Some Useful Unix Commands

This document contains basic information on some of the most frequently used Unix Commands. (Note: Linux is a type of Unix, so everything here applies.) It is intended for Unix beginners who need a guide to the names and details of commands that are likely to be of use to them. It is based on documents found on the web.

The names of commands are printed in bold, and the names of objects operated on by these commands (e.g. files, directories) are printed in typewriter font. To find out more about any one of these commands, use the "man" command (i.e., **`man cp`** will tell you more about the "cp" command).

Index of Commands

The following symbols are frequently used with these and other commands:

& puts a command in the background, so you can type another command in while it is running. For example, to start up Mathematica in the background, type:

```
mathematica &
```

> redirects the output to a file.

`date > myinfo` puts the date into a new file called "myinfo".
`pwd >> myinfo` appends (two >'s) the present directory to the file.
`ps >! myinfo` overwrites the "myinfo" file with the list of processes.

| denotes a pipe, which is used to take the output from one command and feed it into the input of another.

`grep name myfile | sort` finds all lines in which the string "name" appears in myfile and then sorts them in alphabetical order.
`grep name myfile | sort | less` does the same but feeds the output to "less" so you only see a screen at a time.

---

cat - display or concatenate files

`cat` takes a copy of a file and sends it to the standard output (i.e. to be displayed on your terminal, unless redirected elsewhere), so it is generally used either to read files, or to string together copies of several files, writing the output to a new file.

`cat` `ex`
displays the contents of the file `ex`.
`cat` `ex1 ex2 > newex`
creates a new file `newex` containing copies of `ex1` and `ex2`, with the contents of `ex2` following the contents of `ex1`.

cd - change directory

cd is used to change from one directory to another.

cd dir1
: changes directory so that dir1 is your new current directory. dir1 may be either the full pathname of the directory, or its pathname relative to the current directory.

cd
: changes directory to your home directory (where you logged in to).

cd ..
: moves up to the parent directory of your current directory.

chmod - change the permissions on a file or directory

chmod alters the permissions on files and directories using either symbolic or octal numeric codes. The symbolic codes are given here:

u	user	**+**	to add a permission	**r**	read
g	group	**-**	to remove a permission	**w**	write
o	other	**=**	to assign a permission explicitly	**x**	execute (for files), access (for directories)

The following examples illustrate how these codes are used. (Use "man chmod" to find out the numeric versions.)

chmod u=rw file1
: sets the permissions on the file file1 to give the user read and write permission on file1. No other permissions are altered.

chmod u+x,g+w,o-r file1
: alters the permissions on the file file1 to give the user execute permission on file1, to give members of the user's group write permission on the file, and prevent any users not in this group from reading it.

chmod u+w,go-x dir1
: gives the user write permission in the directory dir1, and prevents all other users having access to that directory (by using **cd**. They can still list its contents using **ls**.)

cp - copy a file

The command **cp** is used to make copies of files and directories.

cp file1 file2
: copies the contents of the file file1 into a new file called file2. **cp** cannot copy a file onto itself.

cp file3 file4 dir1
: creates copies of file3 and file4 (with the same names), within the directory dir1. dir1 must already exist for the copying to succeed.

cp -r dir2 dir3
: recursively copies the directory dir2, together with its contents and subdirectories, to the directory dir3. If dir3 does not already exist, it is created by **cp**, and the contents and subdirectories of dir2 are recreated within it. If dir3 does exist, a subdirectory called dir2 is created within it, containing a copy of all the contents of the original dir2.

cp ../other_dir/file1 .
: copy file1 from another directory (located up one level) to the current directory.

date - display the current date and time

date returns information on the current date and time in the format shown below:-

```
Tue Mar 25 15:21:16 GMT 1997
```

It is possible to alter the format of the output from date. For example, using the command line

`date '+The date is %d/%m/%y, and the time is %H:%M:%S.'`

at exactly 3.10pm on 14th December 1997, would produce the output
`The date is 14/12/97, and the time is 15:10:00.`

diff - display differences between text files

`diff` `file1 file2` reports line-by-line differences between the text files `file1` and `file2`. The default output will contain lines such as **`n1 a n2,n3`** and **`n4,n5 c n6,n7`** , (where n1 a n2,n3 means that `file2` has the extra lines n2 to n3 following the line that has the number n1 in file1, and **`n4,n5 c n6,n7`** means that lines n4 to n5 in **`file1`** differ from lines n6 to n7 in `file2`). After each such line, **`diff`** prints the relevant lines from the text files, with **`<`** in front of each line from `file1` and **`>`** in front of each line from `file2`.

There are several options to **`diff`**, including **`diff -i`**, which ignores the case of letters when comparing lines, and **`diff -b`**, which ignores all trailing blanks.

`diff -c`n
: produces a listing of differences within n lines of context, where the default is three lines. The form of the output is different from that given by **`diff`**, with **`+`** indicating lines which have been added, **`-`** indicating lines which have been removed, and **`!`** indicating lines which have been changed.

`diff` `dir1 dir2`
: will sort the contents of directories `dir1` and `dir2` by name, and then run **`diff`** on the text files which differ.

file - determine the type of a file

`file` tests named files to determine the categories their contents belong to.

`file` `file1`
: can tell if `file1` is, for example, a source program, an executable program or shell script, an empty file, a directory, or a library, but (a warning!) it does sometimes make mistakes.

find - find files of a specified name or type

`find` searches for files in a named directory and all its subdirectories.

`find . -name` `'*.f'` **`-print`**
: searches the current directory and all its subdirectories for files ending in **`.f`**, and writes their names to the standard output. In some versions of Unix the names of the files will only be written out if the **`-print`** option is used.

`find` `/local` **`-name`** `core` **`-user`** `user1` **`-print`**
: searches the directory `/local` and its subdirectories for files called `core` belonging to the user `user1` and writes their full file names to the standard output.

grep - searches files for a specified string or expression

`grep` searches for lines containing a specified pattern and, by default, writes them to the standard output.

`grep` `motif1 file1`
: searches the file `file1` for lines containing the pattern `motif1`. If no file name is given, **`grep`** acts on the standard input. **`grep`** can also be used to search a string of files, so

`grep` `motif1 file1 file2 ... filen`
: will search the files `file1`, `file2`, **`...`** , `filen`, for the pattern `motif1`.

`grep` `-c motif1 file1`
: will give the number of lines containing `motif1` instead of the lines themselves.

`grep` `-v motif1 file1`
: will write out the lines of `file1` that do NOT contain `motif1`.

gzip - compress a file

gzip reduces the size of named files, replacing them with files of the same name extended by **.gz** . The amount of space saved by compression varies.

gzip file1
: results in a compressed file called file1.gz, and deletes file1.

gzip -v file2
: compresses file2 and gives information, in the format shown below, on the percentage of the file's size that has been saved by compression:-
file2 : Compression 50.26 -- replaced with file2.gz

To restore files to their original state use the command **gunzip**. If you have a compressed file file2.gz, then

gunzip file2
: will replace file2.gz with the uncompressed file file2.

help - display information about bash builtin commands

help gives access to information about builtin commands in the bash shell. Using **help** on its own will give a list of the commands it has information about. **help** followed by the name of one of these commands will give information about that commands. **help history**, for example, will give details about the bash shell history listings.

info - read online documentation

info is a hypertext information system. Using the command **info** on its own will enter the info system, and give a list of the major subjects it has information about. Use the command **q** to exit **info**. For example, **info bash** will give details about the bash shell.

kill - kill a process

To kill a process using **kill** requires the process id (PID). This can be found by using **ps**. Suppose the PID is 3429, then

kill 3429
: will probably kill the process. If not,

kill -9 3429
: will do it!

less - scan through a text file page by page

less displays the contents of a file on a terminal one screenful at a time.

less file1
: starts by displaying the beginning of file1. It will scroll up one line every time the return key is pressed, and one screenful every time the space bar is pressed. Type **?** for details of the commands available within **less**. Type **q** if you wish to quit less before the end of file1 is reached.

logout - exit the current terminal session ("exit" works, too!)

logout
: and you're done!

lpr - print out a file

lpr is used to send the contents of a file to a printer. If the printer is a laserwriter, and the file contains PostScript, then the PostScript will be interpreted and the results of that printed out.

lpr `-Pprinter1 file1`
will send the file `file1` to be printed out on the printer `printer1`. To see the status of the job on the printer queue use
lpq `-Pprinter1`
for a list of the jobs queued for printing on `printer1`. (This may not work for remote printers.)

ls - list names of files in a directory

ls lists the contents of a directory, and can be used to obtain information on the files and directories within it.

ls `dir1`
lists the names of the files and directories in the directory `dir1`, (excluding files whose names begin with .). If no directory is named, **ls** lists the contents of the current directory.
ls `-a dir1`
will list the contents of `dir1`, (including files whose names begin with .).
ls `-l file1`
gives details of the access permissions for the file `file1`, its size in kbytes, and the time it was last altered.
ls `-l dir1`
gives such information on the contents of the directory `dir1`. To obtain the information on `dir1` itself, rather than its contents, use **ls** `-ld dir1`
ls `-alF`
is a good combination of options for a complete listing.

man - display an on-line manual page

man displays on-line reference manual pages.

man `command1`
will display the manual page for `command1`, e.g **man** `cp`, **man** `man`.
man **-k** `keyword`
lists the manual page subjects that have keyword in their headings. This is useful if you do not yet know the name of a command you are seeking information about.

mkdir - make a directory

mkdir is used to create new directories. To do this you must have write permission in the parent directory of the new directory.

mkdir `newdir`
will make a new directory called `newdir`.

mkdir **-p** can be used to create a new directory, together with any parent directories required.

mkdir **-p** `dir1/dir2/newdir`
will create newdir and its parent directories `dir1` and `dir2`, if these do not already exist.

mv - move or rename files or directories

mv is used to change the name of files or directories, or to move them into other directories. **mv** cannot move directories from one file-system to another, so, if it is necessary to do that, use **cp** instead.

mv `file1 file2`
changes the name of a file from `file1` to `file2` unless `dir2` already exists, in which case `dir1` will be moved into `dir2`.
mv `dir1 dir2`
changes the name of a directory from `dir1` to `dir2`.
mv `file1 file2 dir3`
moves the files `file1` and `file2` into the directory `dir3` (with the same names).

nice - change the priority at which a job is being run

nice causes a command to be run at a lower than usual priority. **nice** can be particularly useful when running a long program that could cause annoyance if it slowed down the execution of other users' commands. An example of the use of **nice** is

nice compress file1
: which will execute the compression of file1 at a lower priority.

passwd - change your password

Use **passwd** when you wish to change your password. You will be prompted once for your current password, and twice for your new password. Neither password will be displayed on the screen.

ps - list processes

ps displays information on processes currently running on your machine. This information includes the process id, the controlling terminal (if there is one), the cpu time used so far, and the name of the command being run.

ps
: gives brief details of your own processes in your current session.

ps -aux | grep firefox
: gives full details of all processes but then selects out those with "firefox" in their name.

ps -fu user1
: obtain full details of all your processes, including those from previous sessions (using your own user name in place of user1).

ps is a command whose options vary considerably in different versions of Unix. Use **man** ps for details of all the options available on the machine you are using.

pwd - display the name of your current directory

The command **pwd** gives the full pathname of your current directory.

rm - remove files or directories

rm is used to remove files. In order to remove a file you must have write permission in its directory, but it is not necessary to have read or write permission on the file itself.

rm file1
: will delete the file file1. If you use

rm -i file1
: instead, you will be asked if you wish to delete file1, and the file will not be deleted unless you answer **y**. This is a useful safety check when deleting lots of files.

rm -r dir1
: recursively deletes the contents of dir1, its subdirectories, and dir1 itself, and should be used with suitable caution.

rmdir - remove a directory

rmdir removes named empty directories. If you need to delete a non-empty directory **rm -r** can be used instead.

rmdir exdir
: will remove the empty directory exdir.

sort - sort and collate lines

The command **sort** sorts and collates lines in files, sending the results to the standard output. If no file names are given, **sort**

acts on the standard input. By default, **sort** sorts lines using a character by character comparison, working from left to right, and using the order of the ASCII character set.

sort -d
uses "dictionary order", in which only letters, digits, and white-space characters are considered in the comparisons.
sort -r
reverses the order of the collating sequence.
sort -n
sorts lines according to the arithmetic value of leading numeric strings. Leading blanks are ignored when this option is used, (except in some System V versions of **sort**, which treat leading blanks as significant. To be certain of ignoring leading blanks use **sort -bn** instead.).

ssh - secure remote login program

ssh is used for logging onto a remote machine and for executing commands on a remote machine, and provides secure encrypted communications between the local and remote machines using an SSH protocol. The remote machine must be running an SSH server for such connections to be possible. For example,

ssh -Y `fox.mps.ohio-state.edu`
will start a login connection to the Physics Department computer named Fox. (If you are connecting from within the Department, you can just use **ssh** `fox`). The "-Y" lets you open windows across the connection.
You can connect using your password for the remote machine, or you can set up a system of passphrases to avoid typing login passwords directly (see the man page for **ssh-keygen** for information on how to create these).
ssh -Y `doe@fox.mps.ohio-state.edu`
will login as user doe (useful if you have a different username on the other computer or if you're ssh'ing from someone else's account).

tar - create and use archives of files

tar is used to create and manage collections of files, including subdirectories, which are called archives. In the following, omit the "z" option if you don't want to use gzip or gunzip (for compressed archives).

tar -cfvz `my_archive.tar.gz my_dir`
will create the compressed archive or "tarball" `my_archive.tar.gz` containing all the files in the directory `my_dir` (and all of its subdirectories).
tar -tfvz `my_archive.tar.gz`
will list the contents of `my_archive.tar.gz`.
tar -xfvz `my_archive.tar.gz`
will unpack the contents of `my_archive.tar.gz`.

Useful Unix Commands. Last modified: 10:57 pm, January 11, 2016.
furnstahl.1@osu.edu