UNIX/Linux Essentials

This document is a short summary of commonly used unix/linux commands and tips.

Command Syntax

- All common UNIX commands are lower-case.
- Options always (almost) start with a dash (-).
- General Syntax:

  $ command [options] [expression] [filename ...]

- Conventions:
  - Square brackets ([ ]) indicate optional items. The brackets are never typed.
  - Ellipses (. . .) indicate that the previous item(s) can be repeated indefinitely.
- You can separate commands on the same line with a semicolon (;).
  Example:

```
$ date; who; pwd
```

- You can continue a long line with a backslash return character pair.
  Example:

```
$ tbl -TX alpha beta gamma \
delta omega | eqn | troff -mm \
-TLJ3 | postprocessor | lp -d3
```

Getting Help

man	an interface to the on-line reference manuals
apropos	search the manual page names and descriptions
whatis	display manual page descriptions
which	locate a command

How to read man pages

NAME

command - one-line title

SYNOPSIS

Detailed command definition
cat [**-v** [**-t**] [**-e**]] [**-** | *file*] ...
Text in brackets is optional.
Options can be nested.
The "|" is a logical "or" function.
The elipsis (...) means anything to the left can be repeated.
Words in *italics* are descriptive.
All headings are not always used.

DESCRIPTION

A detailed description of the command, including all options.

FILES

Related files go here.

WARNING

Cautions and warnings, if appropriate, go here.

SEE ALSO

A list of related commands goes here.

BUGS

Known bugs are reported here.
Bugs are rare – usually reported as features.

CREDITS TO

Author of local programs.

Working with files

`cd`	changing working directory
`mv`	move (rename) files
`rm`	remove files or directories
`ls`	list directory contents
`cp`	copy files and directories

pwd print name of current/working directory

mkdir make directories

rmdir remove empty directories

find search for files in a directory hierarchy

vim a programmers text editor

less view text files

file determine file type

diff compare files line by line

head output the first part of files

tail output the last part of files

tar archiving utility

Examples

ls --help list of all the command options for ls

ls -hl list all files in current directory in human readable and long format. The human readable option means that rather than listing all files in term of bytes, when possible it will list them in term of Kbytes and Mbytes

ls –Slh list all files in current directory in order of size (largest first) detailing the sizes in bytes, Kbytes, Mbytes or Gbytes.

diff –y *file1 file2* place the two lists side by side and highlights lines that are different with a pipe (|)

file –f *filelist* determine the type of the files named in *filelist*

file *filename* determine *filename*'s file type

File Permissions

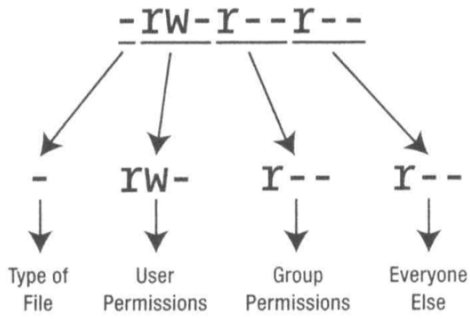

Figure 14-2. *The file permissions part of a file listing can be broken down into four separate parts.*

Table 14-2. *File Type Codes*

Code	File Type
-	Standard file
d	Standard directory
l	Symbolic link (a shortcut to another file)
p	Named pipe (a file that acts as a conduit for data between two programs)
s	Socket (a file designed to send and receive data over a network)
c	Character device (a hardware device driver, usually found in `/dev`)
b	Block device (a hardware device driver, usually found in `/dev`)

Altering Permissions

`chmod` change file mode bits

`chown` change file owner and group

`chgrp` change group ownership

`umask` set file mode creation mask

Network related commands

`ifconfig` view or configure network setting

`ping` check a network connection

`tracepath` trace a network route

Miscellaneous Text Processing Utilities

`cat`	concatenate files and print on the standard output
`wc`	print newline, word, and byte counts for each file
`grep`	print lines matching a pattern
`sort`	sort lines of text files
`uniq`	report or omit repeated lines
`aspell`	interactive spell-checker
`look`	looks up word in the dictionary

Examples

aspell –c *filename*	any questionable words within filename are highlighted and a choice of replacements is offered
look *word*	looks up word in the dictionary (single word spell check)
grep 'hello world' *myfile*	search for the phrase *hello world* within *myfile*

Regular Expressions

Regular Expressions are strings of literal and special characters for string searches in editors and utilities.

A regular expression is taken literally except for the following special characters:

^ Anchor to beginning of line.

^The	Matches lines that start with 'The'.

$ Anchor to end of line.

figure$	Matches lines that end with 'figure'.
^90210$	Matches lines with only '90210'.
^$	Matches empty lines.

. Match any single character.

part.	Matches lines with 'part0', 'part1', 'part9', 'party',

* Match zero or more of preceding character.

Page 23*	Matches lines with 'Page 2', 'Page 23', 'Page 2333'.
.*	Matches any number of any characters.
^ *$	Matches empty lines, even those with any number of space characters.

[*set*] Match any one character in set.
NOTE: *set* is a character list without delimiters.

[aA]lpha	Matches lines with 'alpha' or 'Alpha'.
[aeiou]$	Matches lines that end in a vowel.
^[^I]*$	Matches empty lines, even those with space and/or tab characters.

[^*set*] Match any character not in *set*.

[^NV]	Matches any character except N and V.

[*x-y*]	Match any character between *x* and *y*.
	[A-MW-ZO-U] Matches all upper-case letters except N and V.
	^[-+]*[1-9][0-9]* Matches lines that start with an optionally signed integer without a leading zero, such as 123, -5, -123, +1, +123, but not 0, 01234, -034, +05.
	[1-9][0-9][0-9]* Matches lines with numbers of 2 digits or more.
\c	Literal character *c*.
	***** Matches a string of 5 '*'.
	$[1-9][0-9]*\.[0-9][0-9] Matches non-zero dollar amounts with 2 decimal places.
\<	Start of word (ex/vi and some others)
\>	End of word (ex/vi and some others)
	\<the\> Matches 'the' but not 'them'.

Managing Processes

ps	Report a snapshot of the current processes
jobs	List active background jobs
top	Display system summary and running tasks
kill	Kill an unwanted process.
fg	Move job to the foreground
bg	Move job to the background
pstree	Display a tree of processes (show which parent owns which child processes)

Examples

ps –aux	Display information about other users' processes as well as your own (-a). Display the usernames the processes belong to (-u). When displaying processes matched by other options, include processes which do not have a controlling terminal (-x)

top

Here is an example of a line taken from top, shown with the column headings from the process list:

```
PID  USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
5499 root      15   0 78052  25m  60m S  2.3  5.0   6:11.72 Xorg
```

A lot of information is presented here, as described in Table 16-1.

Table 16-1. *The top Program Process Information*

Column	Description
PID	The first number is the process ID (PID). This is the unique number that the system uses to track the process. The PID comes in handy if you want to kill (terminate) the process (as explained in the next section of this chapter).
USER	This column lists the owner of the particular process. As with files, all processes must have an owner. A lot of processes will appear to be owned by the root user. Some of them are system processes that need to access the system hardware, which is something only the root user is allowed to do. Other processes are owned by root for protection; root ownership means that ordinary users cannot tamper with these processes.
PR	This column shows the priority of the process. This is a dynamic number, showing where the particular process is in the CPU queue at the present time.
NI	This column shows the "nice" value of the process. This refers to how charitable a process is in its desire for CPU time. A high figure here (up to 19) indicates that the process is willing to be interrupted for the sake of other processes. A negative value means the opposite: the process is more aggressive than others in its desire for CPU time. Some programs need to operate in this way, and this is not necessarily a bad thing.
VIRT	This column shows the amount of virtual memory used by the process.[1]
RES	This column shows the total amount of physical memory used.[1]
SHR	This column shows the amount of shared memory used. This refers to memory that contains code that is relied on by other processes and programs.
S	This column shows the current status of the task. Generally, the status will either be sleeping, in which case an S will appear, or running, in which case an R will appear. Most processes will be sleeping, even ones that appear to be active. Don't worry about this; it just reflects the way the Linux kernel works. A Z in this column indicates a zombie process (a child of a process that has been terminated).
%CPU	This column shows the CPU use, expressed as a percentage.[2]
%MEM	This column shows the memory use, again expressed as a percentage.[2]
TIME+	This column shows a measure of how long the process has been up and running.
COMMAND	This shows the actual name of the process itself.

[1] *Both VIRT and RES are measured in kilobytes unless an m appears alongside the number; in which case, you should read the figure as megabytes.*

[2] *The %CPU and %MEM entries tell you in easy-to-understand terms how much of the system resources a process is taking up.*

NOHUP

What if you want to start a command running in a terminal window, but then want to close that terminal window? As soon as you close the window, any processes started within it are also closed. Try this now—type `gcalctool` at the prompt to start the Calculator application and then quit the terminal window.

To get around this, you can use the `nohup` command. This stands for "no hangup," and in simple terms, it tells the command you specify to stick around, even after the process that started it has ended (technically, the command is told to ignore the SIGHUP signal). However, commands run via `nohup` can still be killed.

To use `nohup`, simply add it before the command, for example:

```
nohup unzip myfile.zip
```

If the command requires `sudo` or `gksu` powers, add either of these after the `nohup` command.

Any command output (including error messages) is sent to the file `nohup.out`, which you can then view in a text editor. Note that if you run a command via `nohup` using `sudo` or `gksu`, the `nohup.out` file will have root privileges. If that's the case, you will also have to delete the `nohup.out` file via `sudo` before you can use `nohup` again as an ordinary user, because otherwise, `nohup` will be unable to overwrite the root-owned `nohup.out`.

Useful Shell Tips

`history`	List the most recent commands executed
`!!`	Repeat the last command
`!`*n*	Repeat command *n* from the history list
`!`*PATTERN*	Repeat last command beginning with *PATTERN*
`up-arrow`	Scroll backward through the previous shell commands
`down-arrow`	Scroll forward through the previous shell commands
`Ctrl+u`	Delete the last line you typed
`Tab`	While typing, complete file/path name as much as possible
Tab-Tab	Viewing available options
`Delete` or **`Backspace`**	Erase the last character you typed
`Ctrl+c`	Terminate the active program.
`Ctrl+z`	Suspend the current active program. To bring it back, use %*jobnumber*, e.g. %1 (the number comes from the jobs list)

%	Continue last job suspended. Alternatively type **fg** (foreground)
*	Wildcard. Any number of characters (not "**.**"). Can be used to express patterns matching multiple file names (e.g. typing **ls -l** *dir/*.sp* will list all *dir* files ending with .sp)
?	Any single character (not "**.**").
command **&**	If you put & at the end of a *command,* the process runs in the background, letting you type more commands on the shell (e.g. firefox &)

Other Basic Commands

clear	Clears the screen
eject	Eject a CD/DVD
sftp	Secure file transfer between remote computers
ssh	Secure access to a remote computer
exit	Cause normal process termination
w	Report who is logged in and what they are doing
date	Display the current date and time
cal	Displays a calendar
logname	Print user's login name
dmesg	Display the system's boot up messages
echo	Write arguments to the standard output
last	Shows recent system logins
free	Display information about memory usage (add –m option to see output im megabytes)
help	Show a list of commonly used BASH commands
halt	Shutdown the system (needs to be run as root)
reboot	Restarting the system (needs to be run as root)

uptime Show how long system has been running

touch Change file access and modification times

du Display disk usage statistics

df Display free disk space

tee Read from standard input and write to standard output and files

uname Print operating system info

lscpu CPU architecture information helper

Examples

df –h Display free disk space in human readable format

du –sh *mydirectory* Summarize (-s) disk usage of *mydirectory* in human readable format (-h)

ssh –CY *user@host* Secure access to a remote computer

sftp *user@host* Start secure file transfer session

Table 33-1. *Common sftp Commands*

Command	Function
cd	Change the remote directory
lcd	Change the local directory
get	Download the specified file
ls	List the remote directory
lls	List the local directory
mkdir	Create a directory on the remote machine
lmkdir	Create a directory on the local machine
put	Upload the specified file to the remote machine
pwd	Print the current remote directory
rmdir	Delete the remote directory
rm	Delete the remote file
exit	Quit sftp
!*command*	Execute the specified command on the local machine
!	Start a temporary local shell session (type exit to return to sftp)
help	Show a list of commands

Standard Input and Output

All standard Unix/Linux commands make use of 3 standard I/O file descriptors:

fd (file descriptor)	Description	Name	Default
0	Standard input	Stdin	Keyboard
1	Standard output	Stdout	Screen
2	Standard error	Stderr	Screen

I/O Redirections and Piping

command **>** *filename*	Write the stdout of *command* to *filename*. If *filename* already exists it will be overwritten.
command **>>** *filename*	Append the stdout of *command* to the end of an existing *filename*
command **2>** filename	Write the stderr of *command* to *filename*. If *filename* already exists it will be overwritten.
command **2>>** filename	Append the stderr of *command* to the end of an existing *filename*
command **<** *filename*	Read the *command*'s stdin from *filename*
command **<<** *word*	Read the following lines until a line with only *word* and use these as stdin
command1 **|** *command2*	Redirect the output of *command1* to the input of *command2*. This is called "piping" the output of the first command into the input of the second *command*.
*n***>&***N*	Set the file descriptor *n* to whatever the file descriptor *N* points to. If n is missing, the default used is stdout.

Ubuntu Linux File System

Table 14-3. *Directories and Files in the Ubuntu Root File System*

Directory	Contents
bin	Vital tools necessary to get the system running or for use when repairing the system and diagnosing problems
boot	Boot loader programs and configuration files (the boot loader is the menu that appears when you first boot Linux)
cdrom -> media/cdrom	Symbolic link (shortcut) to the entry for the CD- or DVD-ROM drive in the /dev folder (accessing this file will let you access the CD- or DVD-ROM drive)
dev	Virtual files representing hardware installed on your system
etc	Central repository of configuration files for your system
home	Where each user's personal directory is stored
initrd	Used during booting to mount the initial ramdisk
initrd.img -> boot/ initrd.img-2.6.17-10-generic	Symbolic link to the initial ramdisk, which is used to boot Linux
lib	Shared system files used by Linux as well as the software that runs on it
lost+found	Folder where salvaged scraps of files are saved in the event of a problematic shutdown and subsequent file system check
media	Where the directories representing various mounted storage systems are made available (including Windows partitions on the disk)
mnt	Directory in which external file systems can be temporarily mounted
opt	Software that is theoretically optional and not vital to the running of the system (many software packages you use daily can be found here)
proc	Virtual directory containing data about your system and its current status
root	The root user's personal directory
sbin	Programs essential to administration of the system
srv	Configuration files for any network servers you might have running on your system
sys	Mount point of the sysfs file system, which is used by the kernel to administer your system's hardware
tmp	Temporary files stored by the system

Table 14-3. *Directories and Files in the Ubuntu Root File System (Continued)*

Directory	Contents
`usr`	Programs and data that might be shared with other systems (such as in a large networking setup with many users)[1]
`var`	Used by the system to store data that is constantly updated, such as printer spooling output
`vmlinuz -> boot/ vmlinuz-2.6.17-10-generic`	Symbolic link to the kernel file used during bootup

[1] *The `usr` directory contains its own set of directories that are full of programs and data. Many system programs, such as the X11 GUI software, are located within the `/usr` directory. Note that the `/usr` directory is used even if your system will never act as a server to other systems.*

References

[1.] Mike Lamasney, *Introduction to Unix Usage*, UC Berkeley, 1996

[2.] K. Thomas and J. Sicam, *Beginning Ubuntu Linux*, Apress, 3/e, 2008