

Operating Systems Exercises week 3.

Starting the Linux virtual machine.

Go to Start then VMware then VMware Player. When the VMware Player window comes up, open the Linux 2.6.x virtual machine. This will load up a virtual machine with the Linux operating system.

If it is the first time you run VMPlayer with this virtual machine, a dialog box will appear asking whether or not you moved or copied the virtual machine. Copied is selected by default and is the correct selection. So if you get this dialog box, click on OK.

The operating system will then load. You will see a red screen come up briefly, and then various messages indicating various components being started and checks made. Eventually you will get the Red Hat login screen with a box for entering your user name. To set keyboard/mouse input so that it goes to the virtual machine remember you need to use CTRL+G [CTRL+ALT restores input back to host machine]. The user name to input is

user1

Remember to press the enter key. You will then be prompted for the password. The password is

vmware

Again press the enter key. You will get a blank screen for a short while, but eventually you will get the GUI fully loaded up.

Please note that if you maximise the virtual machine window that the VMPlayer top bar remains in the center at the top of the screen and obscures some of the drop down menus. However, when it is maximised there is an icon of a drawing pin at the left hand side of the top bar. You can click on this to either pin the bar in place so it does not move or to let it slide out of the way. If you get rid of it, you can get it back at any time you need it by simply resting the mouse pointer on the area of the bar that is just showing.

The GUI is conventional and has several menus across the top. Have a look at them. Applications contains a variety of general applications, general utilities, some games, office like applications, etc. Places is a menu which shows various locations and gives conventional folder views of the home directory, the computer (like My Computer on Windows), etc. System has various administrative type menus.

You will notice on the System menu that the GUI we are using is GNOME.

Logging out.

Most important for you to know at the moment is how to log out and close down the system when it is time. You will need to do this only later, but the following is the procedure when you do need to log off.

Go to the System menu. Towards the bottom you have Log Out user1. Click on this. A log out dialog box will appear asking you to confirm whether or not you want to logout. Click OK. The login screen will then re-appear. At the bottom of that screen you will find Shut Down. Click on that and Linux will shut down and the virtual machine close down. You will see a large number of messages which show the various Linux system components being stopped.

Linux exercises.

We are going to do a number of simple exercises using the shell. If you are familiar with Unix or Linux then the exercises will be very straightforward, but if you are unfamiliar with it or have not used it for some time then it will act as an introduction or a revision to remind you of what you may have covered before.

Note - whenever you see '<ret>', it means press the return/enter key after you have typed the command in.

Reminder

Remember that when you power down the virtual machine any changes to the hard disk will be discarded. So if you want to save your work then you need to save your files onto a floppy or USB pen drive.

If you want your USB pen drive to be connected to the virtual machine all you need to do is put it in the USB port as normal and then after it has been recognised by the host operating system, click on the device on the WMPPlayer Device menu and it will be disconnected from the host machine and made available (connected) to the virtual machine. The floppy drive should already be connected to the virtual machine.

Once connected your USB drive will appear as an icon on the desktop as well as within the Computer folder. Clicking on the USB drive icon will bring up a conventional window listing the contents of the USB. You can move or copy files from the user home directory to your USB drive by simply dragging or right clicking and selecting copy and then pasting as usual.

1) Before we proceed you will need a small text file. So we will create one very simply. Go to Applications menu, then Accessories and click on text editor. This will bring up a very simple text editor. Type in your name, address and date of birth. Save the file to the default location (user1 home directory) giving it a suitable name of your choice.

2) Go to the Applications menu, then Accessories and then terminal. This will bring up a window with the command line interface (shell) with the prompt:

```
[user1@localhost ~]$
```

You can now type commands at the prompt as with any command line interpreter.

By default, you will have the *bash* shell.

3) The manual pages

You can use the manual pages to find out about commands. If you type *man command*, e.g. *man cp*, you will get information about the command that you have specified (the *cp* [i.e. copy] command in this instance).

To move forward through the manual entry, you can use the spacebar. Each time you press the spacebar you will advance through the entry a page at a time. Alternatively, the <ret> key will give you a line at a time. If you want to quit before you reach the end of the file that you are viewing, press the q key.

Note – for more detail than that contained in the man pages use *Info command* e.g. *info cp*.

Don't waste your time reading the whole of manual page entries. Just pick out the information that you really need.

4) Some important commands to investigate

Try out the following commands, looking at the entries in the manual pages when you need to, e.g. by typing

```
man ls
```

NB The 'l' in *ls* is a lower-case ell, not the number one.

(a) *ls* - list the files in a directory

List the files in your home directory with *ls* by typing

```
ls
```

Try out *ls -l*, *ls -a* and *ls -s* in succession and see what difference the *-l*, *-a* and *-s* flags make.

Try out *ls -las* (three flags combined). Did it do what you expected?

(b) *pwd* - print working directory

Try out this command. What does it tell you?

(c) *mkdir* - create a directory

Make a directory, using *mkdir newdirectoryname*

(Choose whatever name you like for your new directory.)

(d) *cp* - copy a file

Copy the text file that you created in 1) above into the directory that you have just created using

```
cp myfile.txt newdirectoryname/file2
```

(e) *cd* - change directory

Get into the directory that you have created by typing

`cd directoryname`

(f) `rm` - remove (delete) a file

NB `rm` is an especially dangerous command and you must think before using it! In particular, it is inadvisable to use it with wildcards such as `*`. Never type `rm *.*`, or at least not before thinking about it very carefully first, because it will delete all the files in your directory and you will not be able to get any of them back.

Remove (delete) the file that you copied into your new directory.

Get back into your home directory by typing

`cd`

on its own.

(g) `mv` - move (rename a file)

Rename a file by typing

`mv file1 file2`

NB You will have to substitute the name of a file that you have got in your directory for 'file1'.

(h) `rmdir` - remove a directory

Remove the directory that you created earlier by typing

`rmdir newdirectoryname`

NB You will have to substitute the name of a directory that you have got for 'newdirectoryname'.

5) Redirection of the standard input and output:

What does `ls -l > file.txt` do? Enter this command and look at what is in `file.txt` by using the `cat` command (see the next item).

6) The `cat` (concatenate) command

To view the contents of a text file, you can use `cat filename`, e.g. `cat myfile.txt`

Try out the `cat` command on a suitable file, e.g. `myfile.txt`.

A quick way of creating a short file without using an editor is the following:

Type: `cat > garbage.txt`

Type in the text. Type `ctrl/d` on a new line to finish (i.e. when you've finished entering the text, press the `<ret>` key, followed by `ctrl/d`).

Try creating a brief text file in this manner. Keep it to experiment upon later.

What does `cat file1.txt file2.txt` do?

NB You will have to substitute the names of files that you have got in your directory for 'file1' and 'file2'.

What command could be used to join two text files, `texta.txt` and `textb.txt`, together to produce a third file, `textc.txt`? Try it out.

NB You will have to substitute the names of files that you have got in your directory for 'texta.txt' and 'textb.txt'.

A couple of snags with `cat` are:

(i) If you try to `cat` a binary file, you may make your terminal do strange things. This is because a binary file may contain numbers which the terminal will interpret as control characters.

(ii) Often, information that you want to look at will scroll past at too fast a rate for you to be able to read it.

7) The more command

A good way of dealing with problem (ii) above is to use the more command. The utility called 'more' presents the information to you a screenful at a time. man uses more to present the information on the screen, so the commands that work with man (e.g. <spacebar>, q - See above) will of course work with more too.

Try out the more command, e.g. more myfile.txt

8) Shell Scripts

When you type commands at the Linux prompt, what you type is interpreted by the command interpreter - the shell. Several shells are in common use. The shell that you are using is the bash shell.

In many operating systems, it is possible to put several commands together in a text file and run them as a script. For example, Windows batch files (called file.bat) can be written and then run from the command prompt. In UNIX/Linux too, we can string together several commands in a file (a shell script), make it executable and run it.

9) File protection under Unix/Linux

In order to investigate Unix/Linux file protection, you will need to be familiar with the following commands:

cd - change working directory;

pwd - print working directory;

mkdir - make a new directory;

ls - list the contents of a directory (files and directories);

cat - concatenate;

chmod - change access permissions on a file or directory.

Use man to find out about these commands, if and when you need to. All the aforementioned commands, with the exception of chmod, should be familiar to you if you have been going through this worksheet.

10) Setting up a working area

Use mkdir to make a new directory, cd to move into it and ls to ensure that it is empty.

Use cat > filename to create a file in that directory and type some information into it. When you have typed all you want to put in the file, press control-d on a line on its own to close the file.

Now use ls -alg to display the new file in your directory. Use man to find out what the -g flag does.

Then re-examine the directory with ls and try to interpret the information ls gives you.

11) Access permissions

One of the things `ls -alg` does is to give you protection information. It does this by specifying read (r), write (w) and execute (x) permissions for each file or directory. It shows these for three classes of protection group: the owner (you), your group and anyone else who can log on to the system.

Complete access to a file by anyone on the system will thus appear as:
`rw-rw-rw-`

This is composed of the following elements:

<code>rw-</code>	<code>Rwx</code>	<code>rw-</code>
owner's permissions	Group's permissions	world

The most likely access permission for when you created your file is `rw-r--r--`, i.e. read/write by you yourself, read by other members of your group, read by the rest of the world. The file is plain text, so execute permission on such a file makes little sense.

12) Changing permissions

We will now use `chmod` to change access to the file you have created. In order to do this you need to know the following:

Each `rw-` triplet can be represented as an octal number. 1 for execute, 2 for write and 4 for read,

i.e. `--x` = 1, `-w-` = 2 and `r--` = 4. Thus read and write `rw-` = 4 + 2 = 6, `rw-` = 4 + 2 + 1 = 7.

To use `chmod`, all you have to do is type: `chmod XYZ <filename>`, where X is a number 0-7 for user permissions, where Y is a number 0-7 for group permissions and where Z is a number 0-7 for world permissions.

E.g. `chmod 666 myfile` will change the file `myfile` to have permissions `rw-rw-rw-`.

Use `chmod` to change the file `myfile` to have permissions `r--r--r--`. Check that you have been successful using `ls -alg`.

Now try to use `cat > filename` to change the file. What happens?

Use `cat filename` to display the contents of the file. What happens?

Repeat the above experiments for the following protection values on your file:

`-----rW-`, `---rW----`, `r--r--rW-`, `r--r--r--`.

What does this tell you about Unix/Linux protection?

(13) The sort command

Have a look at the man entry for the `sort` command. When you have read and understood what `sort` does, try to work out what the `-r` and `-n` flags do.

Having constructed a short test file, try out `sort < testfile` on its own.

NB Here is an example of a suitable test file (one word or character per line):

```
five
letter
try
apple
toast
banana
```

6

2

twenty

yacht

sun

(newline + ctrl/d to finish)

Now try out sort with the -r flag on your test file.

Try out sort with the -n flag.

Try out sort with both the -r and -n flags together.

14) Pipes

The Unix/Linux allows you effectively to connect two commands or utilities. A pipe is effectively a one directional channel of information and is represented on the command line by the character |, which is placed between the two commands. For example, the command 'wc' word count command can be used to count the number of lines, words or characters in a file or from the standard input (keyboard).

We can use a pipe connect the standard output of some command such as the ls command to the standard input of the wc command.

Try out the wc command on its own with some files. What do the 3 numbers mean? Look it up in the man pages.

We already know what the ls command does.

Now try doing:

```
ls | wc
```

You may use one or more pipes in your solution to the shell script exercise, if you wish, though it is not essential. Instead of using a pipe you could send the output from one program into a temporary file before you use it as input to another program. For example, to count all the lines/words/characters in a given directory you could use ls to produce the directory listing but redirect it to a temporary file like:

```
ls -l /usr/bin > temp
```

followed by:

```
wc < temp
```

15) Run this and try it again this time using a pipe.

Before we write a shell script we need to set the PATH environment variable so that Linux will look for commands in the current working directory as well as in the standard locations. So you need to type at the command prompt,

```
PATH=$PATH:./
```

16) Writing a short Shell script

Use man to find out what the head command does.

Using an editor (can be the simple editor you used in 1), write a short (two or four line) shell script which applies sort and head to a test file and will give a sorted list of the first five lines in the file.

It is a good idea to begin your script with a comment that explains what your script does. Comment lines begin with a hash (#). (You might have to

experiment with the keyboard to find out which key produces the # on screen. For example, it might be the '£' key if you have got a British keyboard.) Remember that you will have to make your script executable, using the chmod command (come out of the editor first). Run your script by typing its name at the prompt. If it doesn't work, go back into the editor and modify it until it does.

17) The tr filter

The tr filter maps one character to another. Look at the manual entry for it. An example of its use is the following:

```
cat source
```

```
MARY HAD A LITTLE LAMB.  
ITS FLEECE WAS WHITE AS SNOW.
```

```
tr '[A-Z]' '[a-z]' < source > output <ret>  
cat output <ret>
```

```
mary had a little lamb.  
its fleece was white as snow.
```

Try the above out by creating the file *source* (consisting of capital letters only) using an editor. Then type at the system prompt the lines shown above (those starting with 'tr' and 'cat').

18) The uniq command

The uniq command reads the input file, comparing adjacent lines. Investigate the manual entry for uniq. There is a flag which will precede each output line with a count of the number of times each line appears in the file. Try out uniq with the flag referred to above on a suitable file. NB A suitable file would contain something like the following (one word per line)

```
letter  
letter  
three  
three  
time  
sun  
star  
star  
star  
star  
twenty
```

18) Now logout and close down the virtual machine.

Tutorial questions.

- 1) MS-DOS provided no means of concurrent processing. Discuss the major complications that concurrent processing adds to an operating system.
- 2) Describe the differences among short-term, medium-term, and long-term scheduling.
- 3) A DECSYSTEM-20 computer has multiple register sets. Describe the actions of a context switch if the new context is already loaded into one of the register sets. What else must happen if the new context is in memory rather than a register set, and all the register sets are in use?
- 4) What is a process?
- 5) What is a PCB?
- 6) List some of the queues on a typical system.
- 7) What is the “degree of multiprogramming?”
- 8) What is swapping?
- 9) What is spawning?
- 10) Give examples of programs that might be related together as producer and consumer.
- 11) How do you determine if a buffer is full in a circular buffer (array)?
- 12) The ability of a process to create a new process is an important capability, but it has dangers. Consider the consequences of allowing a user to run the process below. Assume that `fork()` is a system call that creates a new child process.

```
int main() {  
    while (true) {  
        fork();  
    }  
}
```

- a) What would the consequences of allowing such a process to run?
- b) Suppose that you are an operating systems designer and have been asked to build in safeguards against such processes. Assume that you have decided that it is inappropriate to reject certain processes, and that the best approach is to place certain runtime controls on them. What controls might the operating system use to detect processes like the above at runtime?

c) would the controls that you propose hinder a process' ability to create new processes?

13) Explain the difference and the relationship between a program and a process. Is this difference important in single-process operating systems? Why or why not?