ASTR 288C - Lab 1

(Based on the online tutorial http://www.ee.surrey.ac.uk/Teaching/Unix/index.html)

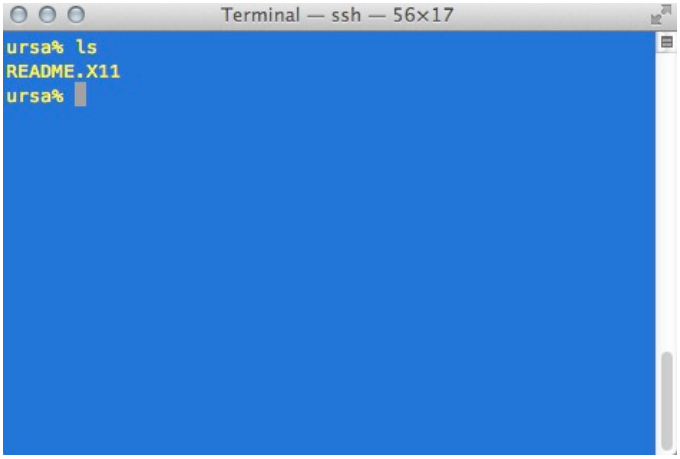
1.1 Listing files and directories

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, for example, jsmith, and it is where your personal files and subdirectories are saved.
To find out what is in your home directory, type:

```
% ls
```

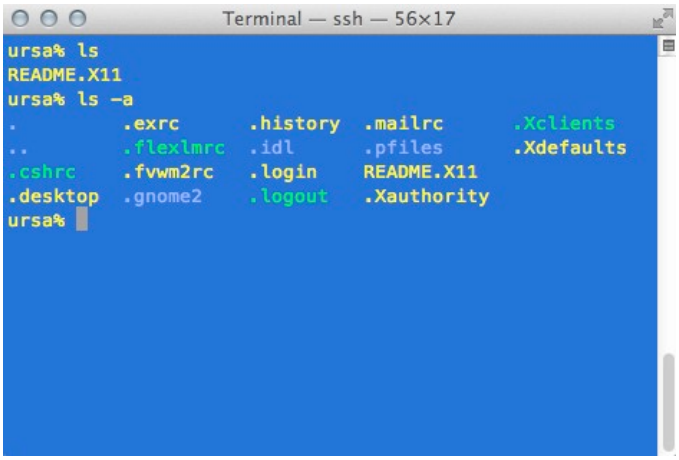
The ls command (lowercase L and lowercase S) lists the contents of your current working directory. There may be no files visible in your home directory, in which case, the UNIX prompt will be returned. Alternatively, there may already be some files inserted by the System Administrator when your account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



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
```

As you can see, ls -a lists files that are normally hidden.



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 directory

cd (change directory)

This command moves your position in the file-system tree from the current working directory to 'directory'. To change to the directory you have just made, type:

```
% cd unixstuff
```

Typing cd with no argument always returns you to your home directory. This is very useful if you are lost in the file system. Try it now.

```
% cd
```

Typing:

```
% cd ..
```

will take you one directory up the hierarchy, in the parent directory of your current working directory. Try it now.

1.3 Path names

pwd (print working directory)

Path names tell you where you are in relation to the whole file-system. For example, to find out the absolute pathname of your current directory, type:

```
% pwd
```

The full pathname will look something like this -

```
/n/ursa/A288C/
```

which means that the sub-directory A288C (the group directory) is located in the ursa sub-directory, which is in the “n” sub-directory, which is in the top-level root directory called " / " . If you now type:

```
% ls unixstuff
```

you will receive an error message like:

```
% ls: cannot access unixstuff: No such file or directory
```

The reason is that **unixstuff** is not in your current working directory. To use a command on a file (or directory) not in the current working directory, you must either change to the correct parent directory, or specify its full pathname. To list the contents of your **unixstuff** you must type:

```
% ls <your_account>/unixstuff
```

or

```
% ls ~/unixstuff
```

Home directories can also be referred to by the tilde ~ character. It can be used to specify paths starting at your home directory, no matter where you currently are in the file system.

1.4 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

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**.

Copy the file *science.txt* in your home directory:

```
% cd
```

```
% cp /n/ursa/A288C/alien/science.txt .
```

Note: Don't forget the dot . at the end. The dot means the current directory, and allows you to copy the file in your working directory without specifying its full pathname.

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. 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.txt to your **unixstuff** directory, and rename it *science.txt*. Type:

```
% mv science.txt unixstuff/science.txt
```

2.3 Removing files and directories

rm (remove), rmdir (remove directory)

To delete (remove) a file, use the **rm** command.
First, copy the *science.txt* file back to the current directory so we can delete it later

```
% cp unixstuff/science.txt .
```

Then type:

```
% rm science.txt
```

this will remove the file *science.txt* from your home directory.

You can use the **rmdir** command to remove a directory. Make sure that the directory is empty first, since UNIX will not let you remove a non-empty directory.

2.4 Displaying the contents of a file on the screen

clear (clear screen)

You may like to clear the terminal window of the previous commands so the output of the following commands can be clearly understood. 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.
Go to your **unixstuff** directory by typing

```
% cd unixstuff
```

Type:

```
% cat science.txt
```

As you can see, the file is longer than the size of the window.

less

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

```
% less science.txt
```

Press the [space-bar] if you want to see another page, and type [q] if you want to quit reading.

head

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

```
% head science.txt
```

To list only the first three lines of the file, type

```
% head -3 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
```

To list only the last three lines of the file, type

```
% tail -3 science.txt
```

2.5 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 for the word 'Astronomy', type:

```
% less science.txt
```

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

```
/Astronomy
```

As you can see, less finds and highlights the keyword. Type [n] to search for the next occurrence of the word.
Note: if you entered *astronomy* instead of *Astronomy*, less returns *Pattern not found*.

grep

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

```
% grep Astronomy science.txt
```

As you can see, grep has printed out each line containing the word *Astronomy*. Also the grep command is case sensitive, and distinguishes between *Astronomy* and *astronomy*.
To ignore upper/lower case distinctions, use the -i option, i.e. type

```
% grep -i astronomy science.txt
```

To search for a phrase or pattern, you must enclose it in single quotes (the apostrophe symbol).

```
% grep -i 'gamma ray' 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.

wc (word count)

To do a word count on science.txt, type:

```
% wc -w science.txt
```

To count the number of lines, type:

```
% wc -l science.txt
```

2.6 Summary

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

3.1 Wildcards

The * wildcard

The character * is called a wildcard, and will match against one or more characters in a file (or directory) name. For example, in your **unixstuff** directory, type:

```
% ls *txt
```

This will list all files in the current directory ending with 'list'. Now try typing:

```
% ls txt*
```

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

The ? wildcard

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

```
% ls sci??ce.txt
```

3.2 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.

```
% cat
```

We have already seen one use of the **cat** command to write the contents of a file to the screen. Now type **cat** without specifying a file to read

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 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.

3.3 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
```

Copy list1 to list2 to save for further use

```
% cp list1 list2
```

3.4 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 fruits, the other contains three fruits.
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 outputting the text to the file **biglist**
To read the contents of the new file, type

```
% cat biglist
```

3.5 Pipes

To see who is on the system with you, type

```
% who
```

To see how many people there are (that is, how many lines listed with the “who” command), type

```
% who | wc
```

The vertical bar | (pipe) means taking the output of the first command as the input of the following command.

3.6 Filename conventions

In naming files, characters with special meanings such as / * & % , should be avoided. Also, avoid using blank 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)
The rules and conventions for naming files apply also to directories.

Good Filename	Bad Filename
project.txt	project
my_program.pro	my program.pro
mark_lisa.jpg	mark & lisa.jpg

File names conventionally end with a dot followed by a group of letters indicating the contents of the file (extension). For example, all files consisting of Python code may be named with the ending .pro, for example, *prog1.py* . Then in order to list all files containing Python code in your home directory, you need only type *ls *.py* in that directory.

3.7 Getting Help

There are online 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 behavior of the command. Type **man** command to read the manual page for a particular command. For example, to find out more about the **grep** command, type:

```
% man grep
```

4.1 Text Editor

There are several text editors available in the terminal environment. Here we introduce you two commonly used text editors.

4.2 vim

Open the text editor by typing

```
% vi test.txt
```

Press the letter **i** to go into the “insert” mode, and type

```
% This is a test.
```

in the file.

When you finish editing, press the escape key (**esc**) to leave the insert mode, and type

```
% :w
```

to save the file, and

```
% :q
```

to quit the file.

Alternatively, you can also just type

```
% :wq
```

to save and quit the file.

4.3 emacs

Open the text editor by typing

```
% emacs -nw test.txt
```

Start editing the file, type

```
% This is a test.
```

in the file.

When you finish editing, type

```
^x^s
```

to save the file, and

```
^x^c
```

to quit the file.

5.1 Shell Script

Shell script is a powerful tool to combine your commands into one single script. When you execute the script, computer will go through the commands in the script. This is very useful if you need to automate your commands.

Here we will create a simple shell script to go to the **unixstuff** directory and print out the current location.

First, go to your home directory by typing

```
% cd ~
```

Use either vi or emacs, open a file called test_shell.sh in your home directory, and type the following lines in the file test_shell.sh.

```
#!/bin/bash

cd unixstuff
pwd

echo "This is a shell script."
```

The first line "#! /bin/bash" indicates which shell you are using. The current line means that we are using the "bash shell", which is the recommended shell to use.

To make the shell script executable, type

```
% chmod 777 test_shell.sh
```

If you type "ls", you should see the name of "test_shell.sh" is now green, meaning that this is an executable file. The number "777" after chmod indicates granting write, read, and execute permissions to everyone.

Now the file is executable, we can execute/run the script by typing

```
% ./test_shell.sh
```

You should see the following output
""
/n/ursa/A288C/<*your_account*>/lab01
This is a shell script.
""

Lab 1 Worksheet

You're required to hand this worksheet in at the end of the lab.

Name: _____ User Name: _____

a) Create a directory **lab01** inside **unixstuff**, and write the full pathname of the directory **lab01**.

b) Copy *science.txt* to the directory **lab01** created in Exercise 1, write down the command you use.

c) Write down the command that lists the first 3 lines in *science.txt*

d) Write down the command that lists the last 5 lines in *science.txt*

e) How many words are there in the file *science.txt*? How many lines?

f) Write down the command to count the number of lines that contain term "physics" in *science.txt?*

g) In your directory **unixstuff**, what is the command to list all the files with names that include "txt"?

h) What does the command **diff** do?

Remember to fill in the doodle poll for alternative office hour:
http://doodle.com/poll/mrfqbp259sraqsyq
(Multiple choices)