# NWEN 241
# Exercise: Introductory Lab

Qiang Fu

March 13, 2010

This exercise is NOT assessed.

## Preamble

This exercise is simply an opportunity for you to become familiar with the computing facilities provided for use with NWEN 241 (and later NWEN courses). These facilities are based on the Unix operating system, and familiarity with this system will be also very useful for more advanced level work both in research and commercial environments throughout the world.

This exercise was written a few years ago. Some of it seems slightly dated in 2010. This is an indication of how quickly computer science evolves.

## 1 Introduction

Unix was originally developed by Ken Thompson and Dennis Ritchie at Bell Labs in the early 1970s. It has been the preferred platform for university research in Computer Science for many years. And today Unix is, by far, the operating system available on the widest variety of machines, architectures, and vendors. It is the foundation for operating system standardisation, and for much of the work in Open Systems.

While the success and availability of Unix has been widespread, there are many different versions of the system, from many different sources. These systems have many names; for example Sun Microsystems call theirs "Solaris", IBM call theirs AIX, and so on. Recently a system called "Linux" (because it was orginated by Linus Torvalds) has been widely acclaimed because it is available free, and is developed through cooperation of many individuals (the "open source" model for development). All these systems are quite similar, and strongly resemble the original Unix system. However, Unix is a system that allows a great deal of customisation. This means that while all Unix systems share a common structure, they differ in many details.

Our undergraduate Unix labs consist of workstations using Pentium PC hardware and the free NetBSD Unix operating system. All the machines on the ECS network are named after Wellington landmarks: hills, restaurants, theatres, streets, bays, etc., e.g.

- `greta-pt`

This is usually followed by our network domain names, e.g.

- `greta-pt.ecs.vuw.ac.nz`

The other Unix system servers (e.g.`rialto` and `state-opera`) run is also NetBSD version of Unix. All these machines use the "X" Window System: a networked graphics system that enables software with graphic user interfaces to work transparently across a network. The Unix design has been very influential, and you may see much that resembles other computer systems. In particular, if you are familiar with MS/DOS or MS/Windows you will see much similar in Unix. However, beware, Unix is a far more sophisticated system than DOS, and many things that seem superficially similar differ significantly in detail.

Elsewhere the School of ECS uses various other machines, other versions of Unix, and a bewildering variety of software. All these machines are connected by an Ethernet local area network, which is used to support a Network File System, which allows files stored on discs attached to any machine as if they were all in a single integrated structure. Many other services also use this local area network, including electronic mail and news, and general remote processing facilities. The local area network is connected to others throughout the university, the Wellington City fibre-optic network CityLink, to the rest of the country, and the world as part of the Internet. (The School of ECS at Victoria was instrumental in connecting New Zealand to the Internet.)

It is very important that all Computer Science students understand the department's philosophy about use of the facilities. The facilities are provided as support for the research and course work of the department. They are designed, equipped, and supported to further that work, and are *shared* by staff and students. This also means that often there is hardware and software that is new or experimental being used, and that the environment changes from time to time. There are several implications of all this. Firstly, while we hope you will take the opportunity to explore the system and learn to use it expertly, we will not tolerate any thoughtless or malicious tampering or interference with the normal smooth operations. Secondly, we will expect you to cooperate with other people using the system with understanding and courtesy. Our system is expertly managed, but relies as much as possible on a spirit of friendly cooperation rather than draconian security and enforcement. We hope and trust that you will join in this community spirit.

# 2 Getting Help on Unix

There are many books on Unix commonly available, but be warned that most books will devote much space to software little used or irrelevant for our system.

More importantly, there is a wealth of documentation available on the system itself. You should quickly learn how this documentation is organised, because you will be expected to learn more and more about Unix in this way. For simple questions, you'll probably find it easiest just to ask someone at a workstation nearby; make sure *you* are friendly and helpful when it is your turn to provide help. In our department there is a *huge* wealth of knowledge about Unix — but *nobody* knows every option of every command: the system is just too diverse.

To get help for Unix commands, you may use Unix manual pages within KDE Help. You will find Unix manual pages on the left hand side of the KDE Help pane. You will learn more about this topic in section 5 of this document.

# 3 Text Interfaces, Graphics Interfaces, and KDE

The original Unix user interface used simple text interaction, where the user entered commands, and the system responded with lines of text. This was once typical of all computer software, but over recent years, graphical user interfaces have become preferred — especially for ordinary office computing (word processors, spreadsheets, etc.) However, because Unix was used more by programmers, the old-style text interaction has persisted. There is much Unix software that does use a graphic user interface, but the text interface is also very common.

Modern Unix systems usually use the X Window system and are capable of running both graphic and text user interface software. The text interface software is usually run in text windows within the graphics screen. It is possible to have several text windows in use all at the same time, and this is a common way of working.

The X Window system is a very versatile and customisable environment, and many systems have a different "look and feel". For our Unix systems at the School of ECS, we have decided to set up the default environment to use KDE, the "K" Desktop Environment. This environment runs within the X window system, but the "look and feel" is similar to Apple's MacOS or Microsoft's Windows environment. It provides menus for common programs, a file browser, and so on. We hope it will help new users quickly become familiar with our system.

Another document is available to help you find you way with KDE. It can be found online at:

- `http://ecs.victoria.ac.nz/Support/TechNoteKdeIntroduction`

You should make sure you have a copy of this document to refer to while starting to use our system. However, if you are familiar with Windows or MacOS, you should have no trouble getting started and becoming productive with KDE.

To become a fully productive Unix user, you will need to become familiar with the text interaction environment of Unix, and the sections that follow are designed to help you.

# 4 Getting Started

To use our system, you will need to have a "userid" set up, and an initial password set. If you have not done this already, see the notice on the wall for how to get this done.

Most machines and terminals will prompt you for your userid and password in turn, and simply repeat the prompts if you get either wrong.

**Do this**

1. Type your username at the Login prompt, and press Tab.
2. Type your password at the Password prompt, and press Enter.

   If you have trouble with your initial password, please ask a course tutor for help. If you have changed your pasword and forgotten it, it will have to be reset by the system manager.

When you get your username and password right, you will be logged into Unix. On an X Windows machine, there will usually be a smaller text interaction window, running the Unix command line interface, known as the "shell". In KDE you can bring up a shell window by clicking on the panel icon with a picture of a monitor with a `‘‘>’’` in it. You can use this window just as if it was a screen — but you can also resize it, move it around, open other windows, and so on. In the shell, you are expected to type a command line, followed the "Enter" key.

When you do successfully "login", the first thing you should do is change your password to one chosen by you. Your password should be at least 6 characters long, and preferably have both upper and lower case letters, and perhaps some punctuation mark. Your password should not be a word or name in *any* language — otherwise you are a target for password attackers with online dictionaries. A good strategy is to use the initial letters of words in a phrase or sentence you will easily remember. You should not tell your password to *anyone*, and should not let *anyone* use *your* userid.

Also, as an alternative, you can use the Web to change your password, using the following web page:

`https://ecs.victoria.ac.nz/cgi-bin/passwd`

**Do this**

1. Type `passwd`.
2. Type your old password.
3. Type the new password. You will be prompted to retype the password, in case you made a mistake.

In the KDE environment, there is a menu item to logout. On non X devices the command to finish your session is `logout`.

*Make sure you are logged out before you leave.* A correctly logged out screen should be displaying a new "login:" message.

# 5 The Shell

The Unix command line interpreter is known as the "shell" (because it conceals the inner kernel of the operating system). Simply speaking: the shell displays a prompt, you type a command possibly followed by some arguments, hit "Enter (return)", and the command is executed. A typical command line might look like this:

```
man -k compiler
```

This shows the `man` command, which will display parts of the online system manual. The `-k` is an option flag that in this case indicates that it should display an index of manual entries that match a certain keyword, and `compiler` is the keyword that will be searched for. So this command will produce an index of manual entries that contain the keyword "compiler". One of the entries it will show is the one for the C++ compiler command "g++"; the command `man g++` will display details about that command.

**Do this**

1. Type `man -k compiler`.

2. Try also looking up all commands that have to do with searching.

Note that our system is quite large, and we often have many commands that are probably of little use to you.

The full reality of commands is complicated. Firstly, the "commands" are not built in to the shell: they are just the names of files which contain executable programs. When you type a command, the shell looks in various places for a file name that matches, then executes that program. From that point on, the shell passes responsibility to the program to do what it is supposed to. Secondly, while the commands are *not* part of the shell itself, the shell does provide many many facilities to make dealing with the commands more easy — these are outlined below, although not completely: in fact the shell is actually an entire programming language when you need it to be. And finally, while we refer to "the" shell, in fact there are several different versions of the shell, and the details differ. Your userid has been set up to use a version of the "C" shell (so called because it has some similarities to the C programming language) — see `man tcsh` for the full (long!) details.

**Do this**

Try these simple instructions, to help orient yourself with the command line based system.
`date` — display the date and time
`who` — display who is using this server
`clear` — clear the screen
`echo` *something* — display the argument *something* on screen

Note that on an X Windows device, you must make sure the mouse cursor is in the window where you wish to type commands.

Some Unix commands expect you to type input to them on following lines (and they usually don't prompt — you're expected to know). When typing input to such commands, the way to signal you are finished is to type the end-of-file character, which is `Control-D` on a new line (just type it *once*). This is often the standard way to finish input in Unix.

**Do this**

Here is an example of such a command. Try it out until you are happy with the way the shell requires such input.

1. `wc` — word count: shows number of lines, words, and characters

The shell provides various facilities for helping you type command lines. For instance, it remembers each line you type, and you can select previous commands using the up and down arrow keys. And you can also use the left & right arrow keys, together with the backspace key, to change old command lines without re-typing the whole thing.

Some special characters indicate input/output re-direction. If a command line has `command >filename` then the output of the command is written to that file, instead of to the screen; `command <filename` takes the input from that file. Joining two commands together with a "pipe" `firstcommand | secondcommand` results in both processes being run simultaneously with the output from the first fed into the second. Again, there are also several more exotic forms of redirection.

**Do this**

1. Store the current date and time in a file called `nowdata`. (Use the '`>`' character.)
2. Count the number of lines in the file `nowdata` (Use the `wc` command.)
3. Use a pipe '`|`' to count the number of characters output by the `date` command.
4. Store the output from the `who` command in a file `whodata`.
5. Press the up arrow key until the command line does not change any more. What was the first command you typed?
6. Now press the down arrow key until the line is blank, then press the up arrow key once more. What was the last command you typed?
7. Press the down arrow key again.
8. The command `more` will display file contents on the screen. Use it to check the file *whodata*.
9. Press the up arrow key and use the backspace and left & right arrow keys to change *whodata* to *nowdata* and enter the command.

Many commands concern files and require file names to be typed on the command line. When you have typed the first few characters of the file name, you can hit the "tab" key, and the shell will automagically complete the name of the file for you — if it can.

As well, there are various other characters that you type as part of the command line that have special meaning. Filenames, for example, may contain "wildcard" characters. The character `?` in a filename matches any single character; `*` matches any *sequence* of characters;

and characters enclosed in square brackets match any single one of those characters — there are other special sequences too.

**Do this** To try these out, use the `echo` command, which just displays what ever it sees on its command line.

1. Use the "tab" key to complete the filename `nowfile` after only typing a few characters.

2. Use the `*` character to echo the names of all files that end with `data`.

## 5.1 Process and Job Control

Because Unix is a multitasking system, you can run processes in the background, letting you keep on doing other things in the foreground. Any command with an `&` at the end of a line is executed in the background, and the shell immediately prompts for another command, without waiting for the background command to execute.

When a command is actually running, some characters typed are intercepted by the shell as requests to interrupt the command program. Typing `Control-C` usually is taken as a request to immediately terminate the program — that's the way to stop infinite loops. Typing `Control-Z` usually is taken as a request to temporarily suspend the program. At that point you can execute commands again, and use the `fg` command to restart the suspended command in the foreground when you are ready — or you can use the `bg` command to restart it as a background process — or the `kill command` to permanently stop it. The `jobs` command will tell you what background and suspended commands you have, and you can specify them by the numbers shown, prefixed with a `%` character.

**Do this**

1. Type `xcalc`, the on-screen calculator. This creates a new process with a new window for a calculator. Move the mouse to the calculator and do some simple calculations.

2. Move back to the original xterm (X terminal) window. Can you run other commands? (No, because `xcalc` is running in the foreground, and the shell is waiting for it to finish.) Type `Control-C` in the xterm window to kill the `xcalc` process. Now you should be able to run commands again.

3. Type `xcalc` again, and make sure the calculator is working.

4. Now move back into the original xterm window by clicking on it, and type `Ctrl-Z`. Move to the calculator and do some more calculations. What happens? (The calculator shouldn't work: why?)

5. Run the `jobs` command in the original window. Type `bg` and run `jobs` again. What has changed, and why?

6. Move into the `xcalc` window and try some calculations again. What happens? (It should work again: why?) When you are done, type `q` to quit the calculator (this is the "ordinary" way of quitting `xcalc`). Note, your cursor should be in the calculator window.

7. Run the `jobs` command again. What has changed, and why?

8. Now run `xcalc` directly in the background with the command `xcalc&`. This is the way we usually start programs when we want them to run independently from the shell. Check that the shell and the calculator may now both be used, and make sure you understand what has happened.

9. Now use `kill` to terminate the calculator process. (You can also use `Control-C`.)

There is much much more to the shell, from the `alias` command to give custom names to commands, to shell programming, special shell variables, and automatic start-up scripts. Much of this is documented in the manual entries on-line (see next page). But you don't need to understand everything to get your work done, so please don't feel overwhelmed by the complexity of the system.

Also, please don't get carried away customising your setup (or letting someone else do that for you) while you are still getting used to the system: it is too easy to make things even more confusing than they already are!

# 6 Finding Documentation On-Line

Some useful commands to remember in Unix provide access to the documentation stored on-line. The most common is the `man` command which was metioned earlier for listing entries in the online manual. Used as `man -k` it allows you to specify a keyword, and it displays a list of commands (with brief descriptions) that include that keyword in their description. So `man -k copy` will display a list of commands that have some connection with copying.

**Do this**

Try

1. `man -k user`. The output from this will go off your screen, but you can see some of it by enlarging the window (see section 9), or by piping the output to the `more` command (see later in this section).

2. `man -k delete`.

3. `man -k password`.

The most important documentation command, however is the ordinary `man` command. You simply specify the name of the command (or name of some other important part of the system), and it displays, page by page, a complete manual for that command. The format for

the manual entries is standard and consistent, and is a simple description of the command, a synopsis of how it is used, a list of all the options that may be specified, and a description of how it works. There are often also sections at the end that mention files that are used, and bugs or limitations in the command. These manual pages are rich with information and an invaluable source of reference. You should get used to reading them now, and they should be the first place you look when you are trying to understand any aspect of the system.

**Do this**

Have a look at the manual entry for these commands.

1. `cp`. What does the -r option do?
2. `date`.
3. `man`.

Do not be alarmed if there is too much detail for you to take in all at once — just scan for what you need. From now on when new commands are mentioned, you will be assumed to be looking up the `man` entry for them. Unfortunately there will be some commands that are not documented with `man`. Sometimes this is because they are grouped together in one manual entry — the entries for the shell (`csh`) document many small commands that are built into the shell. Some commands simply have no documentation written for them: unfortunate, but true.

Finally, on the subject of on-line documentation, many commands have some documentation built in. Many commands will display a brief guide to their options if you specify some option that doesn't exist. Other commands have a special "help" option.

The KDE environment has a panel "help" item, shaped like a life preserver ring. This contains links to the same online man files, and some other documentation as well.

# 7 File Manipulation

The file system is the context for most activity in Unix, and many commands involve activity with files.

Fundamentally all the files on the system are part of a single huge hierarchical structure. Ordinary files contain data, perhaps textual data, or perhaps executable binary data. Directory files, or *directories* give the hierarchy its structure, by containing the names of files and directories at a lower level in the structure. The root of the entire file system is a directory simply called "`/`", and any file in the system may be named by specifying the "pathname" of the file, starting at `/` and stating each directory to be followed to find the file — the `/` character also separates the directory names.

A person with userid `bloggsjoe` may have a file with the pathname

- `/u/students/bloggsjoe/myfile`

In fact the "home" directory of a user may be specified with a special notation, and the name could be specified as ~`bloggsjoe/myfile`. In fact, if *you* were `bloggsjoe`, you could say ~`/myfile`. (Note that because we use a Network File System (NFS), absolute pathnames sometimes differ when shown from the point of view of different machines.)

If you don't start a pathname with `/` or ~ , then the pathname is taken relative to the "current working directory". So when you login, just the name `myfile` is sufficient to name your file.

There are many commands for manipulating and generally dealing with files. The most often used commands are these:
`more` — display a text file on the screen, page by page, allows searching
`less` — a superior version of more, allows backwards scrolling too
`cp` — make copies of files
`mv` — move files to a different place
`rm` — remove files (i.e. delete them – and there is *NO* way back)
`ls` — list the names of files in directory
`chmod` — change the protection mode of the file
`cd` — change your working directory
`pwd` — display your working directory
`mkdir` — make a new directory
`rmdir` — remove a directory (i.e. delete it)
`du` — display the disc usage of a directory
`gzip` — convert files to compressed format to reduce disc usage
`gunzip` — convert the compressed files back to usable format
`diff` — display differences between two files
`grep` — find all occurrences of a text pattern in a file
`sort` — sort a file into order, line by line
`wc` — count the words (and characters and lines) in a file
`lpr` — print out a file on the printer

Please note:

- You can change which users have access to your files and directories with the `chmod` command. However, you should be very careful about doing so; it may well be best just to keep access completely private.
- The KDE environment has a panel item for a file browser; the icon is shaped like a house, and will open a window displaying files in your home directory. This allows a more graphical approach to manipulating files.

# 8 Text Editing

While various straightforward commands are useful for manipulating files in all sorts of ways, there is still a great need for a general purpose "text editor" — a program that allows you to enter text, easily change it, search through it, and so on. The traditional Unix text editor is very simple line editor called `ed`. In much of the Unix world, the most popular editor is the enhanced visual version of ed, called `vi`.

Neither of these editors is used much at VUW, however. Here the editor used by most people is `Emacs` (the name comes from "editor with macros"). Emacs is used in much of the Computer Science research world, and its capabilities and flexibilities that are impressive. Understanding *all* of Emacs is a large task, but learning the essentials is easy. Emacs has pull down menus for common tasks, but also has many control keys. An Emacs reference sheet is available, and you may find it worthwhile to study closely at some time. In KDE, emacs can be started by selecting its icon from the "Editors" sub-menu that can be found from the "Utilities" sub-menu of the "K" menu. The icon is a picture of a Gnu, taken from the name of the project which involved development of Emacs.

# 9 E-Mail

Because all our computers are connected by a network, both throughout the department and throughout the world, they form together a very useful means of communication. For person-to-person communication, we use "e-mail". Using e-mail you can easily send message or files to other computer users anywhere. As is usually the case the Unix, there are in fact several programs that let you read, send, and store e-mail. The KDE porgram is called "Kmail", and there is an icon on the panel to start Kmail.

Another program you can use is the simple text menu-based program `pine`. (The name "pine" is a joke: an earlier system was called "elm" — for ELectronic Mail.)

When sending e-mail within the department's local electronic mail system, the local username (such as `bloggsjoe`) is usually all that's required for the e-mail address. The complete e-mail address, such as *joe.bloggs@ecs.vuw.ac.nz*, can also be used. The complete address is required for people who wish to send you email from outside the Mathematical and Computing Sciences department.

**Do this**

Try out using email now: first look to see if you already have any mail — you should have a message welcoming you to NWEN 241. You may also wish to try sending someone a message: try sending to someone else in the lab, so you can check to see if it arrived properly.

Like almost everything in the Unix system, there are many alternatives to `Kmail` and `pine`. In particular, there is an older system that uses the command `mail`, but it is usually more primitive and difficult to use than `pine`. Some experienced users use a different system called `mh`, but it has many complexities and is again best avoided by those new to Unix.

# 10 World Wide Web

One of the more important developments in networked computing is the "World Wide Web" (WWW). This is a system that allows many resources (documents, photos, sound) to be organised and presented together via a browser program, despite the resources being distributed anywhere on the network (and so, the world). Both VUW and the School of ECS have WWW information available, and we use the Web extensively. To access this information, use the browser program `konqueror`, and see the separate reference sheet for details. There is even a special "home page" for NWEN 241: see the URL

- `http://ecs.victoria.ac.nz/Courses/NWEN241_2010T1/`

We usually use Konqueror and Mozilla to access the web, and KDE has icons on the panel to start these programs. Start Konqueror or Mozilla, and explore for a while. Then open the NWEN 241 home page and explore that. Add the NWEN 241 page to your "bookmarks" so you can easily get to it again.

# 11 Network Access

In reading the above sections, you will have realised that many of the network services (e.g. e-mail, news, the web) are global in their reach. Because of the cost of these services, the department restricts undergraduate students in their access to these services. However, use of the web is allowed anywhere in New Zealand, and reasonable use of e-mail and network news is unrestricted. You must be careful to use these capabilities with courtesy and cooperation.

# 12 What Now?

**Don't Panic!** Although the system is rich with detail, and may seem overwhelming now, it *will* become familiar with a little investment of time and practice. You should read through the notes, and take the time to come back to the lab and try things out.

If you feel you should get a book about using Unix, we recommend *Unix in a Nutshell*, by Daniel Gill (O'Reilly Publishing 1994). Many other good books are available.

Read through the documentation as you need it, and work through any examples using our system. Remember that some details may differ, but the differences should be minor. If you run into problems, make an effort to resolve it by reading the documentation — but if you are really stuck feel free to ask the person next to you, the tutors, the lecturer, or one of the department programmers. In your practical computing work at VUW, and possibly in work beyond VUW, familiarity with this computing environment will be very useful. An investment of time now will pay dividends in the future, so we encourage you to explore and understand the system.

In future labs, various facilities relating to project work will be explained and explored. And if you get lost or confused there are lots of people to give you help and directions. Soon *you* will be giving the help and directions. *Good Luck!*

**Don't Forget ...**

Don't forget to logout. You can be sure you are logged out when the screen displays its login prompt for someone else to login.

Do not turn off the power for any of the lab machines. Because of multitasking and networking, it is possible other users or important background processes are still using the machine at any time of day or night.