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 -R dir1
also lists the contents of any subdirectories dir1 contains.
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 -ltr
Long listing, sorted by time, in reverse order so the newest file you modified appears at the end.

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.
man -Mpath command1
is used to change the set of directories that man searches for manual pages on command1

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.
cd ..
moves to the parent directory of your current directory.
cd -
jumps back to the previous directory you were in

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.

`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`.

date - display the current date and time

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

```
Wed Jan 30 11:27:50 GMT 2008
```

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 11.30am on 30th January 2008, would produce the output

```
The date is 30/01/08, and the time is 11:30:00.
```

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 motif1 a*`

will search all the files in the current directory with names beginning with 'a' 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
```

should kill the process.

pkill - kill a process

To kill a process by name

```
kill bash
```

mkdir - make a directory

`mkdir` is used to create new directories. In order 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.

more - scan through a text file page by page

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

more 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 **more**. Type **q** if you wish to quit more before the end offile1 is reached.
more -n file1
will cause n lines of file1 to be displayed in each screenful instead of the default (which is two lines less than the number of lines that will fit into the terminal's screen).

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 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.

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.
To obtain full details of all your processes, including those from previous sessions use:-
ps -fu user1
using your own user name in place of user1.
ps is a command whose options vary considerably in different versions of Unix (such as BSD and SystemV). 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.

quota - disk quota and usage

quota gives information on a user's disk space quota and usage.

quota
will only give details of where you have exceeded your disc quota on local disks, whereas
quota -v
will display your quota and usage, whether the quota has been exceeded or not, and includes information on disks mounted from other machines, as well as the local disks.

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.).

`sort -kN`

sorts the k'th column of data

ex. **`ls -l | sort -n -k5`** will sort output of ls by size which appears in the 5th column

ssh - secure remote access

`ssh` (also known as `slogin`) is used for logging onto a remote system, and provides secure encrypted communications between the local and remote systems using the SSH protocol. The remote system must be running an SSH server for such connections to be possible. For example,

```
ssh linux.pwf.cam.ac.uk
```

initiates a login connection to a MCS Linux server.
You can authenticate access by using your password for the remote system, or you can set up a passphrase to avoid typing the login password directly (see the man page for `ssh-keygen` for information on how to create these). If you wish to transfer files over an encrypted connection you can use `sftp` (secure remote file transfer program) or `scp` (secure remote file copy program), with authentication being handled as for `ssh`. For example, you could use `sftp` to connect to the remote system sftp.pwf.cam.ac.uk:

```
sftp sftp.pwf.cam.ac.uk
```

Once you have authenticated access to sftp.pwf.cam.ac.uk, you will be in your home directory on the MCS. You can use the command {\bf cd} to change directories on sftp.pwf.cam.ac.uk and `lcd` to change directories on your local system; `get` can be used to transfer files from the remote system, and `put` to transfer files to the remote system. The command `quit` will terminate the `sftp` session.

sudo - superuser do (run root commands)

apt-get - package manager