UNIX 1: Introduction to UNIX

Workbook

Edition 5
October 2001

UNIX 1:

Introduction to UNIX

Edition 5,October 2001

EUCS Document Number: F.5.WB-3184-10/2001

ABOUT THIS WORKBOOK

- One topic to a double-page
  - summary in box
  - concepts on left hand page
  - practical instructions on right hand page
- Different fonts have different meanings

This workbook uses certain conventions of layout.

One topic to a double-page

The information in this workbook is presented as a series of topics, each topic usually presented on a double-page spread with a slide and explanatory text on the left hand page, and practical material on the right hand page. If there are no practical exercises for a topic, the right hand page can be used for another topic, or it may be left blank.

It should be possible to skip topics, because each page is as self-contained as possible, however later pages do assume that you are familiar with the contents of earlier topics.

The practical instructions assume that you are using software that behaves as it does in the EUCS training labs. If you are not using this book in a training lab, some of the instructions may not work as expected.

Different fonts have different meanings

The following typographical conventions are used in this workbook:

Things that you have to type appear in	**`this font`**
Things that appear on the screen are in	`this font`
Places where you must enter a specific value are in	*this font*
Names of keys are enclosed in square brackets	[Space]
The control key is represented by '^', so holding the control key down and pressing 'C' is represented by	^C

➤ *Tips showing different ways of doing things look like this*

PRACTICAL EXERCISES

Practical exercises are always presented on a right hand page. They help you to understand and investigate the information given on the facing left hand page.

This is the title of an exercise

- ❑ Do this step
- ❑ Do this step

  It will have these effects

Title for another task

This technique will allow you to carry out this useful function.

- ❑ Do this step, and this, and this.
- ❑ Do this step

  It will have these effects

➤ *You can also use this short-cut*

ABOUT THE COURSE

- Prerequisites for the course
  - Introduction to Unix

- Learning goals
  - by the end of the course you should be able to
  - carry out simple Unix commands
  - use the Unix filesystem
  - be able to use bash shortcuts

Prerequisites for the course

You need no experience of unix before using this workbook.

Learning goals

When you have completed this book, you should feel comfortable using unix.

You should understand how to create, copy, print, and destroy files.

You should be able to access files created by other people.

Chapter 1

Using Unix

WHAT IS UNIX?

- Unix is an operating system
  - interface between user and computer
  - start programs (electronic mail, spreadsheet, database)
  - send results to printer

- Compare with Windows, System 7 and 8
  - Unix is more powerful, more complex
  - Unix has excellent network facilities

Unix is an operating system. This means that its function is to enable you to use the computer. Without an operating system human beings are not able to interact with computers.

Perhaps the most important task the operating system will carry out for you is to enable you to start programs running. This means for example you can start an electronic mail program, a database program, or a spreadsheet program. The operating system then enables you to print the results.

Unix and microcomputers

Windows is the operating system used by IBM PCs and clones; System 7 or 8 is the operating system used byApple Macintosh computers. Unix is the equivalent of Windows or System 7, but it is much more powerful.

UNIX IS WIDELY USED

- SUN workstations, Hewlett-Packard, many microcomputers
- Unix at Edinburgh - e.g., holyrood
  - many departments
- Network capabilities built into Unix
  - this city
  - this country
  - other side of the world

Unix-based workstations are produced by many manufacturers (SUN, Hewlett-Packard, Digital etc). Unix is also available for many microcomputers.

Unix at Edinburgh

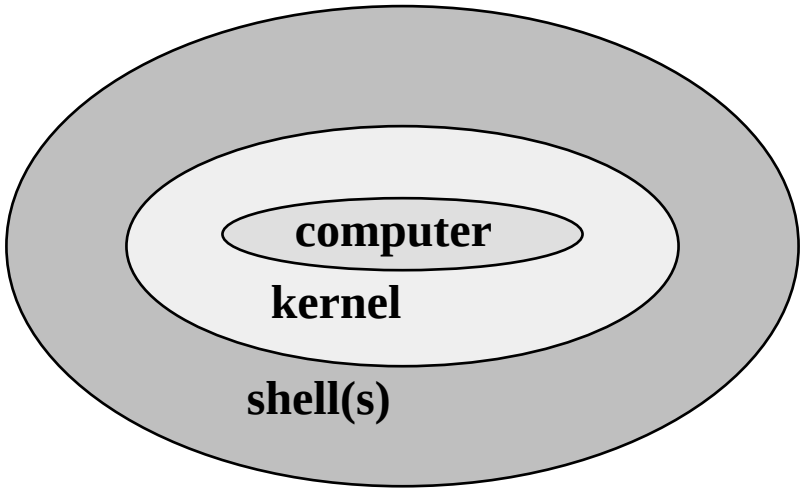
Unix is the operating system for mainframe computers currently in use at Edinburgh such as holyrood.

Many departments at Edinburgh have networks of Unix workstations.

Easy access to remote computers

Networking capabilities are built into Unix, making it very suitable for use with either large or small networks.

SHELL : UNIX'S USER INTERFACE



The part of the Unix operating system which interacts with the user is called the 'shell'. The part which interacts with the computer is called the 'kernel'. The names were coined by analogy with a nut, where the kernel is on the inside, and the shell is on the outside.

Different shells are available, providing a slightly different user interface. This course covers a shell called 'bash'.

Extra information

Several shells have been developed and used widely. These include csh (the 'C' shell), Tcsh (the 'T' 'C' shell, an extension of the 'C' shell) and sh (the Bourne shell).

bash contains features from the C shells and the Bourne shell, and its name stands for 'Bourne-again shell'!

UNIX AND WINDOW SYSTEMS

- Most microcomputers have window-based interfaces
  - Microsoft Windows or Apple System 8
- X Window system is used with Unix workstations
- This course
  - suitable for X window system terminal
  - terminal with no window system

Most microcomputers have window systems (e.g., Microsoft Windows NT for PCs, and a built in window system for Apple computers). More and more Unix computers are using the X window system.

This course

The material in this workbook is suitable both for computers which are using the X window system, and for computers which are not using a window system.

This workbook does not teach usage of the X window system.

MULTI-USER SYSTEM

- Single computer
  - many people using it at the same time
- User name
  - identify yourself to the computer
  - computer can bill you for use of printers, etc!
- Password
  - secret to yourself

The Unix operating system enables a single computer to be used by many people at the same time. Each person is given a user name, and a password.

User name

Your user name is your way of identifying yourself to the computer. Many user names are based on the user's real name (e.g. jeanr for Jean Ritchie). Electronic mail can be sent to your user name; charges for computer usage and for things like printer paper are also made to your user account.

You are also allocated space on the computer's storage disks and your account is your way of gaining access to this.

Password

You will be given a password when you are given your user name. You must keep the password secret to yourself, and change it as soon as possible so that nobody but yourself can know what it is.

Logging in

Before using the computer you must quote your user name and password; this is called 'logging in'.

LOGGING IN

- Before you can use the computer
  - quote your user name
  - quote your password

- Unix is case sensitive
  - capital letters are different from lower case letters!

Before using the computer you must quote your user name and password. Depending on the type of terminal available to you, you will need to use slightly different methods of carrying out this ‘logging in’ process.

The login: prompt

You will be given instructions showing you how to make the computer give you a ‘login prompt’. The login prompt usually takes the form ‘login:’ and means that the computer is ready for use. You respond to the ‘login’ prompt by typing in your user name and then pressing the Return key.

You must type both your user name and password exactly as you were given it, with capital and lower case letters.

The password: prompt

The computer responds with the prompt ‘password:’; your password will not be shown on screen as you type, so if somebody is looking over your shoulder as you type, they cannot learn your password.

The user prompt. You will now see the ‘user prompt’, usually either `holyrood$` or `%`, meaning that the computer is ready for you to start typing in commands.

PRACTICAL EXERCISES

Logging In

- ❑ Obtain the login prompt

  It looks like this

  `login:`

- ❑ Type in your user name exactly as you were given it, using upper and lower case letters

  Press the Return key

- ❑ Type in your password exactly as you were given it

  The symbols will not appear on the screen

  Press the Return key

- ❑ You should now see the user prompt

  `holyrood $` or `%`

  The computer is ready for use

- ❑ Type in commands, then press Return key to activate them
- ❑ If you made a mistake typing either the user account name, or the password, you will receive a message saying `Login incorrect`, and the 'login' prompt will reappear. Try again!

CHOOSING A PASSWORD

- First defence on your filespace
  - change your password regularly
- Choose something nobody can guess
  - include non-alphanumeric characters
  - avoid dictionary words or proper names
  - avoid car registrations and dates of birth
  - not the word "password"
- DO NOT FORGET YOUR PASSWORD
  - you will have to prove your identity to have it reset

Changing your password

Your password is the most basic defence against other people accessing your user account. It is good practice to change it regularly and not to write it down. Even if you do not feel you have data which needs protection, your password helps protect the system and therefore other people's accounts.

Choosing a password

Your password should be at least six characters long. If it contains at least one non-alphanumeric character such as a dot or exclamation mark it will be more secure.

Someone attempting to guess your password will commonly try words of significance to you, such as the names of your friends or your car registration. They may also have a password "cracking" program which can try out every word in the dictionary. This is why you should avoid using any ordinary word. Spectacularly poor choices include the word "password" and your user name.

Some UNIX systems run their own password cracking program to check security. If your password can be guessed by this program, the system administrator will ask you to change it.

Forgetting your password

If you forget your password you will have to ask the system administrator to reset it for you and this will usually involve proving who you are. Avoid this by choosing a memorable password.

PRACTICAL EXERCISES

Changing your password

This is a fairly straightforward process. If you try it now, be sure to remember what you change your password to be.

- ❑ Change your password:

  ```
  holyrood$ passwd
  ```

  The computer will ask you for your current password.

- ❑ Type your current password.

  The computer will now ask you for a new password.

- ❑ Type the new password.

  You will be asked to type it again. This is a check as you will not see the password echoed to the screen as you type it.

- ❑ Type the new password again.

  If the two versions do not match, the password will not be changed.

  Otherwise your password will be changed. You will have to use the new password next time you log in.

LOGGING OUT!

- ALWAYS log out before leaving the terminal
  - otherwise other people can tamper with your account

- Particularly important in public areas
  - best not to leave account open even for a short time
  - antisocial as other people cannot use the equipment
  - security risk

When you have finished using the computer, you should ‘log out’ of the computer. The logging out process tells the computer that you are leaving the terminal, and any further keyboard input will not be made by you! The computer therefore ‘closes down’ your account, and further access to it will require the password to be quoted once again.

Security

It is important to log out every time you leave the computer. If you walk away from your terminal while you are still logged in, this means that anybody else could simply sit down at the terminal and would have access to your files. Most people at this point would be charitable and log you off, but you cannot guarantee that they would not either read sensitive information, destroy some material, or in the worst case gain unauthorised access to other people’s material. Skilful hackers require only an unattended terminal to cause great damage!

How to log out

The logging out process is simple; the command is `exit`. If you still have programs running, you may receive the message `stopped jobs`. You can either do something about these (you will find out how later in the course), or, if you type exit again, you will be logged out, and the ‘stopped jobs’ will be terminated.

PRACTICAL EXERCISES

Logging Out

- ❑ Prepare to leave the computer (for a coffee break perhaps)

  Type:

  ```
  holyrood$ exit
  ```

  The `holyrood$` prompt will disappear.

- ❑ If you want to use the computer again, you must log in giving your password.

  Always log out if you are leaving the terminal, even for a short time!

COMPUTER FILES

- Collections of information
  - text files
  - data files
  - configuration files
  - hidden files
- Stored in directories
  - automatically log in to your own directory

All computers store information in ‘files’.

Text files

Computer files containing ordinary text information (such as reports, letters, etc) are called ‘text files’ and their contents can be displayed on the computer screen, where you can read them. You, the user, can create text files for your own use, and destroy them when you have finished with them.

Data files

Data files contain information which can only be read by the program which created them, or by other programs which have been designed to handle their particular type of data. You, the user, may create data files during the course of your work with certain programs (e.g. with desktop publishing packages, database packages, spreadsheets, or graphics packages). Data files associated with particular programs often have names similar to the program name.

Configuration files

Configuration files are usually text files containing information about things like fonts, colours, mouse behaviour, etc. There are often separate configuration files for different applications, and they are given names which link them to the appropriate applications.

FILE NAMES

- Filenames can have special endings
  - Text files sometimes end in '.txt'
  - C program files end in '.c'
- Names of hidden files start with a dot
  - Configuration files are often hidden files

On some computer systems, text file names end in '.txt'. Unix does not expect special endings to file names, but some programs expect to use files with particular extensions. If you program in Fortran 90, the compiler expects your program files to have names which end in .f90, and compiled versions to have names ending in .o, and so on. Some packages automatically create files with names which have special endings.

Hidden files

Configuration files in Unix usually have names which start with a dot (.login, .profile etc), and their names may end in 'rc' (.bashrc, .xinitrc, etc).

If a file name begins with a dot and ends with 'rc' this means it contains setup information for a package or program; its name will probably tell you which program it is associated with. The letters 'rc' stand for 'Run Commands'.

MANIPULATING FILES

- ls, more, cat, cp, mv, rm
  - list, examine contents, copy, move, remove
- Unix is case sensitive
  - Capital letters, lower case letters
- Capital letters
  - combination of shift key and ordinary key
- LS is different from ls

Unix provides several commands for manipulating files (showing you a list of your files, looking at their contents, changing their names etc). This workbook introduces the following commands:

```
ls, more, cat, cp, mv, rm
```

and includes practical instructions to help you learn about them.

Unix is case sensitive

Unix treats the upper and lower case version of a letter as being different from each other. Thus ‘LS’ is quite different from ‘ls’.

When you type Unix commands, you must use the correct case at all times.

When commands don’t work

If at any time a command does not work as you expect, one of the first things to check is whether you were using the right case when you typed the command!

PRACTICAL EXERCISES

Using commands

Try using the same command (ls) in upper and in lower case.

Always press Return after typing in a command. It will be executed only after you press Return.

Upper case

- type **`LS`**

  you will get a message saying

  `'LS: command not found'`

Lower case

- type **`ls`**

  you will get a list of files. This is because the command 'ls' is always spelt in lower case.

UNIX COMMANDS

- Arguments
  - usually required
  - one or more filenames

    `ls fred bert`
- Options
  - useful but not essential
  - several options can be used at the same time
  - details of how the command should operate

    `ls -Fl` *filenames*

All Unix commands are used in a similar way. First of all you type in the name of the command. This is all that is needed in order for some commands to work.

Arguments

Other commands need to be told where to find data, or need to be given the names of the files on which the command is to operate. This information is called the 'argument' to the command.

Options

Most commands can be made to operate in a different way by typing in 'options' as part of the command line. Options are usually single letters, and are prefixed by a minus sign '-'.

The only complication with options is that some commands allow several options to be specified after one minus sign (-aux) while a few commands require each option to be given its own minus sign (-a -u -x). The first form is the most common, but if a command refuses to perform properly, try the second form!

Further information on each command

Information about the exact details and options available for each command are provided in an on-line manual. Instructions for using this will be presented later on.

PRACTICAL EXERCISES

ls

- Try the command called ‘`ls`’

  Type

  ```
  holyrood$ ls
  ```

  This command does not need any arguments in order to work

  You will see a list of all the files in your directory

- Try the option `-l` (this is the letter l not the number 1)

  Type

  ```
  holyrood$ ls -l
  ```

  this time you will see more information about each file

- Try giving the name of one of the files as an argument to `ls`, as well as using the option `-l`.

  Type

  ```
  holyrood$ ls -l fred
  ```

  This time you will see detailed information of one file only, fred.

UNIX IS FLEXIBLE

- Commands can have
  - none, one, or several arguments
  - none, one or several options
- Shell interprets input
  - runs the command
  - sorts out whether input includes arguments & options
- Using commands
  - Command [list of options] [another list of options] [list of arguments]

Most commands will accept either one argument, or several, and either one option or several.

The shell interprets input

The shell interprets input, and runs the commands having sorted out which part of the input is the command, which is an argument or an option, and how many arguments and options are present.

Using commands

In the following pages, some extra information about commands will be provided. This will start with a line in the format

```
command [list of options] [list of arguments]
```

The sets of square brackets will contain a list of options, indicating that you are free to use any, all or none of the options listed! Similarly you are free to use any all or none of the arguments listed. If an argument or option is shown NOT in a square bracket, the command will not work without it.

Two dots after an argument which itself is not in square brackets indicate that you must have at least one argument, but you could also have more than one!

LS (LIST FILES)

- List of files in the current directory
  - alphabetic order by default
  - options available
- Most commonly used options
  - `-a` show 'dot' files
  - `-l` show details about files
  - `-F` show type of file
- Several options at the same time
  - `ls -al`

The command 'ls' tells the computer to show you a list of the files in your current directory. You can specify how you want the list to appear, and what files to include in the list.

ls -l

You can ask for the list to give more information about each file by using the '-l' option. The details include file type, access permissions, number of links to the file, the file's owner, its size (given in bytes), the date it was last modified, and finally its name! Links and access permissions will be discussed in a later workbook.

ls -a

The '-a' option asks for a list of all the files in a directory, including files which are normally hidden from you. These files have names which begin with a dot (such as .profile) and contain configuration information.

ls -F

This option marks directories with a trailing slash, executable files with an asterisk, and symbolic links with a trailing at-sign (@).

ls -alF

Very often you may wish to use several options at the same time.

FILENAMES AND DIRECTORIES

✸ Use ls to list directory contents or file details

- ls lists files in the current directory if no filenames are given.

If the filename given is itself a directory, ls lists the files inside it. If ‘filename’ is the name of a file, ls repeats its name together with any other information requested. By default, the list is sorted alphabetically.

If no filename is specified, files in the current directory are listed. If several filenames are given, they are sorted appropriately

PRACTICAL EXERCISES

More ls

- List the files in your directory

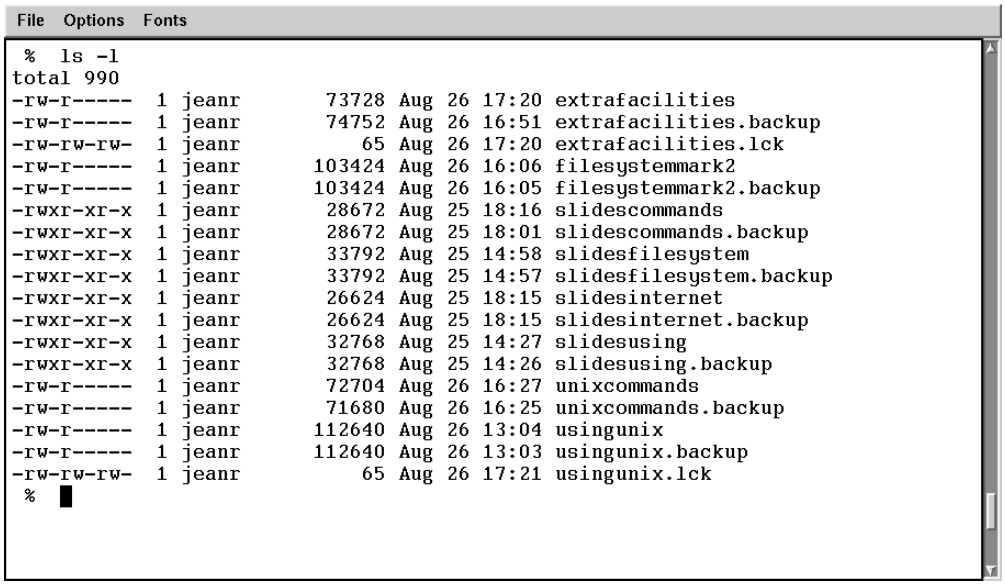
  Type

  ```
  holyrood$ ls
  ```

- Get more details about the files

  Type

  ```
  holyrood$ ls -l
  ```



- Get a list of all files, including those whose names begin with a dot

  Type

  ```
  holyrood$ ls -a
  ```

- Get a list of all files, showing more details

  Type

  ```
  holyrood$ ls -al
  ```

- Get a list of files, marking directory files with a trailing slash

  Type

  ```
  holyrood$ ls -F
  ```

MORE

- A ‘pager’
  - program which shows information a page at a time

- Simple operation
  - press ‘spacebar’ to see next screen
  - press ‘return’ to see next line
  - press ‘q’ to stop

The command ‘more’ is what is known as a ‘pager’ program: in other words it can show you the contents of a text file a screenful at a time. This is useful where the file is a long one. ‘more’ normally pauses after each screenful, and prints the message ‘--More--’ at the bottom of the screen. A two-line overlap is given between screens for continuity.

The percentage of characters displayed so far is shown at the bottom of each screenful.

Commands

There are a few options which you can use when you start ‘more’ running; once you have started ‘more’, commands are available to control its operation.

Next screen

When you have finished looking at the first screenful, you can move on to the next one by pressing the spacebar.

Next line

You can move on to the next line by pressing the ‘return’ key.

Stop

You can stop looking at the file at any time, by typing ‘q’. You will be returned to the command prompt.

MODIFYING MORE

- Use options to modify the action of more
  - Clear the screen before displaying
  - Replace multiple blank lines with a single one

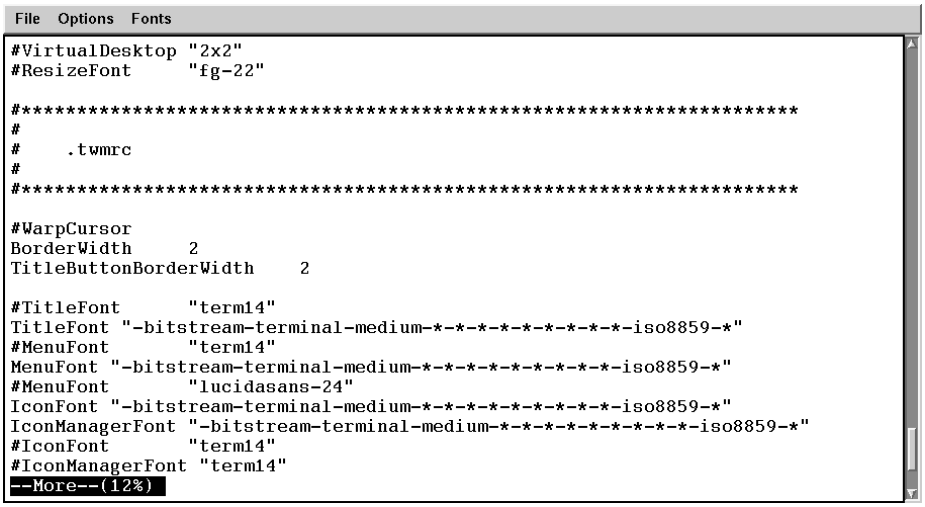
The following options modify the behaviour of more

```
more [-cs] [-lines] filename...
```

-c Clear before displaying - i.e. redraw the screen instead of scrolling.

-s Squeeze. Replace multiple blank lines with a single blank line.

MODIFYING MORE (CONTD.)



Commands

SPACE Display another screenful.

RETURN Display another line.

b Skip back a screenful and then print the screenful.

q, Q Exit from more.

/pattern Search for the next occurrence of the regular expression ‘pattern’.Display a screenful starting two lines prior to the line that contains the match. If there is no such match, the position in the file remains unchanged.

PRACTICAL EXERCISES

Using more

- ❑ Use more to examine the contents of the file called ‘fred’

  Type

  ```
  holyrood$ more fred
  ```

- ❑ Move down a screenful

  Press the spacebar

- ❑ Move back a screenful

  Type **b**

- ❑ Move down one line

  Press the ‘return’ key

- ❑ Leave ‘more’

  Type **q** or **Q**

CAT

- ‘concatenate and display’
  - join a list of files
  - show the joined list on screen

- Redirecting output
  - specify where output of a command is to go
  - send output to a file rather than to the screen

```
cat daisy kate > fred
```

The command ‘cat’ stands for ‘concatenate and display’. This command will join together a list of files, and show the joined files on screen. If only one file is specified, the cat command will display the contents of the file.

cat to another file: redirecting output

Unix allows you to specify where you want the output of a command to appear. Most commands by default direct their results to the screen, as ‘cat’ does. However you can use the symbol ‘>’ to tell Unix to send the output elsewhere.

```
cat daisy kate > fred
```

concatenates the two files daisy and kate, and stores the result in a file called fred. If a file called fred already exists, its contents will be overwritten.

Append not overwrite >>

If you use a pair of right arrow symbols, ‘>>’, information will be added onto an existing file, rather than overwriting it.

Extra information

cat [-nvt] [filename...]

-n Precede each line with its line number.

-vt Display non-printing characters, and in addition display TAB characters as ^I (CTRL-I).

Beware

Beware of ‘cat a b > a’ and ‘cat a b > b’, which destroy the input files (a and b respectively) before reading them.

PRACTICAL EXERCISES

Using cat

- View the contents of the two files called ‘daisy’ and ‘kate’

```
holyrood$ more daisy
holyrood$ more kate
```

- Join the two files called ‘daisy’ and ‘kate’

```
holyrood$ cat daisy kate
```

  The result of the join appears on screen

- Join the two files called ‘daisy’ and ‘kate’ and store the results in a third file called ‘fred’

```
holyrood$ cat daisy kate > fred
```

- Look at the contents of ‘fred’

  type one of the following

```
holyrood$ cat fred
holyrood$ more fred
```

- Append the contents of ‘daisy’ to the file called ‘fred’

```
holyrood$ cat daisy >> fred
```

- Look again at the contents of ‘fred’

COPYING FILES - 'CP' AND 'MV'

- Make a copy of a file
  - two identical versions of the same file

- Renaming files - 'mv'
  - Moving a file to another space in the filesystem
  - only one version of the file
  - version with the old name no longer exists

The command 'cp' is used to make a copy of a file. After this command has been used, there will be two identical versions of the same file.

Renaming files - 'mv'

'mv' is short for 'move', and after this command has been used there is still only one copy of any particular file, but it has changed its name.

Conceptually the file has moved from one place in the disk to another place, acquiring a new name on the way.

PRACTICAL EXERCISES

Using mv

- Make a copy of the file called ‘daisy’, and call the second version ‘buttercup’

```
holyrood$ cp daisy buttercup
```

- Examine the contents of ‘daisy’ and ‘buttercup’ to establish that they are identical

```
holyrood$ ls -l daisy buttercup
```

  you will see that the two files are the same size. Now type

```
holyrood$ more daisy
holyrood$ more buttercup
```

  and you will see that the contents are the same.

- Rename the file called ‘buttercup’; give it the name ‘bindweed’.

```
holyrood$ mv buttercup bindweed
```

  the file called ‘buttercup’ has disappeared, but there is now a file called ‘bindweed’

```
holyrood$ ls bindweed
holyrood$ ls buttercup
holyrood$ more bindweed
```

DELETING FILES - 'RM'

- rm stands for 'remove'
  - `-i` option asks for confirmation before destroying each file

Deleting files

'rm' stands for 'remove' and is the command to remove files.

Confirm before deleting

Once a file has been deleted, it has gone for ever. You can arrange for the computer to confirm that you want to delete each file, by using the option '-i'. In this case '-i' stands for 'interactive', because you must answer 'y' or 'yes' to the computer's question 'destroy filename?' before any file is destroyed.

PRACTICAL EXERCISES

Using rm

- Examine the list of files in your directory

```
holyrood$ ls
```

- Note that there is a file there called ‘bindweed’
- Remove the file called ‘bindweed’

```
holyrood$ rm bindweed
```

- Confirm that this file has now disappeared

```
holyrood$ ls
```

There is now no file called ‘bindweed’

EDITING: UE (MICROEMACS)

- ‘control’ key in combination with other keys
  - **^p**, **^n** move up a line, move down a line
  - **^f**, **^b** forward one character, back one character

- ‘meta’ key followed by other keys
  - ‘Esc’ then **z** to leave ue

You can change the contents of text files by using a program called an editor. This workbook will show you how to use the editor called ‘ue’, which stands for ‘microemacs’.

Microemacs is a very flexible and powerful editor, available on a wide range of computers.

The ‘control’ key

Microemacs uses a combination of the keyboard’s ‘control’ key with other keys to move the cursor about the screen. For example, holding the control key down and pressing **p** moves the cursor up one line; holding the control key down and pressing **n** moves the cursor down one line. This combination of keys is described as ^n and ^p.

Alternatively, the arrow keys on your keyboard can be used to move the cursor.

The ‘meta’ key

ue uses another key, which it calls the ‘meta’ key. On most keyboards, ‘meta’ is in fact the ‘escape’ key, labelled ‘Esc’. The ‘meta’ or ‘Esc’ key is not used at the same time as other keys; instead, you first press the key labelled ‘Esc’, then press another key immediately afterwards.

Leaving ue

Press ‘Esc’ and then **z** to exit from ue.

MICROEMACS (CONTD.)

- Save work frequently using ^x, ^s
- Commands for moving the cursor
  - move forward or backward by one character, line, word or page
  - move to start or end of file

Save as you go

It is very important to make sure that the computer has saved what you have typed onto a disk. The key strokes '^x, ^s' will achieve this.

You should do this frequently, because if the computer stops working and you have not saved your work, you will lose all the text that had not been saved.

Further useful key combinations

^	indicates the 'control' key
ESC	indicates the 'meta' or 'escape' key
^f	move forward one character
^b	move backward one character
^e	move to end of line
^a	move to start of line
ESC f	move forward one word
ESC b	move backward one word
^n	move forward one line
^p	move backward one line
^v	move forward one page
^z	move backward one page
ESC >	move to end of file
ESC <	move to start of file

MICROEMACS (CONTD.)

Further useful key combinations (contd.)

^d	forward delete one character
ESC d	forward delete one word
^k	delete to end of line
^t	transpose characters
ESC space	set mark
^w	wipe region between cursor and mark set with ‘ESC spacebar’
^y	yank text back from wipe buffer
^l	refresh the screen
ESC r	search and replace
^x s	search forward
ESC z	save all buffers and exit
^x ^i	insert file
^x ^s	save current file
^x ^w	write a file to disk
ESC ?	display ‘help’ information about ue
^x 0	leave the ‘help’ information.

PRACTICAL EXERCISES

- Start to edit the file called 'fruitcake'

  ```
  holyrood$ ue fruitcake
  ```

  The screen will go blank, then will look like the screen dump above
- Type in some more text

  Anything will do - type in a poem or a nursery rhyme!
- Experiment with **^p**, **^n**, **^f** and **^b** to move about the text

  Delete characters with the delete key, or the **^d** combination
- Move a chunk of text to a different place in the file

  Go to the start of the chunk

  Press ESC then spacebar

  Go to the end of the chunk

  Press **^w**

  Go to where the chunk is to be inserted

  Press **^y**
- Experiment with some of the other key combinations.

PRINTING FILE CONTENTS

- lpr
  - list file to default printer

- Make sure file is in format suitable for the printer
  - postscript printers require special format

You can print the contents of text files onto paper, if your computer is connected to a printer. The Unix command to print files is `lpr`.

The most commonly used option of the `lpr` command is `-P`, which allows specification of which printer is to be used.

Printers available during the course

You may be able to use a laser printer during the course, or you may have to use a line printer.

It is important to know what sort of printer is available to you, because laser printers usually require files which are in a format known as postscript.

Plain text files

If the file contains ordinary text, you must send it to a line printer; instructions for this are shown on this page.

A command for putting output into a form suitable for postscript printers is the next topic in this workbook. Most word processing packages will automatically invoke a utility that generates output in a format suitable for a postscript printer.

LPR (CONTD.)

- Options available with lpr
  - Specify printer
  - Specify number of copies

Format of the command

```
lpr [ -Pprinter ] [ -#copies ]  [ -filter-option ]
[ filename ... ]
```

Options available with lpr

-P specify a named printer; if none specified, output goes to the default printer

-# specify number of copies

HAS THE FILE PRINTED YET?

- What files are waiting to be printed?

  `lpq -P`*printername*

- Stop a particular file from being printed

  `lprm -P`*printername jobnumber*

What files are waiting to be printed?

You can find out how many files have been sent to a particular printer, who they belong to, and when they were sent, by using the command 'lpq'. You will see a list of files waiting to be printed by a particular printer; this is called that printer's 'print queue'.

The option '-P' is used to specify which printer's queue is to be reported on.

lpq also shows you a job number associated with each file in the print queue; you will need to use this number if you decide to remove one of your files from the queue.

Stop a file being printed

You may decide that you don't want a file to be printed after all (perhaps you discover that there are a lot of files waiting for a particular printer, so you want to send your file off to another printer).

The command to remove a file from the printer's queue is `lprm`. You can only remove files from the queue if they belong to you!

PRACTICAL EXERCISES

Printing

- ❑ Find out the name of the printer to which you have access;

  Find out if this is a line printer or a postscript printer; the following instructions assume you have access to a line printer.

- ❑ Print the contents of the file called ‘daisy’

  ```
  holyrood$ lpr -Pprintername daisy
  ```

- ❑ Find out whether ‘daisy’ has been printed yet

  ```
  holyrood$ lpq -Pprintername
  ```

  You will see a list of files, with their owners, and a job number for each file

- ❑ Remove ‘daisy’ from the printer queue

  ```
  holyrood$ lprm -Pprintername jobnumber
  ```

- ❑ Experiment with some of the other printer options

POSTSCRIPT FILES

- Page description language used by many laser printers
  - postscript printers cannot use ‘plain’ text files
  - postscript files contain details of page layout, fonts etc

- Postscript files start with %!
  - can be printed to line printers or matrix printers
  - TERRIBLE WASTE OF PAPER!
  - result unintelligible except on a laser printer

Postscript

Many laser printers now use a ‘page description language’ called ‘Post Script (T)’, usually referred to as ‘postscript’. Some, but not all, laser printers are not able to print plain text files; they can only cope with files which are in postscript format. Many packages produce output in postscript format.

Postscript files often have names which end in ‘.ps’; they are text files, so you can look at their contents. However, their contents look pretty unappetizing, because they are full of details of page layout and fonts.

Sending files to a laser printer

When you use the ‘lpr’ command you should know what format your file is in, and only send postscript output to laser printers, and plain text output to line printers. Postscript output regularly gets sent to line printers, and this wastes quite a lot of paper!

Finding out if a file is in postscript format

Most postscript files start with the characters ‘%!’ so if a file starts with these characters, it is probably in postscript format.

PRACTICAL EXERCISES

Laserprinting

- ❑ Find out if you have access to a postscript printer. If you have, print the contents of the file called ‘postscriptfile’. The following instructions assume that you do have access to a laser printer. If this is not the case, do not carry out the rest of the instructions on this page!
- ❑ Inspect the contents of the file called ‘postscriptfile’

  Notice the first two characters are %!

ENSCRIPT

- Converts text files into postscript format
  - sends result to a printer
- No clever text formatting
  - no bold, underline, centre
  - can print in 2 columns
  - can rotate through 90 degrees

A command called ‘enscript’ is available to convert plain text files into postscript format, and send the result to a printer. This is a useful way of printing simple files onto a postscript printer.

Enscript does not carry out any clever formatting; it does not have any underline, bold, or centring facilities for example. However it is a good way of getting postscript output quickly!

Options available

A few options are available with enscript. It is possible to rotate the output through ninety degrees, and print in two columns. This is useful for program output where the lines are short and it can be wasteful to print in the ordinary way.

Output can be sent to the named postscript printer, or saved to a file for printing later.

`enscript -2r` *filename(s)* rotate into landscape

`enscript -2R` *filename(s)* don’t rotate (default)

`-Llines`	set the maximum number of lines per page.
`-p` *outfile*	write the POSTSCRIPT output to a file called ‘*outfile*’ instead of printing it.
`-Pprinter`	send the output to the named printer.

PRACTICAL EXERCISES

Using enscript

- Find out if you have access to a postscript printer. If you have, you can carry out the rest of the instructions on this page!

  Use enscript to print the file called `daisy` on the postscript printer called *'lasername'*

  ```
  holyrood$ enscript -Plasername daisy
  ```

- Use enscript but save the output in a file called 'daisyfile'

  ```
  holyrood$ enscript -p daisyfile daisy
  ```

Look at the contents of 'daisyfile' to see what the postscript format looks like

```
holyrood$ more daisyfile
```

Quit from 'more' when you have seen enough!

GETTING HELP!

- man
  - you need to know the name of a command
- help
  - written at Edinburgh
  - shows a list of commands and documents
  - then use man

Two important sources of help are always available to you when you are logged on to a Unix computer at Edinburgh.

All Unix systems have an on-line manual, accessible via the command '`man`'. However, the 'man' command needs you to specify what command you want to know about.

Local help facilities

A 'help' system has been written at Edinburgh; the command '`help`', followed by a topic name, gives a list of commands and also details of documentation written at Edinburgh which may contain information relevant to the topic quoted.

Printing a manual page

If you wish to print manual pages on a line printer, send them to a file first using the '>' character:

```
man ls > lsfile
```

Then print the file:

```
lpr lsfile
```

This should ensure that the printout is readable.

Using man

- Find some information about listing files

  ```
  holyrood$ man ls
  ```

  you will see a lot of detailed information about the ‘ls’ command, presented in your default pager (probably ‘more’).

- Find what command you might use

  ```
  holyrood$ help list
  ```

  you will get a screenful of suggestions of commands that are to do with the word ‘list’.

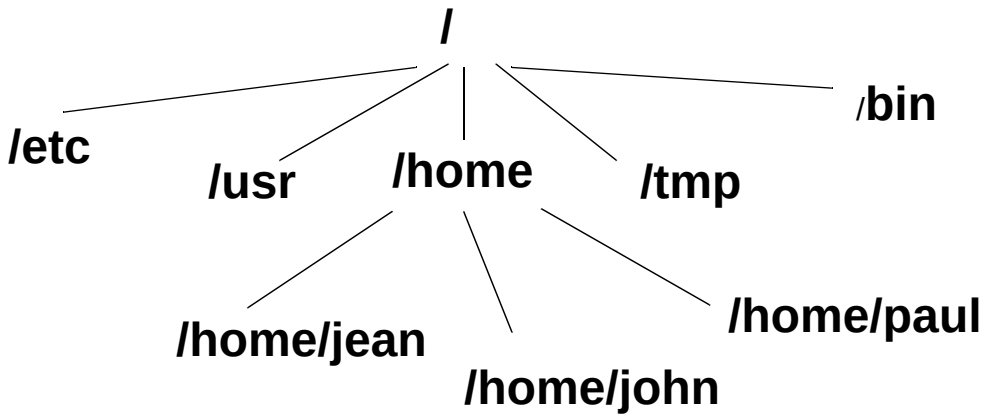
  type ‘q’ to exit from the ‘help’ system

  ```
  help (h returns here): q
  ```

Chapter 2

The Unix Filesystem

DIRECTORY STRUCTURE



All Unix files are grouped together into directories. Directories themselves are also located inside (or below) other directories. The topmost directory of all is called ‘ / ‘, (pronounced either ‘slash’ or ‘the root directory’). All other directories form a hierarchical tree structure underneath ‘ / ’. Immediately underneath the root directory, there are usually 5 directories called `/usr`, `/etc`,`/home`, `/tmp`, and `/bin`.

Home directories - /home

The user disk space is often contained in the part of the filesystem called ‘/`home`’. There are sometimes ‘group’ directories beneath /`home`, and then individuals’ user account directories below those. The group directories are used to keep all users from particular departments in the same administrative group.

Program files - /bin

The directory `/bin` usually contains executable (or binary) files.

/usr

`/usr/bin` and `/usr/lib` contain programs and libraries (as do `/bin` and `/lib`). On some systems `/bin` contains basic commands (`mv`, `cp` etc) and `/usr/bin` has editors etc.

/etc

`/etc` contains system configuration files.

/tmp

`/tmp` is used as ‘scratch’ space which is used by programs to hold intermediate (temporary) files during processing.

FINDING YOUR LOCATION: PWD

✷ Find your present location

- holyrood$ pwd
- /home/jean

You can find your present location at any time using the command ‘pwd’, which stands for ‘print working directory’.

Full pathname

‘`pwd`’ shows you a list of directory names, starting with a ‘ / ’. This is the list of directories through which you would have to pass to move from the root directory (/), to your present directory.

A file’s ‘full name’

The most foolproof method of specifying a file to be used by a Unix program can be to quote the full (or absolute) pathname of the directory containing the file, followed by the file’s name.

Administrator’s activities!

The system administrator may move the position of users’ home directories from time to time, perhaps for reasons of space. This will mean that ‘full’ pathnames will no longer work for files which are within users’ filespace. This problem can be resolved by using pathnames which are relative to the present position of each user’s home directory. Relative pathnames are described on the next page.

Change directory

The command ‘`cd`’ is used to move into another directory.

PRACTICAL EXERCISES

Finding your location

- ❑ Login, if you have not already done so, following the instructions you were given in the first workbook.
- ❑ Find your present location

  ```
  holyrood$ pwd
  ```

  You will see your full pathname
- ❑ Try out the 'ls' command, giving as argument the full pathname for one of the files in your directory

  ```
  holyrood$ ls /pathname/filename
  ```

  Note you should substitute the actual pathname which you obtained in the first part of this exercise, and a real filename, in this use of 'ls'
- ❑ Move into another directory

  ```
  holyrood$ cd directoryname
  ```

  Check that you are now in a different directory

  ```
  holyrood$ pwd
  holyrood$ ls
  ```

RELATIVE PATHNAMES

- Paths assumed to start in current directory
  - unless start with ‘ / ’
- Specify current directory explicitly
  - current directory called ‘ . ’
  - directory above called ‘ .. ’
  - value of ‘.’ and ‘..’ depends on where you are!
- `cd` to change directory

Relative pathnames

Instead of quoting the full pathname of a file or directory, you can specify files or directories with reference to the directory you are currently in. If no directory is specified, Unix will use the current directory.

Current directory and the one above it

Your current directory can be referred to explicitly as ‘ . ’; the directory immediately above you can be referred to as ‘ .. ’.

‘ . ’ and ‘ .. ’ can be used anywhere in the filesystem. A pathname which starts ‘ ./ ’ will have a different absolute value depending on which directory it is used from.

Pathnames starting from ‘ / ’

Note that paths which start ‘ / ’ are specified from the root directory, not from your current directory. This is a common cause of confusion when paths don’t work!

Changing directories: cd

The command ‘`cd`’ is used to move into a different directory. You can specify a particular directory to move to, or you can move to the directory above your present one by referring to it as ‘..’.

The command ‘cd’ on its own, with no arguments, returns you to your home directory. ‘`cd ~`’ also returns you to your home directory.

PRACTICAL EXERCISES

Pathnames

- Find your present location

  ```
  holyrood$ pwd
  ```

  Write down its full pathname

  This is equivalent to ‘ ./ ’

- Move to one of the system directories

  ```
  holyrood$ cd /usr/local/bin
  holyrood$ pwd
  ```

  Write down its full pathname

  This is also equivalent to ‘ ./ ’ for the directory you are now in.

- Return to your home directory

  ```
  holyrood$ cd
  ```

MAKING NEW DIRECTORIES

- Create new directories below present directory
- holyrood$ `mkdir extradir`

In the course of your work you will often find it useful to create new directories - to hold all the files on a particular topic, or all the files associated with a particular project.

The command ‘mkdir’ will create a new directory immediately below the directory from which the command is issued.

Contents of a new directory

A newly made directory is empty, except for two entries called ‘ . ’ and ‘ .. ’. ‘ . ’ signifies the directory itself, and ‘ .. ’ signifies the link between the current directory and the directory above it.

PRACTICAL EXERCISES

Using mkdir

- Establish what your present directory is, and what its contents are

```
holyrood$ pwd
holyrood$ ls
```

- Create a new directory

```
holyrood$ mkdir alfred
holyrood$ ls
```

- Move into the new directory, and examine its contents

```
holyrood$ cd alfred
holyrood$ pwd
holyrood$ ls -al
```

- Move back to the first directory

```
holyrood$ cd ..
```

COMMANDS AND PATHNAMES

- cp ./fred ../fred

```
/home/                    /home/
  jean/                     jean/fred
    extradir/fred             extradir/fred
```

- mv ./fred ../fred

```
/home/                    /home/
  jean/                     jean/fred
    extradir/fred             extradir
```

So far you have used various commands (`ls, cp`, `mv`, etc) always referring to files in your present directory. If no path is given, most commands assume that any files referred to are to be found in the current directory.

These commands can all be used with absolute or relative pathnames, so that they can refer to files anywhere in the filesystem.

Same directory

'.' refers to the current directory, and can be useful in a pathname. Sometimes the system needs this confirmation that a path is relative to the current directory.

PRACTICAL EXERCISES

Moving and Copying from a directory

- Copy a file from your home directory into your new directory, ‘alfred’

```
holyrood$ cp daisy ./alfred
holyrood$ cd alfred
holyrood$ ls
```

- Create a new file in your new directory

  the ‘touch’ command creates a file, which is empty

```
holyrood$ touch bertrand
holyrood$ ls
```

- Copy this file back into your home directory

```
holyrood$ cp bertrand ..
holyrood$ cd ..
holyrood$ ls
```

OTHER PEOPLE'S FILESPACE!

- Possible to move into other people's directories
  - possible to look at their files
- Security implications
  - usually not possible to view contents of files
  - not possible to change system files!
- Getting home again!
  - `holyrood$ cd`

Home directories and the filesystem

Many different users can keep files in the same filesystem. Every user is given a home directory when they are allocated their user account, and this is where they keep their files.

Visiting other people's filespace

It is possible to use the 'cd' command to move into other people's filespace, and then to use the 'ls' command to see a list of their files. However it is polite to obtain that person's permission first.

Returning to your own filespace

The character ' `~` ' on its own can be used to refer to your own home directory. ' `~fred` ' refers to the home directory of user account ' fred '.

Viewing the system files

It is often useful to move into the system directories and look at files there (to check whether particular packages are present, to look at configuration files and so on).

Security implications

It is obviously important to make sure that if other people move into your filespace, they cannot actually view the contents of your files, or, still worse, change the contents, unless you allow them to. The system administrator must make sure that nobody can change the system files!

PRACTICAL EXERCISES

Looking at someone else's files

- Check your present location

  ```
  holyrood$ pwd
  ```

  Move back to your home directory if you are not there already

- Move to the directory above your own

  ```
  holyrood$ cd ..
  holyrood$ pwd
  holyrood$ ls
  ```

  You will see several home directories, including your own

- Move into someone else's home directory!

  ```
  holyrood$ cd someoneelse
  holyrood$ pwd
  holyrood$ ls -l
  ```

- Move to the root directory

  ```
  holyrood$ cd /
  holyrood$ pwd
  holyrood$ ls -l
  ```

- Move back to your own filespace

  ```
  holyrood$ cd
  ```

  Alternatively you could have typed

  ```
  holyrood$ cd ~
  ```

GROUPS

- Facility for sharing resources
  - Collective access to files and printers

UNIX provides facilities for a number of users to work together on a project while keeping their files private from unauthorised users. This is achieved by the use of groups.

A group

A group is a collection of users. Each user on a UNIX system must belong to at least one group. They may belong to several groups.

You can obtain a list of the groups you belong to with the command:

```
holyrood$ groups
```

The function of groups

The most common use of a group is to control shared access to files. Group membership can also be used to restrict access to other shared resources such as printers.

PRACTICAL EXERCISES

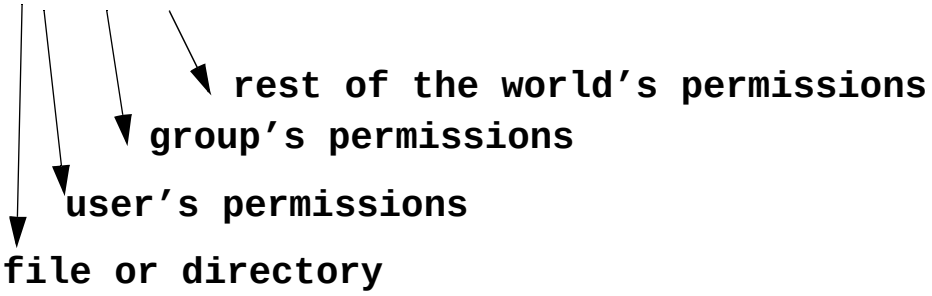
Determining your groups

- Find out which groups you are a member of.

```
holyrood$ groups
```

FILE PERMISSIONS!

```
holyrood$ ls -l
-rwxrwxr-x 1 jeanr eucsup 107 Jun 14 10:55 kate
```



File Permissions

The first piece of information given by the ‘ `ls -l` ’ command is whether or not the entry refers to a file or to a directory. If the first character is ‘ `d` ’, the entry refers to a directory, if it is ‘ `-` ’, it refers to a file.

Access permissions for files

The next 9 pieces of information describe access permissions to the file or directory. There are three sets of permissions: for the user, the user's group, and the rest of the world (others).

For each type of user, separate permission is given to read the contents of the file or directory (indicated by the letter ‘r’), to write to it (in other words, to change its contents), indicated by the letter ‘w’, and to execute the file (only needed if this is a program file), indicated by the letter ‘x’.

Links

After the file permissions the number of links to a file is given. The meaning of ‘links’ will be described later.

Owner

The next piece of information is the account name (the user ID) of the file's owner. Every file is owned by somebody. System files are owned by ‘root’, which is the name in Unix for the system administrator.

Group

The next piece of information is the group ownership of the file. Every file is owned by some group. In the above example members of the group ‘eucsup’ have the same access permissions (read, write and execute) as the actual owner of the file. Other users only have read and execute permission.

PRACTICAL EXERCISES

Examining file permissions

- Examine the access permissions and ownership details for the files in your directory

```
holyrood$ ls -al
```

- Examine these details for the files in the root directory

```
holyrood$ cd /
holyrood$ pwd
holyrood$ ls -al
```

- Move into the directory where a lot of utility files are found

```
holyrood$ cd /bin
holyrood$ pwd
holyrood$ ls -al
```

- Go back to your own directory

```
holyrood$ cd
```

DIRECTORY PERMISSIONS

- Read
  - list files in the directory - need execute permission to cd into it
- Write
  - create and delete files - if have execute permission
- Execute
  - access files - read them, delete them
  - directories need execute permission!

Access permissions to directories have slightly different functions from those for files.

Read permission

Read permission to a directory allows the user (or group, or the rest of the world) to list the files in the directory - but not necessarily to read them. Execute permission is also needed in order to read file contents.

Write permission

Write permission to a directory allows the user to create and delete files from the directory.

Execute permission

Execute permission to a directory allows the user to access files in the directory (i.e. 'go through' the directory to access files and subdirectories). Execute permission is needed in order to read the files in a directory, and to delete files from the directory. Directories are not very useful unless they do have execute permission!

File permissions also needed

Even if a user has read and execute permission to a directory, she will only be able to read those files within it to which she also has read permission at the file level.

PRACTICAL EXERCISES

Examine the permissions of your directory

- Examine the access permissions on your home directory, and on the directories above it

```
holyrood$ cd ..
holyrood$ ls -l
holyrood$ cd ..
holyrood$ ls -l
```

- Return to your home directory

```
holyrood$ cd
```

CHANGING PERMISSIONS

- You can change access permissions
  - for the files and directories that you own
  - chmod
- Specify permissions
  - for user (u), group (g), other (o), default is all (a)
  - add (+), remove (-), explicit (=)
  - read (r), write (w), execute (x)

The command for changing file access permissions is called ‘chmod’. You can change permissions for any of the files which belong to you.

Who is the permission for?

Read, write and execute permissions can be specified for user (u), for group (g), or for others (o). If you do not include any one of these, the default is ‘a’ for all!

Adding or subtracting permission

You can add permissions using the ‘ + ’ sign, remove permissions using the ‘ - ’ sign, or assign permissions explicitly using the ‘ = ’ sign.

What permissions?

Read permission is indicated by the letter ‘r’, write permission by ‘w’ and execute permissions by ‘x’.

Remove all access

You can take away all permissions (so nobody can read, write to, or execute the file) by using the ‘ = ’ sign on its own.

No spaces

It is important not to leave spaces between the user type, the equals sign, and the settings in the ‘chmod’ command.

PRACTICAL EXERCISES

Using chmod

- Find the access permissions for your files

```
holyrood$ ls -l
```

- Add write permission for your group to the file called ‘fred’

```
holyrood$ chmod g+w fred
holyrood$ ls -l
```

- Take away all access permissions to everybody (including yourself) to the file called ‘`daisy`’

```
holyrood$ chmod = daisy
holyrood$ ls -l
```

- Give full permissions to anybody for the file called ‘kate’

```
holyrood$ chmod a+rwx kate
holyrood$ ls -l
```

- Make it so that only you can read a file

```
holyrood$ chmod = kate
holyrood$ chmod u=r kate
holyrood$ ls -l
```

COPYING FILES

- Copy file from another user's directory
- copy of file belongs to you
- need read access to the file
- need execute permission on all directories from root down

Copying files: permissions and ownership

After copying a file from another user's directory, you will be the owner of the copy.

Permission to copy from another user

In order to copy a file from another user, you need to have read access to the file itself, and to have execute permission on all the directories from root down to the directory containing the file.

PRACTICAL INSTRUCTIONS

Using cp

- ❑ Arrange with someone else in the class to copy one of their files.
- ❑ Copy this file into your current directory

  holyrood$ **cp ~***someoneelse/theirfile* **myfile**
- ❑ Check the ownership of the copied file

  holyrood$ **ls -l myfile**

REMOVING DIRECTORIES

- rmdir
  - first remove all files from the directory
  - then use rmdir to remove the directory itself
- rm -r
  - recursively delete the contents of a directory, its subdirectories, and the directory itself.
  - POWERFUL. USE WITH CARE ! !
- Permissions
  - write permission to directory or files

You can also remove directories, using the command ‘rmdir’. This command only works if all files have first been removed from the directory, and if you have write permission to the directory containing the directory.

Recursively remove files and directories

‘rm’ can be used to remove directories if its ‘r’ option is used. This is an extremely powerful option, to be used with great care! ‘rm -r’ recursively deletes the contents of a directory, its subdirectories, and finally the directory itself! The ‘-i’ option, where the computer asks before destroying each file, can be useful here!

Permissions needed

You must have write permission in order to be able to destroy a file or a directory; and you also need write permission to the directory containing the file or directory. You need execute permission further back up the hierarchy than this!

If you try to delete a file or directory for which you do not have write permission, the computer will tell you this, and ask if you want to delete it anyway.

PRACTICAL EXERCISES

Using rmdir

- Establish your present location

```
holyrood$ pwd
holyrood$ ls -l
```

- Remove the new directory:
  first remove any files from it

```
holyrood$ rmdir alfred
holyrood$ ls -l
```

BACKUP FACILITIES

- In case of accidental deletions!
  - in case of hardware problems
- Backup carried out daily, onto tape
  - regular full backup
  - daily incremental backup

Backup of all material which has been saved to disk takes place daily, usually in the small hours of the morning, without intervention from users. Once you have saved material to disk on one of the EUCS maintained computers you can rely on its remaining available to you.

Retrieval

If a backed up file is destroyed by a system failure, it will be replaced as soon as possible.

COMPUTER DISKS

- Partitions
  - an allocation of space when disk is initialised
  - cannot exceed space
- Home directories in same partition
  - user quotas

Computer disks are divided up into sections called partitions. Each partition is given a certain allocation of space when the disk is initialised, and cannot exceed this area. If the files which are stored in the partition grow too large, or there are too many files, the computer will send a message saying something like 'disk full'.

User's files in one partition

User's home directories are all stored in disk partitions, so if certain users store a large number of large files, they will reduce the space available to other users!

User quotas

In order to stop individual users taking up too much file space, every user is given a quota of space on the disk. If their files start to exceed this space, they receive a message to say that they are exceeding their quota, and must reduce their space by a certain amount, in a certain time.

INVESTIGATING YOUR QUOTA

- holyrood$ quota -v
  - Disk quotas for jeanr (uid 1604):

- Filesystem
  - /a/cancer/disk/home/u14
  - usage quota limit timeleft files quotalimit timeleft
  - 10444 13000 14000 595 0 0

You can investigate how much of your own allocation is still available by using the command ‘quota’. You need to use the -v option (for ‘verbose’), otherwise you will only be given information if you are over quota!

Information given

The figures show how much space you have used, what your quota is, your limit (larger than your quota because you are allowed to exceed your quota temporarily), and how much time you have left to reduce your quota when you have exceeded it! If you fail to reduce your quota in the time given, you will no longer be able to create files. All you will be able to do will be to remove files to bring yourself back under quota again.

Your own quota only

The superuser may view the quotas and limits of any user. Other users can only view their own quotas.

PRACTICAL INSTRUCTIONS

Using quota

- Find out how much of your quota you have used

```
holyrood$ quota -v
```

SAVING SPACE

- Files you can safely destroy
  - core
  - most ‘ .log ’ files

- Compress files
  - compressed files have names ending ‘ .Z ’
  - uncompress files when you need them again

If you find that you are exceeding your quota, you will need to reduce the amount of space that your files are occupying on the disk. You can either remove files, or compress your files.

Which files can be removed?

The first thing to look for is any files called ‘core’; core files contain information which is intended to help sort out problems when things go wrong, and are intended for system administrators and programmers. You will not be able to use them, so you can simply remove them.

You might also look at files whose names end in ‘.log’; these files contain day-to-day information on certain programs. However, they can grow large and take up unnecessary space, so after reviewing their contents they can be removed if space is short.

Compressing files

You can make files smaller if you cannot remove them, and ‘compress’ will do just this. Compressed files are given the file extension ‘.Z’; they can be returned to their normal size using the command ‘uncompress’.

Other compression utilities

Utilities called ‘gzip’ or ‘pack’ are available on some systems, and do a similar job to ‘compress’.

PRACTICAL INSTRUCTIONS

Saving space in your account

- Find out how large your quota is, and how much you have left!

  ```
  holyrood$ quota -v
  ```

- Find out if you have a file called ‘core’ in your home directory, or any directory beneath it

  ```
  holyrood$ find . -name core -print
  ```

  Destroy any file called ‘core’

- Compress some files

  ```
  holyrood$ compress filename
  ```

  or

  ```
  holyrood$ compress *
  holyrood$ ls filename.*
  ```

  Files which have been compressed now have the extension ‘.Z’

- Uncompress a file

  ```
  holyrood$ uncompress filename.Z
  ```

DISK USAGE

- holyrood$ df -k ~
  - Filesystem
  - cancer:/disk/home/u14

```
kbytes    used      avail          capacity                        Mounted on
1054422   302464    646516         32%                /a/cancer/disk/home/u14
```

Occasionally you may find that you are unable to save your work because ‘the disk is full’, even though you have not exceeded your quota.

This probably means that the disk partition where your home directory is has been filled up, so that there is simply no way that you can get access to your theoretical quota of space. You can find out how space is being used on the disk using the command df.

df shows the amount of disk space which has been allocated to filesystems, how much of that space has been used and how much is still available. It expresses disk usage as a percentage.

Filesystem containing current directory

df can be used to report on a single filesystem, by specifying as an argument a pathname in that filesystem. Thus ‘df .’ shows the amount of space on the filesystem containing the current directory; ‘df -k ~ ’ shows the space on the filesystem containing the user’s home directory. Used without arguments, df reports on all available filesystems.

Disk full

If you find that the disk is full, you must contact your system administrator. This is not a problem you can solve yourself.

PRACTICAL EXERCISES

Check your disk usage

- Find out how much space is left on the disk where your home directory is stored

  ```
  holyrood$ df ~
  ```

  the output will be similar to that shown on the slide. The example on the slide shows that only 32% of the disk has been used, so there is plenty of space left for everyone.

  If you add up the space quoted as available and the space quoted as used, the figure is less than the amount of total space in the filesystem.This is because the system reserves a fraction of the space in the filesystem to allow its filesystem allocation routines to work well. The amount reserved is typically about 10%.

Chapter 3

Unix Facilities

SAVING TIME USING THE SHELL

- Typing shortcuts
  - filename completion
  - reviewing previous commands (history mechanism)
  - editing the command line
  - wildcards
- Input / output
  - redirection using pipes

The bash shell provides some very helpful mechanisms which can save you a lot of time and make your life a bit simpler!

Typing shortcuts

There are several shortcuts for doing less typing: these include filename completion, reviewing previous commands, editing the command line, and using wildcards.

Input / output

bash also allows a lot of flexibility in the way you specify where program input will come from, and where output is to go to.

FILENAME COMPLETION

- Type the first few characters of a filename
  - system will complete it for you

- If the filename is not unique
  - terminal will bleep (if bleep activated)
  - wait for more characters

If you are typing in a long filename, you can save time by typing in the first few characters and then pressing the ‘Tab’ key. If you have typed enough characters to specify uniquely the name of one of the files in the current directory, bash will finish the name for you.

If the characters you have typed occur at the start of more than one filename, the shell will make the terminal bleep and wait for you to type in enough characters to make it unique.

If you press the ‘Tab’ key again at this point, the shell will provide you with a list of the filenames which begin with the characters which you have so far specified.

PRACTICAL EXERCISES

Using filename completion

- Obtain a list of all the files in your current directory

  Type

  ```
  holyrood$ ls
  ```

- Choose a file, perhaps the file called ‘buttercup’

  type

  ```
  holyrood$ ls -l bertr[Tab]
  ```

  In other words, type the first five characters of ‘bertrand’ then press the tab key

  The rest of the name will be completed for you!

  Press the [Return] key to obtain details about ‘bertrand’

HISTORY MECHANISM

- Review commands already given
- Use commands similar to ue
  - step through previous commands
  - edit them
  - press return to execute
- Cursor keys work
  - if terminal set up properly

Bash enables you to review the commands you have issued, and to use them again without having to retype them. This can be very useful if the command was a difficult one to put together, perhaps involving a long filename or complicated options.

Commands from ue

The editing key combinations that you learnt earlier to edit files can also be used to review and edit previous commands.

Review previous commands

^p is the microemacs command to move back a line in a text file; it can also be used at the command prompt where it will show you the previous command that you typed! If you press ^p a second time you will see the command before that. ^n will then bring you to the next more recent command.

history

Your system saves a list of the commands which you have used, and you can display this list using the command '`history`'. This will display a list of all the commands which are currently stored.

PRACTICAL EXERCISES

Use the History mechanism

- ❑ Review previous commands

  Press **^p** a few times

  Press **^n**

  Press **^u** if necessary to clear the command line

- ❑ Look at the list of commands which you have used

  ```
  holyrood$ history
  ```

EDIT A COMMAND LINE

- ue keystrokes will work on the content of the command line
  - delete characters, move backwards, move forwards

- Press return with cursor anywhere in the command line
  - command will execute

Having obtained the command that you need, you can edit it (using more ue commands), or you can execute it by pressing the Return key. You can press Return when the cursor is anywhere on the command line, and the whole command line will be executed.

Delete the command

If you change your mind, pressing ^u will obliterate text from the present position of the cursor back to the start of the line; ^k will delete from the position of the cursor to the end of the line (these are the ue commands that you will use most often!).

PRACTICAL EXERCISES

Edit a command

- ❑ Delete a command line

  Obtain a command

  `holyrood$ ^p`

  Delete it

  `holyrood$ ^u`

- ❑ Edit one of the previous commands

  Perhaps change a filename

  `holyrood$ ^p ^a ^d ^f,` *etc*

  Execute the new command

  Press [Return]

WILDCARDS

- Asterisk
  - substitutes for any number of characters
  - *sox

Asterisk

The asterisk symbol '*' can be used as a wildcard with all the commands we have seen so far. Thus '*sox' can be used in a command and will specify all the files in the current directory whose names end in 'sox'. The asterisk does not specify how many characters it is standing in for; thus the expression '*sox' will be successful for files called 'popsox' or files called 'psox'.

PRACTICAL EXERCISES

Use the '' wildcard*

- Obtain a list of all the files in your directory whose names begin with 'p'; type

  ```
  holyrood$ ls p*
  ```

- Obtain a list of all the files in your directory whose names end in 'and'; type

  ```
  holyrood$ ls *and
  ```

QUESTION MARK ?

- Substitutes for exactly one character
  - ?sox
  - psox but not popsox!

The question mark ‘?’ can be used to subsitute for exactly one other character. Thus the expression ‘?sox’ can indicate a file called ‘psox’ but will not find a file called ‘popsox’.

PRACTICAL EXERCISES

Use the '?' wildcard

- Obtain a list of all the files in your directory whose names contain exactly four letters, the last one being 'd'

```
holyrood$ ls ???d
```

- Obtain a list of all the files whose names contain the letter 'b' as the second letter

```
holyrood$ ls ?b*
```

REDIRECTING INFORMATION

- Standard output
  - output goes to ‘standard output’
  - the screen, unless otherwise defined
- Redirecting output
  - ‘>’ will overwrite an existing file
  - ‘>>’ will append material to an existing file
- Redirecting input
  - ‘<’ will take input from a named source

Standard output

Output from Unix commands is displayed on the screen unless you direct it elsewhere. The ‘cat’ command sends output to the screen by default, but the ‘>’ symbol can be used to direct output into a file rather than to the screen.

Append or overwrite?

If you use the ‘>’ symbol to redirect output into a file, any file already existing with that name will be overwritten. You must use the ‘>>’ symbol if you want to append material to a file which already exists.

Redirecting input

The ‘<’ symbol can be used to redirect input to any command - so that commands which normally take input from the keyboard, ‘standard input’, will then take input from a file

PRACTICAL EXERCISES

Use '>' and '>>'

- Use the command 'date', sending the output to a file called 'datefile'

  ```
  holyrood$ date > datefile
  ```

  You have created a file called 'datefile', which contains one line giving the date and time

  ```
  holyrood$ more datefile
  ```

- Use ' date' again, appending a new date and time to the end of 'datefile'

  ```
  holyrood$ date >> datefile
  holyrood$ more datefile
  ```

PIPES

- Output from one command as input to another
  - string several commands together in a ‘pipeline’
  - commands separated by vertical bar “ | ”
- Output from ‘ls’ may cover more than one screen
  - view it a screenful at a time
- Pipes and filters

Another powerful feature of Unix is that output from one command can be used as input to another command. This is called ‘piping’ output into the second command, and is done by using the vertical bar symbol ‘|’ between the two commands.

Using ls and more together

One situation where pipes are often useful is when the output from the ‘ls’ command is used as input to the ‘more’ command. This is done so that if there are a lot of files in a directory, the files can be viewed a screenful at a time.

Pipes and filters

There are a number of programs called ‘filters’ which operate on files to prepare them for input to another program, or make them ready to send to a printer.

Text files may have commands embedded in them which will be read by a filter (such as the text formatting filters troff and nroff); the filter will put in control commands to cause the printer to use bold, centre the text and so on.

PRACTICAL EXERCISES

Use '|'

- Use 'more' to view the output from `'ls -l'`

  'more' will show the list of files one screenful at a time

  ```
  holyrood$ ls -l | more
  ```

- Look in some of the system directories

  These are usually very full

  ```
  holyrood$ ls -l /bin |more
  holyrood$ ls -l /etc |more
  ```

GREP

- Search for files containing ‘fred’ in their name
  - holyrood$ ls -l | grep -i fred
  - prints lines containing the pattern ‘fred’, ‘Fred’, ‘fRed’, etc.
- See if someone we know is logged in

```
holyrood$ who | grep tony
tony       pts/80       Jun 27 10:21    (mescalin.ucs.ed.ac.uk)
tony       pts/62       Jun 30 15:30    (mescalin.ucs.ed.ac.uk)
holyrood$
```

The command ‘grep’ is very useful as part of a pipe.

grep searches files for a pattern and prints, on standard output, all lines that contain that pattern.

Options

The useful option ‘ -i’ causes grep to ignore the distinction between upper and lowercase letters when it is searching for the pattern.

PRACTICAL EXERCISES

Using grep and pipes

- Search through the system directory for all files containing the letters ‘ls’

  ```
  holyrood$ ls /usr/bin |grep ls
  ```

- Do a case independent search for all files containing an x in their names. (This is a complex search using a regular expression. These are dealt with in the second Unix course. If you wish to know more about grep just now, look at the man page.)

  ```
  holyrood$ ls /usr/bin | grep -i x
  ```

- See how many bash shells are running on the system just now

  ```
  holyrood$ ps -ef | grep bash | wc -l
  ```

  How many csh shells are running?

PIPES VS REDIRECTION

- Both channel input and output
- Pipes
  - channel results of one command into another command
- Redirection
  - changes flow of standard input / standard output
  - screen, keyboard, files

Comparing pipes and redirection

Both the pipe mechanism and the redirection symbol redefine where commands are to obtain their input, and where they are to send their output.

Pipes

Pipes are concerned only with channelling output from one command to become the input of another command.

Redirection

The redirection symbol is used to change the flow of input or output, usually between files and keyboard or screen.

COMMANDS START PROCESSES

- The process of carrying out a command
  - process starts when you press ‘return’
- Investigate processes started by you
  - ps
  - ps -f
  - ps -fu username
- Terminating processes owned by you
  - kill process-id
  - kill -KILL process-id

When you use a command (such as ls or more), this starts a series of actions inside the computer; in a Unix computer these actions are collectively called a ‘process’. Every process has a number assigned to it, called its PID (process identifier).

Processes started by you

The ‘ps’ command displays information about processes which you have started. It lists the process ID, or PID, the name of the program that is associated with it, and the cumulative time that the process has been running. The -f flag to ps produces a fuller listing.

With this information you can sometimes solve problems, for example if you have started a process which has stopped working for some reason, but which is perhaps not responding to the keyboard.

Kill

The command ‘kill’ can be used to terminate a process. The kill command uses the process ID, as given by ps. If a process proves hard to kill, the option ‘-KILL’ means ‘really kill this process’.

You can only kill processes that belong to you - ones which you started.

Experimenting with processes

- Try out the ps command

  ```
  holyrood$ ps
  ```

  Note the process IDs, the times they each started, and the programs associated with them

- Kill a process

  Choose one of the processes above, and use the ‘kill’ command

  If you kill the shell which was started when you logged in, you will be logged out!

  ```
  holyrood$ kill process-id
  holyrood$ kill -KILL process-id
  ```

- Start a csh

  ```
  holyrood$ csh
  %
  ```

- Note that the prompt changes. Now run ps and kill the csh

FOREGROUND / BACKGROUND

- Commands in foreground
  - computer waits for command to finish before giving prompt
  - default

- Commands in background
  - continue to run while you give further commands
  - ‘command &’

Foreground

By default, if you start a command, the shell will not give you another prompt until that command has finished executing. The command is said to be executing "in the foreground."

Background

However, the computer is quite capable of carrying out several tasks at the same time, and if you type ‘&’ after typing in a command, the computer will return you the prompt while the command is still executing.

The command is said to be executing "in the background." You will receive a message when the command terminates.

Stopping a process

You can stop the current foreground process by typing ‘ctrl z’. If you decide that you would like it to start again, but not to interfere with your work, type ‘bg’ and the command will continue in the background. Typing ‘fg’ will restart the stopped command, but it will again run in the foreground.

All processes, whether background or foreground, have a process ID, and their progress can be monitored using the ‘ps’ command.

PRACTICAL EXERCISES

Backgrounding a process

- Start a process running, then stop it

```
holyrood$ more bigfile
^z
```

- The command prompt will return, enabling you to carry out more commands.

```
holyrood$
holyrood$ pwd
holyrood$ ls
```

- Start the ‘more’ process off again

```
holyrood$ fg
```

  You are back looking at the file at the place where you stopped it

Extra information

You cannot interrupt ‘ue’ with the ^z combination, because ‘ue’ makes special use of the control key and its combinations. ‘ue’ can be stopped however, by the key combination ‘^x’ followed by ‘d’.

PROCESSES

- ps
  - process name
  - process identifier PID
- kill -HUP PID
- kill -TERM PID
- kill -KILL PID

All processes, whether background or foreground, have a process ID, and their progress can be monitored using the ‘ps’ command.

Killing a process

You can stop an individual process using the ‘kill’ command, and giving the PID which you obtained using ‘ps’.

The two options ‘-TERM’ and ‘-HUP’ both give a process the opportunity to write error messages and close files before exiting. If neither of these options succeeds, use the ‘-KILL’ option. This is a last resort, because processes terminated with the ‘-KILL’ option do not have the opportunity to close files, write error messages and so on.

PRACTICAL EXERCISES

Starting commands in the background

- Start a couple of commands running

  Start them as background commands

  ```
  holyrood$ (sleep 20; echo first) &
  holyrood$ (sleep 20; echo second) &
  ```

  'Sleep' tells the computer to pause for a specified number of seconds

- Use ps to look at their PIDs

  ```
  holyrood$ ps
  ```

  Stop one of the processes

  ```
  holyrood$ kill -HUP PID
  ```

  If this doesn't work, use kill -KILL but only as a last resort!

  ```
  holyrood$ kill -KILL PID
  ```

ENVIRONMENT VARIABLES

- Default values used by operating system
  - printer, type of terminal, prompt string
  - stored in ‘environment variables’
- PWD, PATH, HOME, TERM, PRINTER, PS1

When you use commands such as ‘lpr’ to print a file, you can either specify a printer, or allow the output to go to the default printer. The name of a printer to which output will go by default is stored in an ‘environment variable’. All programs have access to the values stored in environment variables and may use them during program execution.

Environment variable names are usually written in capital letters. The value stored in each environment variable can be inspected using the ‘echo’ command, with the variable name prefixed by ‘$’. The ‘$’ sign tells the operating system to display the contents of the variable.

Contents of some variables

PWD - the current working directory as set by the cd command.

PATH - a colon-separated list of directories where programs etc may be found. The order of this list is important, because directories which come early in the list will be searched before those at the end of the list. If different versions of software exist, the version whose directory is earliest in the PATH is the one that will be used.

HOME - the absolute path of the user’s home directory. Default value for the cd command.

TERM - the terminal type currently in use; needed by programs such as editors which make a lot of use of the screen

PRINTER -your default printer

PS1 - a variable which will be expanded to produce the prompt string

ENVIRONMENT VARIABLES

- Inspect using command ‘echo’
  - holyrood$ echo $TERM
- Set value of environment variables
  - holyrood$ TERM=vt100
  - no spaces!
  - holyrood$ export TERM
- export is just magic just now!
  - variables used only by the shell (eg PS1) don’t need to be exported
  - Don’t worry about this!

Contents of environment variables

You can inspect the contents of any environment variable using the command ‘echo’, followed by a space, then the variable name prefixed by a $ sign.

Setting environment variables

You can set the value of an environment variable by typing its name, an equal sign, then the new value.

No spaces

It is important not to leave a space either side of the equal sign.

.

PRACTICAL EXERCISES

Look at some environment variables

- Inspect the contents of some of the environment variables

  ```
  holyrood$ echo $HOME
  ```

  ```
  holyrood$ echo $PATH
  ```

- Change the contents of a variable

  ```
  holyrood$ PS1='hello there!'
  ```

  Note that quotes are needed to protect the spaces in the value being assigned to the variable

  You can change the prompt back to the hostname by using " `\h` "

  ```
  hello there! PS1="\h $"
  ```

DOT FILES

- Commands to be executed when a program starts
  - .bashrc, .emacsrc
- Configuration file for microemacs (ue)
  - redefine key behaviour
  - set fill characteristics
- ls does not usually list "dot" files
  - need to use -a flag

Most programs can be configured to behave differently according to the taste of the current user. Configuration can often be carried out using command line options, or alternatively options can often be stored in configuration files.

Unix configuration files usually have names which start with a dot, and very often end with the letters ‘rc’ (for "run command"). Thus the configuration file for ue (or microemacs) is called ‘`.emacsrc`’, and the configuration file for bash is ‘`.bashrc`’.

Text files

Unix configuration files are text files, and you can change their contents using an editor such as ue. Make sure you know what you are doing - take copies of rc files before you edit them so that you can put back the original if you get it wrong!

.emacsrc

You have learned how to use the editor called ‘ue’. ue can be made to behave differently by means of settings in its configuration file, `.emacsrc`. This file can contain commands to redefine which keys carry out which commands in ue, and to set ‘fill’ characteristics - whether the editor will start a new line when a line reaches a certain length, or whether it will wait for the user to press the ‘return’ key.

PRACTICAL EXERCISES

Examine a dot file

- ❑ Examine the contents of your .emacsrc file

```
more .emacsrc
```

UNIX AND NETWORKS

- Unix networking very mature
- Unix very much a "building block" of the internet

Unix is well suited to being used with computer networks, which permit users to communicate with computers other than their own.

The following pages provide information about ftp, which permits computer users to exchange files with remote machines, and about telnet, which permits users to log in to remote computers.

More detailed information about the many network facilities associated with Unix is provided in the EUCS ‘Enhancing your UNIX skills’ and ‘Starting to use the Internet’ courses. Netscape, which is a very popular communications package, is covered in some detail in the ‘Starting to use the Internet’ course.

TELNET

- Log on to another Internet computer
  - need to have an account on the other computer!
  - The telnet program makes a connection between your terminal and any computer anywhere on the Internet.

Address another computer

You can specify a computer to log in to either by its IP address or by its DNS name. If the computer is on your local network, you will not need to specify its complete address. Within the Edinburgh network, you should only need to specify ‘holyrood’; this computer’s full address is ‘holyrood.ed.ac.uk’.

PRACTICAL EXERCISES

A telnet session

- Log on to holyrood again, using telnet

  ```
  holyrood$ telnet holyrood
  ```

  Log in

  Satisfy yourself that you can use the computer. Use ps and who to verify you are logged on twice.

  Log out again

- You could use telnet to contact another computer on the Internet in exactly the same way, so long as you have permission to access the remote host.

REMOTE MACHINES AND TELNET

- Log in to remote machines using telnet
  - library catalogues
  - information resources
- Emergency exit!

The telnet program, which was introduced earlier on, enables you to log in to a remote computer and to use some of the facilities running on that machine.

For example, you might telnet to a machine running an on-line library catalogue so that you can browse through that catalogue. Other telnet hosts may provide you with information indexed using menus. Once you have logged in to the host, you will need to choose menu items using letters or numbers to move to the information you are interested in.

Should things go wrong, you can perform an emergency exit from telnet by using ^]. Once the telnet prompt (telnet >) has returned, you can type ‘quit’ to return home.

For more details of telnet and other network facilities (including rlogin, which is an alternative method of logging in to remote computers), EUCS runs an ‘Enhancing Your Unix Skills’ course.

PRACTICAL EXERCISES

Log in to a remote machine and use the facilities there.

- Log in to the Edinburgh University on-line library catalogue. Type

  ```
  holyrood$ telnet catalogue.lib.ed.ac.uk
  ```

  Login as User "`library`". No password is required.

  Follow the instructions given to you on the screen by the remote computer. Remember, if things go awry you can type ^]

Some other telnet hosts to explore:

The National Register of Archives

`public.hmc.gov.uk` (login as `public`)

Interactive subway journey planner

`metro.jussieu.fr 10000`

Note that, although many of these hosts provide menus to allow you to reach information, telnet can also be used to log in to any Unix account provided you have a valid user name and password.

Remember that we have no control over these remote hosts, so if they don't respond, don't blame us!

So you could telnet to your own computer from another, using your user name and password to access your machine remotely. In such a case, you will be able to use the usual Unix commands to use your account.

FTP

- File Transfer Protocol
  - connect to another computer
- Simple commands are available
  - ls, cd
- File exchange
  - send files to the remote computer
  - get files from it
- Simple text commands
  - you can't use bash style ^N ^P
  - can only delete and retype!

The ftp program enables users, once logged on to a computer, to contact another computer and to either send files to the other computer or to receive files from it. You normally need a user account on both computers in order to do this.

Anonymous ftp

File exchange is very common on the Internet so some computers have sections which are available to anyone, not requiring them to quote a user name. They are called 'ftp servers', and can be accessed by 'anonymous ftp', a widely used method of obtaining files across the Internet

Anonymous ftp servers take 'anonymous' or 'ftp' as a user name, and then the caller's electronic mail address (usually usercode@host will do) as the password. Giving your user name as a password is a courtesy which lets the administrators of the ftp server know who is using their service.

Obtaining program files

The ftp program has its own prompt, ftp>, and its own commands. You will use the commands 'binary', 'get' and 'put' most frequently. You will need to use the 'binary' command before you obtain program (binary) files, otherwise you will find that the program will not execute. Typing '?' at the ftp> prompt will obtain a list of the commands available.

To place a file on a remote machine, use the 'put' command. To obtain a file from a remote machine, use the 'get' command. To work with several files at once, you can use the 'mput' and 'mget' commands.

To leave the ftp program, type 'quit' at the ftp prompt.

PRACTICAL EXERCISES

Using FTP

- Connect to a fileserver called src.doc.ic, using anonymous ftp

  `holyrood$`**`ftp src.doc.ic.ac.uk`**

  User name ‘ftp’

  Password ‘*username@host*.ed.ac.uk’

  The ftp program’s prompt is ‘`ftp>`’

- Review the ftp commands

  `ftp>` **`?`**

- You can use the commands ls, ls -l, cd to find out what files are there

  `ftp>` **`ls -l`**

- Leave ftp

  `ftp>` **`quit`**

RETRIEVING FILES WITH FTP

- Users often obtain files from remote machines using ftp
- A useful facility for sharing information

The ftp facility available on many Unix machines was introduced earlier. Being able to obtain files from remote computers, whether these be text files or program files, is a great asset to many computer users.

Using ftp it is possible, for example, to obtain a copy of a piece of software written by another user to perform some useful task or to solve a problem. Alternatively, it may be useful to retrieve a copy of a text file which provides information on some particular topic.

Files are stored on Unix machines all over the world and it can often take only a few seconds to obtain a copy of a file from a remote machine.

In the practical exercise which follows, you will be given a chance to transfer a file from a remote machine to your own machine using ftp.

For more details of ftp and other network facilities EUCS runs an 'Enhancing Your Unix Skills' course.

PRACTICAL EXERCISES

Transfer a file from a remote computer to your own.

- Using the ftp protocol, contact the Unix machine at ftp.ed.ac.uk. Type

  ```
  holyrood$ ftp ftp.ed.ac.uk
  ```

- Use the anonymous login facility, typing '`anonymous`' as your login name and giving your email address as your password.
- Change directory to the directory containing the file to be retrieved. Type

  ```
  ftp> cd pub
  ```

- Check the contents of the directory on the remote machine. Type

  ```
  ftp> ls
  ```

  Note that when you type 'ls' here, you are issuing a command to the remote machine rather than to your own. You know this because the prompt you see on the screen is not the normal prompt, but the ftp> prompt, which indicates that commands will be relayed to the remote computer.

  If, however, you should wish to talk to your own machine whilst using the ftp program, this can be achieved by prefixing each command with a 'shell escape' character, which is the '!' character. So, if you type '!ls', you will see a file listing for your own machine.

- Retrieve the file named README

  ```
  ftp> get README
  ```

  If the file is retrieved successfully, ftp will print a message on the screen confirming this.

- Check to see whether the files has arrived at your machine using a shell escape. Type

  ```
  ftp> !ls
  ```

  You could also type **^z** to suspend ftp, then type **ls** as usual, then **fg** to bring ftp back into the foreground. Whatever.

- Leave the remote machine and return to your own. Type

  ```
  ftp> quit
  ```

Chapter 4

Murder at McGumption Mansion

INSTRUCTIONS

This is the final chapter of the workbook for the course *Introduction to Unix.* It presents a longer and more complex practical exercise than those you have covered already, in the form of a puzzle.

You may not have time to complete every step during today's training hours. However the practical has been designed so that you can take this book away and complete it in your own time if you wish.

Some parts of the puzzle can be solved by long, tiresome repetition. However at each point you can make progress with one or two carefully chosen Unix commands.

As you work through the practical you will be reviewing material covered earlier today and you may find it helpful to refer back to the first three books of the course. If you get stuck there are hints at the back of this book.

First you must obtain a copy of the files used in the practical by anonymous ftp. They are kept on `ftp.ed.ac.uk` in the file `pub/Unix1/murder.tar` - download that. Remember to use binary mode. They are packed with the "tar" command and you should extract them by typing the following:

```
tar xpof murder.tar
```

This will build a directory called `case_notes`. Change into that directory and read the file `README` before you proceed further. If you would like to know more about the tar command for future reference, check the manual page.

Good luck.

MURDER REPORT

Sunday 25th August 1996

Major Duncan McGumption threw a small dinner party at his country mansion just outside Penicuik. His wife and son were at table, plus two guests who were to stay the night. The meal was not as pleasant as it might have been - the Major was loud and rude throughout.

The meal began at 9pm and finished late. Guests and family retired to bed. The night seemed uneventful - but the Major was discovered on his bedroom floor in the morning, dead. A bloodstained brass candlestick lay nearby.

It's murder. And there are only four suspects.

How to solve the mystery

On the following pages are three sets of tasks which you will need to complete to gather information about the murder. There is also a short biography of each suspect. As you proceed you can make deductions about what happened. You may wish to use the whitespace in this workbook to make notes.

Once you have completed all three sets of tasks you should have enough clues to eliminate all but one suspect. You can then check your answer (see page 144) to see if you have caught the murderer!

All the Unix knowledge you need has already been covered in the earlier books of the course. However if you get stuck there are hints at the back of this book to point you in the right direction.

MAJOR DUNCAN MCGUMPTION (DECEASED)

The Major volunteered for the battlefields of France and Italy during World War 2. A cool head and more than a little luck eased him up the ranks and kept him alive until 1945. He stayed with the Army and was posted to various colonies around the globe, including a lengthy tour in India. His career became unremarkable and in 1972 he was honorably discharged with failing eyesight.

Throughout his travels the Major acquired a small collection of native artifacts. With these and a considerable inheritance he settled into the twin pursuits of business and anthropology. For a time his shrewd instincts and uncompromising attitude served him well and he was able to buy outright a country mansion near Penicuik.

Sadly in his last few years the Major became erratic, rude and obsessive, a sour conversationalist and an embarrassment to his family. He was found murdered in his bedroom on the morning of 26th August 1996.

PART ONE

A Map of the McGumption Mansion

You need to consult a plan of the first floor of the McGumption mansion. Unfortunately the only one available is in four pieces.

You will find three of the pieces in the directory:

```
map
```

The other one is **in a subdirectory** of the directory:

```
greek
```

It has only one subdirectory.

- ❑ Collect the four map pieces together in one directory. When you have them all, concatenate them together starting with piece one.

- ❑ You can then print the completed map for later reference.

- ❑ Flick forward through this book and read about the four suspects before going on to Part Two.

PART TWO

The Facts

You now know a little about each suspect. You also need to learn some more information about them and some background to the evening of the murder.

- Look in the directory named:

  `facts`

There are four files in that directory containing facts about the murder and the suspects. However it will not be straightforward to read any of them.

- You need to do three search-and-replace operations to decode the file `facts1`. There is a **hidden file** in the same directory which will tell you which substitutions to make.

- You may not have permission to read the file `facts2` and therefore will need to change the file permissions.

- The full filename of the file which begins `facts3` will be very difficult to type. You will need to find a way to refer to the file without typing the entire filename.

- Getting the information from the file `facts4` is likely to involve two steps. However the file itself should help you with one of those.

PART THREE

Statements by the suspects

Each suspect was interviewed the morning after the murder. If you can obtain the statements of all four, you will have enough information to solve the mystery.

- Look in the directory named:
  `statements`

Each of the four statements is available here.

- Sidney's statement is filed in the directory `sidney`. It is a compressed file.

- Martin's statement is in the file `martin`. However some other text has got mixed up with it. To obtain the correct statement you will have to find a way to extract only lines containing the capital letter **I.**

- Daphne's statement is in the file `daphne` but it has been encoded. You must execute the file `filter1` using `daphne` as standard input, then **pipe** the result through the script `filter2` in the same directory.

- You can read Maria's statement by executing the file named `maria`.

SUSPECT: MRS MARIA MCGUMPTION

The daughter of a minor staffer in the Diplomatic Service, Maria cut free from her conservative upbringing and lived the wild life after graduating from Oxford. She met the Major in India, at an embassy party she had gatecrashed with friends.

Maria fell in love with his energy and devil-may-care attitude and they were married in 1969. Only a few years later she realised that he was entrenched in the same stuffy values she had rejected, and he was actually a boring old trout.

Their marriage has not been a happy one for some years and she would leave the Major were it not for the fact that she has grown to like his money.

SUSPECT: MARTIN MCGUMPTION

Martin's upbringing was a haze of mixed messages. His father hammered on about duty and making one's own way in the world, while his mother smoked Turkish cigarettes and told him to grab what he could while he was still young.

He has responded by becoming a waster, with a dismal education and dull friends. Although he lives in Edinburgh, he survives on his parents' money and is a great disappointment to both.

The Major keeps Martin on a short leash by constantly threatening to cut him out of the will. Thus he is often at the mansion for "family time".

SUSPECT: MS DAPHNE POSTLETHWAITE

Orphaned by the Blitz on London, Daphne applied herself to study with a single-mindedness and determination which she retained through her spell at the London School of Economics and into commerce.

A woman trader was a curiosity in the sixties when she began; nevertheless, Daphne's tireless, hard-nosed approach and instinct for the right price won respect for her. She met the Major in 1991 and they found it easy to cooperate. Thus began a profitable partnership which ended around Christmas 1994 when Daphne abruptly retired from the market to live in a quiet London suburb. She now plays tennis seriously and has won two amateur championships.

SUSPECT: MR SIDNEY BUS

Sidney is the curator of a small museum in Derbyshire. The Major wrote to him occasionally after leaving the Army, with specific questions about various artifacts he had acquired. Sidney was happy to help initially and the letters continued, hinting that the Major might donate some of his discoveries to the museum.

However no donations were ever made and the Major received a lot of free advice from Sidney. Over the last year the Major had become increasingly scornful of Sidney and what he regarded as a small, uninteresting museum which nobody visited anyway.

CHECKING YOUR SOLUTION

Once you have made your deductions and are ready to accuse one of the suspects, there are only a few short steps to let you check your solution.

- ❑ Change into the directory

  ```
  check
  ```

- ❑ Type the following command exactly; it begins with a dot and a space.

  ```
  . go
  ```

Now follow the instructions carefully and you will find out whether you are correct!

HINTS

Part One

Having trouble finding the fourth piece of the map?

- Certain options to the "ls" command will show you which files are subdirectories.

Difficulty putting the four pieces together?

- There are at least two ways you could do this. Either you could append pieces two, three and four to piece one, or you could create a new file to put all the pieces in. You will probably have to know about redirection of standard output.

Can’t print?

- The printer in the teaching lab is very probably a laser printer. You may need to convert your plain text map to Postscript.

Part Two

Can't locate the hidden file?

- An option to the "ls" command will make it show hidden files.

No idea how to do a search-and-replace?

- One way to do it is to use the MicroEMACS editor, "ue".

Don't know how to execute a file?

- Just type its pathname (for example: `./file`)

Can’t get the file to execute?

- Perhaps its permissions are not set correctly.

Part Three

Can’t find Sidney’s statement?

- The command "uncompress" looks for a file with a particular extension to its filename which labels it as compressed.

Martin’s statement looks like gibberish?

- There is a command called "grep" which searches a file for a text string and only prints the lines containing that string.

Unsure how to handle the pipe for Daphne’s statement?

- You must first specify what you want to execute, then what file to use as standard input, then where to pipe the result to. There are particular symbols to indicate standard input and pipe.

Having trouble deleting a directory to get Maria’s statement?

- The command "rmdir" deletes an empty directory. However you perhaps need to use an option to the "rm" command along the way. There are a few different ways to complete this section.

Checking Your Solution

Here you are on your own. Rest assured that the commands you need have all been covered in earlier chapters of the course.

Feedback

Please let us know if there is anything in this document which you didn't find clear, or which we have omitted, so that we may improve future editions. Please send your comments (on a photocopy of this page if you like) to:

G.Chetty,
Information Services Section,
Computing Services,
University of Edinburgh,
Main Library,
George Square,
Edinburgh EH8 9LJ

or by email to: G.Chettyt@ed.ac.uk

Document: Unix 1: Introduction to Unix

Edition: 5, October 20001

Your name and address:

Comments: