
A Quick Guide To UNIX

This is an introduction to the UNIX operating system. Unix may seem idiosyncratic, even impenetrable, to begin with but it has the virtue of minimising the number of keystrokes and so speeding up your access to the computer.

The commands listed here are common to different operating systems and shells. They include some of the most useful and frequently used commands in UNIX. The power and utility of most UNIX commands can be enhanced with switches or options preceded by a “-” sign.

More information on the options, the effects and how to use the commands is available by using the **man** command:

man gives manual information on a topic
man grep
displays the manual page about grep
apropos lists all the man(ual) entries relating to a topic (same as **man -k**)
apropos print

Another useful source of information is the on-line EMBnet tutorial which includes a page on UNIX

<http://www.dk.embnet.org/Embnetut/Univsl/unixcmds.html>
or equally

<http://www.uk.embnet.org/Embnetut/Univsl/unixcmds.html>

The general format of this document is that anything in **bold** is a command you can enter. Anything in *italic* is a fake file or directory name you must change according to yours. Anything preceded by a hyphen “-” is an option which will modify the effects of a command. A general description of each command is followed by one or several examples of its use.

FILES

ls lists files in a directory
ls -alF
lists **-a** all files in **-l** long format **-F** identifies directories **/**, executable files ***** and symbolic links **@**, in the current directory
cat concatenates and displays files
cat my.file
displays *my.file* on the screen

chmod modifies the read (**r**), write and delete (**w**), and execute (**x**) permissions of specified files and the search permissions of specified directories. The permission can be set for user (**u**), group (**g**) or other (**o**)
chmod go-w my.file
stops (**-**) anyone else (**go**) changing or deleting (**w**) *my.file*
chmod g+rx my.file
allows (**+**) anyone of my group (**g**) reading, changing, deleting or executing (**rx**) *my.file*
cp copies files
cp orig.file copy.file
cp orig.file subdir/new.file
copies *orig.file* to *new.file* in *subdir* directory
cp subdir/orig.file .
copies *orig.file* from *subdir* to the current directory (**.**) without changing its name
mv moves/renames a file (or directory)
mv oldname newname
mv my.file subdir/my.file
a move (**mv**) is equivalent to a copy (**cp**) followed by a remove (**rm**)
rm removes/deletes a file.
rm oldfile
rm -i *.file
option **-i** (interactive) advised if wildcards (*****) in use
diff compares two files and prints how they differ
diff file1 file2
prints differences to screen options include **-b** to ignore differences in blank space, and **-i** to ignore case
find searches the directory tree for a file
find . -name lostfile -print
will search your current directory (**.**) (and any subdirectories) for *lostfile*
grep searches a file for a string
grep word my.file
grep "two words" my.file
options include **-i** to ignore case and **-n** to print line numbers
vi simple screen oriented text editor

pico simple display oriented text editor
pico myfile.txt
head prints the first few (default = 10) lines of a file
head oddfile
head -20 oddfile
displays first twenty lines of *oddfile*
tail displays last few lines of a file (see head)
more displays a file one screenful at a time
more longfile
hit **<spacebar>** to see the next screen
Note: some people prefer **less**

OUTPUT REDIRECTION

> redirects output of a command to a file
diff file1 file2 > new.file
puts differences into *new.file*
cat one.file two.file > both.file
writes the output of the cat command into *both.file* (overwrites *both.file*)
>> appends a file to the bottom of another
cat three.file >> both.file
appends *three.file* to the bottom of *both.file*
| “pipe” - uses the output of the first command as the input of the second
grep string my.file | wc -l
finds how many lines on which “*string*” occurs (see **grep** and **wc**)

DIRECTORIES

cd changes current directory
cd /etc
go to */etc* directory
cd ..
go up one level in directory tree
cd ../subdir2
go “sideways” to *subdir2*
mkdir creates a new subdirectory
mkdir subdir
rmdir removes a directory - you must delete all the files in it first
rmdir subdir
pwd print working directory, tells your current location (path)