Introduction to UNIX Commands and Scripting

0.0 Getting Help

On-line Manuals

There are on-line manuals which gives information about most commands. The manual pages tell you which options a particular command can take, and how each option modifies the behaviour of the ommand.
Type **man** command to read the manual page for a particular command.

For example, to find out more about the **wc** (word count) command, type

man wc

Alternatively

whatis wc

gives a one-line description of the command, but omits any information about options etc.

Apropos

When you are not sure of the exact name of a command,

apropos keyword

will give you the commands with keyword in their manual page header. For example, try typing

apropos copy

Summary

*	match any number of characters
?	match one character
man *command*	read the online manual page for a command
whatis *command*	brief description of a command
apropos *keyword*	match commands with keyword in their man pages

1

1.1 Listing files and directories

ls (list)

When you first login, your current working directory is your home directory. Your home directory has the same name as your user-name and it is where your personal files and subdirectories are saved.

To find out what is in your home directory, type

ls

The **ls** command lists the contents of your current working directory. There may be no files visible in your home directory, in which case, the UNIX prompt will be returned. Alternatively, there may already be some files inserted by the System Administrator when your account was created.

ls does not, in fact, cause all the files in your home directory to be listed, but only those ones whose name does not begin with a dot (.) Files beginning with a dot (.) are known as hidden files and usually contain important program configuration information. They are hidden because you should not change them unless you are very familiar with UNIX!!!

To list all files in your home directory including those whose names begin with a dot, type
ls -a

ls is an example of a command which can take options: **-a** is an example of an option. The options change the behaviour of the command. There are online manual pages that tell you which options a particular command can take, and how each option modifies the behaviour of the command. (See later in this tutorial).

1.2 Making Directories

mkdir (make directory)

We will now make a subdirectory in your home directory to hold the files you will be creating and using in the course of this tutorial. To make a subdirectory called **unixstuff** in your current working directory type
mkdir unixstuff

To see the directory you have just created, type
ls

1.3 Changing to a different directory

cd (change directory)

The command **cd *directory*** means change the current working directory to 'directory'. The current working directory may be thought of as the directory you are in, i.e. your current position in the file-system tree.

To change to the directory you have just made, type
cd unixstuff

Type **ls** to see the contents (which should be empty)

Exercise 1a

Make another directory inside the **unixstuff** directory called **backups**

1.4 The directories . and ..

Still in the **unixstuff** directory, type
ls -a

As you can see, in the **unixstuff** directory (and in all other directories), there are two special directories called (.) and (..). In UNIX, (.) means the current directory, so typing
cd .
NOTE: there is a space between cd and the dot means stay where you are (the **unixstuff** directory).
This may not seem very useful at first, but using (.) as the name of the current directory will save a lot of typing, as we shall see later in the tutorial.
(..) means the parent of the current directory, so typing
cd ..
will take you one directory up the hierarchy (back to your home directory). Try it now.
Note: typing **cd** with no argument always returns you to your home directory. This is very useful if you are lost in the file system.

1.5 Pathnames

pwd (print working directory)

Pathnames enable you to work out where you are in relation to the whole file-system. For example, to find out the absolute pathname of your home-directory, type **cd** to get back to your home-directory and then type
pwd

The full pathname will look something like this
/usr/abc
which means that **abc** (your home directory) is in the **usr** sub-directory, which is in the top-level root directory called "/".

Use the commands **ls**, **pwd** and **cd** to explore the file system.
(Remember, if you get lost, type **cd** by itself to return to your home-directory)

1.6 More about home directories and pathnames

Understanding pathnames

First type **cd** to get back to your home-directory, then type
ls unixstuff
to list the contents of your **unixstuff** directory.

Now type
ls backups

You will get a message like this -

`backups: No such file or directory`

The reason is, **backups** is not in your current working directory. To use a command on a file (or directory) not in the current working directory (the directory you are currently in), you must either **`cd`** to the correct directory, or specify its full pathname. To list the contents of your backups directory, you must type

`ls unixstuff/backups`

~ (your home directory)

Home directories can also be referred to by the tilde ~ character. It can be used to specify paths starting at your home directory. So typing

`ls ~/unixstuff`

will list the contents of your unixstuff directory, no matter where you currently are in the file system.

Exercise 1b

What do you think **`ls ~`** would list?

What do you think **`ls ~/..`** would list?

Summary

`ls`	list files and directories
`ls -a`	list all files and directories
`mkdir`	make a directory
`cd` ***`directory`***	change to named directory
`cd`	change to home-directory
`cd ~`	change to home-directory
`cd ..`	change to parent directory
`pwd`	display the path of the current directory

2

2.1 Copying Files

cp (copy)

`cp` ***`file1 file2`*** is the command which makes a copy of **file1** in the current working directory and calls it **file2**

What we are going to do now, is to take a file stored in an open access area of the file system, and use the **`cp`** command to copy it to your unixstuff directory.

First, **cd** to your unixstuff directory.

```
cd ~/unixstuff
```

Let's create a file called **sample.txt** by typing the following command

```
man cp > sample.txt
```

Then at the UNIX prompt, type,

```
cp sample.txt sample1.txt
```

The above command means making a copy of the file **sample.txt** and has the new file named as **sample1.txt**

Exercise 2a

Create a backup of your **sample.txt** file by copying it to a file called **sample.bak**

2.2 Moving files

mv (move)

mv ***file1 file2*** moves (or renames) **file1** to **file2**

To move a file from one place to another, use the **mv** command. This has the effect of moving rather than copying the file, so you end up with only one file rather than two. It can also be used to rename a file, by moving the file to the same directory, but giving it a different name. We are now going to move the file sample.bak to your backup directory.

First, change directories to your unixstuff directory (can you remember how?). Then, inside the **unixstuff** directory, type

```
mv sample.bak backups/
```

Type **ls** and **ls backups** to see if it has worked.

2.3 Removing files and directories

rm (remove), rmdir (remove directory)

To delete (remove) a file, use the **rm** command. As an example, we are going to create a copy of the **sample.txt** file then delete it. Inside your **unixstuff** directory, type

```
cp sample.txt tempfile.txt
ls (to check if it has created the file)
rm tempfile.txt
ls (to check if it has deleted the file)
```

You can use the **rmdir** command to remove a directory (make sure it is empty first). Try to remove the **backups** directory. You will not be able to since UNIX will not let you remove a non-empty directory.

Exercise 2b

Create a directory called **tempstuff** using **mkdir** , then remove it using the **rmdir** command.

2.4 Displaying the contents of a file on the screen

clear (clear screen)

Before you start the next section, you may like to clear the terminal window of the previous commands so the output of the following commands can be clearly understood.
At the prompt, type

```
clear
```

This will clear all text and leave you with the prompt at the top of the window.

cat (concatenate)

The command **cat** can be used to display the contents of a file on the screen. Type:

```
cat sample.txt
```

As you can see, the file is longer than than the size of the window, so it scrolls past making it unreadable.

less

The command **less** writes the contents of a file onto the screen a page at a time. Type

```
less sample.txt
```

Press the **[space-bar]** if you want to see another page, type **q** if you want to quit reading. As you can see, **less** is used in preference to **cat** for long files.

head

The **head** command writes the first ten lines of a file to the screen. First clear the screen then type

```
head sample.txt
```

Then type

```
head -5 sample.txt
```

What difference did the -5 do to the head command?

tail

The **tail** command writes the last ten lines of a file to the screen. Clear the screen and type

```
tail sample.txt
```

Exercise 2c

How can you view the last 15 lines of the file?

2.5 Tab completion & command history

Ubuntu offers a great feature called tab completion, which can save you a lot of time. Whenever you're typing the names of files, directories, or commands in the Terminal, you can make it help you auto-complete the name by pressing the [Tab] key. For instance , type

```
tail sa
```

and then press the [Tab] key. If there is only one unambiguous choice, Ubuntu will automatically complete the filename to **`sample.txt`** by looking in the directory you're in (as well as some shared ones) for files, commands, or directories whose name starts with **`sa`**. If there are multiple options, it will not auto-complete. You can make it show you a list of all files that start with the characters you have entered so far by pressing [Tab] once more.

Since there are two files in the directory that start with **`sa`**, so tab completion cannot choose for you. It will however complete it as far as it can. If you press [Tab] again, it will show you a list of the possible options:

```
sample.txt    sample1.txt
```

By typing an additional **`1`**, and then pressing [Tab] again, you can make it auto-complete the filename.

Another feature that can save you a lot of time is cycling through the command history with the [Up] and [Down] arrows. Press the [Up] arrow and the Terminal will show you your previous command. Once you've found the command you want to run again, you can just press [Return] and you will have saved yourself the time it takes to type the command. Press the [Down] arrow to go forward in your command history.

2.6 Searching the contents of a file

Simple searching using less

Using **`less`**, you can search though a text file for a keyword (pattern). For example, to search through **sample.txt** for the word 'copy', type

```
less sample.txt
```

then, still in **`less`** (i.e. don't press [q] to quit), type a forward slash **`/`** followed by the word to search

/copy

As you can see, **less** finds and highlights the keyword. Type **n** to search for the next occurrence of the word.

grep (don't ask why it is called grep)

grep is one of many standard UNIX utilities. It searches files for specified words or patterns. First clear the screen, then type

grep copy sample.txt

As you can see, **grep** has printed out each line containing the word copy. Or has it????

Try typing

grep Copy sample.txt

The **grep** command is case sensitive; it distinguishes between Copy and copy. To ignore upper/lower case distinctions, use the -i option, i.e. type

grep -i copy sample.txt

To search for a phrase or pattern, you must enclose it in single quotes (the apostrophe symbol). For example to search for **destination file**, type

grep -i 'destination file' sample.txt

Some of the other options of grep are:

-v display those lines that do NOT match
-n precede each matching line with the line number
-c print only the total count of matched lines

Try some of them and see the different results. Don't forget, you can use more than one option at a time, for example, the number of lines without the words **copy** or **Copy** is

grep -ivc copy sample.txt

wc (word count)

A handy little utility is the **wc** command, short for word count. To do a word count on **sample.txt**, type

wc -w sample.txt

To find out how many lines the file has, type

wc -l sample.txt

Summary

cp *file1 file2*	copy file1 and call it file2
mv *file1 file2*	move or rename file1 to file2
rm *file*	remove a file

rmdir *directory*	remove a directory
cat *file*	display a file
more *file*	display a file a page at a time
head *file*	display the first few lines of a file
tail *file*	display the last few lines of a file
grep *'keyword' file*	search a file for keywords
wc *file*	count number of lines/words/characters in file

3

3.1 Wildcards

The characters * and ?

The character * is called a wildcard, and will match against none or more character(s) in a file (or directory) name. For example, in your **unixstuff** directory, type

`ls list*`

This will list all files in the current directory starting with **list**

`ls *list`

This will list all files in the current directory ending with **list**

The character ? will match exactly one character.
So **`ls ?ouse`** will match files like **house** and **mouse**, but not **grouse**. Try typing

`ls sample?.txt`

3.2 Filename conventions

We should note here that a directory is merely a special type of file. So the rules and conventions for naming files apply also to directories.

In naming files, characters with special meanings such as **/ * & %** , should be avoided. Also, avoid using spaces within names. The safest way to name a file is to use only alphanumeric characters, that is, letters and numbers, together with _ (underscore) and . (dot).

File names conventionally start with a lower-case letter, and may end with a dot followed by a group of letters indicating the contents of the file. For example, all files consisting of C code may be named with the ending .c, for example, prog1.c . Then in order to list all files containing C code in your home directory, you need only type **`ls *.c`** in that directory.

Beware: some applications give the same name to all the output files they generate.
For example, some compilers, unless given the appropriate option, produce compiled files named a.out. Should you forget to use that option, you are advised to rename the compiled file immediately, otherwise the next such file will overwrite it and it will be lost.

4

4.1 Redirection

Most processes initiated by UNIX commands write to the standard output (that is, they write to the terminal screen), and many take their input from the standard input (that is, they read it from the keyboard). There is also the standard error, where processes write their error messages, by default, to the terminal screen.

We have already seen one use of the **cat** command to write the contents of a file to the screen.

Now type **cat** and press the **[Return]** key without specifying a file to read

```
cat
```

Then type a few words on the keyboard and press the **[Return]** key.
Finally hold the **[Ctrl]** key down and press **[d]** (written as ^D for short) to end the input. What has happened?

If you run the **cat** command without specifying a file to read, it reads the standard input (the keyboard), and copies it to the standard output (the screen). In UNIX, we can redirect both the input and the output of commands.

4.2 Redirecting the Output

We use the > symbol to redirect the output of a command. For example, to create a file called **list1** containing a list of fruit, type

```
cat > list1
```

Then type in the names of some fruit. Press **[Return]** after each one.

```
pear
banana
apple
^D (Control D to stop)
```

What happens is the **cat** command reads the standard input (the keyboard) and the > redirects the output, which normally goes to the screen, into a file called **list1**

To read the contents of the file, type
cat list1

Exercise 4a

Using the above method, create another file called **list2** containing the following fruit: orange, plum, mango, grapefruit. Read the contents of **list2**

The form **>>** appends standard output to a file. So to add more items to the file **list1**, type
cat >> list1

Then type in the names of more fruit
peach
grape
orange
^D (Control D to stop)

To read the contents of the file, type
cat list1

You should now have two files. One contains six fruit, the other contains four fruit. We will now use the **cat** command to join (concatenate) **list1** and **list2** into a new file called **biglist**.
Type
cat list1 list2 > biglist

What this is doing is reading the contents of **list1** and **list2** in turn, then output the text to the file **biglist**
To read the contents of the new file, type
cat biglist

4.3 Redirecting the Input

We use the **<** symbol to redirect the input of a command.
The command **sort** alphabetically or numerically sorts a list. Type
sort

Then type in the names of some vegetables. Press **[Return]** after each one.
carrot
beetroot
artichoke
^D (control d to stop)

The output will be
artichoke
beetroot
carrot

Using < you can redirect the input to come from a file rather than the keyboard. For example, to sort the list of fruit, type

```
sort < biglist
```

and the sorted list will be output to the screen. To output the sorted list to a file, type,

```
sort < biglist > slist
```

Use **cat** to read the contents of the file **slist**

4.4 Pipes

To see who is on the system with you, type

```
who
```

One method to get a sorted list of names is to type,

```
who > names.txt
sort < names.txt
```

This is a bit slow and you have to remember to remove the temporary file called names when you have finished. What you really want to do is connect the output of the **who** command directly to the input of the **sort** command. This is exactly what pipes do. The symbol for a pipe is the vertical bar **|**

For example, typing

```
who | sort
```

will give the same result as above, but quicker and cleaner. To find out how many users are logged on, type

```
who | wc -l
```

Exercise 4b

Using pipes, display all lines of **list1** and **list2** containing the letter 'p', and sort the result.

Summary

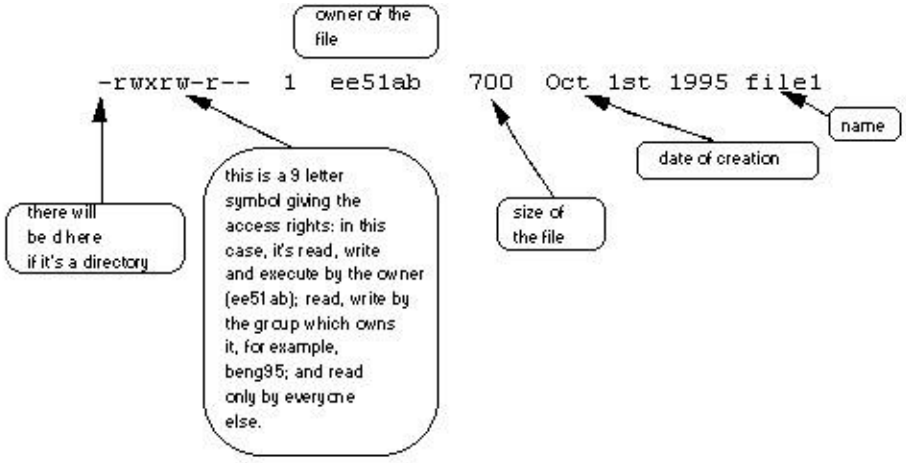
command > file	redirect standard output to a file
command >> file	append standard output to a file
command < file	redirect standard input from a file
command1 | command2	pipe the output of command1 to the input of command2
cat *file1 file2 > file0*	concatenate file1 and file2 to file0
sort	sort data
who	list users currently logged in
a2ps -P*printer textfile*	print text file to named printer
lpr -P*printer psfile*	print postscript file to named printer

5

5.1 File system security (access rights)

In your unixstuff directory, type
`ls -l (l for long listing!)`

You will see that you now get lots of details about the contents of your directory, similar to the example below.



Each file (and directory) has associated access rights, which may be found by typing **`ls -l`**. Also, **`ls -lg`** gives additional information as to which group owns the file (beng95 in the following example):
`-rwxrw-r-- 1 ee51ab beng95 2450 Sept29 11:52 file1`
In the left-hand column is a 10 symbol string consisting of the symbols d, r, w, x, -, and, occasionally, s or S. If d is present, it will be at the left hand end of the string, and indicates a directory: otherwise - will be the starting symbol of the string.

The 9 remaining symbols indicate the permissions, or access rights, and are taken as three groups of 3.

- The left group of 3 gives the file permissions for the user that owns the file (or directory) (ee51ab in the above example);
- the middle group gives the permissions for the **group** of people to whom the file (or directory) belongs (eebeng95 in the above example);
- the rightmost group gives the permissions for all others.

The symbols r, w, etc., have slightly different meanings depending on whether they refer to a simple file or to a directory.

Access rights on files.

r (or -), indicates read permission (or otherwise), that is, the presence or absence of permission to read and copy the file
w (or -), indicates write permission (or otherwise), that is, the permission (or otherwise) to change a file

x (or -), indicates execution permission (or otherwise), that is, the permission to execute a file, where appropriate

Access rights on directories.

r allows users to list files in the directory;
w means that users may delete files from the directory or move files into it;
x means the right to access files in the directory. This implies that you may read files in the directory provided you have read permission on the individual files.

So, in order to read a file, you must have execute permission on the directory containing that file, and hence on any directory containing that directory as a subdirectory, and so on, up the tree.

Some examples

`-rwxrwxrwx` a file that everyone can read, write and execute (and delete).
`-rw-------` a file that only the owner can read and write - no-one else can read or write and no-one has execution rights (e.g. your mailbox file).

5.2 Changing access rights

chmod (changing a file mode)

Only the owner of a file can use **`chmod`** to change the permissions of a file. The options of **`chmod`** are as follows

Symbol	Meaning
u	user
g	group
o	other
a	all
r	read
w	write (and delete)
x	execute (and access directory)
+	add permission
-	take away permission

For example, to remove read write and execute permissions on the file **biglist** for the group and others, type

```
chmod go-rwx biglist
```

This will leave the other permissions unaffected.

To give read and write permissions on the file **biglist** to all,

```
chmod a+rw biglist
```

Exercise 5a

Try changing access permissions on the file **sample.txt** and on the directory **backups**
Use `ls -l` to check that the permissions have changed.

5.3 Processes and Jobs

A process is an executing program identified by a unique PID (process identifier). To see information about your processes, with their associated PID and status, type

```
ps
```

A process may be in the foreground, in the background, or be suspended. In general the shell does not return the UNIX prompt until the current process has finished executing. Some processes take a long time to run and hold up the terminal. Backgrounding a long process has the effect that the UNIX prompt is returned immediately, and other tasks can be carried out while the original process continues executing.

Running background processes

To background a process, type an **&** at the end of the command line. For example, the command **sleep** waits a given number of seconds before continuing. Type

```
sleep 10
```

This will wait 10 seconds before returning the command prompt. Until the command prompt is returned, you can do nothing except wait.

To run **sleep** in the background, type

```
sleep 10 &
[1] 6259
```

The **&** runs the job in the background and returns the prompt straight away, allowing you do run other programs while waiting for that one to finish. The first line in the above example is typed in by the user; the next line, indicating job number and PID, is returned by the machine. The user is be notified of a job number (numbered from 1) enclosed in square brackets, together with a PID and is notified when a background process is finished. Backgrounding is useful for jobs which will take a long time to complete.

Backgrounding a current foreground process

At the prompt, type

```
sleep 100
```

You can suspend the process running in the foreground by holding down the **[control]** key and typing **[z]** (written as **^Z**) Then to put it in the background, type

```
bg
```

Note: do not background programs that require user interaction.

5.4 Listing suspended and background processes

When a process is running, backgrounded or suspended, it will be entered onto a list along with a job number. To examine this list, type

```
jobs
```

An example of a job list could be

```
[1] Suspended sleep 100 [2] Running netscape [3] Running
nedit
```

To restart (foreground) a suspended processes, type

```
fg %jobnumber
```

For example, to restart **`sleep 100`**, type

```
fg %1
```

Typing **`fg`** with no job number foregrounds the last suspended process.

5.5 Killing a process

kill (terminate or signal a process)

It is sometimes necessary to kill a process (for example, when an executing program is in an infinite loop)

To kill a job running in the foreground, type **`^C`** (control c). For example, run

```
sleep 100
^C
```

To kill a suspended or background process, type

```
kill %jobnumber
```

For example, run

```
sleep 100 &
jobs
```

If it is job number 4, type

```
kill %4
```

To check whether this has worked, examine the job list again to see if the process has been removed.

ps (process status)

Alternatively, processes can be killed by finding their process numbers (PIDs) and using

```
kill PID_number
```

```
sleep 100 &
```

```
ps
PID TT S TIME COMMAND
20077        pts/5        S        0:05        sleep        100
21563          pts/5          T          0:00          netscape
21873 pts/5 S 0:25 nedit
```

To kill off the process **sleep 100**, type

```
kill 20077
```

and then type **ps** again to see if it has been removed from the list. If a process refuses to be killed, uses the **-9** option, i.e. type

```
kill -9 20077
```

Note: It is not possible to kill off other users' processes !!!

Summary

ls -lag	list access rights for all files
chmod [*options*] *file*	change access rights for named file
***command* &**	run command in background
^C	kill the job running in the foreground
^Z	suspend the job running in the foreground
bg	background the suspended job
jobs	list current jobs
fg %1	foreground job number 1
kill %1	kill job number 1
ps	list current processes
kill 26152	kill process number 26152

6

6.1 Other useful UNIX commands

df

The df command reports on the space left on the file system. For example, to find out how much space is left on the fileserver, type

```
df
```

du

The du command outputs the number of kilobyes used by each subdirectory. Useful if you have gone over quota and you want to find out which directory has the most files. In your home-directory, type

```
du
```

compress

This reduces the size of a file, thus freeing valuable disk space. For example, type

```
ls -l sample.txt
```

and note the size of the file. Then to compress sample.txt, type

```
compress sample.txt
```

This will compress the file and place it in a file called sample.txt.Z To see the change in size, type ls -l again.

To uncomress the file, use the uncompress command.

```
uncompress sample.txt.Z
```

gzip

This also compresses a file, and is more efficient than compress. For example, to zip sample.txt, type

```
gzip sample.txt
```

This will zip the file and place it in a file called sample.txt.gz To unzip the file, use the gunzip command.

```
gunzip sample.txt.gz
```

file

file classifies the named files according to the type of data they contain, for example ascii (text), pictures, compressed data, etc.. To report on all files in your home directory, type

```
file *
```

history

The C shell keeps an ordered list of all the commands that you have entered. Each command is given a number according to the order it was entered.

```
history (show command history list)
```

If you are using the C shell, you can use the exclamation character (!) to recall commands easily.

```
!! (recall last command)
!-3 (recall third most recent command)
!5 (recall 5th command in list)
!grep (recall last command starting with grep)
```

You can increase the size of the history buffer by typing

```
set history=100
```

Parts of this lab has been adapted from M. Stonebank's UNIX Tutorial for Beginners (http://www.ee.surrey.ac.uk/Teaching/UNIX/).