School of Computer Science and Software Engineering
Clayton Campus, Monash University
CSE2303 and CSE2304
Semester I, 2001

Practical Session 0
Introduction to Linux

Novell accounts. Every Monash student can get an account on a Novell network. At Clayton the Novell servers begin with **MFS**. If you already have one then you simply use it. If you don't have one or if you have forgotten your password you will need to take your ID card to IT Services Helpdesk where you can be issued with your username and password.

Linux accounts. To do your pracs you will need to obtain a Linux account. You will need to register to activate your Linux account, even if you had one last year. As with your Novell account, you need to get both a username and password. Your username is the same as the one for your Novell account, **except** that it is in lower case. To get your password, follow these steps:

- Have a pen and paper handy.
- Log into your Novell account.
- Do **only one** of the following:
  - type at the prompt: **TELNET ALPHA1.CC** then log in as **REGISTER** (*not* your own username), or
  - open Netscape in Windows and point the browser at **http://alpha1.cc.monash.edu.au/htbin/getpass**

Both methods ask for username, ID number and date of birth. The password disappears after 40 seconds so make sure you write it down. You can only register once so if you lose your password you will need to go to IT Services Helpdesk to have your password changed.

Booting Linux. Once you have obtained your Linux password you can log into your account. To do this, you must first start up Linux on the computer in front of you. To do this, follow these steps:

- Turn on the computer, or if it is already on press **CTRL + ALT + DELETE**
- The computer will prompt you for an operating system to load as it is booting. Simply select Linux and press **ENTER**. You may be prompted to do other things along the way.

Due to how the IT Services has setup their system, Linux takes a while to load. When it is ready, it displays a prompt like this:

clay-b19g13-10 login:

Every computer has a different name; this prompt tells you the name of the computer in front of you.

Logging into Linux.

To log in: At the prompt, type your username and password.

To log out: When you're finished type **logout** or **exit** to close your session.

Changing your password. Since the linux password you were given by the registration program is difficult to remember, it's probably wise to change your password right away to something that you can remember. To change your password, type:

clayton-b19g13-10> passwd

You will be prompted for you old password, then you should type a new password and confirm it by typing it again.

X Windows. By default, Linux has a text-only command-line interface. However, there is a graphical environment you can run in Linux. It is call the *X Window System*, or usually just *X Windows*. You can start X Windows from the command prompt by typing:

clayton-b19g13-10> startx

To exit from X windows you should close all the applications that you are running. Then click on the **Start Menu** button. Pick **exit fvwm** at the bottom of the menu. This will return you to the text-mode screen which you will also need to log out of by typing **logout** or **exit**.

Help Commands. In the following sections we will introduce you to a number of commands. For most of these commands there is online documentation, which describes the command, its options and how to use it. To read these manual pages use the command **man**. For example to read about *startx* type:

clayton-b19g13-10> man startx

There is also a X window version of **man**. To run it type the following in a terminal window after you have started X windows:

clayton-b19g13-10> xman

Other helpful commands are:

apropos *string*
: Searches the short manual page descriptions in the *whatis* database for the occurrence of each *string*. Like **whatis**, except that it searches for strings instead of words. Equivalent to **man -k** *string*.

info
: GNU hypertext reader: displays online documentation previously built from texinfo input.

whatis *keyword*
: Searches the short manual page descriptions in the *whatis* database for each *keyword*. Equivalent to **man -f** *keyword*.

whereis [*options*] *filenames*
: Locates the binary, source, and manual page files for a specified command/file.

which *commands*
: Lists the full pathname of the files that would have been executed if the named *commands* had been run.

Directory Commands. Linux groups files into directories (like folders in Windows), and a directory can contain other directories as well as ordinary files. Also, each user is assigned a *home directory* that contain the files and directories that belong to him or her. The following are basic commands which will enable you to move around and manage directories.

cd [*dir*]
: Stands for *change directory*. With no arguments, it changes the working directory to the home directory. Otherwise, it changes the working directory to *dir*.

ls [*options*] [*names*]
: Lists the contents of directories. If no *names* are given, it lists the files in the current directory. With one or more *names*, it lists the files in the directories *names* or match the files *names*.

mkdir [*options*] *directories*
: Create one or more *directories*. You must have write permission in the parent directory in order to create a new directory.

pwd
: Prints the full name of the current directory.

rmdir [*options*] *directories*
: Delete the named *directories* (not the contents). The *directories* must be empty and you must have write permission in their parent directories.

Exercise:

- Make two new directories **pracs** and **pracs/prac0**.
- Find where the file **stdio.h** is stored. Hint start looking in **/usr**.

File Commands. Linux provides a selection of commands for working with ordinary files rather than directories. The following are the basic commands that allow you to manage files.

chmod [*options*] *mode files*
: Changes the access *mode* (permission) of one or more *files*.

cp [*options*] *sources target*
: Copy file to file, or files into a directory.

less [*options*] *filename*
: Allows you view any text file a screenful at a time.

lpr [*options*] *filenames*
: Allows you to print files. Before you can print any files on the university system you will need to select your printer using **setprint**.

mv [*options*] *sources target*
: Renames files and directories, or moves files into a directory.

rm [*options*] *files*
: Deletes one or more *files*. To remove a file you must have write permission in the directory that contains the file, but you need not have permission on the file itself.

setprint
: Selects a printer on the university system.

Text Editing. There are several text editing programs available on Linux. On the system you will be using on campus the simplest is **nedit**. To learn how to use it read its manual page or run the program and select its Help feature. Other editors you may wish to try are **emacs**, **vi**, or **jove**.

Exercise:

- Create a file called **hello.c** which contains the following program.

```
#include <stdio.h>

main()
{
  printf(``Hello World!\n'');
}
```

Compiling C Programs In Linux to edit, compile and run your C programs you will need to use different commands. First you need to use an editor to create a program. Once you have finished editing your program you need to turn your source code into executable code. The compiler **gcc** (which stands for *GNU C Compiler*) turns C programs into executable code. For example to compile **hello.c** type:

> **clayton-b19g13-10> gcc -o hello hello.c**

To run the program type:

> **clayton-b19g13-10> ./hello**

Note

1. If you leave out **-o hello** from the above command then the executable file will be called **a.out**.
2. If you include **math.h** in your program you will need to use the option **-lm** when you call **gcc**.

   Exercise:

   - Get the file **commandLine.c** from the internet via the web page

     http://www.csse.monash.edu.au/courseware/cse2303/pracs/prac.html

   - The program **commandLine.c** prints out all the command line arguments. If there are no arguments it will echo what you write until you type **CNTR+D**.[1] Compile **commandLine.c** and run the program with various arguments.

Commands. A command usually ends with a newline. Another command terminator is the ampersand **&**. It's exactly like the newline, except it tells the shell not to wait for the command to complete. Typically, **&** is used to run commands in the background. For example to run **xman** in X windows in the background, start X windows and type:

> **clayton-b19g13-10> xman&**

Many commands take standard input print output to the standard output. Using the following redirection commands the input and output can be redirected.

> *file*
: Directs standard output to *file*.

>> *file*
: Appends standard output to *file*.

[1] Other books and notes use other notations including **control-D**, cntr-d, **ˆD**, and < **ctrl-D** >.

< *file*
: Takes standard input from *file*.

p1 | *p2*
: The | is called a *pipe* and it connects the standard output of *p1* to the input of *p2*.

Exercise:

- Often you need to look at more material than can be viewed in a single screenful. Try typing **ls /etc** then try typing **ls /etc** | **less**.
- You don't need to start an editor to create a file. Try typing:

  clayton-b19g13-10> cat > junk

  Then write whatever you want to and finish by typing **CNTR+D**.
- Write a C program, **reverse**, which takes each line from the standard input and prints it out in reverse order. You may assume each line of input has at most **1023** characters including the newline character **n**.
- Create a text file, **tst.txt**, which contains 10 lines of text. Then try typing:

  clayton-b19g13-10> ./reverse < tst.txt

- Try typing:

  clayton-b19g13-10> ./reverse < tst.txt | ./reverse

Filters Many Linux programs read some input, perform a transformation on it, and write it to some output. These programs are called filters and when used together with pipes can produce powerful programs. Some useful filters are:

comm [*options*] *file1 file2*
: Compare lines common to the sorted files *file1* and *file2*.

diff [*options*] *file1 file2*
: Compare two text files and reports the difference between *file1* and *file2*.

head [*options*] *files*
: Prints the first few lines (default is 10) of one or more *files*.

sort [*options*] *files*
: Sorts the lines of the named *files*.

spell
Prints the words in the standard input which do not appear in its dictionary.

tail [*options*] *files*
Prints the last few lines (default is 10) of one or more *files*.

uniq [*options*] *file*
Removes duplicate adjacent lines from a sorted *file*.

wc [*options*] *files*
Prints the number of characters, words and lines for each *file*.

Exercise: The point of this exercise is to find all the 9-letter words which contains all of the following letters **u, c, r, o, e, n, i, s, r**.

- Get the file **premute.c** from the internet via the web page

  http://www.csse.monash.edu.au/courseware/cse2303/pracs/prac.html

- Compile **permute.c** and call the executable **permute**. Then use the program **permute** to write all permutations of the above letters to a file **perms**, by typing:

  clayton-b19g13-10> **./permute ucroenisr** | **sort** | **uniq** > **perms**

- Run **spell** on the file **perms** and put the output in the file **rejects**, by typing:

  clayton-b19g13-10> **spell** < **perms** > **rejects**

- Now type:

  clayton-b19g13-10> **comm -23 perms rejects**

  to find all the *valid* 9-letter words.

dwa 2001.