UNIX Exercises

Initial Exercises

1. Logon to a UNIX machine using your userid and supplied password.
2. Change your password to something that you will actually remember by using **yppasswd**. Do not pick a password that is an actual word *in any language* and try to include some numbers and special characters.
3. If you want to change your name and other personal information use **ypchfn**.
4. Logout and then login again using your new password. (Note that it may take about a minute for your password to be updated.)

Editing and printing files

1. Find emacs on the menu, or start emacs by using the **emacs** command in an xwindow.
2. Start typing in the emacs window, just write a couple of test sentences, and then save the file as `test.txt`. If you like, try out a few of the movement and search commands in the documentation.
3. Quit emacs and restart it to edit `test.txt`. Add a few more lines of text and then spell check your file.
4. Use the **printer** command to set your default printer to be the nearest one. Ask a demonstrator which one this is.
5. Print your file, using the nearest printer. Make sure that you can check the print queue to see whether your file is being printed or not.

Filesystem exercises

1. Have a look at your home directory using **ls**. Also try **ls -a** and **ls -l**.
2. Examine the contents of some of the files in your home directory using both the **cat** and **more** commands.
   Is there a difference?
3. Use the **pwd** to examine the full path to your home directory.
4. Create a new directory called tmp and change into it.
5. Copy some of the files from your home directory into tmp.
6. Rename one of the files in tmp and then delete it.
7. Delete the entire tmp directory.
8. Move around the rest of the directory tree. Try **cd** / and have a look at the top of the tree. If you get lost remember that **cd** returns you safely to your home directory.

Command history and filename completion

Many of the basic emacs editing commands can be used to edit the command line. N.B. This is shell specific. Perhaps the most useful is CTL-t which transposes two letters.

1. Try using the cursor keys to move about the command line. What does the up arrow do ?
2. Use the **history** command to look at previous commands. Each command has a number and can be executed by typing !n, where n is the number of the command. Try it for yourself.
3. Other useful history commands are !! (repeat the last command) and !$ (the last word of the previous command). Try using them to save time in the following exercises.
4. Type **ema** and then hit the TAB key. The rest of the command emacs should be filled in for you. This is command completion. Try it after typing the letter e only, you may need to type **set autolist** or hit TAB a couple of times to get this to work.
5. Type **ls** followed by a space and then hit TAB twice. You should be presented with a list of files (if not type **set autolist**, hit return and try again). Type the first letter of one of your files, perhaps the ‘t’ of test.txt and then hit TAB again. The complete filename should be filled in.

Processes and job control

1. Start emacs from an xwindow and type a few lines. Suspend the emacs process by using CTL-Z in the original xwindow and then exit the xwindow. What happens to the emacs window ? Why ?
2. Restart emacs and suspend the process again. Now move it into the background and kill the xwindow. What happens to the emacs window ? Why ?
3. Restart emacs yet again and suspend the process. Use the **jobs** commands to examine the jobs running from your xwindow. Move emacs into the background. Now use **jobs** again. What’s different ? Finally kill emacs by using the **kill** command in the xwindow.
4. Start emacs in the background with the command **emacs &**. Now use the **ps** command to examine the processes that *you* are running. Find emacs and kill it.
5. Use **uptime** to find the load average on your machine. Now use **top** to examine all the programs running on your machine.

6. Create a file called what.time containing the following lines. Note that the spaces ARE important in this file:

```
#!/bin/bash
#A script to test nohup
i=10
while [ $i -gt 0 ]; do
sleep 15
date
i=$(expr $i - 1)
done
```

   This is a shell script. Make the file executable (**chmod +x what.time**) and run it by typing **./what.time** What does the script do ? Interrupt the program using CTL-C.

7. Start what.time using the **nice +19** command. Is there any difference ?

8. Start what.time in the background using **nohup** and **nice** and then logout quickly. Login again, has the program continued to run ?

Remote machines

1. Have a look at loads and users on remote machines using the **rup** and **rusers** commands.

2. Find out the name of the machine next to you and logon to it using the **ssh** command. Try out a few simple commands.

3. Have a look at the running processes and load on the machine.

4. Use the **who** command to see who else is using the machine.

5. Try to run a program that requires a separate window, i.e. **emacs**.

Redirection and pipes

1. Type **ls -l** > **ls.txt**. Why don't you see the directory listing ? Examine the new file ls.txt.

2. Now use **sort** to sort ls.txt into alphabetical order. Use the input redirector < (although it's not strictly necessary).

3. Do it all in one go using a pipe **ls -l** | **sort** > **sort.txt**

4. The most used pipe construct is **ls -l** | **more**. Try it on a directory containing lots of files i.e /usr/bin. Make sure that you also try **ls -l** on its own.

General exercises

1. Use **firefox** to find the departmental computer documentation

2. Use the **info** command to find out about UNIX commands. Firstly, you should work though the info tutorial by typing **info** *info*. Once you have the hang of it, try **info** *yppasswd*.

3. Also, try the **man** command to find out about UNIX commands. N.B. On Linux systems man is often out of date.

4. Try searching for UNIX help online, again using firefox.

5. Work through the emacs tutorial