1. Entering Unix Commands

A simple command is a sequence of (one or more) words separated by blanks or tabs. The first word of the command is the command's name. Subsequent words are the commands arguments, e.g.

```
stella:oradba??> pine
stella:oradba??> pine  -i
```

Normally, a user enters commands one at a time (i.e. in one line), but he/she may also type in several commands on one line as long as they are separated by semicolons, e.g.

```
stella:oradba??> pwd  ;  ls  ;  who
```

2. Input/output/standard error

The *standard output* (stdout) of an user is his/her terminal screen, while the *standard input* (stdin) is the terminal keyboard. There is another output connection called *standard error* (stderr) which displays the error and diagnostic messages. For instance, by default, the outcome of a command will be shown on the user's screen since it is the standard output, e.g.

```
stella:oradba??> cat nonexistent
```

Assuming the file nonexistent does not exist, then an error message will be displayed on your screen.

3. alias

It is possible to assign alias to Unix commands, e.g.

```
stella:oradba??> alias dir ls
stella:oradba??> alias del 'rm -i'
```

The effect of these will be assigning the Unix command `ls` to a command called `dir`, and assigning the Unix command `rm -i` to a command called `del` so that, for example, entering `del` will be equivalent to entering the Unix command `rm -i`

4. ls

The Unix command `ls` (short for *list* or *listing*) lists the files in the current directory:

```
stella:oradba??> ls
```

There are several **options** you can use with `ls`. The `-l` option displays a **l**ong listing of your files, including size and date information.:

```
stella:oradba??> ls -l
```

The `-a` option displays **a**ll the files; filenames that start with `.` aren't usually displayed, because they're utility files, and only clutter up the display.

```
stella:oradba??> ls -a
```

The `-F` option prints an extra character at the end of some filenames. A `/` indicates that the file is a directory.

```
stella:oradba??> ls -F
```

If you've used the `pine` command, you have a `mail` directory, which you can list:

```
stella:oradba??> ls -l mail
```

Listing a directory shows you all its files. Sometimes, you only want to list information about the directory itself. You use the `-d` option to do this:

```
stella:oradba??> ls -ld mail
```

5. cat

There are several ways to create a new text file in Unix. Short files can be created with `cat > filename`. Before you try this command, you will enter the command `set noclobber` to prevent accidental overwriting of existing files:

```
stella:oradba??> set noclobber
stella:oradba??> cat > new
This method is ok for creating short files.
Like copy con, you can't correct mistakes on earlier lines.
To end the file, type Ctrl-D (NOT Ctrl-Z!!!) on a new line.
^D
```

`cat`'s normal behaviour (without the >) displays your new file:

```
stella:oradba??> cat new
```

What happens if you enter the command **`cat > new`** again, now that the file `new` exists? If you want to *append* (add more lines) to **`new`**, use the command **`cat >> new`**, type more lines and end with **`^D`**.

6. echo

The echo command just repeats its argument:

```
stella:oradba??> echo new2
new2
```

As with `cat`, the output of `echo` can be *redirected* into a file:

```
stella:oradba??> echo This method is ok for creating 1-line files.> new2
```

7. Creating files using pico

You can use the program `pico` to create and edit files, as shown below. With `pico` you can move the cursor with the arrow keys, perform other actions with `Ctrl` keys (as indicated at the bottom of the screen).

```
stella:oradba??> pico goodbye
Dear Alex,
This is a very hard letter to write and I'm not sure where to begin but I
hope you'll understand. It really is better this way.
Yours truly,
Nik
^X
```

8. Edit an existing file using pico

In this exercise, you'll edit a copy of your `.cshrc` file. It's always safest to edit a copy of important files, in case you accidentally delete a lot of it. The Unix command to copy files is `cp`:

```
stella:oradba??> cp .cshrc mycshrc
stella:oradba??> pico mycshrc
```

First, we'll change your Unix prompt, so that it tells you the current directory. Move the cursor to the end of the file, and add the following line (the space after `$cwd` is necessary; the other spaces make it look nice):

```
set prompt="$cwd ! % "
```

Exit `pico`, saving the changes to `mycshrc`. The commands in your `.cshrc` file run automatically when you login, but you can use the `source` command to run the commands in any file, anytime. E.g., to run the commands in `mycshrc`:

```
stella:oradba??> source mycshrc
```

You should notice a change in your Unix prompt. If everything else seems to work ok, rename `.cshrc` to `cshrc.bak` then rename `mycshrc` to `.cshrc`:

```
stella:oradba??> mv .cshrc cshrc.bak
stella:oradba??> mv mycshrc .cshrc
```

9. rm

Unix *removes* files using the command `rm`. Whenever you remove files, be very careful to remove **only** the files you really don't want. E.g., `rm *` is a **very dangerous** command!

You can remove more than one file at a time:

```
stella:oradba??> rm new new2 goodbye
```

10. pwd, cd, mkdir & rmdir

Unix directories are tree-like, just like Windows directories; however, there are no drives, so there's only one root directory. The name of the root directory is `/` (not `\` as in Windows). The absolute path name of your home directory is the output of the following command:

```
stella:oradba??> pwd                  (print working (i.e., current) directory)
```

`cd` **c**hanges the working **d**irectory to its argument. The directory name `.` means *current* directory and `..` means *parent* directory. The command `cd` without an argument takes you home.

```
stella:oradba??> cd ../..             ( )
stella:oradba??> pwd                  (where are you?)
stella:oradba??> cd                   (go home)
stella:oradba??> pwd                  (just to make sure)
```

The Unix commands to create and remove directories are `mkdir` and `rmdir`. Try using these commands to create and remove directories.

11. wc

The `wc` command counts the number of characters, words and lines in the specified file(s), or you can specify which counts to display:

```
stella:oradba??> wc filename         (display number of characters, words and lines in filename)
stella:oradba??> wc -l filename      (display number of lines in filename)
stella:oradba??> wc -lw filename     (display number of words and lines in filename)
```

Like all the other filters, `wc` can also accept input from a pipe. Enter a command to count the number of lines in the output of the command **`ls`** (what info does this tell you?)

12. sort

The `sort` command sorts the lines in its input on different "logical" columns (*fields* separated by white space). By default, it sorts on the first field, but you can tell it to sort on the next (`+1`) or the `n`th next (`+n`) field:

```
stella:oradba??> sort +2 marks              (sort on the third column of marks)
```

This command should sort by the marks in the third column, but the smallest number is out of order. This is because `sort` normally sorts *lexicographically* (in ASCII order); in this order, "9 " (`9[Space]`) comes after "87". You can tell `sort` to sort *numerically* by using the option `-n`:

```
stella:oradba??> sort +2 -n marks           (sort numerically on the third column of marks)
```

13. Unix File Protection

Unix file protection is based on three types of file access and three groups of file users.
Types of access to a file are:

- *read the file (r)*
- *write the file (w)*
- *execute the file (x)*

The three groups of users are

- *file owner (u, user)*
- *group members (g, group)*
- *public (o, others)*
- *all of the above (a)*

Access	Binary	File_name
rwx --- r-x	111 000 101	program1
rw- r-- r--	110 100 100	text1

Ignoring the left most bit, which indicates the type of file, e.g. `d` for directory, `-` for ordinary file, we have: 1st 3 bits for owner (u), 2nd 3 bits for group (g), last 3 bits for public (o). (In this lab sheet, unless otherwise indicated, we would disregard the left most bit which indicates the file type.)

A command that can be used for changing file access is **`chmod`**

Only the owner of a file (and superuser) can change the access rights to that file, e.g.

```
stella:oradba??> chmod  o+rw  filename          (adds read and write permission to public)
stella:oradba??> chmod  u=rx  filename          (sets read and execute permission to owner)
stella:oradba??> chmod  o+x, ug-r  filename     (adds execute permission to public, removes read
                                                 permission from owner and group)
```

Appendix : Unix Commands

Managing Files

Ls	Outputs info about (specified) file(s) in current/specified directory	ls -a
		ls -l asst*
Cp	Copies existing file(s) to different name(s) or directory	cp .cshrc cshrc.old
		cp ../data/menu .
Mv	Renames file(s) and/or moves file(s) to another directory	mv .cshrc cshrc.old
		mv recipe* Recipes
		mv pieRecipes Recipes/Pies
Rm	Removes the specified ordinary file(s). -r (recursive) option on a directory removes all files and subdirectories	rm cshrc.old
		rm -r Recipes
chmod	Changes the modes (permissions) for file(s)	chmod a+x findnum
		chmod g-r *.*
Pwd	Prints (outputs full path name of) working directory	pwd
Cd	Changes to specified directory, or to home directory	cd Recipes
		cd
mkdir	Make a new directory with the specified name	mkdir Recipes
		mkdir Recipes/Cakes
rmdir	Removes the specified empty directory	rmdir backup

Viewing (Part of) the Contents of Files

Cat	Outputs entire file	cat filename
more	Pages through entire file	more filename
pico	Loads entire file into editor	pico filename
grep	Outputs lines of file that match the specified regular expression	grep -i apple filename
head	Outputs the first 10/specified number of lines of file	head -3 filename
tail	Outputs the last 10/specified number of lines of file	tail -5 filename
Wc	Outputs counts of lines, words, characters in file	wc -w filename
sort	Outputs lines of file, sorted	sort +2 phone.lst

Changing the Contents of Files

... >	Saves output of preceding command into specified file	finger > users.now
... >>	Appends output of preceding command into specified file	date >> users.now
echo... >	Saves argument of echo in specified file	echo One line > message
cat >	Saves following standard input in specified file□End with Ctrl-D (NOT Ctrl-Z!) on new line	cat > shortfile
		Line one
		Line two
		^D
pico	Loads specified file into editor	pico shortfile

Personal & Interpersonal Matters

passwd	Prompts you for your old password, then your new one	passwd
logout	Logs you out	logout
kill	Kills processes in current command shell	kill -9 0
Whoami	Outputs your userid	whoami
who am i	Outputs your userid and other info	who am i
who	Outputs list of users who are currently logged in	who
finger	Outputs list of logged-in users with real-life names	Finger
	Outputs info about specified user	finger userid@matilda
pine	Allows you to read/reply to your mail	pine
	Prompts you for Subject; sends mail to userid	pine userid
write	Writes to specifed logged-in user	write userid
mesg	Outputs current mesg status, or switches it to y or n	mesg; mesg n; mesg y
history	Outputs the last several commands you entered	history
set	Outputs shell variable settings, or (re)sets value	set prompt = "Yes "
alias	Outputs all aliases or specified alias, or creates a new one	alias
		alias m
		alias dir ls -l

Other Useful Commands & Files

date	Outputs current date and time	date
cal	Outputs current or specified calendar month	cal; cal 12 1993
man	Outputs manual entry for specified UNIX command	man cal