An Introduction to Unix

by Bernard Doyle

ver 4.2 (Web Systems) February 2015 Bernard.Doyle@uts.edu.au

Operating Systems

An Operating System is a program (or group of programs) running on a computer that performs 2 basic tasks :

- Management of the computer's resources (memory, hard drive, keyboard, mouse, cd-rom, etc.)
- Provide an interface for users to use the computer.

Common Computer Interfaces

There are two widely used interfaces for users to interact with computers :

- Graphical User Interface (GUI)
  This is a visual interface that uses a “point-and-click” i.e. windows, icon, mouse pointer environment.
- Command Line Interface (CLI)
  This is an interface that uses a keyboard for input and (typically) a 25x80 text display for output.

Examples of GUIs and CLIs

- GUI Interface
  Microsoft Windows, Mac OS/X, X-Windows.
- CLI Interface
  Windows DOS Box, MS-DOS, Cisco IOS CLI, Unix Command Line Interface.

Comparing the Unix and DOS/Windows Command Line Interfaces

- Windows CLI is very limited in scope. Users are encouraged to use Windows GUI for all tasks.
- Unix CLI is very rich. There is a large number of commands and an extensive built in help system (the man pages). Most commands have a large number of options adding to their versatility.
- Also, the Unix CLI allows users to chain together commands allowing the creation of quite powerful mini-programs on the command line.

The Unix CLI

In this workshop we are going to look at the command line interface that unix operating systems provide to users.

Unix Users and Accounts

- People who use a unix system are known as as “users”
- Users have usernames and passwords. The combination of a username and a password is an account. For example username:”fred” and password “HIcr0N!”
- Passwords should be cryptic but memorable.One way to create a password is to use the first letter of a phrase e.g. Here I come, ready or not!

Unix users and accounts the root user

- There is one special user “root” who can do anything on a unix machine
- You will never be root on the university’s machines
- You can be root on your own computer running unix at home.

Practical Exercise 1

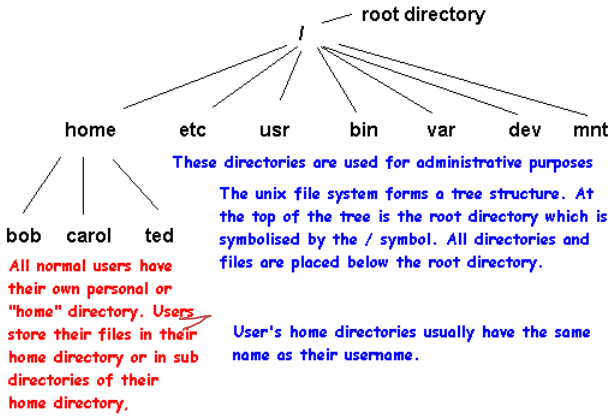
- Log on to the machine in front of you using the supplied user name and password.

Unix Directory Structure and Paths

- To specify the location of a file in windows we use terminology like:
  C:\Program Files\Office\word.exe
- In Unix we specify the location of a file in the following manner /home/fred/documents/job_applications/cv.txt
- Note that windows uses a backslash while unix uses a forward slash to separate directory and file names.
- The full specification of a file's location in unix is known as a pathname.

Unix Directory Structure (1)

A simplified picture of the unix directory structure



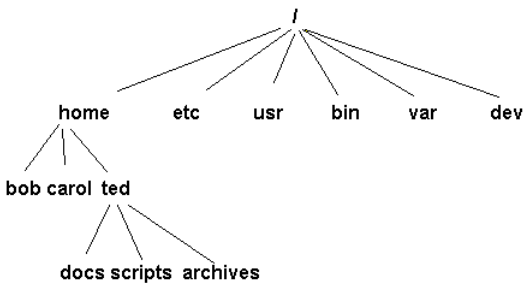
These directories are used for administrative purposes

The unix file system forms a tree structure. At the top of the tree is the root directory which is symbolised by the / symbol. All directories and files are placed below the root directory.

All normal users have their own personal or "home" directory. Users store their files in their home directory or in sub directories of their home directory.

User's home directories usually have the same name as their username.

Unix Directory Structure (2)



The diagram above shows a small portion of the directory structure for a unix computer. Directories are the unix equivalent of windows folders. In the diagram above the home directories of the three users bob, carol and ted are shown. Ted's home directory has 3 subdirectories - docs, scripts and archives.

Home Directories

- Every ordinary user (like you) has a home directory.
- The home directory and its subdirectories of the home directory is where users store their personal files.
- The home directory has the same name as the user. In most unix systems (including the ones at UTS) the user's home directory is a subdirectory of the /home directory

Specifying your home directory

A user's home directory can be specified in at least two ways:

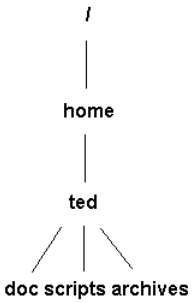
(1) By the absolute path to the directory

e.g. /home/fred

(2) By the using the tilde (~) character
(pron. "tild-duh"

i.e. ~ symbolises a users home directory

Unix directories, subdirectories and parent directories



The diagram at the left shows the file and directory structure for the user ted. The diagram also illustrates the fact that every directory (except for the root directory) is a subdirectory of another directory. Similarly, all files are contained in directories.

The term "parent" directory is used to describe the directory immediately above a file or directory.

A directory may have zero, one or many subdirectories and files below it. However, a directory or file can have only ONE directory above it. The directory immediately above a file or directory is known as the "parent directory" and it uses the .. symbol.

Using Directories

- Unix Directories are like Windows folders
- Directories can contain files and/or other directories.
- If you have an account on a computer running unix, you will almost certainly create subdirectories of your home directory to keep your files organised.

Interacting with Unix

- Users can interact with unix either through a Graphical User Interface (GUI) or through a CLI using a terminal or terminal emulator.
- Terminals interact with unix in text mode only. Input is through the keyboard only, output is in text through a 25x80 character display. (Sometimes the 25x80 can be increased).
- Terminal emulators (such as putty, terminal, konsole and xterm) run in a GUI but emulate a standard terminal.
- Almost everything we do in this practical will be through a terminal emulator program

Unix Commands options and arguments

Unix commands have a structure

command argument1 argument2 argument3 ...

There are two types of arguments:

(1) Command line options

(2) Ordinary arguments

Unix Commands options and arguments

Options begin with a - sign

e.g. `uname -a` — Command line option

Ordinary arguments don't begin with a - sign

e.g. `ls *txt` — Ordinary command line argument

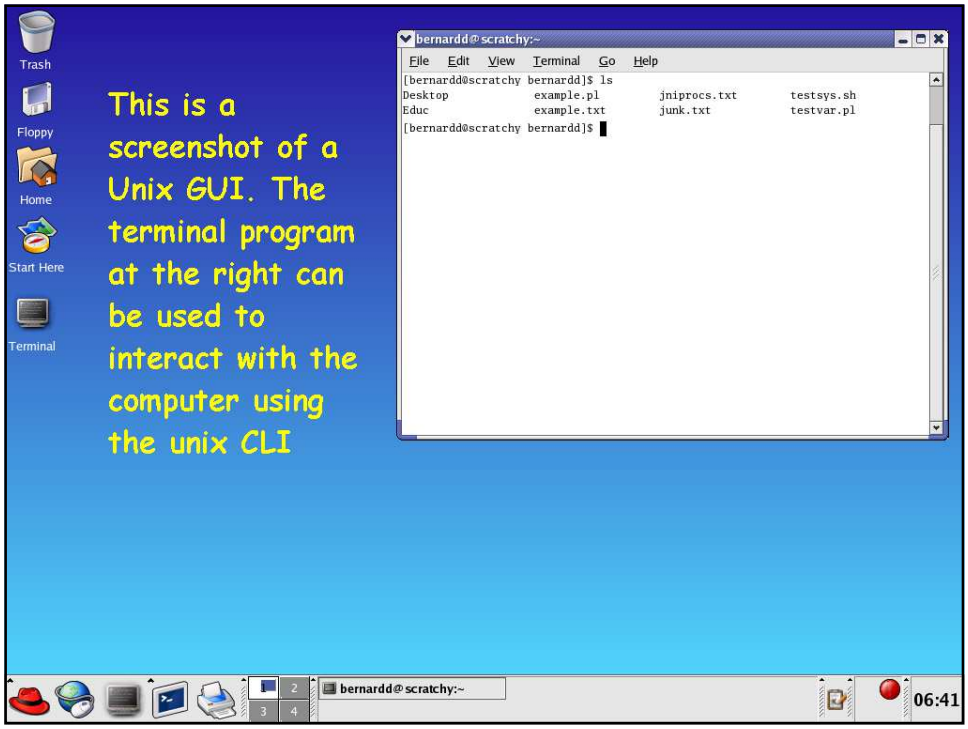
Options and ordinary arguments can both be used with the one command. Options always precede ordinary arguments.

e.g. `ls -a *pdf`

Depending on the command, more than one option and argument may be used. Options can sometimes be combined as well.

e.g. `ls -a -l *doc`

`ls -al *doc` — These two commands are the same



Getting Oriented in Unix (1)

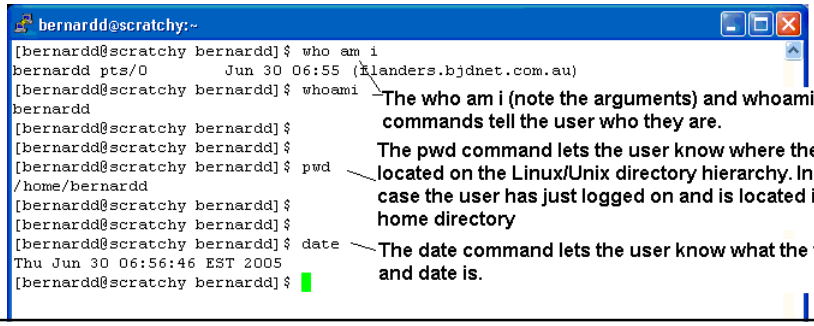
When we are logged onto a computer running unix we are always located somewhere in the unix directory tree. Usually when we first log in, we are located in our home directory. Our current location is our present working directory

Five commands are useful for orienting ourselves in a unix system. These are pwd, whoami, who, uname and date.

Action	Unix Command
Where are we ? i.e. what's our present working directory ?	`pwd`
Who are we ?	`whoami`
Who else is logged on ?	`who`
What variety of unix are we using?	`uname`
What is the date and time ?	`date`

Getting Oriented in Unix (2) an example session

The picture below shows the effect of running the whoami, pwd and date commands in a terminal session on a unix computer.



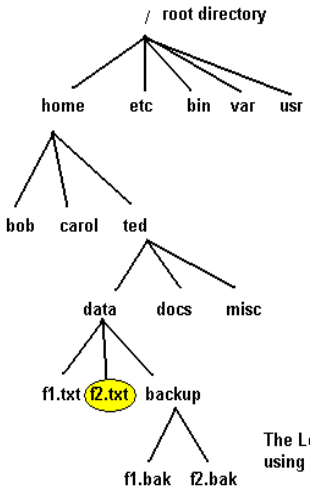
Practical Exercise 2

- Use the commands pwd, date, uname, whoami, and who to orient yourself and find out who else is on the computer you are logged into.

Some more unix commands

Command	What it does
`cd`	changes the working/current directory
`mkdir`	creates a directory
`ls`	lists the contents of a directory
`cp`	copies a file.
`rm`	removes a file (be careful using this)
`mv`	renames a file (equivalent to copy a file and delete the original)
`rmdir`	removes an empty directory (be careful using this)

Navigating Directories – Absolute and Relative Paths



In this diagram of part of the file and directory structure of a computer running unix, we see that the user ted has 3 subdirectories. In the subdirectory data, he has 2 files and another subdirectory called backup.

The location of files and directories can be specified using absolute and relative pathnames. Absolute pathnames specify the location of a file relative to the root directory.

The absolute location of f2.txt is /home/ted/data/f2.txt

From the misc directory the relative location of f2.txt is ../data/f2.txt

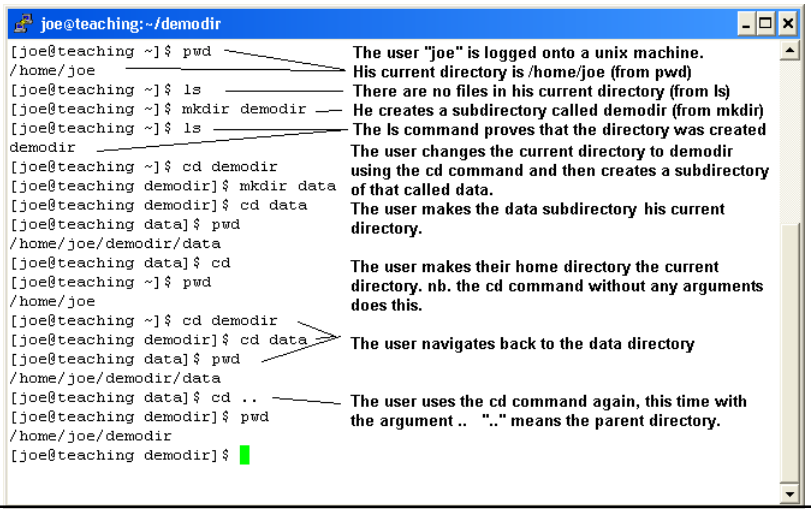
The .. notation means the parent directory of the working directory.

The Location of a file or directory can be specified using an absolute or relative path.

Relative and Absolute Paths

- An absolute path starts at the root directory and navigates through the directory structure to a particular file or directory. An absolute path always leads to the same location.
- A relative path starts from where the user is currently located i.e. their present working directory. A relative path can lead to different locations depending on where the user is located.

Creating directories and navigating through them



Copying Files

Files are copied in unix using the cp command

example :

To make a copy of the file f1.txt called f1.bak the command is :

```
cp f1.txt f1.bak
```

The syntax of cp is

```
cp original_file_name destination_file_name
```

Getting Help
the man command

There is an online help facility in all unix installations. This is accessed using the man command and the data the command provides are known as the “man pages”

As an example, to access the man pages for the ls command type in `man ls`

Other sources of help:

A brief summary of a command can often be obtained by typing in the name of the command with the argument `--help`

Some unix systems (such as linux) have an extra help system known as “info”. This can be accessed by typing in info followed by the name of the command.

Some Tips

- To clear the screen, press the control and L keys simultaneously.
- Previously executed commands can be accessed by pressing the up arrow key on the keyboard.
- Tab completion enables users to key in long and confusing filenames with a minimum of keystrokes.

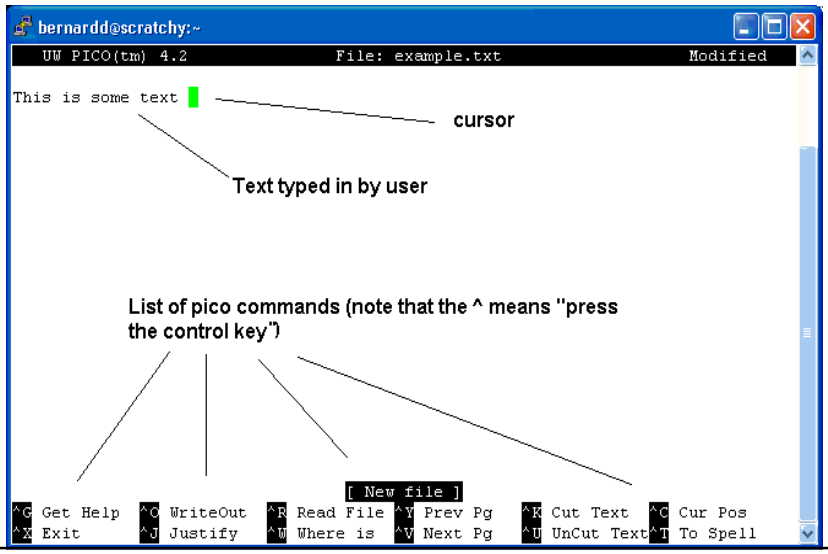
Editors

Editors are an important tool that enable users to create and alter text files. Unix systems often have a number of editors:

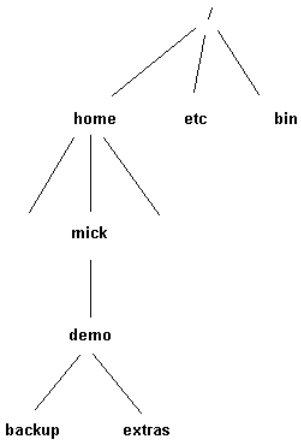
- nano (sometimes called pico) – easy to use editor. (The same editor with 2 different names).
- vi – powerful and installed on most unix systems, but it has a steep learning curve. Learning vi is compulsory in some UTS courses.
- emacs – powerful and usually installed on most unix systems, reasonably steep learning curve.

Each of these editors can be started by typing in the name of the editor followed by the name of the file to be edited.

The nano/pico interface



Practical Exercise 3



Directory structure for user "mick"

(1) Create a subdirectory called "demo" in your home directory. Make demo your current directory (i.e. your working directory) and create subdirectories of demo called backup and extras.

(2) Using pico, create a a file called f1.txt in the demo directory.

(3) Copy f1.txt to the subdirectory backup under the same name. Then copy f1.txt to the subdirectory backup but this time name the copy f1a.txt Copy f1.txt again to backup this time naming the copy in backup f1.asc

(4) While demo is still your current directory, copy f1a.txt from backup to your home directory.

(5) Make backup your current directory and copy f1.txt from backup to extras, renaming the copy in extras f1_extras.txt

(6) Make demo your current directory. Delete all files in backup that end in "txt"

Answer to Practical Exercise

mick@ulysses:~

```
[mick@ulysses ~]$ pwd
/home/mick
[mick@ulysses ~]$ mkdir demo
[mick@ulysses ~]$ cd demo
[mick@ulysses demo]$pwd
/home/mick/demo
[mick@ulysses demo]$ ls
[mick@ulysses demo]$ mkdir backup
[mick@ulysses demo]$ mkdir extras
[mick@ulysses demo]$ ls
backup extras
[mick@ulysses demo]$ pico f1.txt
[mick@ulysses demo]$ ls
f1.txt backup extras
[mick@ulysses demo]$cp f1.txt backup
[mick@ulysses demo]$cp f1.txt backup/f1a.txt
[mick@ulysses demo]$cp f1.txt backup/f1.asc
[mick@ulysses demo]$cp backup/f1a.txt ../
[mick@ulysses demo]$cd backup
[mick@ulysses backup]$ pwd
/home/mick/demo/backup
[mick@ulysses backup]$ cp f1.txt ../extras/f1_extras.txt
[mick@ulysses backup]$ cd demo
[mick@ulysses demo]$ ls backup
f1.asc  f1.txt  f1a.txt
[mick@ulysses demo]$ rm backup/*txt
[mick@ulysses demo]$ ls backup
f1.asc
[mick@ulysses demo]$
```

User "mick" logs in and is located in his home directory

Subdirectory demo is created and user makes demo his current directory

User creates subdirectories backup and extras

User creates a text file called f1.txt using pico

User copies f1.txt to backup under the same name, also copies it and renames it f1a.txt and f1.asc

User copies f1a.txt from backup to home directory

User makes backup his current directory

User copies f1.txt to extras directory and renames it f1_extras.txt

User makes demo the current directory and gets a listing of the files in the backup directory

User removes all files in the backup directory ending in "txt"

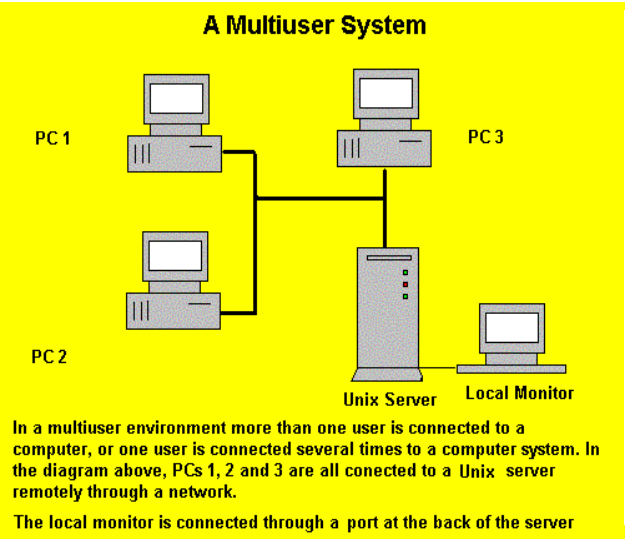
User checks that all files ending in "txt" have been removed.

Unix File Details

Unix Files have the following attributes.

- Filename – This can be any string of characters you like, but must be unique in the directory it is in.
- Filetype – this usually an ordinary file or a directory but other types are possible.
- Creation and last access dates
- Size – the size of the file in bytes
- Owner – a user who owns the file
- Group – what group is associated with this file
- Permissions – we will explain this further on.

Unix is a multiuser operating system



Because unix is a multiuser system more than one person can have access to the system this means that the security and privacy of files and directories becomes an issue that must be addressed.

Unix File Ownership Concepts

Because unix is a multiuser operating system security issues are relevant. As part of its inbuilt security with respect to files, unix divides users into three groups. These are:

- File's owner
- File's group
- The rest of the world

Unix File Permission Concepts

Another aspect of unix's inbuilt security is the concept of file permissions. There are three types of file permissions:

- Read permission
- Write permission
- Execute permission

File Permission Concepts (2)

Permission	Description
Read	If granted, allows a file or directory to be read
Write	If granted, allows a file to be altered i.e. written to
Execute	If granted, allows a file to be run as a program i.e. executed

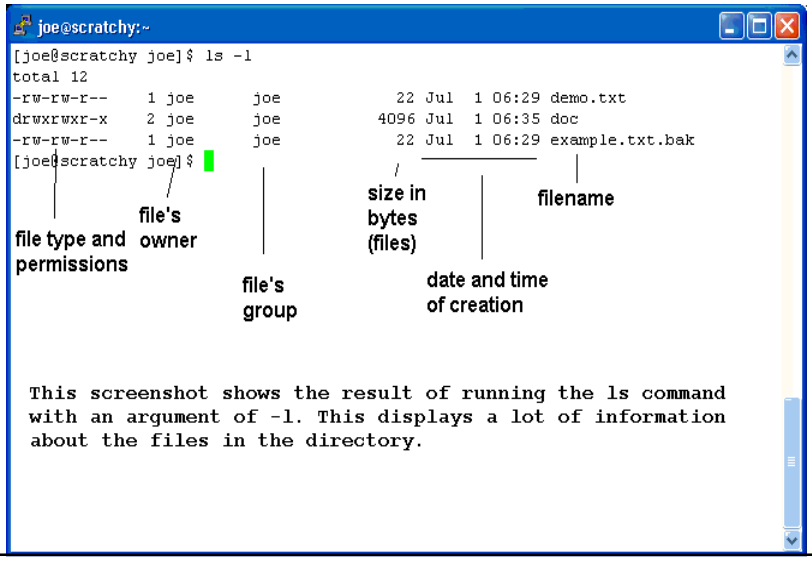
Ownership and Permissions

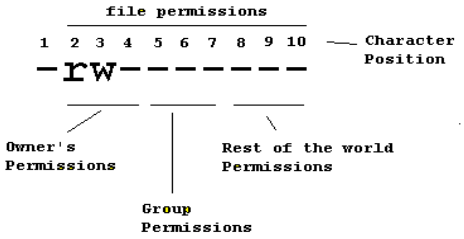
We have seen that there are three classes of user with respect to files and three types of permissions.

Each one of the 3 classes of user has a setting for the three types of permissions.

3 classes of user x 3 types of permission = 9 permissions

Finding out more about files





Unix file permissions as displayed by the `ls -l` command occupy the leftmost column of the output of the command. The character at position 1 is the file type (- for a normal file, d for a directory).

The remaining 9 characters show the permissions of the 3 sets of users. A - means the permission is not granted, an r means read permission is granted, a w means write permission is granted, an x means execute permission is granted.

In the example above, `-rw-------` means that we are looking at a file where the owner has read and write permission, but not execute permission. Neither the owner's group nor the rest of the world has any permissions at all (as shown by the - in their respective positions)

drwxr-xr-x

The example permission set immediately above shows that the owner has read, write and execute permission. The group has read and execute permission and the rest of the world has read and execute permission. The fact that the leftmost letter is a d shows that we are dealing with a directory rather than a normal file.

Changing File Permissions using the chmod command

Permission	Score
Read	4
Write	2
Execute	1

```
chmod  nnn  file_or_directory_name
  /     |
command |
        3 digit number setting permissions
```

Each of the digits of the three digit number setting permissions has a value between 0 and 7. The first digit sets the owner's permissions, the second the group permission and the third sets the rest of the world permission.
Each digit is derived by starting with zero and adding in the score for each permission as set out in the table above. So, to set read and execute permission the digit will be 4+1 = 5. To set read and write permission, the digit will be 4+2 = 6. Each number between 0 and 7 specifies a unique combination of permissions. 0 means no permissions, 7 means all.

Numbers and permissions

Value	Permission	Explanation
0	No Permissions	no numbers, no permission
1	Execute Permission	1 = execute permission
2	Write Permission	2 = write permission
3	Write+Execute Permission	3 = 2+1 = write+execute
4	Read Permission	4 = read permission
5	Read and Execute Permission	5 = 4+1 = read + execute permission
6	Read and Write Permission	6 = 4+2 = read + write permission
7	Read, Write and Execute Permission	7 = 4 + 2 + 1 = read+ write + execute permission

An alternative way to use chmod

The previous slides outlined the use of numbers as arguments to the chmod command to change file and directory permissions.

The chmod command also has an alternative mode where it uses letters and the + and – symbols to change permissions. If you are more comfortable using that mode, by all means do so.

Practical Exercise 4

- Create a directory called permissions_demo
- Make permissions_ demo your present working directory.
- Use nano or pico to create a file called p.txt
- Make 4 copies of p.txt called p1.txt, p2.txt, p3.txt and p4.txt.
- Set the permissions of the files as set out below
- Check the permissions using the appropriate command ie. `ls -l`

File	Owner	Group	Rest
p.txt	**rwx**	**---**	**r--**
p1.txt	**-w-**	**r--**	**rwx**
p2.txt	**r-x**	**rw-**	**-wx**
p3.txt	**--x**	**--x**	**rw-**
p4.txt	**-wx**	**rwx**	**---**

(1) Create a subdirectory in your home directory called public_html

(2) Make public_html your present working directory

(3) In public_html create a file called index.html

```
<html>
<head>
<title>My first web page</title>
</head>
<body>
Hello World!<br>
This is my first web page<br>
</body>
</html>
```

(4) Set the permissions on public_html and on your home directory to owner can read write and execute, group can read and execute and rest of world can read and execute.

(5) Set the permissions of index.html to owner can read and write, group can read, rest of world can read. Also chmod 755 ~

(6) Start up a browser and point it at

`http://rerun.it.uts.edu.au/~myusername/index.html`

Summary of Unix Commands Learnt

whoami	rmdir
uname	cp
date	mv
pwd	chmod
ls	nano
cd	vi/vim
mkdir	rm

Summary of Concepts

Command Line interface	Graphical user interface
file	directory
Unix file system hierarchy	Present working directory
Home directory	Command options
Command arguments and options	Owner, group, rest of world
Pathnames	Absolute path
Relative path	Permissions – read, write and execute

Useful practical tips

Command Line tab completion	Command line history
command --help gives a quick and dirty summary of command options (usually)	Cntrl-l to clear a terminal screen
Cntrl-c to kill a command that has taken over the screen	Cntrl-d to log off

Where to from here ?

There are a number of things you can do:

- Get your own unix system (Linux, FreeBSD, NetBSD, OpenBSD, Solaris, minix) *
- Buy a Macintosh – the operating system is a variety of Unix
- To get your own version of unix, download linux/freebsd iso files from the internet. A good place to start is http://distrowatch.com
- The easiest option is to use virtualbox or VMWare. These can also be downloaded from the net. A more difficult approach is to burn the iso files to a DVD and then installing onto a physical computer (partitioning may be necessary).
- Use the computers at UTS running unix systems (Solaris and Linux)

* You can run linux or freebsd on a windows system by using virtualization software like VirtualBox or VMWare