Tutorial 1: Basic Linux commands

Help: the **man** command

First of all the **man** gives you information about the linux commands (generic help).

Type **man –h** (help)

man command

example**: man ls**

will give you all options concerning the ls (listing) command (see below for details)

List: the **ls** command

When you first login, your current working directory is your home directory. Your home directory has the same name as your user-name, for example, **ee91ab**, and it is where your personal files and subdirectories are saved.

To find out what is in your home directory, type **ls**

Each command can be associated with options:

Example type **ls –ltra** (the –l provides a list as lines, t means it's sorted as a function of time (from the older to the lat modified file) and a display hidden files (system ones, be carrefull when you modify these ones). If you want to know all the available options type **man ls**

Making directories: the **mkdir** command

Example: **mkdir my_first_directory** (in Linux never ever use spaces use underscore..)

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 (Where am I?)

The full pathname will look something like this -

/home/macleod

which means that **macleod** (your home directory) which is in the **home** sub-directory, which is in the top-level root directory called " / " .

Changing directory: cd

Example**:** after creating my_first_directory type: cd my_first_directory

Directories . and ..

In UNIX, (**.**) means the **current** directory, so typing

% cd . means stay where you are

In UNIX, (**..**) means the **parent** directory, so typing

% cd .. will take you one directory up the hierarchy (back to your home directory)

Cd by itself will take you back to your home directory ($HOME) this is typically /home/Macleod

More details:

http://www.ee.surrey.ac.uk/Teaching/Unix/unix1.html

(do the exercices)

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 new directory.

First create the unixtuff dir by typing **mkdir unixstuff**

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

% cd ~/unixstuff

Then at the UNIX prompt, type,

% cp /home/caminade/science.txt .

Note: Don't forget the dot **.** at the end. Remember, in UNIX, the dot means the current directory.

The above command means copy the file **science.txt** to the current directory, keeping the name the same.

You can do as well:

% cp /home/caminade/science.txt /home/macleod/unixstuff/.

Or

% cp /home/caminade/science.txt /home/macleod/unixstuff/science.txt

mv (move and/or rename as well)

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 science.bak to your backup directory.

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

% mv science.bak backups/.

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

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 science.txt

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

Less or more

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

% less science.txt

% more science.txt

Press the [**space-bar**] if you want to see another page, and 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 science.txt

Then type

% head -5 science.txt

tail

The tail command writes the last ten lines of a file to the screen.

Clear the screen and type

% tail science.txt

Simple searching using less

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

% less science.txt

then, still in less, type a forward slash [/] followed by the word to search

/blabla

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 blabla science.txt

As you can see, grep has printed out each line containg the word **science**.

Or has it ????

Try typing

% grep Blabla science.txt

The grep command is case sensitive; it distinguishes between Science and science.

To ignore upper/lower case distinctions, use the -i option, i.e. type

% grep -i blabla science.txt

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

% grep -i 'This is just' science.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 science or Science is

% grep -ivc blabla science.txt

wc (word count)

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

% wc -w science.txt

To find out how many lines the file has, type

% wc -l science.txt

More details:

http://www.ee.surrey.ac.uk/Teaching/Unix/unix2.html

(do the exercices)

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 without specifing 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 specifing a file to read, it reads the standard input (the keyboard), and on receiving the 'end of file' (**^D**), copies it to the standard output (the screen).

In UNIX, we can redirect both the input and the output of commands.

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 {this means press [Ctrl] and [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

Appending to a file

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 outputing the text to the file **biglist**

To read the contents of the new file, type

% cat biglist

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 animals. Press [Return] after each one.

dog
cat
bird

ape
^D (control d to stop)

The output will be

ape
bird
cat
dog

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**

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

The * wildcard

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 pr*

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

Try typing

% ls /home/caminade/Sim/ENSEMBLES/stream2/monthly/pr*

This will list all files in the current directory beginning with **pr....**

The ? wildcard

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

ls
/home/caminade/Sim/ENSEMBLES/stream2/monthly/prlr_1m_1960??_2005??_ENSEMBLES.nc

More details:

http://www.ee.surrey.ac.uk/Teaching/Unix/unix4.html

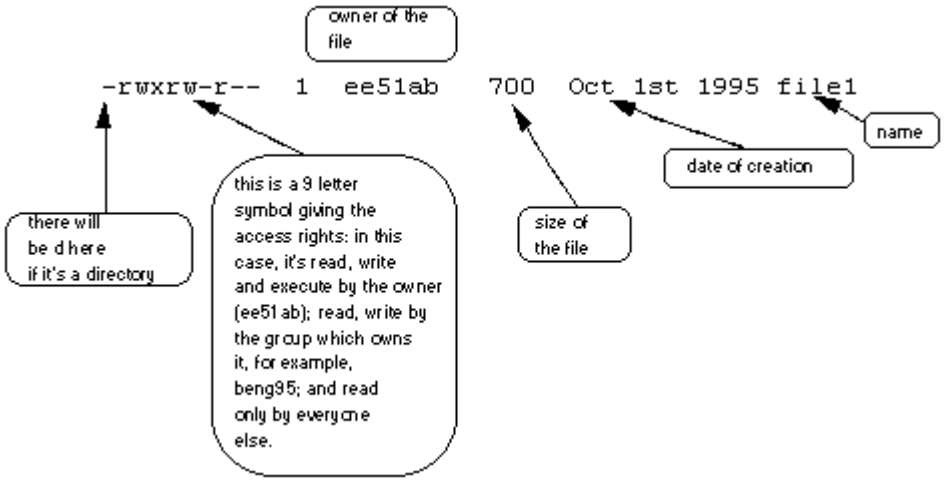
do the exercices...

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 **science.txt** and on the directory **backups**

Use ls -l to check that the permissions have changed.

More details:

Browse the UNIX tutorial:

---

http://www.ee.surrey.ac.uk/Teaching/Unix/

Summary:

Command	Meaning
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
Command	**Meaning**
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
less *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
Command	**Meaning**
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

Command	**Meaning**
*	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

Command	**Meaning**
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

Important to launch an application / soft:

Appli & (the & means you launch the application in the background)

Examples: nedit &

mozilla&

xemacs &

gv &

xterm&

Keep the summary on your desk

Good luck