1 - The Command Line

1. Learning Outcomes

After completing the exercises in this lab you should be able to use a Linux shell (command line interpreter) to compose and execute some simple commands in the Linux environment

2. Organisation

Please attempt this lab individually as you will need the skills acquired in this lab in order to complete later labs in this module.

3. Grading

This worksheet is worth up to 10% of your overall module grade. You must attend and sign in at the labs in order to obtain the credit for the associated worksheets. You may work on this worksheet during lab 1 and lab 2 with instructor assistance. You must also demonstrate your submission in order to receive credit - see below.

4. Submission

The deadline for submission is Wednesday Sep 28, 2016 @23:59 through Webcourses.

5. Demonstration

You will give a brief demonstration of your submission to the lab instructor in lab 3.

6. Requirements

For this lab you will need to sign up for the following services if you've not already done so:

- Access to a Linux/Mac Terminal environment on your laptop or in the cloud or a free account with a hosted environment such as Nitrous.IO (https://nitrous.io) or TutorialsPoint (http://tutorialspoint.com) or Cloud 9 (https://c9.io) or similar
- An educational or paid account with Github (https://github.com) **or** a free account with Bitbucket (https://bitbucket.org) which you can use securely store your code

7. Resources

You are free to research whatever you need to solve the problems in this lab. Some recommended resources include:

- Online course on the *Command Line* here

8. Quick Primer

a. The Linux Shell (bash)

When you log into Linux, you are immediately placed into a command line interpreter called a shell. There are many shell programs available for Linux but the default shell in Ubuntu Linux is called bash. In a shell you may enter commands where you see the command prompt, like so:

```
user:~$
```

The prompt above conveys certain information about the server and environment as follows:

`user`	The name of the user who is logged in
`@`	Means "on" or "at"
`ubuntu`	The system's host name
`:`	Separator (could be anything)
`~`	The current directory (or folder if you prefer). ~ is an alias for "/home/user"
`$`	Conventionally, the character denoting a command prompt

Run the following commands and the command prompt in the following sequence. When you do this there will be certain output displayed after each, not shown here.

NOTE 1: When typing commands into a shell, you must type them exactly as shown. Spaces are significant. Commands are case-sensitive. All punctuation is significant.

NOTE 2: Do not type the "`user:~$`" in any of the following exercises. This is shown only as a guide here.

```
user:~$ echo "hello from Ubuntu Linux"

user:~$ pwd

user:~$ mkdir lab1

user:~$ ls

user:~$ cd lab1

user:/lab1$ pwd

user:/lab1~$ cd

user:~$ history
```

What do you think the previous commands did? The Linux help command is called *man* (manual).

```
user:~$ man man

user:~$ man echo

user:~$ man mkdir

user:~$ man ls
```

Note the format of a man page. Also, note the conventions used for command line arguments and the use of the dash (-) to prefix command line flags.

b. Command line navigation and editing

In the like event that you make a mistake when typing in commands you need to know how best to correct them. Typing everything in again is an option but definitely not the most efficient. Commit the following to memory to save you time when on the command line

Key sequence	Full name	What it does
←	Left Arrow	Move left one character if not already fully left
→	Right Arrow	Move right one character if not already fully right
↑	Up Arrow	Recall the previous command typed
↓	Down Arrow	Go to the next command typed
Ctrl-a	Control-a	Go to the start of the command line
Ctrl-e	Control-e	Go to the end of the command line
Ctrl-r	Control-r	Start a search for a previously entered command. Followed by typing one or more characters from that command and pressing ENTER when selected
Ctrl-l	Control-l	Form-feed. Clear the screen
Tab	Filename completion	Pressing the TAB key when typing a filename may complete the typing of it for you if bash can guess what you intent

c. The Linux host characteristics

Now, let's explore the server machine itself. Run the following commands. When you do this there will be certain output displayed after each, not shown here

```
user:~$ cd

user:~$ hostname

user:~$ uname -a

user:~$ cat /proc/cpuinfo

user:~$ cat /proc/meminfo

user:~$ df -H

user:~$ ps -ef | more
```

The last command is an example of a pipeline. There are two commands being issued at the same time. The first one's output is "piped" to a screen pager which allows you to see all of the output of a command which would otherwise exceed the number of visible lines on your screen.

What did each of the previous commands do?

Now, let's look a little more at the user environment.

```
user:~$ echo $PATH
```

The variable PATH is called an environment variable. It's value is available to you in your shell commands and to all commands (processes) executed in your shell. PATH contains a colon-separated list of system directories that are searched, in the order given, for commands you type in. That brings up an important learning point for this lab. All commands you type are actually implemented as files somewhere in the Linux file system. To find out where a file for a particular command is location you can use the *which* command

```
user:~$ which echo

user:~$ which ls
```

---

9. Problem Sets

To prepare for the problem sets, you will need to download a couple of files you will be working with as follows:

```
curl -Ls https://www.dropbox.com/s/9i9gjfh0sxm4gf1/labfiles.tar.gz?dl=0
| tar zxf -
```

If successful, you should see a new folder called lab files in which there are two additional files. Using those files, provide command-line fragments to answer the questions below. For reference, you should look at the following utilities to help you with this task:

`ls, cat, grep, awk, find, wc, file, ps, echo, sort, head, tail`

1	How many words are in the “words” file?	
2	How many words in the the “words” file start with the letter ‘a’?	
3	How many words in the the “words” file end with the letter ‘z’	
4	How do you create a copy of the “words” file sorted in reverse order?	
5	How many words do not contain vowels?	
6	What is the length of the shortest word?	
7	What is the length of the longest word?	
8	How many words are there the “words” file with only words having 4 and 5 letters?	
9	How many bytes are output in solution 8?	
10	How many processes are there in “processes.txt”?	
11	How many processes, listed in “processes.txt”, do not belong to the “root” user?	
12	From the “processes.txt”, what is the process ID of the ssh daemon?	
13	From the “processes.txt”, what is the parent process name of the ssh daemon?	
14	From the “processes.txt”, how many interactive login sessions are there on this machine?	
15	From the “processes.txt”, what is the sum total of the process ids when added all together?	

Providing answers for the problems above is all you need to submit. You can include your shell command fragments for each in a single file and submit that on Webcourses or you can create 15 individual files and submit a zipped version of this on Webcourses.