CSE 3202
Operating Systems
Lab 1: Basic UNIX Commands

Overview

UNIX administration is not normally achieved using the graphical user interface (GUI). The reasons for this are several folds.

- Many systems have a 'stripped-down' operating system with no GUI.
- It is common for systems to be held on "dark sites" (sites where no-one goes in the normal pattern of things) and the systems are managed remotely using telnet, ssh etc.
- GUI's limit the power of UNIX, you can do a lot more on a command line than you can do through a GUI. GUI's are getting better but still the average UNIX expert finds the GUI limiting for all but a few tasks.

The user of the machine must know how to interact with it by using UNIX command that are typed in next to the command prompt.

The aim of this workshop session is to get you familiar with basic UNIX commands so that you are able to interact with a UNIX machine from a simple command line prompt instead of by using a friendly and colorful GUI.

Logging in and out of the system.

To use a UNIX machine a user needs a ***user account***. This comprises of:

- A user account name.
- A user account password.
- A directory that the user is automatically 'placed in' when he/she logs onto the system. This directory, referred to as the user's home directory, is where the user keeps his/her work.
- A number of administration files placed in the home directory that configure the user's working environment.
- Information entries for the user in certain system files (e.g. /etc/passwd)

Each UNIX system has a "superuser account" called **root**. People charged with managing UNIX systems are usually referred to as UNIX systems administrators. One obvious UNIX system administration task is to create a user account for each user of the system.

System directory hierarchy

The directory structure of a UNIX system can be viewed as a tree. At the base of the tree is 'the root directory' (which is the home directory of the user root) this directory will contain a number files and other directories, each of these directories will contain additional files and other directories ..... and on it goes ..... it is a hierarchical directory structure. Don't get to lost in the tree analogy, the root directory is generally considered to be at the top of the hierarchy and all other directories are siblings of it.

The following example shows the top level directory structure for Ubuntu-

Top level	**Second level**	**Purpose**
/		The root directory
	bin	Contains all of the code that constitutes the operating system
	boot	Contains the boot sector
	dev	Contains '**dev**ice files'. These are special files that the system uses to interact with hardware devices.
	etc	The etc directory contains files that are changed to configure the operating system
	home	Users' home directories go under here. Each home directory is named with the same name as the user account name. E.g. /home/os
	initrd	
	lib	
	lost+found	If directory tree gets corrupted and UNIX finds any lost files, it sticks them in here so the administrator can decide what to do with them.
	misc	
	mnt	
	opt	
	proc	
	root	
	sbin	
	tftpboot	
	tmp	Contains temporary files - all programs written for UNIX should use this directory to store temporary files. It sort of keeps things tidy, many UNIX variants will delete all files in tmp when rebooted
	usr	

	var	Contains files and directories that change size quite a lot. For example, print queues and e-mails waiting to be sent are held in var

Notice that the top level directory is / **, that** is root.

Issuing commands at the command prompt

The 'shell' is an interface between the user and the UNIX operating system. Commands are typing in at the command prompt and the shell interprets them and sends instructions to the operating system to perform the appropriate actions.

There are many different types of shell, such as, Bourne, Korn, C, Bash. Ubuntu by default uses the Bash shell which provides a convenient means of repeating previously entered commands...

- Use the up and down arrows to view and select a previously entered command.
- Use double pling (**!!**) to repeat the last command entered.
- Enter the **history** command to see a full list of previously entered commands. Choose a previously entered command from the history list by using its displayed index number prefixed with a pling character (**!**)

UNIX commands that give information

	Command	Description
1	hostname	Displays the name of the machine. Each machine on the network is given a unique name by the system administrator.
2	who	Displays who is logged onto the machine.
3	date	Displays the current date and time.
4	cal	Displays a calendar.
5	env	Displays a list of environment variables and their values. We will meet these again later
6	uptime	Displays the time duration that the machine has been turned on.
7	uname	Displays information about the computer system.
8	man uname	The man command gives information on the command that follows it.

Exercise 1 - Using UNIX commands that give information

Try out the following commands and make observation notes as you go along ...

	Command	Notes
1	**Hostname**	
2	**who**	
3	**date**	
4	**cal** **cal 2015** **cal 11 2015**	
5	**Env**	Scroll up and down the screen and make a note of the values of the following Environment variables: HOSTNAME = SHELL= TERM= USER= PATH=
6	**Uptime**	What pieces of information are presented by this command? You may need to find out more about the uptime command using the man pages
7	**uname –a**	What pieces of information does this command display.

UNIX commands relating to directories

	Command	Description
1	Pwd	**P**rint (on the screen) the pathname of the **w**orking **d**irectory (i.e. the directory you are presently in).
2	ls	**L**i**s**t contents of directories.
3	cd	**C**hange working **d**irectory.
4	mkdir	**M**a**k**e a **dir**ectory.
5	rmdir	**Rem**ove **d**irectory.
6	mv	**M**o**v**e a directory (also used to rename a directory or file).

Exercise 2 - Using UNIX commands relating to directories.

	Command	Notes
1	**pwd**	Write down the pathname of your current directory. (The directory you are automatically placed in after logging on is referred to as your '**home**' directory.)

2	**ls**	List the contents of your current directory (should be empty at this stage).
3	**mkdir testdir**	Make a subdirectory named 'testdir'.
4	**cd testdir**	Change directory to the new directory.
5	**pwd**	Write down the pathname of your current directory.
6	**ls**	List the contents of your current working directory (should be empty at this stage).
7	**cd**	The change directory command, cd, on its own will move you back to your 'home directory'. Another way to return to your home directory is to use the command **cd ~**
8	**pwd**	Write down the pathname of your current directory (it should be your home directory)
9	**ls**	List the contents of your current directory (this should now show the name of the subdirectory named testdir)
10	**cd /** **pwd** **ls** **ls -p**	Change to the top most level directory (i.e. the root directory) Confirm where you are using the pwd command. List the contents of the current directory List the contents of the current directory and put a forward slash at the end of any entry that is a directory.
11	(Task)	Check that the names of the directories under the root directory agree with those listed in the theory page. If not, make a note of any differences.
12	**cd** **pwd** **ls**	Change to your home directory Confirm where you are using the pwd command. List the contents of the current directory
13	**cd testdir**	Change to the directory named testdir
14	**cd ..** **pwd** **ls**	Change to the directory one level up (as specified by the use of two dots). Confirm where you are using the pwd command. List the contents of the current directory
15	**pwd** **ls** **mkdir testdir/play** **ls** **cd testdir/play** **pwd** **ls** **cd ..** **pwd** **ls** **mv play play2** **ls**	Check you are in your home directory. List the contents of the current directory. Create a directory named **play** underneath the one named testdir. List the contents of the current directory. Change directory into the one just created. Confirm where you are. List the contents of the current directory. Change up one level (ie. back up to the testdir directory) Confirm where you are. List the contents of the current directory. Rename the directory named **play** to **play2.** List the contents of the current directory to check that the renaming has been done.
16	Note:	The command **cd testdir/play** uses the name of the directory relative to the current directory. To be more precise you could have specified the command as: **cd ./testdir/play** where the single dot means the current directory.
17	**cd** **pwd** **ls**	Move back up to your home directory. Confirm where you are. List the contents of the current working directory.
18	**rmdir testdir**	Remove directory. You will find that this command will not work since the directory contains subdirectories. Overcome this by first removing the lower level directories. Later you will be shown a potentially dangerous command that will remove any directory and anything underneath it.
19	**pwd** **ls**	Check where you are. List the contents of the current directory.
20	**cd** **mkdir testdir** **mkdir testdir/play**	Recreate the directories again.

UNIX commands relating to files

	Command	**Description**
1	cat	A way of displaying the contents of a text file. (Command comes from the word 'con**cat**enate').
2	more	A way of displaying the contents of a text file. (This is a 'file perusal' filter and displays files a page at a time).
3	cp	**Co**py file
4	mv	**Mo**ve a file (also used to rename a file).
5	rm	**Rem**ove a file. (Can also be used with the appropriate option to removed a directory and all its contents)

6	touch	This can be used to quickly create an empty file (or to date stamp an existing file).
7	file	To determine the file type.
8	head	Displays the first few lines of a file.
9	tail	Displays the last few lines of a file.
10	wc	Used to display how many lines, words and characters are held in a file.
11	find	Find the location of a file.
12	du	Used to display how big a file is (disk usage). The figure given is the number of memory blocks. Each block on our system is 1024 bytes.
13	df	Used to display the amount of disk space that is free (unused).

Commands, arguments and options.
Shell commands lines entered at the shell command prompt consist of one or more words separated by spaces.
The first word is the ***command*** and the remaining words (if any) are the ***command arguments*** (sometimes known as ***command parameters***).
The command arguments are the 'things' that the command acts upon.
An option is a special kind of argument (it is usually a letter prefixed with a minus sign) that tells the command how it should behave or what it has to do.
The following table shows how commands, arguments and options work. The example assumes a user named os who has a home directory /home/os and a subdirectory under that named /home/os/cwork.

Command	Details
cd	Change to the home directory (/home/os). The cd command on its own will always return a user to his/her home directory.
ls	List the contents of the directory
ls cwork	List the contents of the directory named cwork
ls -t cwork	List the contents of the directory named cwork but show the information sorted by modification time before sorting alphabetically.

Examples of command options...

	Command	Description
1	ls mywork	List the contents of a directory named mywork
2	ls **-l** mywork	List directory contents in **l**ong format (this displays additional useful information)
3	ls **–p** mywork	List directory and **p**ut a slash (/) after each entry that is a directory name.
4	ls **–s** mywork	List directory and show the **s**ize in blocks.
5	ls **–t** mywork	List directory and sort by the **t**ime modified before sorting alphabetically.
6	ls **–u** mywork	List directory and sort by the time of the last access instead of last modification. (i.e. sort by the last time **u**sed)
7	ls **–x** mywork	List directory and display the entries sorted **x**across the screen rather than down it.
8	ls **–R** mywork	List directory and **r**ecursively list the contents of any subdirectory encountered.

So... although there are only a few basic commands that you need to know to get started, each command can be used with many options that control how the command behaves. It is impossible to remember all the commands and their options so a user will frequently need to refer to the on-line help pages referred to as the 'man pages' (i.e. the **man**uals)

Using the man pages
The ***man pages*** are simple to use. For example, if you want to know all about the **ls** command simply enter the command:

man ls

Everything you ever wanted to know (and don't want to know) about the ls command is then displayed on the screen When viewing a man page remember to:

- Use the return key to scroll down through the information one line at a time
- Use the space bar to scroll down through the information one page at a time
- Use **q** to quit the man page and return to the dollar prompt.

Unix system administrators refer to man pages all the time. There are too many Unix commands and options for anyone to remember them. Take your time when reading man pages as they often contain the information you are often looking for.

Exercise 3 - Using UNIX commands relating to files.
Before starting:

- Move into the directory named **testdir**.
- List the contents of that directory.
- Move down into the directory named **play**.

- List the contents of that directory (it should be empty at this stage).

	Instruction and/or Unix command	Observations
1	**man ls > lsmanpage**	Create a file containing the contents of the man page for the **ls** command. Note the use of the command output redirection symbol (>). This sends the ouput of the command 'man ls' to the file instead of to the screen.
2	**Ls**	List the contents of the directory.
3	**file lsmanpage**	Use the file command to find out what type of file **lsmanpage** is.
4	**cat lsmanpage**	Display the contents of the file name **lsmanpage**. It will fly by so quickly that you will probably only see the last few lines of it.
5	**more lsmanpage**	Use the **more** command to see the contents of a file a page at a time. When viewing a file using this command you will need to use the return key to scroll down the file by one line at a time, the spacebar to scroll a page at a time and use q to finish viewing.
6	**cp lsmanpage lsmanpage2** **ls**	Make a copy of the file List the contents of the directory (it should contain **lsmanpage** and **lsmanpage2**).
7	**mv lsmanpage2 lsmanpage3** **ls**	Rename the file. List the contents of the directory (it should contain **lsmanpage** and **lsmanpage3**)
8	**head lsmanpage** **tail lsmanpage**	Display the first 10 lines of the file. Dislay the last 10 lines of the file.
9	**wc lsmanpage**	Make a note of the 3 numbers that are displayed and write down what they represent.
10	**du lsmanpage**	Write down the number of disk blocks that this file uses. How many bytes does this represent?
12	**rm lsmanpage3** **ls**	Remove a file. List the contents of the directory (it should now only contain **lsmanpage**).
13	**cat > mydetails** **Surname: Bloggs** **Firstname: John** **Age: 21** ***<now press Ctrl/Z to terminate text entry>***	The cat command can be used with the redirection operator (>) to create a new text file 'on-the-fly'. Enter the command **cat >mydetails** then press the return key, then continue to enter text on the next blank line and to terminate text entry press the Ctrl and Z keys at the same time.
14	**cat mydetails**	Now use the cat command without any redirection to display the contents of the file.
15	**cat -n mydetails**	Display the text file and show line **n**umbers.
16	**cat -n lsmanpage** **cat -n lsmanpage | more**	Display the original large text file and show line **n**umbers. Again the file goes by too quickly. Try the **cat -n** command again but send its output to another command, ie. the **more** command. This is referred to as piping (think of a pipe between 2 commands). The vertical bar character (to the left of the z key) is the pipe symbol.
17	**cat -n lsmanpage > num-lsmanpage** **ls** **more num-lsmanpage**	Redirect the output of the **cat -n lsmanpage** command to another file named **num-lsmanpage**. List the contents of the directory. Display the contents of the new file using the **more** command.
18	**cat > mycar** **Car make: Ford** **Model: Escort** **Colour: Silver** ***<now press Ctrl/Z to terminate text entry>***	Create another file using the on-the-fly technique.
19	**cat mycar >> mycar2** **cat mycar >> mycar2**	Use the >> operator to **append** the contents of one file to another file. Examine the contents of mycar and mycar2.
20	**touch quickone** **ls** **du quickone** **cat quickone**	Touch can be used to quickly create an empty text file.

Courtesy: Rushdi Shams Sir