write ("w"), and execute ("x"). There are three sets of permissions: for the owning user of the file ("u"), for the group of the file ("g"), and for other users ("o").

For a file, "execute" means to run it as a program. For a directory, "execute" permission is required to use anything in that directory tree, so doing anything with "/usr/share/doc/README" requires execute permissions on all of "/", "/usr", "/usr/share", and "/usr/share/doc".

If you are interested in more advanced topics like the set-uid, set-gid, sticky bits and octal numbers, try reading the FreeBSD manual page at http://www.FreeBSD.org/cgi/man.cgi (type "chmod" in the form and submit).

A useful option is:

-R

recursively change or set permissions on an entire directory tree

Examples:

We wrote a shell script called "configure". We make it executable ("+x") and then execute it as a command. Usually, "+x" is the same as "u+x" or "ug+x", depending on the status of the *file mode creation mask*.

```
$ chmod +x configure
$ ./configure
```

Allow the owning user to run "configure":

```
$ chmod u+x configure
```

Deny the group and other users from running "configure":

```
$ chmod go-x configure
```

For all users except the owner ("gw"), disable all access to "~/mail" and "~/private" ("-rwx"). This way, the contents are private and only their owner (or root) can access them.

```
$ chmod go-rwx ~/mail ~/private
```

Note that in the previous example, "-R" was not specified. By disabling the execute bit ("-x"), all files inside `~/{mail,private}` are protected even if their group and other read bits are enabled. Thus, simply moving some file from inside `~/{mail,private}` to some public place like `/tmp` can make the files available to other users again.

The "root" user wants to set up `/usr/local/src` so that all users in group "wsrc" (including "tux") can create files there. Root will continue to own the directory. This is done by changing the group of `/usr/local/src` to "wsrc" and then by granting to the group ("g") the read, write, and execute permissions ("+rwx").

```
# chown :wsrc /usr/local/src
# chmod g+rwx /usr/local/src
```

All Unix-like systems should allow all users to create temporary files in "/tmp" and "/var/tmp". Thus root gives everyone ("a", short for "ugo") all permissions ("+rwx") on the files.

```
# chmod a+rwx /tmp /var/tmp
```

The problem with the above is that because all users have write access to /tmp and /var/tmp, every user can delete and rename files, even ones not created by them. For example, "tux" could create "/tmp/socket.3908" and another user could delete it or rename it to "/tmp/garbage", thus annoying Tux. To keep temporary files safe, we use the *sticky bit* called "t". This limits the deletion and renaming of files in /tmp to root, the owner of /tmp (also root), and the owner of the file (Tux for "/tmp/socket.3908"). It does the same for /var/tmp. So what we should do is: # **chmod a+rwxt /tmp /var/tmp**

References

[1] http://www.freebsd.org/cgi/man.cgi?query=ls&sektion

[2] http://netbsd.gw.com/cgi-bin/man-cgi?ls+1

[3] http://www.openbsd.org/cgi-bin/man.cgi?query=ls&sektion

[4] http://www.gnu.org/software/coreutils/manual/html_node/coreutils_52.html

[5] http://www.gnu.org/software/coreutils/manual/coreutils.html

[6] http://www.freebsd.org/cgi/man.cgi?query=

[7] http://netbsd.gw.com/cgi-bin/man-cgi?

[8] http://www.openbsd.org/cgi-bin/man.cgi?query=

[9] http://www.gnu.org/software/coreutils/manual/html_node/coreutils_{{{coreutils}}}.html

[10] http://www.freebsd.org/cgi/man.cgi?query=pwd&sektion

[11] http://netbsd.gw.com/cgi-bin/man-cgi?pwd+1

[12] http://www.openbsd.org/cgi-bin/man.cgi?query=pwd&sektion

[13] http://www.gnu.org/software/coreutils/manual/html_node/coreutils_114.html

[14] http://www.freebsd.org/cgi/man.cgi?query=chroot&sektion

[15] http://netbsd.gw.com/cgi-bin/man-cgi?chroot+8

[16] http://www.openbsd.org/cgi-bin/man.cgi?query=chroot&sektion

[17] http://www.gnu.org/software/coreutils/manual/html_node/coreutils_145.html

[18] http://www.freebsd.org/cgi/man.cgi?query=rm&sektion

[19] http://netbsd.gw.com/cgi-bin/man-cgi?rm+1

[20] http://www.openbsd.org/cgi-bin/man.cgi?query=rm&sektion

[21] http://www.gnu.org/software/coreutils/manual/html_node/coreutils_68.html

[22] http://www.freebsd.org/cgi/man.cgi?query=shred&sektion

[23] http://netbsd.gw.com/cgi-bin/man-cgi?shred+1

[24] http://www.openbsd.org/cgi-bin/man.cgi?query=shred&sektion

[25] http://www.gnu.org/software/coreutils/manual/html_node/coreutils_69.html

[26] http://www.freebsd.org/cgi/man.cgi?query=chown&sektion

[27] http://netbsd.gw.com/cgi-bin/man-cgi?chown+8

[28] http://www.openbsd.org/cgi-bin/man.cgi?query=chown&sektion

[29] http://www.gnu.org/software/coreutils/manual/html_node/coreutils_79.html

[30] http://www.freebsd.org/cgi/man.cgi?query=chmod&sektion

[31] http://netbsd.gw.com/cgi-bin/man-cgi?chmod+1

[32] http://www.openbsd.org/cgi-bin/man.cgi?query=chmod&sektion

[33] http://www.gnu.org/software/coreutils/manual/html_node/coreutils_81.html

Guide to Unix/Commands/Finding Files

find

find searches a given path for a file or folder. The syntax is: find [path...] [expression...]

Examples: On some of the latest Unix-like OS's, the -print option is a default and can be omitted. The following command searches for the file 'grub.conf' starting at the root ('/') directory.

```
$ find / -name grub.conf
/etc/grub.conf
```

If you are not the administrator of the computer, you get error messages for all the directories you are not allowed to read. In this case do it like this for a bash shell:

```
$ find / -name grub.conf 2>/dev/null
/etc/grub.conf
```

Or like this for a csh/tcsh:

```
$ find / -name grub.conf >& /dev/null
/etc/grub.conf
```

If you want to ignore the case of the characters, you can use **-iname**. The following will find files with extensions *txt*, *TXT*, *Txt*, and so on:

```
$ find / -iname '*.txt'
/home/rtm/hacking.txt
/home/bok/Documents/MyLetter.TXT
/home/bok/tmp/found.tXt
```

The following command will search for all directories named 'local'.

```
$ find / -name local -type d
/usr/X11R6/lib/X11/fonts/local
/usr/local
/var/cache/man/local
/var/local
```

The above example combined two tests - filename (-name) and filetype (-type) - and returned files satisfying both (logical AND).

It's also possible to specify two or more tests, and return files satisfying any of them, with the **-o** (logical OR) directive. The tests to be ORed must be grouped together between **(...)** (which must be escaped with **** as parentheses are special to the shell). The following return files with the extensions *txt* or *doc*, as well as any file larger than 5MB (megabyte) in size.

```
$ find / \( -name '*.txt' -o -name '*.doc' -o -size +5M \)
/home/rtm/hacking.txt
/home/bok/report.doc
/home/bok/backup/may.tar.bz2
/home/koppe/BiggerThan5Megs.doc
```

Where the last file satisfied two tests.

Note: Each directive must be complete, tests cannot be omitted. E.g. **-name '*.txt' -o '*.doc'** (omitting the 2nd **-name**) is not valid.

Tips: Using 'exec' option executes certain commands for each file found by find:

```
$ find . -name '*bak' -exec rm -i {} \;
rm: remove regular empty file `./file1.bak'? y
rm: remove regular empty file `./file2.bak'? y
rm: remove regular empty file `./file3.bak'? y
```

Using 'ok' has same effect but it will prompt for every file:

```
$ find . -name '*~' -ok rm {} \;
< rm ... ./RMAIL~ > ? y
```

whereis

whereis searches the normal executable and man page locations for a specified file.

Examples:

```
$ whereis ls
ls: /bin/ls /usr/bin/ls /usr/man/man1/ls.1.gz
/usr/share/man/man1/ls.1.gz
```

which

which searches the locations in your PATH variable for a specified file. If you know a program is in your path (i.e you can run it) this is faster than **whereis**.

```
$ which pine
/usr/bin/pineddd
```

locate

locate finds all filenames that match the specified query.

Examples:

```
$ locate make.conf
/etc/make.conf
/etc/make.conf.orig
/etc/make.conf.example
/usr/qt/3/mkspecs/linux-g++/qmake.conf
/usr/share/man/man5/make.conf.5.gz
```

locate however, is a GNU software and the command is not a standard in traditional UNIX systems like Solaris. The **locate** command comes standard with Linux based systems.

Note that **locate** use a database of *already collected* filenames, that is typically updated once every 24 hours. As such, using **locate** will not correctly show newly created or deleted files/directories.

Guide to Unix/Commands/File Viewing

cat

cat displays the contents of a file to screen. It can also display multiple files con*cat*enated together, and using the shell, its standard output can be redirected from the screen to the file.

Some useful options are:

```
-n  line numbering
-s  squeeze sequences of blank lines into one
-v  show non-printing characters
```

Examples

```
$ cat example.txt
The contents of the file  example.txt are now
displayed.
```

```
$ cat -n example.txt
     1  The contents of the file
     2  example.txt are now
     3  displayed.
```

Concatenating multiple files (in this example, the same file twice):

```
$ cat example.txt example.txt
The contents of the file
example.txt are now
displayed.
The contents of the file
example.txt are now
displayed.
```

Concatenate the files, but use the shell to redirect the result to a new file.

```
$ cat example.txt example.txt > double.txt
```

We can concatenate binary files, too. Some programs limit downloads to 2 GB; this is bad for larger files like some DVD images. Suppose a web site helpfully divides the file "sodalinux.dvd" into three parts, 2 GB or less, for downloading and later concatentation. We combine the files, and use the > shell redirection output to put the DVD image in a file:

```
$ cat sodalinux.dvd1 sodalinux.dvd2 sodalinux.dvd3 > sodalinux.dvd
```

If we want to type less, then most shells also allow this:

```
$ cat sodalinux.dvd{1,2,3} > sodalinux.dvd
```

The -v option is useful for viewing control characters embedded mostly in text. In this example, the file "/usr/share/man/cat1/pax.0" is mostly text but contains control characters which the pagers "less" and "more" use to make text bold. Using cat -v we can see the control characters. Here are the first four lines:

```
$ cat -v /usr/share/man/cat1/pax.0 | head -4
PAX(1)                     OpenBSD Reference Manual
```

```
                    PAX(1)

N^HNA^HAM^HME^HE
     p^Hpa^Hax^Hx - read and write file archives and copy directory
hierarchies
```

Using "cat" with no arguments makes it copy standard input to standard output. Combined with shell redirection, this makes it easy to write a very short text file. All one needs to know is to press Control-D (^D) to indicate end of input, finish the file, and return to the shell. Here is how to write "example.txt":

```
$ cat > example.txt
The contents of the file
example.txt are now
displayed.
^D $
```

If you put "cat" with no arguments in a pipe, it only copies standard input to standard output. This might seem useless. For example, the following two pipes have the same function:

```
$ dmesg | less
$ dmesg | cat | less
```

However, "cat" can be used as insulation to make programs think that they are not running on terminals. In the next example, GNU bc does not print its copyright message on startup. We enter one calculation ("3 + 9") and then quit (^D):

```
$ bc | cat
3 + 9
12
^D
```

tac

tac (`cat` spelled backwards) works like `cat`, but reverse the order of the lines (last line is written out first). When multiple files are given, they are printed out in the order they appear.

```
$ tac foo
Third Line
Second Line
First Line
```

more

more paginates output. The problem with "cat" is that if a file is too long, then it falls beyond the top of the screen. The job of "more" is to stop and wait when it fills the screen. Most users find it easier to use "less", but on some systems "more" has all of the features of "less".

Keys:

- *return* read next line
- *space bar* read next screen
- *q* quit

Examples: The pager will act like "cat" if the file is short enough.

```
$ more hello.txt
Hello World
```

less

less paginates output. The program is called "less" because of the joke that "less is more", "less" actually has several features which "more" lacks.

Keys:

- *h* read help. You might forget the other commands, but remember this one!
- *j* go down one line. The down-arrow key might also work.
- *k* go up one line. The up-arrow key might also work.
- *d* go down one-half screen.
- *u* go up one-half screen.
- *f* go forward one screen.
- *b* go back one screen.
- *p* return to the top of the file.
- *q* quit the pager.

Number arguments:

- *0* through *9*: type a number. The number will be used as the argument N to the next command.
- *j* go down N lines.
- *k* go up N lines.
- *p* jump to the N% position, where 0% is the first line and 100% the last line of the file.

Examples:

Read some file:

```
$ less example.txt
```

Pipe "dmesg" into "less" so that the dmesg does not scroll off the screen:

```
$ dmesg | less
```

od

od is a utility that lets you view binary files.

Examples:

View a file in octal format:

```
$ od wordlist.dat
0000000 064506 071562 006564 051412 061545 067157 006544 040412
0000020 070160 062554 005015 072522 061155 062554 005015 061501
```

View a file in hex format:

```
$ od -x wordlist.dat
0000000 6946 7372 0d74 530a 6365 6e6f 0d64 410a
0000020 7070 656c 0a0d 7552 626d 656c 0a0d 6341
```

View a file in character format:

```
$ od -c wordlist.dat
0000000   F   i   r   s   t  \r  \n   S   e   c   o   n   d  \r  \n   A
```

```
0000020   p   p   l   e  \r  \n   R   u   m   b   l   e  \r  \n
```

head

head displays 10 lines from the head (top) of a given file

Examples:

```
$ head wordlist.dat
First
Second
Third
Fourth
Fifth
Sixth
Seventh
Eighth
Ninth
Tenth
```

Display the top two lines:

```
$ head -2 wordlist.dat
First
Second
```

tail

tail displays last 10 lines of the file

Examples:

```
$ tail wordlist.dat
Ninety-First
Ninety-Second
Ninety-Third
Ninety-Fourth
Ninety-Fifth
Ninety-Sixth
Ninety-Seventh
Ninety-Eighth
Ninety-Ninth
One Hundredth
```

Display the bottom two lines:

```
$ tail -2 wordlist.dat
Ninety-Ninth
One Hundredth
```

Tips: The -f option displays the tail, then waits for and displays any new options to the file. This is normally used to watch log files. (The next example has only three lines from tail, but the 80-column terminal was too narrow, so the lines were broken into five lines.)

```
$ tail -f /var/log/messages
Apr 14 00:05:33 redserver sshd[1575]: Accepted password for rumbear
from 24.52.1
45.23 port 33372 ssh2
Apr 14 00:05:34 redserver sshd[1594]: subsystem request for sftp
Apr 14 00:06:35 redserver sshd[1594]: Received disconnect from
242.122.35.47: 11
```

: Disconnect requested by Windows SSH Client.

Guide to Unix/Commands/File Editing

Basic editing of system files.

pico

Nano is a clone of Pico. Pico (text editor) [1] is an easy-to-learn text editor originally designed for composing e-mail in Pine.

zile

A light-weight, feature-reduced clone of emacs.

vi

A powerful editor based on ex. For details see the Wikibooks Learning the vi editor.

emacs

A very powerful editor that is used by many programers.

References

[1] http://en.wikipedia.org/wiki/Pico_%28text_editor%29

Guide to Unix/Commands/File Compression

gzip

gzip compresses files. Each single file is compressed into a single file. The compressed file consists of a GNU zip header and deflated data.

If given a file as an argument, gzip compresses the file, adds a ".gz" suffix, and deletes the original file. With no arguments, gzip compresses the standard input and writes the compressed file to standard output.

Some useful options are:

```
-c  Write compressed file to stdout. Do not delete original file.
-d  Act like gunzip.
-1  Performance: Use fast compression (somewhat bigger result)
-9  Performance: Use best compression (somewhat slower)
```

Examples:

Compress the file named `README`. Creates `README.gz` and deletes `README`.

```
$ gzip README
```

Compress the file called `README`. The standard output (which is the compressed file) is redirected by the shell to `gzips/README.gz`. Keeps `README`.

```
$ gzip -c README > gzips/README.gz
```

Use gzip without arguments to compress `README`.

```
$ < README gzip > gzips/README.gz
```

gunzip

gunzip uncompresses a file that was compressed with "gzip" or "compress". It tries to handle both the GNU zip format of gzip and the older Unix compress format. It does this by recognizing the extension (".gz" or ".Z" or several others) of a file.

Some useful options are:

```
-c  Write uncompressed data to stdout. Do not delete original file.
```

Undo the effect of **`gzip README.gz`** by replacing the compressed version of the file with the original, uncompressed version. Creates `README` and deletes `README.gz`.

```
$ gunzip README.gz
```

Write the uncompressed contents of `README.gz` to standard output. Pipe it into a pager for easy reading of a compressed file.

```
$ gunzip -c README.gz | more
```

Another way to do that is:

```
$ gunzip < README.gz | more
```

Some people name files `package.tgz` as short for `package.tar.gz`.

zcat

zcat is same thing as **`uncompress -c`**, though on many systems it is actually same as "gzcat" and **`gunzip -c`**.

gzcat

gzcat is same as **`gunzip -c`** which is **`gzip -dc`**.

tar

tar archives without compression.

An archive contains one or more files or directories. (If archiving multiple files, it might be better to put them in one directory, so extracting will put the files into their own directory.)

Modes:

```
-c  create an archive (files to archive, archive from files)
-x  extract an archive (archive to files, files from archive)
```

Options:

```
-f FILE  name of archive - must specify unless using tape drive for
archive
-v       be verbose, list all files being archived/extracted
-z       create/extract archive with gzip/gunzip
-j       create/extract archive with bzip2/bunzip2
```

Examples:

Compress (gzip) and package (tar) the directory `myfiles` to create `myfiles.tar.gz`:

```
$ tar -czvf myfiles.tar.gz myfiles
```

Uncompress (gzip) and unpack compressed package, extracting contents from `myfiles`:

```
$ tar -xzvf myfiles.tar.gz
```

There are two different conventions concerning gzipped tarballs. One often encounters .tar.gz. The other popular choice is .tgz. Slackware packages use the latter convention.

If you have access to a tape device or other backup medium, then you can use it instead of an archive file. If the material to be archived exceeds the capacity of the backup medium, the program will prompt the user to insert a new tape or diskette.

Use the following command to back up the `myfiles` directory to floppies:

```
$ tar -cvf /dev/fd0 myfiles
```

Restore that backup with:

```
$ tar -xvf /dev/fd0
```

You can also specify standard input or output **`-f -`** instead of an archive file or device. It is possible to use copy between directories by piping two "tar" commands together. For example, suppose we have two directories, `from-stuff` and `to-stuff`

```
$ ls -F
from-stuff/
to-stuff/
```

As described in *Running Linux*, one can mirror everything from `from-stuff` to `to-stuff` this way:

```
$ tar cf - . | (cd ../to-stuff; tar xvf -)
```

Reference: Welsh, Matt, Matthias Kalle Dalheimer and Lar Kaufman (1999), *Running Linux.* Third edition, O'Reilly and Associates.

cpio

cpio is used for creating archives. When creating an archive, a list of files is fed to it's standard-input (rather than specifying the files on the commandline). This file-list is typically created by **ls**, **find** or **locate** and then piped directly to **cpio**; but it can also first be filtered/edited with commands like ***grep**, **sed**, **sort** and others. A (pre-edited) list stored as a file can also be used, by using **cat** to feed the pipeline or simply by redirecting the shell's standard-input (<).

cpio works in one of three modes:

- **cpio -o** - *Copy-Out mode*: Files are copied **out** from the filesystem to *create* an archive. Usually the archive is created by simply using the shell to redirect cpio's output to a file (with **>**).
- **cpio -i** - *Copy-In mode*: Files from an existing archive are *restored/extracted*, and copied back **in** to the filesystem.
- **cpio -p** - *Pass-Through mode*: **cpio** is used to copy files from one location in the directory-tree to another, without an actual archiving being made.

In addition comes:

- **cpio -t** - *List archive*: The content of an archive is listed without extracting it.
- **cpio -tv** - Here the verbose-option (**-v**) will cause a "long listing", with permissions, size and ownership.

Adding the verbose-option (**-v**) in Copy-In, Copy-Out and Pass-Through mode, will cause **cpio** to list the files as they're extracted/archived/copied.

Using **ls** to create an archive (verbosely) with all doc-files in the current directory:

```
$ ls *.doc | cpio -ov > word-docs.cpio
```

Using **find** to create an archive with all txt-files in and below the current directory:

```
$ find . -name "*.txt" | cpio -ov > text-files.cpio
```

Using **find** and **fgrep** to create an archive of just the txt-files containing the word *wiki* (any case):

```
$ find . -name "*.txt" -exec fgrep -l -i "wiki" {} \; | cpio -ov >
wiki.cpio
```

For **fgrep** the option **-i** means "ignore case", and the option **-l** cause it to just list the filenames of files matching the pattern.

Using an existing list of files:

```
$ cpio -ov < file-list.txt > archive.cpio
```

Using several list of files, but first after **sort**-ing and **uniq**-ing them:

```
$ cat files1 files2 files3 | sort | uniq | cpio -ov > myfiles.cpio
```

To add more files, use the append-option (**-A**). Specify the file with the file-option (**-F**):

```
$ cat files4 | cpio -ovA -F myfiles.cpio
```

To extract files (being verbose):

```
$ cpio -iv < myfiles.cpio
```

cpio doesn't create directories by default, so use the option **-d** to make it.

To extract files, while creating directories as needed:

```
$ cpio -ivd < myfiles.cpio
```

To list the content of an archive, short listing:

```
$ cpio -t < myfiles.cpio
```

To list the content of an archive, long listing:

```
$ cpio -tv < myfiles.cpio
```

pax

pax is like "tar" but with different command-line syntax. Because "pax" does not assume the tape device, some prefer it to "tar".

bzip2

bzip2 and **bunzip2** are similar to "gzip"/"gunzip" but with a different compression method. Compression is generally better but slower than "gzip". Decompression is somewhat fast.

An option of *-1* through *-9* can be used to specify how good **bzip2** should compress. The number tells how large "chunks" in steps of 100kB should compress at a time, so using **`bzip2 -5 foo.bar`** will compress foo.bar in chunks of 500kB each. Generally, larger chunks means better compression (but probably slower). Only *undamged* "chunks" can be recoverd with **`bzip2recover`** from a damaged bzip2-file, so if you've compressed 900kB chunks, you'll loose 900kB of your file if one chunk is damaged - but only 100kB if you used 100kB chunks (`bzip2 -1`). By default **bzip2** uses 900kB chunks for best possible compression.

bzcat is same as **`bunzip2 -c`** which is **`bzip2 -dc`**.

zip

zip is an archive which compresses the members individually. (Imagine gzip of every file before tar-ing them, but with a different format.) The "zip" format is a common archiving file format used on Microsoft Windows PCs.

Like for *gzip* the quality of the compression can be specified by giving a number between 1 and 9 as an option (e.g. **`zip -5`**). 1 is quickest, but gives a low-quality compression. 9 gives the highest quality of compression, but is slow. In addition 0 can be used (i.e. **`zip -0`**) to specify that the files should just be "stored" and *not* compressed (a compression of 0%), thus making it possible to use **zip** to make uncompressed archives.

Note that a **zip**-archive contains *individualy compressed* files collected into a single file. This is the oposite of how it's done for most other compressed *Unix*-archives (e.g. `tar.gz` and `tar.bz2`), where the files/directories are *first* collected into a single file -- an archive (e.g. **cpio** or **tar**), and *then* this *single file* is compressed (e.g. using **gzip** or **bzip2**).

compress

compress is a compressed file format that is popular on UNIX systems. Files compressed with **compress** will have a ".Z" extension appended to its name.

Guide to Unix/Commands/File Analysing

file

file displays the file type. To get the mimetype, use the -i option.

Examples

```
$ file Unix.txt
Unix.txt: ASCII text
```

```
$ file -i Unix.txt
Unix.txt: text/plain; charset=us-ascii
```

wc

wc tells you the number of lines, words and characters in a file.

Examples:

```
$ wc hello.txt
2       6      29 hello.txt
```

```
$ wc -l hello.txt
2 hello.txt
```

```
$ wc -w hello.txt
6 hello.txt
```

```
$ wc -c hello.txt
29 hello.txt
```

cksum

cksum gives you the CRC checksum of some files.

Checksums can be used to protect against accidental modifications to files: if the checksum has not changed, then the file is probably undamaged. The default CRC checksum is not cryptographic.

Cryptographic checksums are those checksums which protect against both accidental modifications and malicious modifications. Use these to verify that there is no trojan inserted into your file. The "md5" algorithm is beginning to show weaknesses against attacks, so "sha1" is preferred.

Examples:

```
$ cksum /etc/passwd
3052342160 2119 /etc/passwd
```

Some "cksum" implementations provide other algorithms, such as "md5" and "sha1":

```
$ cksum -a sha1 /etc/passwd
SHA1 (/etc/passwd) = 816d937ca4cdb4dee92d5002610fae63b639d224
```

Some "cksum" implementations let you take checksums of strings specified as arguments:

```
$ cksum -s 'Guide to UNIX'
2195826759 13 Guide to UNIX
$ cksum -a sha1 -s 'Guide to UNIX'
```

SHA1 ("Guide to UNIX") = 0e9c1779e61c7fdb473d2e55eb878a82c37eecea

Guide to Unix/Commands/Multiuser Commands

who

who gives information about the users logged into the machine. The information includes the user's terminal, login date ,login time and the location they are connecting from.

Examples:

```
$ who
alice    pts/0        Mar 23 08:05 (213.23.423.24)
bob      pts/2        Apr 10 22:06 (domain.aol.com)
carol    pts/3        Apr 10 18:34 (space.com)
```

The option "-w" shows wheter or not a user's tty is accessable with commands like **write** or **talk**. **+** indicates that the tty is accessable, and **-** that it's not:

```
$ who -w
root     - tty3         Jan 19 02:26
koppe    - tty4         Jan 19 17:10
bok      + pts/1        Jan 19 23:03
```

Using "who" with two non-option words gives your username. On some systems, this gives your actual username, and using "su" or "sudo" to switch user does not change this name.

```
$ who am i
puffy
```

On other systems, this gives more information:

```
$ who am i
puffy    ttyp2    Oct 27 10:08
```

finger

finger finds out information about a user. If the user has created a .plan (several lines) and/or a .project (one line) file in their home directory this will also be displayed.

Examples:

```
$ finger alice
Login: alice                    Name: Alice Makemerry
Directory: /home/alice          Shell: /bin/bash
On since Sat Apr 10 18:34 (BST) on pts/3 from ip.fakedomain.com

  1 hour 25 minutes idle
Mail last read Sat Apr 10 23:57 2004 (BST)
No Plan.
```

su

su switch user

Examples:

Become another user:

```
user> su bob
Password:
bob>
```

Become *root*... then become another user:

```
user> su
Password:
root#
root# su bob
bob>
(Note: root is not asked for password to become bob!)
```

Switching user and using the new user's enviroment (shell, shell-variables, home-directory) as if after a normal log-in:

```
user> su - bob
Password:
bob$
```

Run a program as another user (as *root* unless otherwise specified):

```
user> su -c 'apt-get update'
Password:
```

Note: The permission and owner/group of **su** - as well as other config-files (typically */etc/su* and */etc/login.defs*) - may prevent users not belonging to certain groups from switching user even with the correct password, or even being able to execute **su** at all (E.g. in BSD it's traditionally been restricted to members of the *wheel*-group only).

Guide to Unix/Commands/Self Information

whoami

whoami tells you your current username.

Examples:

```
$ whoami
abicool
```

groups

groups states the groups the current user is a member of

Examples:

```
$ groups
wheel slocate www
```

id

id gives you the same information as the **whoami** and **groups** commands, but also includes the user id (uid) and group id (gid) integers associated with the login.

Examples:

```
$ id
uid=3426(alice) gid=10(wheel)
groups=10(wheel),21(slocate),401(www)
```

tty

tty tells you the terminal device that is assigned to your interactive login. The tty represents your console device, network connection ("ssh", ...), or terminal emulator process ("xterm", "konsole", ...).

Examples:

```
$ tty
```

/dev/pts/14

Guide to Unix/Commands/System Information

uptime

uptime tells you how long the computer has been running since its last reboot or power-off.

Example:

```
$ uptime
22:27:49 up 10:14,  2 users,  load average: 0.03, 0.32, 0.28
```

uname

uname displays the system information such as hardware platform,system name and processor, Operating System type.

Example:

```
$ uname -a
Linux DarkBox 2.4.27-1-k6 #1 Wed Apr 14 19:00:29 UTC 2004 i586 GNU/Linux
```

dmesg

dmesg display the messages from the kernel, since boot.

Example:

```
$ dmesg
```

Tips:

While a UNIX system is booting, usually a lot of messages flash on the console screen in rapid succession; to view those messages after the system is booted, use the following command:

```
$ dmesg | less
```

Using a command option, **dmesg** can filter the kernel messages, based on priority. The '-n 1' arguments will display only the panic messages:

```
$ dmesg -n 1
```

free

free display used and free memory

Example:

```
$ free
             total       used       free     shared    buffers
cached
Mem:        123260     119540       3720          0       8752
58096
-/+ buffers/cache:      52692      70568
Swap:       369452      63212     306240
```

Display in human readable form using MegaByte block sizes:

```
$ free -m
             total       used       free     shared    buffers
```

```
cached
Mem:           120        116          3          0          8
56
-/+ buffers/cache:          51         68
Swap:          360         61        299
```

Tips: Display system memory usage every 5 seconds, use Ctl+c to exit:

```
$ free -m -s 5
             total       used       free     shared    buffers
cached
Mem:           120        116          3          0          8
56
-/+ buffers/cache:          51         68
Swap:          360         61        299
```

```
             total       used       free     shared    buffers
cached
Mem:           120        116          3          0          8
55
-/+ buffers/cache:          52         68
Swap:          360         61        299
```

vmstat

vmstat displays a compact summary of overall system activity (processes, memory, and cpu information).

Example:

```
$ vmstat
procs -----------memory---------- ---swap-- -----io---- --system--
----cpu----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in    cs us
sy id wa
 2  0  63108   4484   7432  56480    8   11    93    45 1110   622 41
11 48  0
```

Tips: Print out vmstat summaries every two seconds, for five iterations.

```
$ vmstat -n 2 5
procs -----------memory---------- ---swap-- -----io---- --system--
----cpu----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in    cs us
sy id wa
 1  0  63100   5172   7440  55892    8   10    90    44 1110   622 41
10 49  0
 2  0  63100   5168   7440  55892    0    0     0     0 1120   559 32
3 65  0
 1  0  63100   5160   7440  55892    0    0     0     0 1111   499  8
6 86  0
 1  0  63100   5160   7440  55892    0    0     0     0 1113   505 12
3 85  0
```

```
 1  0  63100   5168   7440  55892    0    0     0     0 1121   532 20
3 77  0
```

top

top displays system process in real time

Example:

```
$ top
Tasks:  50 total,   2 running,  45 sleeping,   2 stopped,   1 zombie
Cpu(s):  40.9% user,  10.5% system,   0.0% nice,  48.7% idle
Mem:    123260k total,   119508k used,     3752k free,     7420k buffers
Swap:   369452k total,    63036k used,   306416k free,    57212k cached
```

```
  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND

 5340 arky    15   0   968  968  780 R 13.8  0.8   0:00.22 top

 1408 root       6 -10 23712 6692 3252 S  1.5  5.4   3:39.58 [XFree86]

    1 root       8   0   500  472  448 S  0.0  0.4   0:00.31 init [2]

    2 root       9   0     0    0    0 S  0.0  0.0   0:01.60 [keventd]

    3 root      19  19     0    0    0 S  0.0  0.0   0:00.02
[ksoftirqd_CPU0]
    4 root       9   0     0    0    0 S  0.0  0.0   0:07.03 [kswapd]

    5 root       9   0     0    0    0 S  0.0  0.0   0:00.00 [bdflush]

    6 root       9   0     0    0    0 S  0.0  0.0   0:00.44 [kupdated]

  154 root       9   0     0    0    0 S  0.0  0.0   0:00.00 [khubd]

  562 root       9   0   604  588  508 S  0.0  0.5   0:05.09
/sbin/syslogd
  565 root       9   0  1152  492  448 S  0.0  0.4   0:01.24
/sbin/klogd -c 3
  .........
```

df

df reports the amount of free disk space available on each partition.

```
$ df
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/md0               5763508    207380   5263352   4% /
/dev/md1              78819376  13722288  61093296  19% /home
/dev/md4              23070564   4309572  17589056  20% /usr
/dev/md2               5763508   1757404   3713328  33% /var
```

```
/dev/md3               2877756    334740   2396832  13% /tmp
```

To report the number of free i-nodes

```
$ df -i
Filesystem            Inodes   IUsed   IFree IUse% Mounted on
/dev/hda3             321952   32558  289394   11% /
/dev/hda2              67320      67   67253    1% /boot
/dev/mapper/vg00-home
                      372352   34227  338125   10% /home
/dev/mapper/vg00-tmp  242784   11649  231135    5% /tmp
/dev/mapper/vg00-usr 1821568  208669 1612899   12% /usr
/dev/mapper/vg00-var 1282560   75704 1206856    6% /var
```

Reports disk usage in human readable format with block-sizes in Kilo,Mega,Gigabytes. This option is specific for Wikipedia:GNU version of **df**.

```
$ df -h
Filesystem            Size  Used Avail Use% Mounted on
/dev/hda1             2.3G  2.1G  133M  95% /
tmpfs                  61M  8.0K   61M   1% /dev/shm
/dev/hda2             2.0G  1.8G  113M  94% /usr
```

In some of Unix systems (SYS V family ie. HP-UX) **df** displays the information in a different way:

```
$ df
/home                (/dev/vg01/lvol2     ):   262478 blocks   2647709
i-nodes
/tmp                 (/dev/vg00/lvol5     ):   952696 blocks    125941
i-nodes
/usr                 (/dev/vg00/lvol6     ):   132842 blocks     17633
i-nodes
/var                 (/dev/vg00/lvol7     ):   131704 blocks     17288
i-nodes
/stand               (/dev/vg00/lvol1     ):    47548 blocks     13390
i-nodes
/                    (/dev/vg00/lvol3     ):   160772 blocks     21215
i-nodes
```

In such cases try to use bdf command.

hostname

hostname displays and set system host name

Example:

Display the host name:

```
$ hostname
Darkstar
```

Display the IP address of the system:

```
$ hostname -i
61.95.196.52
```

Set the host name of the system to 'DarkHorse':

```
$ hostname DarkHorse
```

DarkHorse

Guide to Unix/Commands/Networking

NAME

```
      ifconfig - configure a network interface
```

SYNOPSIS

```
      ifconfig [-v] [-a] [-s] [interface]
      ifconfig [-v] interface [aftype] options | address ...
```

DESCRIPTION

```
      Ifconfig  is  used to configure the kernel-resident network
interfaces.
      It is used at boot time to set up interfaces as necessary.  After
 that,
      it  is  usually  only  needed  when  debugging or when system
tuning is
      needed.
```

```
      If no arguments are given, ifconfig displays the  status  of  the
  cur-
      rently  active interfaces.  If a single interface argument is
given, it
      displays the status of the given interface only; if a single  -a
  argu-
      ment  is  given,  it  displays the status of all interfaces, even
 those
      that are down.  Otherwise, it configures an interface.
```

Address Families

```
      If the first argument after the interface name  is  recognized
as  the
      name  of  a  supported  address family, that address family is
used for
      decoding and displaying all protocol  addresses.   Currently
supported
      address  families  include  inet  (TCP/IP, default), inet6
(IPv6), ax25
      (AMPR Packet Radio), ddp (Appletalk Phase  2),  ipx  (Novell
IPX)  and
```

```
      netrom (AMPR Packet radio).
```

OPTIONS

```
      -a     display  all  interfaces  which are currently available,
even if
             down
```

```
      -s     display a short list (like netstat -i)
```

```
      -v     be more verbose for some error conditions
```

```
      interface
             The name of the interface.  This is usually a driver  name
  fol-
             lowed  by a unit number, for example eth0 for the first
Ethernet
             interface. If your kernel supports  alias  interfaces,
you  can
             specify  them  with  eth0:0 for the first alias of eth0.
You can
             use them to assign a second address. To delete an  alias
inter-
             face use ifconfig eth0:0 down aliases are deleted, if you
delete
             the first (primary).
```

```
      up     This flag causes the interface to be activated.  It  is
implic-
             itly specified if an address is assigned to the interface.
```

```
      down   This  flag causes the driver for this interface to be shut
 down.
```

```
      [-]arp Enable or disable the use of the ARP protocol on this
interface.
```

```
      [-]promisc
             Enable  or  disable  the  promiscuous mode of the
interface.  If
             selected, all packets on the network will  be  received
by  the
             interface.
```

```
      [-]allmulti
             Enable  or  disable all-multicast mode.  If selected, all
multi-
             cast packets on the network will be received by the
interface.
```

```
      metric N
             This parameter sets the interface metric.
```

```
      mtu N  This parameter sets the Maximum Transfer Unit (MTU) of an
inter-
             face.

      dstaddr addr
             Set  the  remote  IP  address for a point-to-point link
(such as
             PPP).  This keyword is now obsolete; use the pointopoint
keyword
             instead.

      netmask addr
             Set the IP network mask for this interface.  This value
defaults
             to the usual class A, B or C network mask (as derived
from  the
             interface IP address), but it can be set to any value.

      add addr/prefixlen
             Add an IPv6 address to an interface.

      del addr/prefixlen
             Remove an IPv6 address from an interface.

      tunnel aa.bb.cc.dd
             Create  a new SIT (IPv6-in-IPv4) device, tunnelling to the
 given
             destination.

      irq addr
             Set the interrupt line used by this device.  Not all
devices can
             dynamically change their IRQ setting.

      io_addr addr
             Set the start address in I/O space for this device.

      mem_start addr
             Set  the  start  address  for shared memory used by this
device.
             Only a few devices need this.

      media type
             Set the physical port or medium type to be used by  the
device.
             Not all devices can change this setting, and those that
can vary
             in what values  they  support.   Typical  values  for
type  are
             10base2 (thin Ethernet), 10baseT (twisted-pair 10Mbps
Ethernet),
```

```
             AUI (external transceiver) and so on.  The special  medium
  type
             of  auto can be used to tell the driver to auto-sense the
media.
             Again, not all drivers can do this.
```

```
      [-]broadcast [addr]
             If the address argument is given,  set  the  protocol
broadcast
             address  for  this  interface.   Otherwise,  set  (or
clear) the
             IFF_BROADCAST flag for the interface.
```

```
      [-]pointopoint [addr]
             This keyword enables the point-to-point mode  of  an
interface,
             meaning  that  it  is  a  direct  link between two
machines with
             nobody else listening on it.
             If the address argument is also given, set the protocol
address
             of  the  other  side of the link, just like the obsolete
dstaddr
             keyword does.  Otherwise, set or clear the IFF_POINTOPOINT
  flag
             for the interface.
```

```
      hw class address
             Set the hardware address of this interface, if the device
driver
             supports this operation.  The keyword must be  followed
by  the
             name of the hardware class and the printable ASCII
equivalent of
             the hardware  address.   Hardware  classes  currently
supported
             include  ether  (Ethernet), ax25 (AMPR AX.25), ARCnet and
netrom
             (AMPR NET/ROM).
```

```
      multicast
             Set the multicast flag on the interface. This  should  not
  nor-
             mally   be   needed  as  the  drivers  set  the  flag
correctly
             themselves.
```

```
      address
             The IP address to be assigned to this interface.
```

```
      txqueuelen length
             Set the length of the transmit queue of the device. It is
useful
             to  set  this  to  small  values  for slower devices with
a high
             latency (modem links, ISDN) to prevent fast bulk transfers
  from
             disturbing interactive traffic like telnet too much.
```

NOTES

```
      Since kernel release 2.2 there are no explicit interface
statistics for
      alias interfaces anymore.  The  statistics  printed  for  the
original
      address  are shared with all alias addresses on the same device.
If you
      want per-address statistics you should add  explicit  accounting
  rules
      for the address using the ipchains(8) or iptables(8) command.
```

```
      Since  net-tools  1.60-4  ifconfig  is printing byte counters and
 human
      readable counters with IEC 60027-2 units. So 1 KiB are 2^10 byte.
 Note,
      the  numbers  are  truncated to one decimal (which can by quite a
 large
      error if you consider 0.1 PiB is 112.589.990.684.262 bytes :)
```

```
      Interrupt problems with Ethernet device drivers fail with EAGAIN
(SIOC-
      SIIFLAGS:  Resource temporarily unavailable) it is most likely a
inter-
      rupt conflict.  See  http://www.scyld.com/expert/
irq-conflict.html  for
      more information.
```

Guide to Unix/Commands/Process Management

nohup

nohup lets you run a program in a way which makes it ignore hangup signals. This can be used to make a program continue running after a user has logged out. The output of the program is redirected from the standard output to the file nohup.out.

Examples:

```
$ nohup wget http://foo.org/foo_list.gz'''
nohup: appending output to `nohup.out'
$
```

ps

ps displays a list of current processes and their properties.

Examples:

Processes owned by the current user:

```
$ ps
  PID TTY          TIME CMD
17525 pts/0    00:00:00 su
17528 pts/0    00:00:00 bash
17590 pts/0    00:00:00 ps where pid is process id.
```

All processes:

```
$ ps -A
```

kill

kill is used to send termination signals to processes.

Examples

To send the kill signal to the processes with process id 17525,

```
$ kill -9 17525
```

To send the kill signal to all processes,

```
$ kill -9 -1
```

see Guide to Unix/Commands/Process Management/Kill

pgrep

pgrep search and kill system processes

Example: Check if apache webserver is running.

```
$ pgrep -f apache
 5580
 5581
 5582
 5583
```

```
 5584
 5585
or
$ svcs -a | grep -i apache.
```

Stop xterm program with 'pkill' program:

```
$ pkill -9 xterm
```

Tips: Display all the process of a user

```
$ pgrep -l -u arky
894 bash
895 bash
897 bash
898 bash
899 bash
1045 links
1396 startx
1407 xinit
1411 openbox
1412 xterm
1413 xfaces
1414 xsetroot
1415 emacs
```

pidof

pidof display Process ID (PID) of a task

Example: Display the PID of emacs process:

```
$ pidof emacs
1415
```

killall

killall kill a process by name

Example:

Kill the 'xfaces' program: $ **killall xfaces**

Guide to Unix/Commands/Devices

fuser

fuser tells you what process is using an indicated filesystem object (ordinary file, device, etc.)

```
$ fuser /dev/dsp
/dev/dsp:             8369
```

lsof

lsof lists all open files,is more detailed than fuser.

Example:

```
$ lsof /dev/dsp
COMMAND  PID USER   FD   TYPE DEVICE SIZE NODE NAME
mplayer 8406 alex    7w   CHR   14,3       389 /dev/sound/dsp
```

Tips:

Using -i 4 option will report all programs currently using IPv4 network, it is useful for watching the programs accessing the network and Internet resources.

```
$ lsof -i 4
COMMAND    PID   USER   FD   TYPE DEVICE SIZE NODE NAME
btdownloa 2618 arky    3u  IPv4   9524       TCP *:6886 (LISTEN)
btdownloa 2618 arky    6u  IPv4   9544       TCP
dsl-KK-229.53.101.203.ttel.net:1539->cpc1-leed3-3-0-cust10.ldst.cable.ntl.com:59074
 (ESTABLISHED)
```

fstat

fstat lists all open files.

The previous two commands (fuser and lsof) do not exist on all systems. The 4.3BSD-Tahoe system introduced the "fstat" command that is found on many *BSD systems. Unlike the previous two commands, it seems not to know the exact path of each file, but only what filesystem it is on?

Some options are:

```
 -p PROCESSID  show open files of this process
 -u USERNAME   show open files of this user
```

Examples: Lists every open file by every user, including root! Pipe it into a pager.

```
$ fstat | less
```

Get the process ID of the running Bourne shell and then list the files it opened.

```
$ echo $$
5283
$ fstat -p 5283
USER     CMD          PID   FD MOUNT      INUM MODE       R/W    DV|SZ
...
```

The init process always has ID of 1. List its open files. In this example it only opened one file somewhere on the / filesystem.

```
$ fstat -v -p 1
USER     CMD          PID   FD MOUNT      INUM MODE       R/W    DV|SZ
root     init           1   wd /             2 drwxr-xr-x   r      512
```

Reference:

- FreeBSD manual page for fstat at http://www.FreeBSD.org/cgi/man.cgi?query=fstat

Guide to Unix/Commands/Kernel Commands

lsmod

lsmod lists the modules loaded by the Linux kernel.

modprobe

modprobe loads a Linux kernel module. You can specify only the name of the module, and modprobe will load it from the correct location and also load any dependent modules.

Many modules load automatically. For example, Linux loads a USB keyboard module when a USB keyboard is attached. It also loads the base USB modules as dependencies. Some modules must be loaded manually, and "modprobe" is the easiest way to do this.

sysctl

sysctl sets a parameter to change the behavior of the kernel. The available parameters vary by kernel, so check the man page for sysctl in your distribution.

Examples:

Check the setting for the "vm.swapencrypt.enable" parameter:

```
$ sysctl vm.swapencrypt.enable
vm.swapencrypt.enable=0
```

Root can set the parameter to 1.

```
$ sysctl vm.swapencrypt.enable=1
```

vm.swapencrypt.enable: 0 -> 1

Guide to Unix/Commands/compress Commands

command syntex- to compress

1. tar cvf file.tar source_file_directory

to uncompress

1. tar xvf tar_file_name

Guide to Unix/Commands/Miscellaneous

sync

sync write memory buffers to disk

Example: Sync has no options, doesn't display any messages

```
$ sync
```

Tips:

It is always good to type sync a couple of times, one the important functions of *sync* is to update your superblock information.

The *sync* calls sync Unix system call and exits with success code '0' or '1' if it fails. These exit codes stored in $? variable.

```
$ sync
$ echo $?
 0
```

The above example shows that *sync* was successful.

echo

echo outputs its parameters to the standard output.

Examples:

```
$ echo "hello world"
hello world
```

Tips: Some common echo usage:

Check a shell variable:

```
$ echo $EDITOR
emacs
```

Check the parameters passed in the previous command:

```
$ ls -l
 .........
$ echo $_
-l
```

Check the current parent process:

```
$ echo $0
bash
```

Check the exit code of the last command:

```
$ echo $?
0
```

Create a empty file (same as **touch /tmp/newfile**):

```
$ echo "" > /tmp/newfile
```

Create a new file with some text:

```
$ echo "exec fluxbox" > ~/.xinitrc
```

Add (append) a new line to end of file:

```
$ echo "A New Line" >> /tmp/newfile
```

cal

cal displays a calender for the current month. If the command is followed by a date (a month or a year) it will return a calender for that period.

Examples:

```
$ cal
     April 2004
Su Mo Tu We Th Fr Sa
              1  2  3
 4  5  6  7  8  9 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29 30
```

```
$ cal 01 2007
    January 2007
Su Mo Tu We Th Fr Sa
    1  2  3  4  5  6
 7  8  9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30 31
```

```
$ cal -3
Shows current, previous and next month (default on some implementations)
```

```
$ cal -3 04 2004
Shows April 2004, as well as the previous (March) and following (May)
month
```

```
$ cal -1
Shows the current month (default on some implementations)
```

```
$ cal 2004
Shows a calendar for the whole year 2004
```

```
$ cal -j
Shows this months calendar with day-of-year number (counted from
January 1st) rather than the date
```

Tips: The Gregorian Calendar was adopted in the British Empire in 1752. The 2nd day of September 1752 was immediately followed by the 14th day of September, as shown by the example below.

```
$ cal 9 1752
  September 1752
Su Mo Tu We Th Fr Sa
       1  2 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30
```

date

date displays the current date and time.

Example:

```
$ date
Mon Jun 26 12:34:56 CDT 2006
```

time

time time a program

Example:

```
$ time gcc apache.c

 real    0m1.818s
 user    0m0.770s
 sys     0m0.210s
```

from

from display the names of those who sent you mail recently

Example:

```
$ from
  From andy@box.po Sat Feb 05 08:52:37 2005
  From andy@box.po Sat Feb 05 08:53:52 2005
```

Count the number of mail in your mailbox

```
$ from -c
There are 2 messages in your incoming mailbox.
```

mail

mail allows you to read and write emails.

Example:

```
$ mail
No mail for user.
$ mail user2
Subject: What's up?
Hi user2, you can delete this rubbish by pressing 'd'.
Cc: user
```

Tips: Note that you need to press enter then ctrl+d to confirm.

```
$ mail
Mail version 8.1 6/6/93. Type ? for help.
"/var/spool/mail/user": 1 message 1 new
>N 1 user@unix.com Tue Jun 27 12:34 16/674 "What's up?"
&
```

Tips: Press enter to read.

clear

clear clears the screen.

Example:

```
$
```

PS1

PS1 is an environment variable which defines the shell prompt. If not defined, the prompt defaults to "$"

Example:

```
$ PS1='yes? '
yes? PS1='$ '
$
```

A more complicated example:

```
$ PS1='*\u@\H* \t \d [\W]\n$ '
*user@unix.com* 12:34:56 Tue Jun 27 [home]
```

$

Guide to Unix/Environment Variables

An **environment variable** is a setting normally inherited or declared when a shell is started. You can use shells to set variables; the syntax varies but Bourne shells use:

```
$ VARNAME=new value
$ export VARNAME
or
$ export VARNAME=new value
```

Each program started from that shell will have *VARNAME* set to *new value*. The names of environment variables are case-sensitive; by convention they are uppercase.

A **shell variable** is like an environment variable, except that it is not exported to new programs started from that shell. (You could export it, but normally you just write a shell initialisation script to set it in each shell.)

EDITOR

The editor program called by sudoedit, vipw, and other such programs when you tell them to edit a file.

Examples:

```
EDITOR=vi
```

```
EDITOR=emacs
```

Also see **VISUAL**.

HOME

The home directory of the user. Most programs use this shell variable to find your home, thus you can set this variable to override the setting in `/etc/passwd` for your home directory. This way, you can start programs that put dotfiles or other files in a different directory than your usual home directory.

In most shells, `~` refers to your home directory. In C shell, and some more recent versions of Bourne shell, `~tux` always refers to the home directory of user tux as specified in `/etc/passwd`, while `~` (without a username after it) always refers to the value of HOME, even if it differs from your home directory in `/etc/passwd`.

LOGNAME

The name of the user. This is an easy way for a user to get own username. However, programs must not trust this variable because it can be set to an arbitrary value.

Both LOGNAME and USER should be set to the username.

Examples:

```
LOGNAME=tux
```

```
LOGNAME=puffy
```

MAIL

The location of incoming local email. When **mail** or another local email reader inherits this environment variable, it uses this variable to find the inbox.

Some users do not have email at their local Unix box, but instead use the Internet to access their mail server, in which case the MAIL environment variable is irrelevant.

Many users do not have MAIL set, in which case the email reader uses the default setting. The default value for user "tux" would be `/var/mail/tux`, which is where many systems deliver mail.

MAILCHECK

This is a shell variable, not normally exported as an environment variable.

The frequency for which "bash" checks and alerts you for new local email.

PAGER

The pager called by man and other such programs when you tell them to view a file.

Examples:

```
PAGER=less
```

```
PAGER=more
```

PATH

A space or colon separated list of directories in which the shell searches for executables when a command is run without an absolute path. For example **ls** doesn't have an absolute path, but **/bin/ls** does).

Some systems set PATH using the system shell initialistation files, such as `/etc/profile` for Bourne shells. Some systems set PATH before this as part of the login procedure, for example in `/etc/login.conf` for OpenBSD systems. For example, a Linux box could set the PATH at login, then add `/usr/X11R6/bin to the path using /etc/profile`, then add `/home/ambler/bin` to the path using `~/.bash_profile`.

The system boot scripts also set PATH. On some Linux boxes, the first command to set the path would seem to be in /etc/rc.d/rc.sysinit, which is one of the shell scripts invoked by the init process (inittab).

Examples:

```
PATH=/bin:/sbin:/usr/bin:/usr/sbin:/home/puffy/bin
```

If this PATH is set and you type the shell command

```
$ uname -r
```

then the shell searches for the "uname" executable program. First it searches in /bin, then /sbin, then /usr/bin. If /bin/uname is an executable (which it should be), then the shell stops searching and runs it. If /home/puffy/bin/uname also is executable, it is not run, because the search never reached that directory.

```
Q: How do you remove, for example, "/usr/bin" from PATH  set as above?
```

PS1

This is a shell variable, not normally exported as an environment variable.

The **bash** and public domain **ksh** shells use this as the prompt string.

Things that can be put in the prompt string include `\h` (hostname), `\u` (username), `\w` (absolute pathname of working directory), `\W` (name of working directory w/o path), `\d` (date), `\t` (time).

On some Red Hat boxes, the primary prompt string is set in the `/etc/bashrc` file. The prompt is also set in `/etc/profile`, but the setting in bashrc seems to take precedence. A `~/.bashrc` file runs /etc/bashrc, which sets the prompt. Because every instance of "bash" runs `~/.bashrc`, the prompt also appears in X sessions started from a display manager such as "xdm".

On some Slackware boxes, the command line prompt is set in /etc/profile. The xterm and rxvt prompts are different. The prompt is not set for X sessions, but it would be if you write a `~/.bashrc` to do that. Prompts are shell variables set from shell initialisation scripts. They are not xterm settings set by X resources such as `/usr/X11R6/lib/X11/app-defaults/XTerm`.

- *A Practical Guide to Linux*, by Mark G. Sobell and published by Addison-Wesley (1998), has more information on prompt strings at page 331.

PS2

This is a shell variable, not normally exported as an environment variable.

The **bash** and public domain **ksh** shells use this as a secondary prompt string.

USER

This variable should have the same setting and purpose as LOGNAME.

VISUAL

This variable is used to specify the "visual" - *screen-oriented* - editor. Usually you'd want to set it to the same value as the EDITOR variable. I imagine that originally EDITOR would've be set to **ed** (a line-based editor) and VISUAL would've been set to **vi** (a screen-based editor), these days though (when people uses screens and not teletype-machines) there is no need to choose different editors for the two.

References

- environ(7) manual page (FreeBSD [1], NetBSD [2], OpenBSD [3])
- Sobell, Mark G. (1998), *A Practical Guide to Linux*. Addison-Wesley
- SSC (2000), "Bash Reference Card", http://www.digilife.be/quickreferences/QRC/Bash%20Quick%20Reference.pdf (updated link)

References

[1] http://www.freebsd.org/cgi/man.cgi?query=environ&sektion=7

[2] http://netbsd.gw.com/cgi-bin/man-cgi?environ+7

[3] http://www.openbsd.org/cgi-bin/man.cgi?query=environ&sektion=7

Guide to Unix/Files

/etc/

/etc/fstab

The **fstab** (for *file systems table*) file is commonly found on Unix and Unix-like systems and is part of the system configuration. The fstab file typically lists all used disks and disk partitions, and indicates how they are to be used or otherwise integrated into the overall system's file system.

Traditionally, the fstab was only read by programs, and not written to. However, more modern system administration tools can automatically build and edit fstab, or act as graphical editors for it. It is the duty of the system administrator to properly create and maintain this file.

The file may have other names on a given Unix variant; for example, it is `/etc/vfstab` on Solaris.

Example

The following is an example of a fstab file on a Red Hat Linux system:

```
# device name            mount point            fs-type      options
            dump-freq pass-num
LABEL=/                  /                      ext3    defaults
        1 1
none                     /dev/pts               devpts  gid=5,mode=620
  0 0
none                     /proc                  proc    defaults
        0 0
none                     /dev/shm               tmpfs   defaults
        0 0

# my removable media
/dev/cdrom               /mnt/cdrom             udf,iso9660
noauto,owner,kudzu,ro 0 0
/dev/fd0                 /mnt/floppy            auto
noauto,owner,kudzu 0 0

# my NTFS Windows XP partition
/dev/hda1                /mnt/WinXP             ntfs    ro,defaults
     0 0

/dev/hda6                swap                   swap    defaults
        0 0

# my files partition shared by windows and linux
/dev/hda7                /mnt/shared            vfat    umask=000
       0 0
```

(kudzu is an option specific to Red Hat and Fedora Core)

The first column indicates the device name or other means of locating the partition or data source. The second column indicates where the data is to be attached to the filesystem. The third column indicates the filesystem type, or

algorithm to use to interpret the filesystem. The fourth column gives options, including if the filesystem should be mounted at boot. The fifth column adjusts the archiving schedule for the partition (used by dump). The sixth column indicates the order in which the fsck utility will scan the partitions for errors when the computer powers on. A value of zero in either of the last 2 columns disables the corresponding feature (http://www.humbug.org.au/talks/fstab/fstab_structure.html).

To get more information about the fstab file you can read the man page about it.

The Kfstab graphical configuration utility is available for KDE for editing fstab.

See also

- mtab

/etc/group

/etc/group stores the definitive list of the users groups and their members.

A typical entry is:

```
root::0:root,alice
```

It has four sections which going from left to right are,

- *(root)* The group name.
- *()* The group password in a hashed form. Normally not. When it is, it allows *any* user knowing the password access to the group, as such it *lessens* security. Group-passwords may be shadowed and stored in a seperate file.
- *(0)* The unique id assigned to the group. Group ids below 10 are reserved for system use. Some unixs such as HP-UX reserve other groups numbers as well.
- *(root,alice)* The list of users who are members of that group.

/etc/passwd

/etc/passwd is the user authentication database, it contains a list of users and their associated internal user id numbers. Historically it also included passwords, however as this file needs to world readable (so all programs can use it to convert between username and user id) it is no longer considered secure to keep passwords in this file.

An entry in this file is of the form:

```
alice:*:134:20:Alice Monkey:/home/alice/:/bin/bash
```

It has seven sections which going from left to right are,

- *(alice)* The username.
- *(*)* The password in a hashed form. In modern systems a star indicates shadowing is in use and hence the password can be found in /etc/shadow/.
- *(134)* The unique id assigned to the user. Some unique ids have special purposes. For example the user id 0 is used for the root user.
- *(20)* The group that the user is assigned to upon login.
- *(Alice Monkey)* The GCOS field, can be used for anything or left blank. Normally used for personal information abou the user such as full name.
- *(/home/alice/)* The home directory of the user.
- *(/bin/bash)* The users default shell.

/etc/profile

/etc/profile contains the system default settings for users who login using the Bourne shell, "/bin/sh". When these users login, the Bourne shell runs the commands in this file before giving the shell prompt to the user. Most of these commands are variable assignments which configure the behavior of the shell.

Some Bourne-compatible shells also use this file, but other shells, such as the C shell, do not.

/etc/shadow

/etc/shadow contains the passwords for users in systems which use shadowing.

```
alice:43SrweDe3F:621:5:30:10:100:900:
```

The sections are:

- *(alice)* The username.
- *(43SrweDe3F)* The password in hashed form.
- *(621)* date of last password change.
- *(5)* the mimimum number of days before the password may be changed.
- *(30)* the maximum number of days before the user is forced to change their password.
- *(10)* the number of days after which a user is advised to change their password.
- *(100)* the maximum number of days an account can be inactive for before it is suspeneded.
- *(900)* the date the account will expire, if left blank the account will remain indefinitely. Most often used for the purpose of temporary accounts.

/etc/sysctl.conf

/etc/sysctl.conf configures the behavior of the running Unix kernel. During system boot, the scripts read this file and use "sysctl" to set the parameters shown in the file. Changing the file has no effect before the next reboot.

Files to be merged in to the list

- /etc/aliases - file containing aliases used by sendmail and other MTAs (mail transport agents). After updating this file, it is necessary to run the newaliases utility for the changes to be passed to sendmail.
- /etc/bashrc - system-wide default functions and aliases for the bash shell
- /etc/conf.modules - aliases and options for configurable modules
- /etc/crontab - shell script to run different commands periodically (hourly, daily, weekly, monthly, etc.)
- /etc/DIR_COLORS - used to store colors for different file types when using ls command. The dircolors command uses this file when there is not a .dir_colors file in the user's home directory. Used in conjunction with the eval command (see below).
- /etc/exports - specifies hosts to which file systems can be exported using NFS. Man exports contains information on how to set up this file for remote users.
- /etc/fstab - contains information on partitions and filesystems used by system to mount different partitions and devices on the directory tree
- /etc/HOSTNAME - stores the name of the host computer
- /etc/hosts - contains a list of host names and absolute IP addresses.
- /etc/hosts.allow - hosts allowed (by the tcpd daemon) to access Internet services
- /etc/hosts.deny - hosts forbidden (by the tcpd daemon) to access Internet services
- /etc/group - similar to /etc/passwd but for groups
- /etc/inetd.conf - configures the inetd daemon to tell it what TCP/IP services to provide (which daemons to load at boot time). A good start to securing a Linux box is to turn off these services unless they are necessary.

- /etc/inittab - runs different programs and processes on startup. This is typically the program which is responsible for, among other things, setting the default runlevel, running the rc.sysinit script contained in /etc/rc.d, setting up virtual login terminals, bringing down the system in an orderly fashion in response to `[Ctrl][Alt][Del]`, running the rc script in /etc/rc.d, and running xdm for a graphical login prompt (only if the default runlevel is set for a graphical login).
- /etc/issue - pre-login message. This is often overwritten by the /etc/rc.d/rc.S script (in Slackware) or by the /etc/rc.d/rc.local script (in Mandrake and Red Hat, and perhaps other rpm-based distributions). The relevant lines should be commented out (or changed) in these scripts if a custom pre-login message is desired.
- /etc/lilo.conf - configuration file for lilo boot loader
- /etc/motd - message of the day file, printed immediately after login. This is often overwritten by /etc/rc.d/rc.S (Slackware) or /etc/rc.d/rc.local (Mandrake/Red Hat) on startup. See the remarks in connection with /etc/issue.
- /etc/mtab - shows currently mounted devices and partitions and their status
- /etc/passwd - contains passwords and other information concerning users who are registered to use the system. For obvious security reasons, this is readable only by root. It can be modified by root directly, but it is preferable to use a configuration utility such as passwd to make the changes. A corrupt /etc/passwd file can easily render a Linux box unusable.
- /etc/printcap - shows the setup of printers
- /etc/profile - sets system-wide defaults for bash shell. It is this file in Slackware that sets up the DIR_COLORS environment variable for the color ls command. Also sets up other system-wide environment variables.
- /etc/resolv.conf - contains a list of domain name servers used by the local machine
- /etc/securetty - contains a list of terminals on which root can login. For security reasons, this should not include dialup terminals.
- /etc/termcap - ASCII database defining the capabilities and characteristics of different consoles, terminals, and printers
- /etc/X11/XF86Config - X configuration file. The location in Slackware is /etc/XF86Config.

/proc/

- /proc/cpuinfo - cpu information
- /proc/filesystems - prints filesystems currently in use
- /proc/interrupts - prints interrupts currently in use
- /proc/ioports - contains a list of the i/o addresses used by various devices connected to the computer
- /proc/kcore - The command ls -l /proc/kcore will give the amount of RAM on the computer. It's also possible to use the free command to get the same information (and more).
- /proc/version - prints Linux version and other info

/var/

- /var/log/messages - used by syslog daemon to store kernel boot-time messages
- /var/log/lastlog - used by system to store information about last boot (can be read with **lastlog**)
- /var/log/wtmp - contains binary data indicating login times and duration for each user on system (can be read with **last**)
- /var/log/btmp - contains binary data indicating *failed* attempts to login (can be read with **lastb**)
- /var/run/utmp - contains binary data indicating *currently logged-in* users (can be read with **who**)

/boot/

- /boot/vmlinuz - the typical location and name of the Linux kernel. In the Slackware distribution, the kernel is located at /vmlinuz.
- /boot/grub/menu.lst - has configuration settings for users of the GRUB bootloader for the available kernels and OS's which can continue the boot process.

/dev/

/dev/cdrom

/dev/cdrom is not an actual device, but on many systems it is a symbolic link to the actual CD device. For example, a Linux system with `/dev/hdb` for its floppy drive is likely to have a link `/dev/cdrom` which redirects to `/dev/hdb`.

/dev/fd*

At Linux, `/dev/fd0` is the first floppy disk drive at the system. Use `/dev/fd0H1440` to operate the first floppy drive in high density mode. Generally, this is invoked when formatting a floppy drive for a particular density. Slackware comes with drivers that allow for formatting a 3.5" diskette with up to 1.7MB of space. Red Hat and Mandrake do not contain these device driver files by default.

Likewise, `/dev/fd1` is the second floppy disk drive.

/dev/hd*

At Linux, `/dev/hda` is the first IDE hard drive. The second drive is either `/dev/hdb` or `/dev/hdc`, depending on the hardware configuration. Some IDE hardware allows up to four drives, including `/dev/hdd`.

Many machines have one hard drive (hda) and one cdrom drive (hdc on many machines, but hdb on some). Often, `/dev/cdrom` is a symbolic link to the cdrom drive.

Partitions are numbered from 1, like `/dev/hda1`, `/dev/hda2`, ...

/dev/null

/dev/null is a do-nothing device to use when one wants to ignore or delete program output. This file is useful when a program expects to save to a file, but you want not to save anything. This file can also be used as input to a program to represent an empty file.

There is no actual hardware associated with the `/dev/null` device.

Examples:

Deleting file called "x" (command **rm x**) sometimes causes an error, for example if the file does not exist:

```
$ rm x
rm: x: No such file or directory
```

One can hide the error by redirecting it to a file. By using `/dev/null` as the file, the error never saves to an actual file.

```
Bourne shell:
$ rm x > /dev/null 2>&1
```

In the Bourne shell, the "2>&1" redirects the standard error of "rm" (where the error appears) to standard output, then the ">" redirects the standard output to `/dev/null`.

One way to make an empty file called "y" is:

```
$ cat /dev/null > y
```

The "cat" command copies the file "/dev/null" to standard output, and the shell operator ">" redirects this output to "y". The "/dev/null" file seems empty when read, so the file "y" appears, but is also empty. (Note that in this case, simply "> y" will do the same thing.)

Dot files

There is some redundancy across these programs. For example, the look and behavior of emacs can be customized by usinng the .emacs file, but also by adding the appropriate modifications to the .Xdefaults file. Default versions of these files are often installed in users' home directories when the software packages that use them are installed. If a program doesn't find its configuration file in the user's home directory, it will often fall back on a sytem-wide default configuration file installed in one of the subdirectories that the package lives in.

- .bash_logout - file executed by bash shell on logout
- .bash_profile - initialization of bash shell run only on login. Bash looks first for a .bash_profile file when started as a login shell or with the -login option. If it does not find .bash_profile, it looks for .bash_login. If it doesn't find that, it looks for .profile. System-wide functions and aliases go in /etc/bashrc and default environment variables go in /etc/profile.
- .bashrc - initialization command run when bash shell starts up as a non-login shell
- .cshrc - initialization commands that are run automatically (like autoexec.bat) when C shell is initiated
- .emacs - configuration file for emacs editor
- .fvwmrc - configuration file for fvwm window manager
- .fvwm2rc - configuration file for fvwm2 window manager
- .jedrc - configuration file for the jed text editor
- .lessrc - typically contains key bindings for cursor movement with the less command
- .login - initialization file when user logs in
- .logout - commands run when user logs out
- .wm_style - gives choice of default window manager if one is not specified in startx
- .Xdefaults - sets up X resources for individual user. The behavior of many different application programs can be changed by modifying this file.
- .xinitrc - initialization file when running startx. Can be used to activate applications, run a given window manager, and modify the appearance of the root window.
- .xsession - configuration file for xdm

Directories

Different distributions have different directory structures, despite attempts at standardization such as the the Linux Filesystem Hierarchy Standard (FHS) organization.

- /bin - essential UNIX commands such as ls, etc. Should contain all binaries needed to boot the system or run it in single-user mode
- /boot - files used during booting and possibly the kernel itself are stored here
- /dev - contains device files for various devices on system
- /etc - files used by subsystems such as networking, NFS, and mail. Includes tables of disks to mount, processes to run on startup, etc.
- /etc/profile.d - contains scripts that are run by /etc/profile upon login.
- /etc/rc.d - contains a number of shell scripts that are run on bootup at different run levels. There is also typically an rc.inet1 script to set up networking (in Slackwar), an rc.modules script to load modular device drivers, and an

rc.local script that can be edited to run commands desired by the administrator, along the lines of autoexec.bat in DOS.

- /etc/rc.d/init.d - contains most of the initialization scripts themselves on an rpm-based system.
- /etc/rc.d/rc*.d - where ``*'' is a number corresponding to the default run level. Contains files for services to be started and stopped at that run level. On rpm-based systems, these files are symbolic links to the initialization scripts themselves, which are in /etc/rc.d/init.d.
- /etc/skel - directory containing several example or skeleton initialization shells. Often contains subdirectories and files used to populate a new user's home directory.
- /etc/X11 - configuration files for the X Window system
- /home - home directories of individual users
- /lib - standard shared library files
- /lib/modules - modular device driver files, most with .o extensions
- /mnt - typical mount point for many user-mountable devices such as floppy drives, cd-rom readers, etc. Each device is mounted on a subdirectory of /mnt.
- /proc - virtual file system that provides a number of system statistics
- /root - home directory for root
- /sbin - location of binaries used for system administration, configuration, and monitoring
- /tmp - directory specifically designed for programs and users to store temporary files.
- /usr - directory containing a number of subdirectory with programs, libraries, documentation, etc.
- /usr/bin - contains most user commands. Should not contain binaries necessary for booting the system, which go in /bin. The /bin directory is generally located on the same disk partition as /, which is mounted in read-only mode during the boot process. Other filesystems are only mounted at a later stage during startup, so putting binaries essential for boot here is not a good idea.
- /usr/bin/X11 - most often a symbolic link to /usr/X11R6/bin, which contains executable binaries related to the X Window system
- /usr/doc - location of miscellaneous documentation, and the main location of program documentation files under Slackware
- /usr/include - standard location of include files used in C programs such as stdio.h
- /usr/info - primary location of the GNU info system files
- /usr/lib - standard library files such as libc.a. Searched by the linker when programs are compiled.
- /usr/lib/X11 - X Window system distribution
- /usr/local/bin - yet another place to look for comon executables
- /usr/man - location of manual page files
- /usr/sbin - other commands used by superuser for system administration
- /usr/share - contains subdirectories where many installed programs have configuration, setup and auxiliary files
- /usr/share/doc - location of program documentation files under Mandrake and Red Hat
- /usr/src - location of source programs used to build system. Source code for programs of all types are often unpacked in this directory.
- /usr/src/linux - often a symbolic link to a subdirectory whose name corresponds to the exact version of the Linux kernel that is running. Contains the kernel sources.
- /var - administrative files such as log files, used by various utilities
- /var/log/packages - contains files, each of which has detailed information on an installed package in Slackware. The same file can also be found at /var/adm/packages, since the adm subdirectory is a symbolic link to log. Each package file contains a short description plus a list of all installed files.
- /var/log/scripts - package installation scripts in Slackware are stored here. You can inspect these scripts to see what special features are included in individual packages.
- /var/spool - temporary storage for files being printed, mail that has not yet been picked up, etc.

External link: Modified Directory Structure [1]

References

[1] http://markhobley.yi.org/mdirs/index.html

Guide to Unix/BSD/Introduction

This page summarizes the main features of the *BSD kernel and system, especially in comparison to other Unix-like systems. Currently, this page covers the free variants of *BSD, especially DragonFly BSD, FreeBSD, NetBSD, and OpenBSD. There are also nonfree variants.

The Wikibook, A Neutral Look at Operating Systems, gives an overview of BSD in its Berkeley Software Distribution chapter.

General

- The *BSD systems are descendants of AT&T Unix (though non-free Unix code was removed) so they feel more like Unix than GNU/Linux. AT&T had cheaply licensed the code to universities including Berkeley where major enhancements such as TCP/IP was then developed. The last BSD release from Berkeley had non-free code removed so more persons could use it. Today, the *BSD projects honor their ancestry putting BSD at the end of their names.
- The four big *BSD systems are **free software** and **open source**. While *BSD does use some copylefted GNU programs, most of *BSD is **not copyleft**, which leaves everyone free to make non-free versions of *BSD and distribute binaries without source code. However, some parts of *BSD have extra terms in the license that annoy some people, and which are not found in the GNU licenses. In particular, there was enough dislike for the "advertising clause" (which requires marks on certain ads) that the Regents of the University of California removed it from their license; some other *BSD copyright holders still use the clause.
- The core system, including kernel and userland, is maintained in one CVS tree. This is similar to OpenSolaris and OpenDarwin, but GNU/Linux programs and kernels are in separate trees, and the development trees of non-free Unix-like systems are normally not public. Compare:
  - DragonFly BSD http://www.dragonflybsd.org/cgi-bin/cvsweb.cgi/
  - FreeBSD http://www.freebsd.org/cgi/cvsweb.cgi/
  - NetBSD http://cvsweb.netbsd.org/bsdweb.cgi/
  - OpenBSD http://www.openbsd.org/cgi-bin/cvsweb/
  - OpenSolaris http://cvs.opensolaris.org/source/
  - OpenDarwin http://cvs.opendarwin.org/index.cgi/
- The *BSD sources are stored in **/usr/src/** and the kernel in **/usr/src/sys/**. The *BSD kernel and system programs are heavily integrated, and so must be upgraded together.

Kernel

- Recall that a *kernel* is the interface between programs and hardware. Unix-like kernels provide device drivers and networking support and allow multiple users and programs to share the system.
- The *BSD kernel (often installed at "/bsd/") is *monolithic*, which means that it is one program in one memory-addressing space. Thus the kernel avoids forming and sending messages between parts of itself.
  - DragonFly BSD is changing some of this.
  - The original AT&T Unix and Linux are also monolithic, while mkLinux and the kernel of Mac OS X are modular.
- The kernel is actually the original AT&T Unix with all of its files replaced. The BSD university project had added or replaced so much stuff (the VAX port, the networking features, the fast file system, ...) that *BSD projects could take the free parts and produce completely free kernels without needing a Unix license. A consequence of this is that the BSD kernel has a similar structure to commercial Unix kernels also descended from AT&T Unix.
- The kernel contains a "securelevel" feature which attempts to permanently restrict what all users (including root, the superuser) can do after a certain point in the boot process.
- The kernel boot messages (also visible with **dmesg**, on all Unix systems) are organized and shows where each device was detected.
  - In contrast, the Linux drivers seem to give any boot messages that they feel like. The mounting and examining of "/proc" and "/sys", or the use of tools like "lspci", is a better strategy on Linux.

Userland

- Recall that *userland* consists of all the software above the kernel. This section describes the userland included with the base system.
- The programs in `/bin/` and `/sbin/` are statically linked.
  - This is because `/usr/` might not be mounted, so the shared library `/usr/lib/libc.so` cannot be used.
  - *Static linking* is when each program is in one file, without the need for other files (shared libraries) containing code shared by programs. The library code is copied into the programs.
- The main text editor in the base system is "vi". This is nvi included with BSD, and ultimately the original vi, and is not some other vi implementation such as vim. Actually, nvi is a clone of the original vi. The clone was necessary to remove some non-free Unix code.
  - OpenBSD also includes "mg", an editor resembling Emacs 17 but without any free but copylefted GNU code. FreeBSD includes "ee", the "easy editor" with some similarities to nano [1] and pico [2] (both of which are in the ports tree, although nano is free software and pico isn't).

Ports

- There is a ports tree or packages tree originally from FreeBSD. This consists of Makefiles that automate the downloading, extracting, patching, and building of software for *BSD. This is the main way of installing stuff that is not part of the base system.
  - On NetBSD this is called **pkgsrc**. A unique feature of the pkgsrc tree is that it also works on other operating systems. DragonFly BSD also uses pkgsrc.
  - Thus installing a program is often two easy steps. First, type a "cd" command to the directory containing the port (for example, `/usr/ports/games/nethack/`). Then, type a "make install" command and wait for everything to finish.
  - Building software takes a long time, so many prefer to use a binary package instead of a "make install".
- The system compiler is the GNU Compiler Collection (GCC) with C, C++, Objective C, and Fortran 77.

- By default, gcc does not look in `/usr/local/` for header files and libraries; it only looks at the base system.
- The base system can rebuild itself with gcc.

References

[1] http://www.nano-editor.org/
[2] http://www.washington.edu/pine/

Guide to Unix/BSD/OpenBSD

This module contains information that is specific to OpenBSD.

Background

OpenBSD calls itself the "multi-platform, ultra-secure operating system". The OpenBSD team believes in strong security and code correctness. The OpenBSD team has a between six and twelve developers working on finding bugs and security holes in the system. OpenBSD also strives to be secure by default, meaning that the user does not have to be a security expert to secure the system. The OpenBSD project normally makes a new version every six months.

OpenBSD is known for having strict rules regarding bugs. Such as an application, not having a manual is considered a bug and therefore the application will not be included in the port (or package) tree. These rules are also included in the operating system. Should an application pose a security risk OpenBSD will kill the process on the spot. If there is a bug, and it is announced on the mailing list, then the OpenBSD team will come out with a patch in a matter of days. OpenBSD has only had two remotly exploitable bugs (both were in ssh and soon there afterwards patched), after a default install (with no additional services turned on) in over 10 years.

Notable Security Features

- strlcpy() and strlcat()
- Memory protection purify
  - W^X
  - .rodata segment
  - Guard pages
  - Randomized malloc()
  - Randomized mmap()
  - atexit() and stdio protection
- Privilege separation
- Privilege revocation
- Chroot jailing
- New uids
- ProPolice

Installation

Main article: Guide to Unix/BSD/OpenBSD/OpenBSD Installation

OpenBSD has a easy although non-graphical installer.

Customization

Main article: Guide to Unix/BSD/OpenBSD/Customize Installation

You can change the default configuration files and install additional packages, by creating your own custom iso file.

As a firewall

Main article: Guide to Unix/BSD/OpenBSD/As a Firewall

OpenBSD uses **pf** ("packet filter") as a firewall. Though the authors originally contributed pf to OpenBSD, because it is free, other operating systems are including pf.

As a desktop

Main article: Guide to Unix/BSD/OpenBSD/As a Desktop

Despite is reputation for being only for servers OpenBSD can also serve as a great workstation/desktop. OpenBSD uses pkg_add as their binary package management system. The pkg_add automatically resolves dependencies. If you get the packages from ftp, pkg_add is able to resolve all of the dependencies for you. Some of the desktop related packages in the binary package system include: KDE (3.5), Xfce, Gnome, fluxbox, blackbox, and e16. Many other packages are available through the port system which also resolves and compiles the dependencies for you.

References

- "OpenBSD Security." 31 Oct. 2008. OpenBSD. 31 Oct. 2008 <http://www.openbsd.org/security.html>.

Guide to Unix/Explanations/Shell Prompt

The *shell prompt* (or *command line*) is where one types commands. When accessing the system through a text-based terminal, the shell is the main way of accessing programs and doing work on the system. In effect, it is a shell surrounding all other programs being run. When accessing the system through a graphical environment such as X11, it remains possible to open a terminal emulator and do useful work with the shell.

This chapter describes how to find a shell prompt and start using it.

Finding a shell prompt

People get shell prompts in different ways, such as:

- They use a graphical environment (such as Aqua, GNOME, or KDE) and a terminal emulator.
- They do not use GUI, but simply use TTY device; sometimes also use GUI and get to a TTY device with Ctrl+Alt+F[number] (most GNU/Linux systems allow 1-6 for [NUMBER]). To get back to The X Window System, use Ctrl+Alt+F[number 1 higher than the number of TTY devices].

Using the TTY device

Unix systems can use TTY devices for a shell. The first line looks like

```
login:
```

Type your username at this prompt, then type your password. This gives a shell.

Opening a terminal emulator

A **terminal emulator** (or console emulator) is a program that emulates the terminal hardware that early users traditionally used to login to Unix. It appears as a window in the graphical environment and allows access to the shell prompt.

There are several ways to open a terminal emulator:

- With The X Window System, try **xterm**. Most setups either start you with an xterm or two when you login, or provide a menu from which you can start an xterm.
  - KDE provides a **Terminal Program (Konsole)**. You can find it in the K Menu > System menu. If you right click on an empty part of your icon panel, you can Add an Application Button for the terminal.
  - GNOME provides a terminal emulator somewhere in the **Programs** menu.
  - Using Enlightenment 17, right click on the desktop, then go to Enlightenment>Eterm.
- On Mac OS X, use `/Applications/Utilities/Terminal`. That is, go to the Applications folder on your hard disk, and to the Utilities folder inside it, and double-click the icon for the Terminal program. You may want to keep the Terminal icon on your Dock.

Most terminal emulators, like other graphical programs, provide a menu bar to configure the terminal. For example, they allow you to change the font and colors; some people prefer white text on a black background. xterm is more difficult to configure; its menus can be found by holding the Control key and clicking with each of the three mouse buttons.

Using SSH to access a remote shell

The **ssh** program is a secure way to connect to a shell account on a remote server. The server must be running the **sshd** servers software to accept the connection. See the chapter on ../Connecting to Remote Unix/.

Appearance of the prompt

The shell prompt normally ends in a $ sign. For simplicity, the examples in this book use a shell prompt like this:

```
$
```

Some older shell prompts end in % instead:

```
%
```

The C shell sometimes uses > instead:

```
>
```

Several shells have prompts that give more information, such as:

```
localhost:puffy {1}
```

You can also customize your shell prompt. For bash, use these [1] special characters in the variables $PS[1-4]. $PS1 is what you usually see and $PS2 is what you see when you are doing a multi-line command with a backslash ('\'). For more see the manual [2].

Never copy/type the shell prompt used in this book. The shell will always give you a prompt if it is ready to accept commands.

Root shell prompt

If you become root or login as root, most systems change the shell prompt to end with #. The root account is allowed to do *anything* (delete or change any file) so the # is a reminder of the power of the prompt. Avoid using the # prompt when necessary; see the chapter on ../Becoming Root/. In this book, root shell prompts look like this:

```
#
```

The basics of using the shell

Arguments and Options

When you enter a command, the shell does a few things in this order (if it succeeds, it executes the found command):

1. The shell checks if the command is an absolute path (such as **/bin/ls**) and if that path is an executable file.
2. If the command is not an absolute path, the shell:
    1. searches through its builtin commands for the entered command.
    2. Looks in the directories in enviornment variable PATH for the entered command. It starts its search with the first directory listed in PATH, then the second and so on.

For examples, we'll be using the command **ls**, which lists files and directories. This command lists the contents of `/var` (which may differ on your computer):

```
$ ls /var
account backups db      lib     msgs    run     tmp
audit   crash   empty   log     named   rwho    www
authpf  cron    games   mail    quotas  spool   yp
```

The first word, "ls", is the name of the program or command built into the shell to run. In this case, the program /bin/ls is run. The "/var" in this case is an *argument*; it tells ls what to list. Arguments are separated by whitespace, usually one space.

There are some special arguments called options. Each command decides what is an option, but for many commands, options with only one hyphen to start usually are short for a more descriptive option that starts with two hyphens and some that start with two hyphens don't have shorthands. Here is an example (read it as "el es dash el slash var"):

```
$ ls -l /var
drwxr-xr-x   2 root    wheel      512 Mar 20  2005 account
drwxrws---   2 root    wheel      512 Mar 20  2005 audit
drwxrwx---   2 root    authpf     512 Mar 20  2005 authpf
...
drwxr-xr-x   2 root    wheel      512 Jun 11 02:09 yp
```

Note that these commands require that options come before other arguments. For example, the following does not work (unless you have a file or directory called `-l`):

```
$ ls /var -l
ls: -l: No such file or directory
/var:
account backups db      lib     msgs    run     tmp
audit   crash   empty   log     named   rwho    www
authpf  cron    games   mail    quotas  spool   yp
```

Exception: on systems with the GNU C library (GNU and GNU/Linux), many programs (not all of them) will automatically treat the options as if they were in the front, so **`ls /var -l`** is the same as **`ls -l /var`**. This is nice for users who forget to type some option.

What if there really is a file called "-l"? Then it must be specified that "-l" is not an option. One does this using the "--" argument, which means "end of options". This is why it is inconvenient to have filenames start with hyphens.

```
$ ls -- -l
-l
```

Editing Commands

If the shell has command-line editing, then the arrow, End, and Home keys are useful. The left and right arrows allow the user to move the text cursor to edit the command and the Home and End keys allow the user to move the to the beginning or end of the line. For example, we want to list some specific files, like this:

```
$ ls -l /etc/passwd /etc/profile
```

But we typed:

```
$ ls l- /etc/passwrd /etc/profile
```

We can press the Home key to move the cursor to the beginning of the line, then use the right arrow to move the cursor to the right to delete the "-l". After this, we press Return to run the command as normal. (This will not work in many non-shell programs that lack command line editing!)

Shells with history features allow using the up arrow to recall previous commands to the shell prompt. These previous lines are not run again unless the user presses Return. The down arrow returns down the list.

If the arrow keys are broken, or you actually find some keyboard without arrows, then the Control Emacs navigation keys (Ctrl+B, Ctrl+F, Ctrl+P, Ctrl+N, Ctrl+E and Ctrl+A) also work in most shells.

The current/working directory

Use **cd** to change the directory you are "in." The syntax is cd followed by the pathname. cd bin would take you to the bin directory (located in the directory you are currently in), cd .. would take you up one level. cd ~ would take you to your home directory, and cd ~ followed by a username would take you to that user's home.

Although many shell prompts have the current directory's name or the end of the current directory's name right in the prompt (like '[user@localhost ~]$'), you can use the command **pwd** to Print the Working Directory.

```
$ cd /etc
$ pwd
/etc
$ ls passwd profile
passwd profile
$ cd ~john
$ pwd
/home/john
```

Finding help for commands

The first word of the command is the name of the command. For example, in the following command, "ls" is the command name, and "-l", "/etc/passwd", and "/etc/profile" are the arguments.

```
$ ls -l /etc/passwd /etc/profile
```

But how do we know what the "ls" command does? Most Unix-like systems provide *online manual pages* for each command. For example,

```
$ man ls
```

This opens the manual page in a program called the *pager*. The most common pagers are **less** and **more**. These let the user type space bar to scroll down and 'q' to quit the pager.

However, the manual pages are often not useful for persons who know almost nothing about the commands. The chapter on Commands will help. The section Guide to UNIX/Commands/Getting Help contains strategies for how to use man and the other help tools effectively.

Continue

1. ../Shell Prompt/
2. ../Quoting and Filename Expansion/
3. ../Pipes and Job Control/

References

[1] http://www.gnu.org/software/bash/manual/bashref.html#SEC83
[2] http://www.gnu.org/software/bash/manual/bashref.html

Guide to Unix/Explanations/Quoting and Filename Expansion

Special characters and substitution

The shell recognizes several special characters. The following are the special characters:

- \ (backslash)
- " (double quote)
- ' (single quote)
- # {number sign, pound sign, or hash)
- $ (dollar sign)
- ` (tick)
- ~ (tilde)
- { and } (braces or curly brackets)
- (and) (parentheses)
- * (asterisk, star or splat)
- ? (question mark)
- < (less than sign)
- > (greater than sign)
- & (ampersand)
- | (pipe)
- ; (semicolon)
- ! (exclamation point or bang)

They are all special characters in the Bourne shell, except for the exclamation point. However, many Bourne and non-Bourne shells also make special the exclamation point. When the shell sees special characters, it does something more complex than simply running the command that you typed.

Some special characters trigger *substitution*, when parts of the command are replaced with other text. One common form of substitution is *filename expansion*, which saves you work when typing longer filenames and lists of filenames.

To test substitution, we need the **echo** command. This command simply echoes its arguments (including any changes made to the arguments by the shell). All options except "-n" are ignored.

```
$ echo -lnQ arg1 arg2 arg3
-lnQ arg1 arg2 arg3
```

Quoting

Quoting is used to preserve the literal meaning of special characters.

Here is an example of several types of quoting, which will be refered to in the rest of this section (there are many unnecessary things in this which you should find after reading this section, but are used for the sake of example):

```
$ echo There is \"a small possibility\" that '/etc/*tab' are " not "
text and a backslash will not be printed after this. \\
There is "a small possibility" that /etc/*tab are  not  text and a
backslash will not be printed after this. \
```

Backslash

A backslash (\) simply stops the shell from thinking that certain characters are special. In the example above, it was used to print literally the doublequote character.

To print a backslash, use two backslashes in a row.

Now commands that want special shell characters for input will work.

Paired quoting characters

Paired characters affect whatever is between them.

Single quotes

Single quotes perserve the literal meaning of everything except single quotes.

In the example, single quotes were used to prevent the * from expanding.

Note single quotes can't be within single quotes, even with a preceding backslash.

Double quotes

Double quotes are like single quotes, but don't preserve the literal meaning of $, \ when followed by a dollar sign, tick, double quote, or backslash, and `.

In the example, double quotes were used to preserve the literal meaning of spaces.

Filename expansion

The *filename expansion* is used in avoiding the typing of long lists of files. These characters are used for filename expansion:

```
* ? { }
```

* and ?

The *shell globbing characters* of * and ? are used to form *patterns*. The shell searches for existing files that match the patterns and does a subsitution.

- `*` matches 0 or more characters, except /
- `?` matches any 1 character except /

When more than one file is matched, the files are separated by spaces, as separate arguments.

For example, `/etc/ss*` means all of the files inside the `/etc` directory which begin with the letters `ss`. The echo command demonstrates the substitution. The results may differ on your system, depending on what files you have.

```
$ echo /etc/ss*
/etc/ssh /etc/ssl
```

The echo command did this because it thinks that you ran **`echo /etc/ssh /etc/ssl`**. The shell *substituted* those names for `/etc/ss*`. Though `/etc/ssh/ssh_config` exists on many systems, it appeared not above, because `*` never matches `/`.

Now here is something more useful. We will use substitution with a command other than "echo". We will combine filename expansion with the ls command.

```
$ ls /etc/ss*
/etc/ssh:
```

```
ssh_config                ssh_host_key
ssh_host_rsa_key.pub
ssh_host_dsa_key          ssh_host_key.pub          sshd_config
ssh_host_dsa_key.pub      ssh_host_rsa_key

/etc/ssl:
lib          openssl.cnf  private      x509v3.cnf
```

Notice how the * saved typing. If we did **`ls /etc/s*`**, we would probably save much typing.

Here is an example of the ? character at work:

```
$ ls /etc/ssh/ssh_host_???_key
/etc/ssh/ssh_host_dsa_key  /etc/ssh/ssh_host_rsa_key
```

When expansion fails

Sometimes, when using * or ?, no files are found. In this case, the shell does not give an error; it simply does no substitution. Suppose that in the machine on the example above, we make a mistake and typed `/etc/sss*` which matches nothing. Then the shell would do no substitution:

```
$ echo /etc/sss*
/etc/sss*
```

Curly brackets expansion

The curly brackets help when typing several similar arguments, especially filenames. The filenames do not need to actually exist. Example:

```
$ echo /etc/ss{h,l,ssnakes}
/etc/ssh /etc/ssl /etc/ssssnakes
```

They are very useful for long lists of files in the same directories:

```
$ echo /bin/{ls,mv,cp} /sbin/{halt,reboot}
/bin/ls /bin/mv /bin/cp /sbin/halt /sbin/reboot
```

You can also use them to print the alphabet (or numbers):

```
$ echo {z..a}
z y x w v u t s r q p o n m l k j i h g f e d c b a
```

Guide to Unix/Explanations/Pipes and Job Control

Pipes and Redirects

We can connect standard input, output, and error of commands to other commands or to files.

Job Control

You can run multiple commands at once, but only one can be in the foreground. The other commands can be in running at the background or suspended.

Create the shell script "date_loop.sh"

```
#!/bin/bash
while :
do
    date > datefile
    sleep 10
done
```

Make **date_loop.sh** executable:

```
$ chmod +x date_loop.sh
```

Run the command in foreground:

```
$ ./date_loop.sh
```

Wait a while (about 30 seconds) then use CONTROL-Z to send **SIGTSTP** signal to the process:

```
^Z[2] + Stopped (SIGTSTP)        date_loop
```

Look at the last line of **datefile**:

```
$ tail -1 datefile; sleep 10; tail -1 datefile
Thu Apr 27 10:46:06 BST 2006
Thu Apr 27 10:46:06 BST 2006
```

Note, that while the process was *runnable*, it was suspended, thus new "times" were not appended to the file in the 10 second interval between respective **tail** commands.

Put the process into background:

```
$ bg
[2]     date_loop.sh&
```

The process is resumed and **datefile** is being updated again:

```
$ tail -1 datefile; sleep 10 ; tail -1 datefile
Thu Apr 27 10:48:34 BST 2006
Thu Apr 27 10:58:54 BST 2006
```

Note that you can type commands at the command prompt.

Put the process back into foreground:

```
$ fg
date_loop.sh
```

Send a *'SIGINT* to the process by typing "control-C". This terminates the process. We confirm that **datefile** is not being updated anymore:

```
^C
$ tail -1 datefile; sleep 20; tail -1 datefile
Thu Apr 27 10:59:55 BST 2006
Thu Apr 27 10:59:55 BST 2006
```

Guide to Unix/Explanations/Signals

Signals

Signals are software interrupts that notify processes that some condition or event has occurred. Examples of events that generate signals are:

- a process attempts a divide by zero
- a user presses the INTERRUPT key at the terminal
- a process sends another process a signal with the kill command of the kill function

A list of all the different signals can be displayed using

```
$ kill -l
```

The **kill** command can also be used to send signals to other processes. Start a background process that sleeps for 1000 seconds:

```
$ sleep 1000 &
[1]     14936
$ ps
   PID TTY      TIME CMD
 14936 pts/7    0:00 sleep
 12203 pts/7    0:01 bash
```

Note that the process id will (most likely) be different when you run it. The command-line below sends the **SIGKILL** signal to process 14936 (in this case):

```
$ kill -KILL 14936
$ ps
   PID TTY      TIME CMD
 12203 pts/7    0:01 ksh
[1] + Killed                  sleep 1000 &
```

Upon receiving the **SIGKILL** signal the **sleep** process terminates.

Traps

Processes can respond to signals in three different ways:

- perform the default action
- ignore the signal.
- execute a specified function (called a signal handler)

The default action performed by a process when it receives a signal depends on the signal type. Most (like **SIGKILL**) cause the process to terminate. Others like **SIGQUIT** cause the process to terminate and generate a core dump:

```
$ sleep 1000 &
[1]     14950
$ kill -QUIT 14950
$ ps
   PID TTY      TIME CMD
 12203 pts/7    0:01 bash
[1] + Quit(coredump)           sleep 1000 &
$ ls core
core
```

Core files can be large so delete it:

```
$ rm core
```

Signals can be caught using the trap command. Create the following shell script in the file **catch_signal.sh**:

```
#!/bin/bash
trap 'echo "GOTCHA!"' INT KILL
while :
do sleep 10
done
```

Run **catch_signal.sh** in background:

```
$ catch_signal.sh &
[1]     15053
$ kill -INT 15053
$ GOTCHA!
```

The signal was caught and instead of terminating (which is the default action for **SIGINT**) the message "GOTCHA!" was echoed to standard output.

Some signals however *can't* be caught, **SIGKILL** for example:

```
$ kill -KILL 15053
[1] + Killed                   catch_signal &
```

Run **catch_signal.sh**; in foreground:

```
$ catch_signal
```

Hit the interrupt key (control-C usually):

```
^CGOTCHA!
```

Now hit the quit key:

```
^\catch_signal[3]: 15102 Quit(coredump)
$ ls core # will have generated a core file
core
```

Nohup

Commands run with **nohup** do not terminate when the user logs off.

Create the script in the file **dont_hangup.sh** and make it executable:

```
#!/bin/bash

while :
do
    date
    sleep 100
done
```

Run it in background preceded by the command **nohup**:

```
$ nohup dont_hangup.sh &
[1]     15224
Sending output to nohup.out
```

Check its running: id (12203 in this case):

```
$ ps -fu aholt
aholt 15224 12203 pts/7  sh ./dont_hangup.sh
aholt 15232 15224 pts/7  sleep 100
```

Make a note of the parent process id (12203 in this case) of "dont_hangup" and the terminal device and log out of your session.

Log back in again. See that **dont_hangup.sh** is still running:

```
$ ps -fu aholt
aholt 15224      1 ? 0:00 sh ./dont_hangup.sh
aholt 15237 15224  ? 0:00 sleep 100
```

While "dont_hangup.sh" is still running, its parent process has changed. The process id of the parent is 1 (not 12203 as before) and it is no longer associated with a terminal device. Using **nohup** is equivalent to *ignoring* the SIGHUP, that is:

```
trap '' HUP
```

Bourne Shell Scripting/Quick Reference

This final section provides a fast lookup reference for the materials in this document. It is a collection of thumbnail examples and rules that will be cryptic if you haven't read through the text.

Useful commands

Command	Effect
cat	Lists a file or files sequentially.
cd	Change directories.
chmod ugo+rwx	Set read, write and execute permissions for user, group and others.
chmod a-rwx	Remove read, write and execute permissions from all.
chmod 755	Set user write and universal read-execute permissions
chmod 644	set user write and universal read permissions.
cp	Copy files.
expr 2 + 2	Add 2 + 2.
fgrep	Search for string match.
grep	Search for string pattern matches.
grep -v	Search for no match.
grep -n	List line numbers of matches.
grep -i	Ignore case.
grep -l	Only list file names for a match.
head -n5 source.txt	List first 5 lines.
less	View a text file one screen at a time; can scroll both ways.
ll	Give a listing of files with file details.
ls	Give a simple listing of files.
mkdir	Make a directory.
more	Displays a file a screenfull at a time.
mv	Move or rename files.
paste f1 f2	Paste files by columns.
pg	Variant on "more".
pwd	Print working directory.
rm	Remove files.
rm -r	Remove entire directory subtree.
rmdir	Remove an empty directory.
sed 's/txt/TXT/g'	Scan and replace text.
sed 's/txt/d'	Scan and delete text.
sed '/txt/q'	Scan and then quit.
sort	Sort input.
sort +1	Skip first field in sorting.

sort -n	Sort numbers.
sort -r	Sort in reverse order.
sort -u	Eliminate redundant lines in output.
tail -5 source.txt	List last 5 lines.
tail +5 source.txt	List all lines after line 5.
tr '[A-Z]' '[a-z]'	Translate to lowercase.
tr '[a-z]' '[A-Z]'	Translate to uppercase.
tr -d '_'	Delete underscores.
uniq	Find unique lines.
wc	Word count (characters, words, lines).
wc -w	Word count only.
wc -l	Line count.

Elementary shell capabilities

Command	Effect
shvar="Test 1"	Initialize a shell variable.
echo $shvar	Display a shell variable.
export shvar	Allow subshells to use shell variable.
mv $f ${f}2	Append "2" to file name in shell variable.
$1, $2, $3, ...	Command-line arguments.
$0	Shell-program name.
$#	Number of arguments.
$*	Complete argument list (all in one string).
$@	Complete argument list (string for every argument).
$?	Exit status of the last command executed.
shift 2	Shift argument variables by 2.
read v	Read input into variable "v".
. mycmds	Execute commands in file.

IF statement

The if statement executes the command between `if` and `then`. If the command returns not 0 then the commands between `then` and `else` are executed - otherwise the command between `else` and `fi`.

```
if test "${1}" = "red" ; then
  echo "Illegal code."
elif test "${1}" = "blue" ; then
  echo "Illegal code."
else
  echo "Access granted."
fi
```

```
if [ "$1" = "red" ]
then
  echo "Illegal code."
elif [ "$1" = "blue" ]
then
  echo "Illegal code."
else
  echo "Access granted."
fi
```

Test Syntax Variations

Most test commands can be written using more then one syntax. Mastering and consistently using one form may be a programming best-practice, and may be a more efficient use of overall time.

String Tests

String Tests are performed by the `test` command. See `help test` for more details. To make scripts look more like other programming languages the synonym `[ ... ]` was defined which does exactly the same as `test`.

Command	Effect
test "$shvar" = "red" ["$shvar" = "red"]	String comparison, true if match.
test "$shvar" != "red" ["$shvar" != "red"]	String comparison, true if no match.
test -z "${shvar}" test "$shvar" = "" ["$shvar" = ""]	True if null variable.
test -n "${shvar}" test "$shvar" != "" [-n "$shvar"] ["$shvar" != ""]	True if not null variable.

Arithmetic tests

simple arithmetics can be performed with the `test` for more complex arithmetics the `let` command exists. See `help let` for more details. Note that for `let` command variables don't need to be prefixed with '$' and the statement need to be one argument, use `'...'` when there are spaces inside the argument. Like with `test` a synonym - `(( ... ))` - was defined to make shell scripts look more like ordinary programs.

Command	Effect
test "$nval" -eq 0 let 'nval == 0' ["$nval" -eq 0] ((nval == 0))	Integer test; true if equal to 0.
test "$nval" -ge 0 let 'nval >= 0' ["$nval" -ge 0] ((nval >= 0))	Integer test; true if greater than or equal to 0.
test "$nval" -gt 0 let 'nval > 0' ["$nval" -gt 0] ((nval > 0))	Integer test; true if greater than 0.
test "$nval" -le 0 let 'nval <= 0' ["$nval" -le 0] ((nval <= 0))	Integer test; true if less than or equal to 0.
test "$nval" -lt 0 let 'nval < 0' ["$nval" -lt 0] ((nval < 0))	Integer test; true if less than to 0.
test "$nval" -ne 0 let 'nval != 0' ["$nval" -ne 0] ((nval != 0))	Integer test; true if not equal to 0.
let 'y + y > 100' ((y + y >= 100))	Integer test; true when $x + y \geq 0$

File tests

Command	Effect
test -d tmp [-d tmp]	True if "tmp" is a directory.
test -f tmp [-f tmp]	True if "tmp" is an ordinary file.
test -r tmp [-r tmp]	True if "tmp" can be read.
test -s tmp [-s tmp]	True if "tmp" is nonzero length.
test -w tmp [-w tmp]	True if "tmp" can be written.
test -x tmp [-x tmp]	True if "tmp" is executable.

Boolean tests

Boolean arithmetic is performed by a set of operators. It is important to note then the operators execute programs and compare the result codes. Because boolean operators are often combined with `test` command a unifications was created in the form of `[[ ... ]]`.

Command	Effect
test -d /tmp && test -r /tmp [[-d /tmp && -r /tmp]]	True if "/tmp" is a directory and can be read.
test -r /tmp || test -w /tmp [[-r /tmp || -w /tmp]]	True if "tmp" can be be read or written.
test ! -x /tmp [[! -x /tmp]]	True if the file is not executable

CASE statement

```
case "$1"
in
  "red")      echo "Illegal code."
              exit;;
  "blue")     echo "Illegal code."
              exit;;
  "x"|"y")    echo "Illegal code."
              exit;;
  *)          echo "Access granted.";;
esac
```

Loop statements

```
for nvar in 1 2 3 4 5
do
  echo $nvar
done
```

```
for file            # Cycle through command-line arguments.
do
  echo $file
done
```

```
while [ "$n" != "Joe" ]     # Or:   until [ "$n" = "Joe" ]
do
  echo "What's your name?"
  read n
  echo $n
done
```

There are "break" and "continue" commands that allow you to exit or skip to the end of loops as the need arises.

Instead of [] we can use test. [] requires space after and before the brackets and there should be spaces between arguments.

Credit

This content was originally from http://www.vectorsite.net/tsshell.html and was originally in the public domain.

Guide to Unix/Explanations/Introduction to Editors

The **Introduction to Editors** briefly introduces the reader to the common Unix text editors and provides links to more information.

Many readers will be familiar with text editors that have graphical user interfaces similar to Notepad from Windows, TextEdit (in unstyled text mode) from Mac OS X, GEdit from GNOME, or KEdit or KWrite from KDE. Other readers will only know about word processors, which are like text editors, but have additional features for applying style and layout to the text. Text editors only deal with sequences of text characters, all in the same font.

The approach of this chapter is to introduce the earliest Unix text editors and progress to the Notepad-style editors. The early editors lack many common features of editors.

The need for a text editor

One who uses the command line, but knows not how to use a text editor, can still create text files using the **cat** tool and the shell redirection feature.

```
$ cat > newfile
We the people, promoting the common keyboard
and preserving a more perfect docmument for all,
do ordain and establish three lines of text.
```

Suppose there is a want to change this file. In this example, "docmument" has an incorrect spelling; it should be document. Other wants could be revisions to this sentence and the addition of more sentences.

One can append to the file with **`cat >> newfile`**, but that leaves the first lines unmodified. One can replace it with **`cat > newfile`** again, but that requires retyping the entire file. (Users of graphical user interfaces such as X11 have an advantage: they can use the mouse to copy and paste the text before "docmument", type the correction, then copy and paste the remainder of the text.)

An interesting possibility is if there is a way to delete the extra "m" from "docmument", and to handle more complex tasks like inserting words and rearranging text.

ed

One of the first editors that did this was **ed**, short for "edit". This has many features of Windows Notepad, but also lacks many. Observe what happens when one starts "ed". Here, "newfile" is the name of the text file to edit.

```
$ ed newfile
139
```

A number (here "139") appears. Then the program seems to stop, but no shell prompt appears. This is actually "ed" waiting for commands. On many computers, "ed" actually lacks a prompt.

The "139" indicates that "ed" read in 139 characters that are now ready for editing. We say that the file is *open*. To be more correct from the perspective of a C programmer, the file was opened and copied into a buffer. The buffer, not the file is open. This means that the disk or storage device will not be bothered until we *save* the file. Other text editors still follow this behavior, opening files, copying them into buffers, and requiring the user to *save* to write the

file back to disk.

Notice that unlike many editors, "ed" has not yet shown the text of the file. We type a command `,p` to do this.

```
$ ed newfile
139
,p
We the people, promoting the common keyboard
and preserving a more perfect docmument for all,
do ordain and establish three lines of text.
```

The command, roughly translated, is that for every line in the file (","), "p"rint that line to standard output.

Now to do some actual editing, we use a command **`2s/docmument/document`**.

```
$ ed newfile
139
,p
We the people, promoting the common keyboard
and preserving a more perfect docmument for all,
do ordain and establish three lines of text.
2s/docmument/document
and preserving a more perfect document for all,
```

For line "2", "s"ubstitute the first instance of "docmument" with "document". Here "ed" prints the changed line from the buffer.

We run two more commands, "w" to write the buffer back into `newfile` on disk, and "q" to quit "ed". If we forget "w", then our edit is lost.

```
$ ed newfile
139
,p
We the people, promoting the common keyboard
and preserving a more perfect docmument for all,
do ordain and establish three lines of text.
2s/docmument/document
and preserving a more perfect document for all,
w
138
q
```

We made only one minor change, but this already required four commands. Even worse, "ed" error messages are not useful when we mistype commands. Many Unix users never bother to learn "ed". For those readers with interest, this book has a chapter ../ed and sed/ (when someone writes that chapter).

vi

At some point, Unix systems introduced video screens that allow Unix to draw anywhere on the screen. Someone decided that it would be good to create a "visual editor" that allows the user to move the cursor through the file (as it appears with "cat" or the ed "p" command) and make changes. The name of this program is **vi**, which is short for "VIsual editor". Thus, the namers of this program intended one to call it "vee eye", not "vee" nor "six".

Observe what happens when one starts "vi". First you type the command:

```
$ vi newfile
```

Then the screen clears and becomes like this:

```
We the people, promoting the common keyboard
and preserving a more perfect document for all,
do ordain and establish three lines of text.
~
~
~
~
newfile: unmodified: line 1
```

The screen might look different if you have a different version of "vi". Your screen also probably has more than eight lines. However, all versions of "vi" have these two features:

- a "~" shows a nonexistant line (though you could type a line with only "~" to be confusing)
- there is a status line at the bottom, here it reads "newfile: unmodified: line 1"

On many computers, you can use the arrow keys to move the cursor. If that does not work, you can use the standard "vi" keys:

- **[h]** moves left, **[j]** moves down, **[k]** moves right, **[l]** moves down

For example, one can press **[l]** for fifteen times to move the cursor from the "W" in "We" to the "p" in "promoting". (In fact, as a shortcut, vi lets one press **[1] [5] [l]** (one, five, ell). This gives the number "15" to the **[l]** command, which in this case means to repeat the command fifteen times.) So it is possible to move the cursor through the file, which was not possible with ed.

But what if we want to type the letter "l" instead of moving the cursor? Press **[i]**, which is a command to switch the vi editor from *command mode* to *insert mode*. Then type something. Here, the user, with the cursor at "p" in "promoting", typed **[i]** and then "nominally " (including one space):

```
We the people, nominally promoting the common keyboard
and preserving a more perfect document for all,
do ordain and establish three lines of text.
~
~
~
~
```

Note that some copies of "vi", such as this one, by default never show at the screen whether the editor is in command or insert mode.

To exit insert mode, press **[ESC]** escape. To save and quit the editor, type **[:] [w] [q] [RETURN]** which inputs "wq" to the colon prompt.

To learn more about vi, read the Wikibook, Learning the vi editor.

vim

In new generation of Linux and Unix operating systems, the more improved version of VI editor was released called vim (version 7 latest). VIM incorporates almost all the features of VI and more, including color coding screen, highlights, and spell check within the document.

Learning the vi editor/vi Reference

The following conventions are used in this reference.

<c>

A single character, such as 'a' or '1'.

<ESC>, <Ctrl-[>

Indicates that the Escape (Esc) key on your keyboard should be pressed, which is identical to Control and '['.

<CR>

Indicates that the Return (Enter) key should be pressed.

<TAB>

Indicates that the Tabulator key should be pressed

<Ctrl-x>, <C-x>

Indicates that the Control key and the 'x' key should be pressed simultaneously. 'x' can be almost any other key on your keyboard.

<Shift-x>, <S-x>, <X>

Indicates that the Shift key and the 'x' key should be pressed simultaneously

<Meta-x>, <M-x>

Indicates that the Meta or Alt key and the 'x' key should be pressed simultaneously.

quit, **:q**: An Ex command. started with <:>, followed by the command and ends with <CR>. For many Ex commands there is a long form (**:quit**) and a short form (**:q**).

set nocompatible : represents a setting.

strlen ()

represents a function.

/pattern/*, *?pattern?

A Search pattern. Search pattern in vi are regular expressions.

*range***s**/*search*/*replace*/*options*, **:global** /*pattern*/ **delete**: A Search pattern combined with an Ex command.

All commands in *vi* are case sensitive.

c	A single character, such as 'a' or '1'.
m	A single lowercase letter, used to mark text.
string	Several characters, such as 'abc bed'.
pattern	A string used in searching, which may contain regular expressions. For example 'abc' or '^ab[123]'.
myfile	The name of a file to be edited.

Invocation

vi myfile	Open the file *myfile* for editing. If it does not exist, a new file is created. Multiple files can be opened at the same time.
vi +line myfile	Open the file *myfile* with the cursor positioned at the given line. • **vi +5 myfile** opens *myfile* at line 5. • **vi + myfile** opens *myfile* at the last line.
vi +/string/ myfile	Open the file *myfile* with the cursor positioned at the first line containing the string. If the string has spaces it should be enclosed in quotes. • **vi +/"search string"/ myfile** opens *myfile* at the first line containing *search string*.
vi -r	Lists recovery copies of files. A recovery copy is taken if a **vi** session is killed or the system crashes.
vi -r myfile	Opens a recovery copy of the file *myfile*.
view myfile	**view** is a read only version of **vi**. All **vi** commands, include those to change the file are allowed and act as in **vi**. The difference is that normal attempts to save, **ZZ** or **:wq** do not work. Instead **:x!** or **:w** need to be used.

vi Commands

Movement

vi can be set up on most systems to use the keyboard movement buttons, such as *cursor left*, *page up*, *home*, *delete*, etc.

<G>	Move to the last line of the file. Can be preceded by a number indicating the line to move to, <1><G> moves to the first line of the file.
<h>	Move left one character, or cursor left. Can be preceded by a number, <5><h> moves left 5 places.
<j>	Move one line down, or cursor down. Can be preceded by a number, <5><j> moves down 5 lines.
<k>	Move one line up, or cursor up. Can be preceded by a number, **5k** moves up 5 lines.
<l>	Move forward one character, or cursor right. Can be preceded by a number, **5l** moves right 5 places.
<H>	Moves to the line at the top of the screen.
<M>	Moves to the line in the middle of the screen.
<L>	Moves to the line at the bottom of the screen.
-	Moves to the first non-whitespace character of the line above. Can be preceded by a number. • **10-** moves up 10 lines.
<+>	Moves to the first non-whitespace character of the line below. Can be preceded by a number. • **10+** moves down 10 lines.
<CR>	Same as <+>.
<>	**moves to column 10.**
<w>	Moves to the start of the next word, which may be on the next line.

<W> As **w** but takes into account punctuation.

<e> Moves to the end of the current word or to the next word if between words or at the end of a word.

<E> As **e** but takes into account punctuation.

<b> Moves backwards to the start of the current word or to the previous word if between words or at the start of a word.

<B> As **b** but takes into account punctuation.

<f>*c* Find first occurrence of character *c* on the same line. This command may be repeated using <;> or <,> (reverse direction).

- <3><f><x> moves forward on the third occurrence of x (if present).

  Same as <f><x><;><;>

<F>*c* Same as **f** but backward.

<t>*c* Find the character before the first occurrence of character *c* on the same line.

<T>*c* Same as **t** but backward, placing the cursor after character *c*.

<0> Moves to the start of the current line.

<^> Moves to the first non-whitespace character on the current line.

<$> Moves to the end of the current line.

<Ctrl-F> Move forwards one page.

- **5<Ctrl-F>** moves forwards five pages.

<Ctrl-B> Move backwards one page.

- **5<Ctrl-B>** moves backwards five pages.

<Ctrl-D> Move forwards by half a page.

<Ctrl-U> Move backwards by half a page.

<Ctrl-E> Display one more line at the bottom of the screen.

<Ctrl-Y> Display one more line at the top of the screen.

Inserting

All insert commands put **vi** into *insert mode*. *Insert mode* is terminated by the ESC key.

<i> Enters *insert mode* at the cursor position.

<I> Enters *insert mode* at the start of the current line.

<a> Enters *insert mode* after the cursor, or appends.

<A> Enters *insert mode* at the end of the current line, or append to the end of the current line.

<o> Inserts a new line underneath the current line and then goes into *insert mode*.

<O> Inserts a new line above the current line and then goes into *insert mode*.

Replacing

r Replaces the character underneath the cursor with the next character typed. Can be preceded by a number, **5ra** replaces 5 characters with the letter *a*.

R Enters *replace mode*. Each time a letter is typed it replaces the one under the cursor and the cursor moves to the next character. *Replace mode* is terminated by the ESC key. Can be preceded by a number, **5Rab** followed by ESC replaces the character under the cursor by *a*, the next character by *b* and then inserts another 4 *ab*s. The original line is placed into the buffer, replacing any text already there.

Deleting

Each time a delete command is used, the deleted text is placed into the buffer, replacing any text already in the buffer. Buffered text can be retrieved by **p** or **P**.

dd Deletes the current line. Can be preceded by a number.

- **5dd** deletes five lines. **d5d** is the same as **5dd**.

de Deletes from the character underneath the cursor to the end of the word. Can be preceded by a number.

- **5de** deletes five words. **d5e** is the same as **5de**.

dE As **de** but takes into account punctuation.

dw Deletes from the character underneath the cursor to the start of the next word. Can be preceded by a number.

- **5dw** deletes five words. **d5w** is the same as **5dw**.

dW As **dw** but takes into account punctuation.

db Deletes from the left of the cursor to the start of the previous word. Can be preceded by a number.

- **5db** deletes five words to the left of the cursor.

dB As **db** but takes into account punctuation.

dt*c* Deletes from the cursor position to before the first instance of the character.

- **dta** deletes text up and to, but not including, the first letter 'a'.

df*c* Deletes from the cursor position to the first instance of the character.

- **dfa** deletes text up and to, and including, the first letter 'a'.

dG Deletes the current line and everything to the end of the file.

d/string Deletes from the cursor to the string, either forwards or backwards.

D Deletes from the cursor to the end of the line.

d$ Same as **D**.

d^ Deletes from the left of the cursor to the start of the line.

---

x Delete the character underneath the cursor. Can be preceded by a number.

- **5x** deletes the character underneath the cursor and the next 4 characters.
- **xp** swaps the character underneath the cursor with the one to the right of it.

X Delete the character to the left of the cursor, but will not delete the end of line marker or any characters on the next line. Can be preceded by a number.

- **5X** deletes 5 characters to the left of the cursor.

Changing

The change commands all select text to be removed, the end of which is indicated by a **$**. Insert mode is entered and new text overwrites or extends the text. When the <ESC> key is pressed to terminate the insert, any remaining original text is deleted.

Text deleted during a change is placed into the buffer, replacing any text already there. Buffered text can be retrieved by **p** or **P**.

C Change from the cursor position to the end of the line. Can be preceded by a number.

- **5C** changes 5 lines, the current line and the next 4 lines.

---

c*M* Generally the same as **d***M***i**, where *M* is any movement command.

cc Change the current line. Can be preceded by a number.

- **5cc** changes 5 lines, the current line and the next 4 lines.

ce Change the current word. Can be preceded by a number.

- **5ce** changes five words. **c5e** is the same as **5ce**.

cw Exactly the same as **ce**.
This command is inconsistent with the usual vi moving: **cw** and **ce** is the same as **dei** but **dwi** removes also the spaces until the next word.

ct*c* Changes from the cursor position to the first instance of the character.

- **cta** changes text up and to, but not including, the first letter 'a'.

cf*c* Changes from the cursor position to the first instance of the character (including the character *c*).

cG Changes from the start of the current line to the end of the file.

---

s Change the character underneath the cursor. Can be preceded by a number.

- **5s** changes 5 characters, the one under the cursor and the next 4.

Cut and Paste

The *yank* commands copy text into the *vi* buffer. Text is also copied into the buffer by delete and change commands. The *put* or *place* commands retrieve text from the buffer.

yy Yanks the current line into the buffer. Can be preceded by a number.

- **5yy** yanks five lines.

Y Same as **yy**.

yw Yanks from the cursor to the start of the next word into the buffer. Can be preceded by a number.

- **5yw** yanks five words.

---

p If the buffer consists of whole lines, they are inserted after the current line. If it consists of characters only, they are inserted after the cursor.

P If the buffer consists of whole lines, they are inserted before the current line. If it consists of characters only, they are inserted before the cursor.

Searching

Searching uses regular expressions.

/*pattern*/ Searches for the string, which could be a regular expression. Searching is from the cursor position downwards, stopping at the first match. If not found, it will continue from the start of the file to the cursor position. The trailing slash character is optional.

- /abc/ seaches for the first occurrence of *abc*.

/*pattern*/+ Goes to the line after the one containing the search string.

- /abc/+3 goes to the third line after the one containing *abc*.

/*pattern*/e Leaves the cursor on the last character of the string that *pattern* matched.* By adding **+***num* or **-***num* after **e** you can supply an offset in characters to where the cursor gets left. For example: /foo/e+3 will leave the cursor 3 characters past the next occurrence of **foo**.* By using **b** instead of **e** you can specify a character offset from the beginning of the matched string.

/\c*pattern*/ Does a case insensitive search.

?*pattern*? As */pattern/* but searches upwards. The trailing question mark character is optional.

?*pattern*?- Goes to the line above the one containing the search string.

- ?abc?-3 goes to the third line above the one containing *abc*.

<n> Repeat last search.

<N> Repeat last search but in the opposite direction.

<f>*char* Search forward on the current line for the next occurrence of *char*.

<F>*char* Search backward on the current line for the next occurrence of *char*.

<t>*char* Search forward on the current line for the next occurrence of *char* and leave the cursor on the character before *char*.

<T>*char* Search backward on the current line for the next occurrence of *char* and leave the cursor on the character after *char*.

<;> Repeat the last **f** or **F** search.

Search and Replace

Search and replace uses regular expressions and the Ex command **:substitute** (short **:s**) which has syntax similar to the sed utility - which is not surprising sed, Ex and w:Vi have common roots - the Ed editor.

:.s/*pattern*/*replacement*/ Replaces the first occurrence of *pattern* on the current line with *replacement*.* If *pattern* contains **\(** and **\)** they are used to remember what matched between them instead of matching parenthesis characters. For example **:.s/\(\d*\)-\(\d*\)/\2:\1/** could match the string **12345-6789** and substitute **6789:12345** for it.

:.s/*pattern*/*replacement*/g Replaces all occurrences of *pattern* on the current line with *replacement*.

:%s/*pattern*/*replacement*/g Replaces all occurrences of *pattern* in the whole file with *replacement*.

:x,y**s/***pattern*/*replacement*/g Replaces all occurrences of *pattern* on lines *x* through *y* with *replacement*.* For example: **:**14,18**s/**foo/bar/g will replace all occurrences of **foo** with **bar** on lines 14 through 18.

- The character **.** can be used to indicate the current line and the character **$** can be used to indicate the last line. For example: **:**.,$**s/**foo/bar/g will replace all occurrences of **foo** with **bar** on the current line through the end of the file.

The following meta-characters have special meaning in the replacement pattern:

& Replaced by the text that matched the search pattern.

\n Replaced by the text matching the search pattern between \(and \), where *n* is in the range of 1 through 9 with *1* being replaced by the match from the first set.

\u Capitalize the next character (if the character is a letter).

\l Lowercase the next character (if the character is a letter).

\U Turn on capitalization mode, where all of the following characters are capitalized.

\L Turn on lowercase mode, where all of the following characters are lowercased.

\E Turn off capitalization or lowercase mode.

\e Turn off capitalization or lowercase mode.

For example **.:s/\(foo\) \(bar\) \(baz\)/\u\1 \U\2\E \3/** could match the string **foo bar baz** and substitute **Foo BAR baz** for it.

Mark Text

Marked lines can be used when changing or deleting text.

<m>*m* Mark the current line with the letter.

- <m><a> marks the current line with the letter *a*.

<'>*m* Move to the line marked by the letter.

- <'><a> moves to the line marked by *a*.

Screen Refresh

<Ctrl-L> Refresh the screen.

z<CR> Refreshes the screen so that the current line is at the top. Can be preceded by a line number.

- **35z** refreshes the screen so that line 35 is at the top.

/pattern/z Finds the line with the first occurrence of *string* and then refreshes the screen so that it is at the top.

z. Refreshes the screen so that the current line is in the middle of the screen. Can be preceded by a line number, in which case the line is at the middle. The sequence **zz** also has the same effect.

- **35z.** refreshes the screen so that line 35 is in the middle.

/string/z. Finds the line with the first occurrence of *string* and then refreshes the screen so that it is in the middle.

z- Refreshes the screen so that the current line is at the bottom. Can be preceded by a line number, in which case the line is at the bottom.

- **35z-** refreshes the screen so that line 35 is at the bottom.

/string/z- Finds the line with the first occurrence of *string* and then refreshes the screen so that it is at the bottom.

Others

<~> Changes the case of the character underneath the cursor and moves to the next character. Can be preceded by a number, so that **5~** changes the case of 5 characters.

<.> Repeats the last insert or delete. Can be preceded by a number, **dd** followed by **5.** deletes a line and then deletes another 5 lines.

<%> Moves the cursor to the matching bracket, any of (), [] or {}.

<Ctrl-G> Temporarily displays a status line at the bottom of the screen.

:f Same as <Ctrl-G>.

<J> Joins the next line to the end of the current line. Can be preceded by a number. Both **1J** and **2J** do the same as **J**.

- **3J** joins three lines together, the current line and the next two lines.

<u> Undoes the last change. A second **u** puts the change back.

<U> Undoes all changes to the current line.

<Ctrl-Z> Puts *vi* into the background, that is control is returned to the operating system. In UNIX, the *vi* session can be returned to the foreground with **fg**.

Saving and Quitting

<Z><Z>	Saves and quits. It is symbolic of sleep, indicating the end of work.
:quit **:q**	Quits, but only if no changes have been made.
:quit! **:q!**	Quits without saving, regardless of any changes.
:write **:w**	:write!}} **myfile** saves to the file called *myfile*.
:write!*filename* **:w!***filename*	Saves to the file, overwriting any existing contents.
:wq **:write|quit**	Saves and quits.
:exit **:xit** **:x**	Saves and quits.
:exit! **:xit!** **:x!**	Used to save and quit in **view**.

Files

:e filename	Quits the current file and starts editing the named file.
:e + filename	Quits the current file and starts editing the named file with the cursor at the end of the file. • **:e +5 myfile** quits the current file and begins editing *myfile* at line 5.
:e!	The current file is closed, all unsaved changes discarded, and the file is re-opened for editing.
:e#	Quits the current file and starts editing the previous file.
:n	When multiple files were quoted on the command line, start editing the next file.
:n files	Resets the list of files for **:n**. The current file will be closed and the first file in the list will be opened for editing.
:r filename	Read a file, that is insert a file. • **:r myfile** inserts the file named *myfile* after the cursor. • **:5r myfile** inserts the file after line 5.

vi Options

All options are *ex* options, and so require an initial colon.

Default options may be placed into a file in the user's home directory called **.exrc**. Options in this file do not have the initial colon, e.g.

set ic

:set all	Displays all the current settings.

! Set on	! Set off	! Meaning
:set ignorecase **:set ic**	**:set noignorecase** **:set noic**	Ignore case. Makes searching case insensitive.
:set list	**:set nolist**	Shows control characters. *<Ctrl-I>* is tab, *$* is linefeed.
:set number **:set nu**	**:set nonumber** **:set nonu**	Turns on line numbering.
:set term		Displays the terminal type.

ex Commands

ex commands start with **:**, which puts *vi* into *last line mode*, entered on the last line of the screen. Spaces within the command are ignored.

:!command Executes the named operating system command and then returns to **vi**.

- **:! ls** runs the UNIX *ls* command.

:sh Starts up a shell. **exit** returns to the **vi** session.

:vi Exit *last line mode* and return to normal *command mode*.

ex line commands

These commands edit lines and have the following syntax:

1. No line number, meaning work on the current line.
2. With **%**, meaning work on all lines.
3. A pair of line numbers, such as '3,5' meaning work on lines 3 to 5 inclusive. Either number can be replaced with **.**, standing for the current line or **$** standing for the last line. So **.,$** means from the current line to the end of the file and **1,$** means the same as **%**. Additionally simple arithmetic may be used, so **.+1** means the line after the current line, or **$-5** means 5 lines before the last line.

co Copy, followed by the line position to copy to.

- **:co 5** copies the current line and places it after line 5.
  - **:1,3 co 4** copies lines 1 to 3 and places after line 4.

d Delete.

- **:d** deletes the current line.
  - **:.,.+5d** delete the current line and the next 5 lines.
  - **:%d** deletes all lines.

m Move, followed by the line position to move to.

- **:m 10** moves the current line and places it after line 10.
  - **:1,3 m 4** moves lines 1 to 3 and places after line 4.

Mapping / Remapping vi Commands

:map Create new command or overwrite existing in vi command mode.

- **:map v i--<Ctrl-[>** new command **v** will insert -- and return to command mode. <Ctrl-[> is the escape character typed as <CTRL-V><ESC>.

:map! Create new command in both command and insert mode.

- **:map! ;r <Ctrl-[>** typing *;r* in insert mode will return to command mode.

External link

- vim Official Reference Manual [1]
- An introduction to the vi editor [2]
- Learning Unix in 10min [3]

References

[1] http://vimdoc.sourceforge.net/htmldoc/ref_toc.html
[2] http://planzero.org/tutorials/introduction_to_the_vi_editor.php
[3] http://freeengineer.org/learnUNIXin10minutes.html

Guide to Unix/GNU Free Documentation License

Version 1.2, November 2002

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of

historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements."

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free

Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

Article Sources and Contributors

Guide to Unix/Commands *Source*: http://en.wikibooks.org/w/index.php?oldid=1619471 *Contributors*: A thing, Adrignola, Arky, CFeyecare, Codeman, DavidCary, Davidbspalding, Djamund, Eibwen, Fractal3, Fudz, Herbythyme, Hyad, Imran, Jomegat, Kernigh, Krischik, Perl, Reisio, RrsunN, Siddharthc, Uncle G, 22 anonymous edits

Guide to Unix/Commands/Summary *Source*: http://en.wikibooks.org/w/index.php?oldid=1619487 *Contributors*: Adrignola, CFeyecare

Guide to Unix/Commands/Getting Help *Source*: http://en.wikibooks.org/w/index.php?oldid=1619479 *Contributors*: Adrignola, Arky, DukeEgr93, Kernigh, Reisio, Waratah, 4 anonymous edits

Guide to Unix/Commands/File System Utilities *Source*: http://en.wikibooks.org/w/index.php?oldid=1619595 *Contributors*: Adrignola, Arky, Kernigh, Pavium, Perpsectoff, PurplePieman, Reisio, RrsunN, Spoon!, Waratah, 12 anonymous edits

Guide to Unix/Commands/Finding Files *Source*: http://en.wikibooks.org/w/index.php?oldid=1619478 *Contributors*: Adrignola, Al17, Arky, Kernigh, Pavium, Reisio, 17 anonymous edits

Guide to Unix/Commands/File Viewing *Source*: http://en.wikibooks.org/w/index.php?oldid=1619477 *Contributors*: Adrignola, Arky, Geocachernemesis, Hyad, Kernigh, Reisio, 9 anonymous edits

Guide to Unix/Commands/File Editing *Source*: http://en.wikibooks.org/w/index.php?oldid=1619583 *Contributors*: Adrignola, Fudz, Hagindaz, Jguk, Jomegat, Krischik, 9 anonymous edits

Guide to Unix/Commands/File Compression *Source*: http://en.wikibooks.org/w/index.php?oldid=1619475 *Contributors*: Adrignola, Djamund, Fan, Kernigh, Koppe, Reisio, Spoon!, Wm, 11 anonymous edits

Guide to Unix/Commands/File Analysing *Source*: http://en.wikibooks.org/w/index.php?oldid=1619474 *Contributors*: A thing, Adrignola, Arky, Kernigh, Reisio

Guide to Unix/Commands/Multiuser Commands *Source*: http://en.wikibooks.org/w/index.php?oldid=1619483 *Contributors*: Adrignola, Arky, Kernigh, Reisio, 12 anonymous edits

Guide to Unix/Commands/Self Information *Source*: http://en.wikibooks.org/w/index.php?oldid=1619485 *Contributors*: Adrignola, Arky, Djamund, Kernigh, Reisio, RrsunN, Shiva kg123, 1 anonymous edits

Guide to Unix/Commands/System Information *Source*: http://en.wikibooks.org/w/index.php?oldid=1620136 *Contributors*: Adrignola, Arky, Djamund, Kernigh, Reisio, 6 anonymous edits

Guide to Unix/Commands/Networking *Source*: http://en.wikibooks.org/w/index.php?oldid=1619580 *Contributors*: Adrignola, Kane77

Guide to Unix/Commands/Process Management *Source*: http://en.wikibooks.org/w/index.php?oldid=1619484 *Contributors*: Adrignola, AlbertCahalan, Arky, Iamunknown, Jguk, Kernigh, Reisio, 4 anonymous edits

Guide to Unix/Commands/Devices *Source*: http://en.wikibooks.org/w/index.php?oldid=1619472 *Contributors*: Adrignola, Arky, Kernigh, Reisio

Guide to Unix/Commands/Kernel Commands *Source*: http://en.wikibooks.org/w/index.php?oldid=1619481 *Contributors*: Adrignola, Arky, Kernigh, Reisio, 1 anonymous edits

Guide to Unix/Commands/compress Commands *Source*: http://en.wikibooks.org/w/index.php?oldid=1619586 *Contributors*: Adrignola, 3 anonymous edits

Guide to Unix/Commands/Miscellaneous *Source*: http://en.wikibooks.org/w/index.php?oldid=1619482 *Contributors*: Adrignola, Arky, Kernigh, Pavium, Reisio, RrsunN, 8 anonymous edits

Guide to Unix/Environment Variables *Source*: http://en.wikibooks.org/w/index.php?oldid=1425263 *Contributors*: A thing, CFeyecare, Kernigh, Reisio, 6 anonymous edits

Guide to Unix/Files *Source*: http://en.wikibooks.org/w/index.php?oldid=1252200 *Contributors*: DuLithgow, Imran, Jguk, Kernigh, Reisio, Spoon!, Waratah, 9 anonymous edits

Guide to Unix/BSD/Introduction *Source*: http://en.wikibooks.org/w/index.php?oldid=1619458 *Contributors*: A thing, Adrignola, Gronau, Jguk, Kernigh, NuShrike, Reisio, 6 anonymous edits

Guide to Unix/BSD/OpenBSD *Source*: http://en.wikibooks.org/w/index.php?oldid=1619461 *Contributors*: Adrignola, CFeyecare, Darklama, Gronau, Kernigh, Nmontague, Reisio, 11 anonymous edits

Guide to Unix/Explanations/Shell Prompt *Source*: http://en.wikibooks.org/w/index.php?oldid=1619579 *Contributors*: A thing, Adrignola, Chridd, Hagindaz, Kernigh, PurplePieman, Reisio, Vinchi007, 5 anonymous edits

Guide to Unix/Explanations/Quoting and Filename Expansion *Source*: http://en.wikibooks.org/w/index.php?oldid=1619578 *Contributors*: A thing, Adrignola, Kernigh, Reisio, 3 anonymous edits

Guide to Unix/Explanations/Pipes and Job Control *Source*: http://en.wikibooks.org/w/index.php?oldid=1619567 *Contributors*: Adrignola, Aholt, Kernigh, Reisio, 1 anonymous edits

Guide to Unix/Explanations/Signals *Source*: http://en.wikibooks.org/w/index.php?oldid=1619590 *Contributors*: Adrignola, Aholt, 2 anonymous edits

Bourne Shell Scripting/Quick Reference *Source*: http://en.wikibooks.org/w/index.php?oldid=1497129 *Contributors*: Adrignola, Jguk, Krischik, Mihai Capotă, 22 anonymous edits

Guide to Unix/Explanations/Introduction to Editors *Source*: http://en.wikibooks.org/w/index.php?oldid=1619566 *Contributors*: A thing, Adrignola, Calabimanifold, Kernigh, Reisio, Vinchi007

Learning the vi editor/vi Reference *Source*: http://en.wikibooks.org/w/index.php?oldid=1481982 *Contributors*: Jguk, Jonathan Webley, Jusjih, Krischik, Mahanga, Van der Hoorn, 28 anonymous edits

Guide to Unix/GNU Free Documentation License *Source*: http://en.wikibooks.org/w/index.php?oldid=510353 *Contributors*: Jguk, Kernigh, Reisio

License