


4 Basics of Unix

You should read through this section, ideally sitting in front of a terminal so you can try out the various commands.

4.1 Introduction

The Unix operating system is accessed partly through the X-Window interface and partly through *Terminal Window* into which you type *commands*. This section covers the basic UNIX commands that you will need to use in the *Terminal Window*.

4.2 Entering Commands

Commands are typed into an *active* terminal window, however the command is *not* executed by the system until you press the RETURN or  key. Note also that all commands *are* case sensitive (capital and lower case letter *are* different).

4.3 The Directory Structure

All information, be it a program source, letters, programs, or system applications, are all held in “files”. These files are grouped together in “directories”. The directories are then arranged as a “tree”, starting from the system *root* directory. This is shown schematically in figure 1

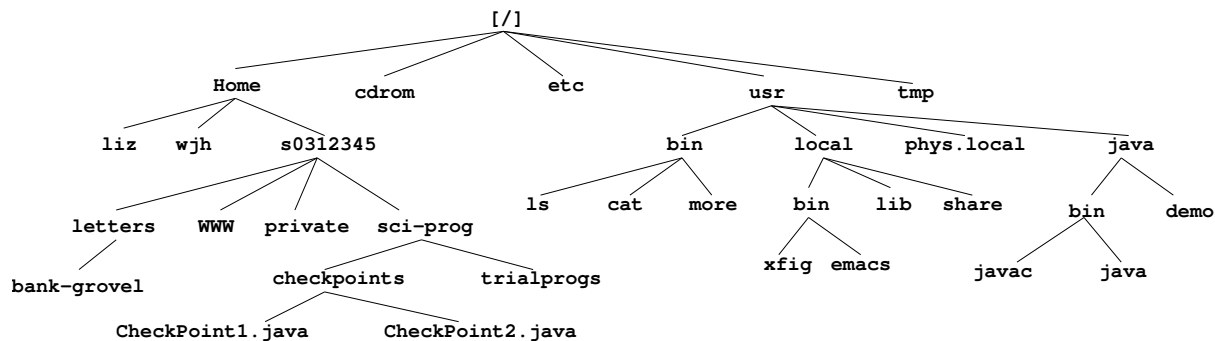


Figure 1: Schematic of part-of the UNIX file system.

For example the file that contained the source code for the solution to your first checkpoint may be located in:

```
/Home/s0312345/sci-prog/checkpoints/Checkpoint1.java
```

while the JAVA compiler `javac` is located in:

```
/usr/java/bin/javac
```

The structure is divided into *user* and *system* files, being the files that belong to the users of the system and the files that are part of the general system. On this system the users file are under `/Home`, while the system one are mostly under `/usr` (the name is historical!).

Your files are located under `/Home` in a directory which have the same name as your *username*. So in the above example if your *username* is `s0312345` then your files are located under

`/Home/s0312345`

This is known as your *home directory*. All files below this directory belong to *you*, and *you* can do anything to them, (read, write, delete etc.).

Path Names

Filenames can either be *absolute*, where the file names defines the whole *path* from the system *root* (`/`) directory. Such filenames start with (`/`) as above. Alternatively filenames can be *relative* to where “you are” in the file structure. In this case the name does not have a leading (`/`).

For example, if your username is `s0312345` then when you login the terminal screen is initially in your *home directory*, being `/Home/s0312345`. This is called your *present working directory*. So when in this directory the file `checkpoint1.c` can be referred to as **either**:

`/Home/s0312345/sci-prog/checkpoints/CheckPoint1.java`

or

`sci-prog/checkpoints/CheckPoint1.java`

where the first example is its *absolute* name and the second is its name *relative* to your *present working directory*.

This all sounds rather complex, but is actually very intuitive once you have tried it out for yourself!

4.4 File and Directory Names

File and directory names can be up to 32 characters long and can be *any* combination of letters, digits and punctuation characters `-+.,_ : ;`. There are however some rules, and basic guidelines,

1. Directories or files names that start with a dot (“.”) are called *hidden*. These will not appear on normal directory listings, see below. Such files are *normally* used to hold user options to various applications.
2. Most filenames are of the form `<basename>.<extn>` where `<extn>` is used to specify the type of information in the file.
3. It *is* possible to use other punctuation characters but they can clash with wild-card characters and other special characters, so should be avoided.
4. Spaces are *not* allowed.

4.5 Directory Manipulation

The following commands are used to check and set your *present working directory*:

```
pwd          show the present working directory
cd           change directory to home directory
cd newdir    change directory to newdir (newdir can be absolute or relative)
cd ..        change directory to parent directory (one-up)
```

Your prompt changes as you change directory to show the basename of the *present working directory*.

New directories can be created, and deleted with:

```
mkdir newdir  make directory newdir (newdir can be absolute or relative)
rmdir olddir  remove directory olddir.
```

You will get an error message if the directory you try and remove contains either files or other directories.

4.6 List of Files and Directories

Generating lists of the contents of directories is obtained with the very flexible command `ls` which has many dozens of options. The most common are:

```
ls          list of file and directories in current directory (simple output)
ls -l       list of file and directories in current directory (long format)
ls -a       list of all files, including hidden files. (see section on filenames)
```

4.7 Viewing and printing files

The contents of a file can be “paged” to the screen with the utility `more`. If you want to view the contents of file `Fred.java` then the command is,

```
more Fred.java
```

If the file `Fred.java` is longer than the current terminal screen the following keys come into play, these being.

```
SPACE      Forward one screen
D          Forward half screen
RETURN     Forward one line
B          Back one screen
H          Rather cryptic help screen
Q          Quit
```

Plain text files can be printed using

```
lp <filename>
```

where `<filename>` is the name of the file to be printed. This printer is free and useful for short program listings.

If using the CPLab LINUX this will appear on the laser printer in the CPLab, if using the remote connection via the Microlab, this will appear on the EUCS “lineprinter” in room 3305.

4.8 Manipulation files

Files are normally created with an editor `emacs`, or from the output of programs or applications. Other useful file manipulation commands are:

```
cp file1 file2    make a copy of file1, named file2
mv file1 file     move file1 to file2 (rename)
rm file1          remove, or delete file1
```

These utilities can also be used with directories in the filenames, in particular `mv` can be used to move files into a directory for example if `checkpoints` is the name of a directory, then

```
mv CheckPoint-1.java checkpoints
mv CheckPoint-2.java checkpoints
```

will move the two files, `CheckPoint-1.java` and `CheckPoint-2.java` into the directory `checkpoints`.

Warning: files that are deleted with `rm` are really gone. There is **no** WASTEBASKET to fish things “out-off”.

4.9 Wild-cards in file names

Wild-cards characters allow matching of a range of filenames. The two commonly used wild-cards are:

- * matches any string of characters
- ? matches any single character

so for example

```
rm *.class        removes all files with extension .class
ls CheckPoint-?.java lists all files called CheckPoint-<any>.java where <any> is
                    any character
```

4.10 E-Mail

You do not have a separate E-MAIL account on this system. You could continue to use your University SMS account which is available through MOZILLA.

man pages

All UNIX commands, most applications and most functions are all documented via the on-line system manual pages. These are very complete, usually being written by the actual programmer who implemented the command or application. As a result they usually assume a level of knowledge about the system well above that possessed by the average (never mind the novice) user. The quality, readability and length of `man` pages is very variable, however they do *usually* contain the correct information!

The `man` pages are accessed by,

- Type `man <command>` in the terminal window. For example to get the `man` page for `ls` you simply type `man ls`. This will output the manual page to the terminal window one page at a time with the same page commands as `more` (see above).

4.11 The Shell

When you type commands they are interpreted by the program called the `shell`. Under UNIX there are many different `shells` (*eight* I know of) all of which do the same basic task, but with subtle and annoying differences. All your accounts have been set-up to use `bash` (**b**ourne **a**gain **s**hell), which is used on about 50% of all UNIX system including all the EDINBURGH COMPUTER SERVICES (ECS) machines.

4.12 More Information

This section contains the “absolute minimum to get by” with UNIX. Extensive additional documentations are available:

1. CP Lab “on-line” HELP pages, link from *Scientific Programming Home Page*.
2. On-line via the UNIX HELP pages written and maintained by the EDINBURGH COMPUTING SERVICE (ECS). Link from *Scientific Programming Home Page*.
3. ECS documentation available at a modest cost from JOB RECEPTION IN ROOM 3210 or GEORGE SQUARE LIBRARY.
4. Any and varied books on UNIX in the library and for purchase in bookshops.

Note: There are many “flavours” and “dialects” of UNIX which are superficially similar but with subtle and annoying differences. These differences mainly show-up in the system management tools which most users never need or use. For each system the definitive documentation is located in the `man` pages!

What Next?

Now you have some practice of using the basic UNIX commands you should now read the next section on the `emacs` editor. You will need to use this to write your first JAVA program.