Preface

This guide introduces you to the basic features of the UNIX operating system. It aims to give you enough information to get you started with UNIX. It covers the most common UNIX commands and explains how to use the system efficiently.

EUCS provide a range of documents on UNIX, available Online at http://www.ucd.ed.ac.uk/usd/iss/docs/; paper copies are available from EUCS reception areas.

The guide assumes that you are using the central UNIX services offered by the Edinburgh University Computing Services, running on the holyrood and waverley host computers, although it can also be applied to other UNIX systems.

The guide has had several contributors over the years, including Helen Chisholm, Keith Farvis, Mike Holmes, Gavin Inglis, Rob McCrone, David Neilson, Jean Ritchie, Arthur Wilson and Sam Wilson - to whom all due acknowledgement is gratefully made.

Gill Chetty (Ed)
August 2002

Copyright © EUCS 2002

Permission is granted to any individual or institution to use, copy or redistribute this document whole or in part, so long as it is not sold for profit and provided that the above copyright notice and this permission notice appear in all copies.

Where any part of this document is included in another document, due acknowledgement is required.
## Contents

Preface .................................................................................... iv

1 Introduction ........................................................................... 1
  1.1 Aim of this Guide ................................................................. 1
  1.2 Printing conventions used in this guide ............................. 1
  1.3 Special Keys ......................................................................... 1
  1.4 UPPER & lower case ............................................................ 2
  1.5 The System prompt .............................................................. 2

2 Getting Access to a UNIX computer ..................................... 3
  2.1 Registering as a user ............................................................ 3
  2.2 Accessing UNIX Hosts .......................................................... 3

3 Logging on to UNIX, Passwords, logging off ................. 5
  3.1 Logging on ............................................................................. 5
  3.2 Your password ...................................................................... 5
  3.3 Logging off ........................................................................... 6

4 Getting on-line Help .......................................................... 7
  4.1 The man command ............................................................... 7
  4.2 UNIXhelp ............................................................................. 8

5 UNIX Commands .................................................................. 9
  5.1 Commands ............................................................................. 9
  5.2 Commands and arguments .................................................. 9
  5.3 Options ................................................................................ 10
  5.4 Redirecting standard input and output ............................... 10
  5.5 Pipelines ............................................................................... 11
  5.6 UNIX shells ......................................................................... 12

6 Files and how to create them ............................................ 13
  6.1 What is a file ......................................................................... 13
  6.2 File names ............................................................................ 13
  6.3 How to create a file ............................................................. 13

7 Finding out what files you have ....................................... 15
  7.1 Looking for related sets of filenames: wild cards ............. 15
  7.2 Getting more information about your files ....................... 16
  7.3 Hidden files ........................................................................ 17

8 Finding out what is in a file ............................................... 18
  8.1 Using the file command ....................................................... 18
  8.2 Inspecting a file ................................................................... 18
  8.3 Using the cat command ....................................................... 18
  8.4 Using a pager ....................................................................... 19
  8.5 Using an editor .................................................................... 19

9 Printing your files ............................................................ 20
  9.1 Types of printer ................................................................... 20
  9.2 Printer names ...................................................................... 20
9.3 Sending files to a printer ................................................................. 20
9.4 PostScript Files ............................................................................. 21
9.5 Checking if your file has been printed .......................................... 22
9.6 Cancelling a print request ............................................................ 22

10 Manipulating your files ................................................................. 23
10.1 Copying a file .............................................................................. 23
10.2 Joining files together ................................................................. 23
10.3 Renaming a file ........................................................................... 23
10.4 Deleting a file ............................................................................. 24
10.5 Using wild card symbols ......................................................... 24

11 Grouping your files ...................................................................... 25
11.1 The UNIX file system ................................................................. 25
11.2 Path names .................................................................................. 25
11.3 Important directories ................................................................. 26
11.4 Creating new directories ............................................................ 26
11.5 Moving to another directory .................................................... 27
11.6 Putting files into a directory ...................................................... 27
11.7 Listing directory contents .......................................................... 28
11.8 Deleting directories .................................................................. 29

12 Exchanging files with another user ........................................... 30
12.1 File security .............................................................................. 30
12.2 Changing permissions: the chmod command ......................... 32
12.3 Setting modes numerically ....................................................... 33
12.4 Accessing files belonging to another user .............................. 34

13 Customising your Environment .................................................. 35
14 MicroEMACS: a text editor .......................................................... 37
14.1 Introduction to MicroEMACS ....................................................... 37
14.2 Starting the ue Editor ................................................................. 37
14.3 Moving the cursor around the file ............................................. 38
14.4 Deleting characters in a line ...................................................... 38
14.5 Quitting ue ................................................................................ 39
14.6 Inserting another file ................................................................. 39
14.7 Deleting chunks of text .............................................................. 39
14.8 Searching .................................................................................. 40
14.9 Search and Replace .................................................................. 40
14.10 Repeating a ue command many times ................................... 41
14.11 Redrawing the Screen ............................................................... 41
14.12 Reference Card ....................................................................... 41
14.13 Text processing: TeX and LaTeX ............................................. 41

15 Communicating by Mail and News ............................................. 42
15.1 Electronic Mail .......................................................................... 42
15.2 Network News ........................................................................... 42

Appendix A: Further Reading .......................................................... 43
UNIX operating system .................................................................. 43
Editors ......................................................................................... 43
Electronic Mail ............................................................................. 43
1 Introduction

1.1 Aim of this Guide

This guide introduces all the basic features of UNIX, and aims to tell you everything you need to know when starting to use UNIX. It also provides pointers to sources of information which will be required for learning more advanced skills.

Versions of UNIX are in use on workstations, personal computers and mainframes at business and academic sites all over the world. This guide applies to the large central UNIX service offered by Computing Services (EUCS) running on the holyrood and waverley host computers. Most of the information given can also be applied to other UNIX systems, notably workstations made by Sun and Compaq, and to other mainframe computers and supercomputers, which are available in individual departments rather than through EUCS. UNIX systems, are also available for personal computers, for example Linux.

UNIX is so widely available that there are many books and manuals about it. A list of EUCS documentation on UNIX is given in Appendix A.

1.2 Printing conventions used in this guide

In this guide we use various conventions for items you type or particular keys you need to press, and for information you see displayed on your computer screen. These are as follows:

- Examples of output which you would see on your terminal screen or line printer listing, will be shown:

  in this font

- Examples of input which you type in to the computer are shown:

  in this font

- At some points, you do not need to type in the words shown but information of that type. This is indicated by this font. For example if the following is shown in the document:

  bash$ cp from to

you would replace from and to with the names of the files you want to work with; for example, to make a copy of file1 called file2 you would actually type:

  bash$ cp file1 file2

1.3 Special Keys

[Return]

[Return] means that you press the return key – usually marked ‘return’, ‘enter’ or ↓. Information is usually sent to UNIX in complete lines, so when you have finished typing a line you should press the [Return] key. Reminders to use the [Return] key are included until the end of Chapter 3, after which it is assumed that readers will remember it. Some UNIX programs act on single keystrokes and do not need the [Return] key to be pressed: these will be noted when they are covered in this guide.
**[Del] or [Delete]**

If you make a mistake while typing and you notice it before you press the [Return] key you can change what you have typed. The key marked ‘delete’, ‘del’ or ← (usually near the top right of the keyboard) will erase the last character typed. Pressing it for a second time will erase the character before that and so on back to the start of the line. Programs which act on single keystrokes will generally ignore this key.

**[Control]**

Sometimes you need to use two keys together. Just as you can hold down the [Shift] key while typing a letter to send the upper case version of that letter to the computer, so you can hold down the [Control] key to send a different, usually invisible, form of that letter. The [Control] key is often marked CTRL or CNTL, and is usually in the bottom left hand corner of the keyboard. For example, CTRL-N means “hold down the [Control] key, press and release the N key, then release the [Control] key”. *(Note that although control characters are usually shown in upper case, as here, you do not have to hold down the [Shift] key when typing them.)*

1.4 UPPER & lower case

UNIX recognises the difference between UPPER and lower case letters, so the correct case must always be used. UNIX commands are usually in lower case. Usernames are always in lower case.

1.5 The System prompt

The examples in this guide are for a UNIX ‘shell’ called Bash (the ‘Bourne again shell’), and show the system prompt as bash$. In practice the prompt is more likely to be $ preceded by the name of the host computer you are using, for example holyrood$. You will have a different system prompt if you use another type of UNIX system, or ‘shell’. Shells on UNIX are described at the end of Chapter 5, and in the document *Shells and Shell scripts – a Guide to Tailoring your UNIX Environment*. 
2 Getting access to a UNIX computer

UNIX is a multi-user operating system, in other words it allows more than one user to have access to a computer at the same time. The users share the computer's various resources. Although each user has only a small portion of the resources of the machine at their disposal, the computer operates so fast that users have the impression of complete control over it.

To use a UNIX service you need to be registered as an authorised user, and you need some kind of desktop machine (PC, AppleMac, X terminal or workstation) through which to access it.

2.1 Registering as a user

To register for any service operated by EUCS use the Service Registration form which is available from the EUCS Sales Points. In order to be able to use UNIX you need a username and password. These will be given to you by the System Administrator when you register as a user.

Username

Your username is a unique code which identifies you and your files. In common UNIX parlance your username is also known as your ‘login’ or ‘account’. A common convention on UNIX systems is to use given names or initials as usernames.

Your username gives you access to an account – a share of all the resources of the computer. Within your account you can store information and data, and have complete and exclusive control of whatever is stored, ensuring that your work is protected from the activities of other users using the system at the same time.

Password

When you are first accredited to the system you will be given a password. You have to use this password to gain access to the UNIX system.

Warning: Your password is private to you, should be changed regularly, and should never be revealed to anyone else.

More information about passwords and how to change them is given in section 3.2, and a useful note called Passwords is available from EUCS Sales Points.

2.2 Accessing UNIX Hosts

You can access the central UNIX services through:

- a PC running telnet or eXceed
- an AppleMac running telnet
- an X-terminal or workstation
You can access the central UNIX services through any of the machines in the Open-access Computing Labs. The publications *Open-access Computing Labs, How Computers can Help your Studies* and *Computing at Edinburgh* give their locations, opening hours and the type of equipment available.

Appendix C describes how to connect to a UNIX computer from a PC or Macintosh in one of the Open Access Computing Labs.
3 Logging on to UNIX, Passwords, logging off

Getting connected to your UNIX account is known as ‘logging on’. Disconnecting your desktop machine from UNIX is called ‘logging off’. (Some people use the terms ‘logging in’ and ‘logging out’.)

3.1 Logging on

You log on to a UNIX computer as follows:

➲ If necessary, switch on your desktop machine, and if it’s a personal computer or AppleMac, run a suitable communications program (see Appendix C for help with this, or if logging on from the Open Access Labs).

➲ When prompted for your login, identify yourself by typing your username and then press the [Return] key.

➲ Type your password, then press [Return]; notice that the computer does not show your password, so that there’s no chance of anyone overseeing it.

You may get the message: Login incorrect

This means that you have not managed to log in. You may have:

• typed your username incorrectly, spelling it wrongly or using upper case letters.
• typed your password incorrectly (as for username).

Either way, you will be prompted with login: so you can try again.

If your username and password are accepted, the screen displays some information about the computer, finishing with a line prompting you for a command. The ‘command prompt’ will probably be in the form of the computer’s name followed by $ (for example holyrood$).

You should keep an eye on the messages which appear when you log on. In particular, the ‘Message of the day’ gives you information about the system, perhaps telling you about a new package, or warning you that the computer is going to be inaccessible during a maintenance period. You might also see the line: You have new mail, which means that there are new email messages waiting for your attention (documentation about electronic mail is available from EUCS Sales Points).

When you log on to an EUCS central service for the first time, you will be led through a self-explanatory dialogue which will set up your UNIX ‘environment’ to suit your personal needs.

3.2 Your password

When you are first accredited to a UNIX system you are given a password. You need to use this password to gain access to the UNIX system.
In order to reduce the risk of other people using your account, and perhaps altering your files or running up charges for computer time, you should take the following precautions:

- never reveal your password to anyone. Giving away your password has security implications for the whole UNIX system, and is a breach of the University's Computing Regulations.
- change your password periodically, by using the command `passwd` (see next section)
- choose an obscure password (not your initials or the name of your department) preferably between 6 and 8 characters long, of mixed case and containing at least one non-alphanumeric character. (The note *Passwords*, published by EUCS, suggests how to choose a good password.)

### Changing your password

The command to use is `passwd`. Type the command:

```bash
passwd
```

then press the [Return] key).

You are prompted for your old password first, as a security check that it is really you that’s asking. If you give the old password correctly, you are prompted for a new password. Type this in. Note that passwords are never shown on the screen, so to guard against typing mistakes you are next asked to retype the new password (if you had made a mistake you would be locked out of your account without this verification). The password is changed if the verification is correct. Note that UNIX passwords are case-sensitive (watch out for the ‘Caps Lock’ key being inadvertently left on).

There is no way in which the system can ever reveal your password. If you forget your password, you can neither change it (if you are logged on), nor log on (if you are not logged on), nor can support staff ‘look it up’ for you. In this case contact your departmental computing representative or your User Support Team who will arrange for a new password to be registered on your UNIX account.

### 3.3 Logging off

It is important to log off when you finish your work so that nobody else can misuse your account, and also so that other people can log on to the desktop machine. To log off, simply type the command:

```bash
logout
```

This is as usual followed by the [Return] key.

So far [Return] has been shown wherever it is necessary. It is not shown in subsequent illustrations, but do remember that usually no information is transmitted to the computer until you complete a line by pressing the [Return] key. One effect of this is that you can correct errors in a line using the [Delete] key and retyping, before sending the line to UNIX.
4 Getting on-line Help

4.1 The man command

You will probably need help from time to time on the UNIX commands and facilities available. You can consult written documents, or use the on-line help system.

The UNIX help system is based on the UNIX system manuals, which are held on-line. These manuals are generally aimed at more experienced users and programmers, but comprehensive descriptions of all the usual commands are included. As you gain experience you will be able to interrogate the system yourself to find new features.

The command for accessing the manual information is man (short for manual), which gives you the page of the UNIX manual describing a particular command.

To get help about any UNIX command, use the command:

```
man commandname
```

For example, to get help about the command cp (which is used to copy files):

➲ Make sure you are at the usual Shell prompt (for example holyrood$)

➲ Type:

```
man cp
```

You will see a screenful of information about the cp command, ending with the prompt —More—, as shown here:

![man cp output]

If you want to see the next screenful, press [Space].

If you want to see just the next line, press [Return].

If you find the information you need, you can type q to quit at any time.

“More” is described further in “Using a pager” in section 8.4.
Note: arguments and options in square brackets \[ \] are optional, anything outside brackets is required. There is more information on arguments and options in Section 5.

4.2 UNIXhelp

‘UNIXhelp’ is a self-teaching tutorial on the World Wide Web: it is located at \http://unixhelp.ed.ac.uk\, and you can find information in it simply by clicking on items of interest with the mouse (if using a Mac, PC or X Windows), or by moving the cursor around menus if using a character display.
5 UNIX Commands

5.1 Commands

A UNIX command name is usually quite terse, often consisting of just two or three letters, but often resembling the English verb which best describes the command’s purpose. Typical examples are `cp` for copy and `rm` for remove.

This Guide introduces you to some of the basic commands you will need as you start to use UNIX.

5.2 Commands and arguments

UNIX commands usually take the form:

```
bash$ command options argument [argument argument ...]
```

Most commands require arguments which are the names of files or other information to which the command applies. For some commands arguments are obligatory, whilst for others they are optional.

Commands with optional arguments

The following are some commands where arguments are optional.

The `date` command prints out the date, time and time zone:

```
bash$ date
Tue Aug 29 10:04:47 BST 1999
bash$
```

The `who` command lists users who are logged on, gives their connection number, the date and the time at which they logged on and – on the central EUCS UNIX services – whereabouts on the network their desktop machine is:

```
bash$ who
jfc pts/157 Aug 14 09:51 (incip.ee.bru.ed.ac.uk)
rah pts/570 Aug 29 09:41 (rairdh.dcs.ed.ac.uk)
angusr pts/547 Aug 24 09:34 (tori.ucs.ed.ac.uk)
bash$
```

The `ls` command (short for ‘list’) gives a list of your files in alphabetical order:

```
bash$ ls
file1   file2   file3
bash$
```

The `ls` command is described more fully in “Finding out what Files you have”, in chapter 7.
5.3 Options

Options (also known as flags) modify the way commands work. They usually consist of a single letter preceded by a hyphen. A command with an option has the following form:

```
bash$ command options argument [argument argument...]
```

For example the `wc` command counts the number of words, characters or lines in a file. By using the options with this command you can choose what is counted:

```
bash$ wc -w file1           (counts the words)
bash$ wc -c file1           (counts the characters)
bash$ wc -l file1           (counts the lines; that's letter L, not digit one)
```

Multiple Options

Unfortunately the way different commands deal with options is not standard. Most commands accept multiple options grouped together with one hyphen, others require each option to have its own hyphen:

```
bash$ command -abc argument
bash$ command -a -b -c argument
```

You can use the `man` command to find in which form a particular command requires its options: see “Getting on-line Help” in Chapter 4.

With the `wc` command the default is that all three options are assumed, or they can be combined with a single hyphen:

```
bash$ wc file1               (counts words, characters and lines)
bash$ wc -wcl file1          (counts words, characters and lines, too)
bash$ wc -cl file1           (just counts characters and lines)
```

With the `lpr` command the options are given separately. For example, to print a file called `fred` on `lp15` and delete `fred` after printing you would use the command:

```
bash$ lpr -Plp15 -r fred
```

The rest of this Guide will introduce you to new useful commands but will assume that you know how to use commands with options and arguments.

5.4 Redirecting standard input and output

All information stored for you by the system is organised into files; see “Files and how to create them” in Chapter 6 for further details. UNIX considers any collection of information to be a file. It also regards every device attached to the computer as a file – so even the input from your desktop machine is a file to UNIX.
Unless you instruct UNIX otherwise, many commands will read input from your desktop machine and write the output to it. Therefore, by default, your desktop machine is connected as the “standard input file” and “standard output file”. This can be changed by redirecting input and output from and to any other file.

The > (greater than) character redirects output to a specific file, as in this example:

```
bash$ date > file1
```

Here the output from the date command is redirected to `file1`. If you then used the `cat` command to list this file you would see the command’s output:

```
bash$ cat file1
Tue Aug 29 10:04:47 BST 1999
```

Redirecting output to a file creates the file or, if it already exists, will overwrite the previous contents. If you wish to avoid this you can use the `>>` operator (two greater than signs) and the output will be added to the end of the file if it already exists.

Similarly the < (less than) character is used to redirect the input from any file into the given command, rather than input coming from the desktop machine.

In this example both redirection characters are used:

```
bash$ cat < file1 > file2
```

This means “print the contents of `file1` and direct it into `file2” — that is, copy `file1` into `file2`. This makes use of the fact that `cat` with no argument simply reads from its standard input and writes to its standard output. There is, of course, a better command to copy a file (it is called `cp` and is described in Chapter 10).

### 5.5 Pipelines

As well as directing output to files, UNIX commands can send their output to other commands. Piping allows you to connect the standard output from a command on the left-hand side of the pipe (the | character) to the standard input of another command on the right-hand side. The two commands run concurrently.

For example the command `ls` used with no options gives a list of the files you have. The command `wc` used with the -w option counts the number of words in a file. Thus if you were to redirect the list of file names produced by `ls` into a file, you could use `wc` to count how many files you have (one ‘word’ per file name). However, this would require you to create an extra file to contain the list of file names and, to be tidy, you would want to delete that file again after you had used it. UNIX pipes can often be used to avoid the use of temporary files like this. For example you can pipe the output from the `who` command directly into the `wc` command, and the output from `wc` gives the number of users who are logged on:

```
bash$ who | wc -l
109
```
5.6 UNIX shells

When you think you are talking to UNIX you’re actually talking to a very versatile program called the Shell. The Shell accepts your commands and works out what you want done before passing on instructions to other programs which do the real work. Nearly all commands in UNIX are simply programs of one sort or another.

With UNIX you can place a number of UNIX commands in a file and run them at one time. A file containing shell commands is known as a shell script.

High-level programming features such as flow control and conditional statements can be included in a shell script; this gives you the power to write shell scripts which behave like high-level programs. For more information about this, see the EUCS document *Shells and Shell scripts – a Guide to Tailoring your UNIX Environment*.

Several varieties of shell are available, the most common being the Bourne shell, the C shell, the Korn shell and ‘bash’ (Bourne again shell). The last two are based on the Bourne shell syntax but have a much more sophisticated interface. The TC shell is another shell with a sophisticated interface but based on the C shell.

The default for interactive use on central EUCS UNIX services is bash (which is used for this Guide). Bash provides a flexible and powerful user interface, including:

- screen-editor style functions to edit command lines
- a history mechanism, which stores a list of previously-used commands so they can be edited and re-submitted
- the automatic completion of partially specified file and command names

Bash is described in the on-line text accessed by man bash and in the document *Shells and Shell scripts – a Guide to Tailoring your UNIX Environment*. 
6 Files and how to create them

6.1 What is a file

Your computing work will include handling collections of data or text kept in files. A file is a collection of information with a name. Typical examples of things that might be kept in files are computer programs, experimental data, address lists and documents such as this one. One of the features of UNIX is that it treats almost everything as a file, including your desktop machine and other devices such as printers. UNIX recognises many types of file, for example, ordinary, special and directory.

6.2 File names

Each file has a name. A file name may be any sequence of up to 255 characters, and UNIX distinguishes between upper and lower case characters. The only character not allowed in a file name is the slash character / (and if you use some characters such as * ? - etc you may find some difficulty). Thus the following are valid file names:

animal47
89memos
newdat
output.from.SPSS.on.960715

The following file names are different:

Animal47
animal47

Each file that belongs to you has a name that must be unique, that is, you cannot have two files within the same directory with the same name.

6.3 How to create a file

You will very rarely have to create a file explicitly. File creation normally happens as a by-product of another operation that you do. Thus you can:

• use an editor with a file which does not already exist. (See Chapter 14 for a brief introduction to the MicroEMACS editor);
• make a copy of a file (for details on how to copy files see “Manipulating your files” in Chapter 10, or “Exchanging files with another user” in Chapter 12);
• run a program, such as a statistics package, which puts its results in a file (you will need to consult the package’s documentation to find out how this is done);
• compile a program (programming is not covered in this Guide);
• use the cat command to put text into a file. The following example shows how to create a file called fred and insert the text “a line of text” into it:

bash$ cat > fred
a line of text
CTRL-D
The sequence of events in this last example is as follows:

➲ type the command line:

\texttt{cat > fred}

The \texttt{cat} command with no argument reads from the standard input (your desktop machine) and writes to the standard output – which is normally your screen, but is here redirected with the \texttt{>} operator to the file \texttt{fred};

➲ type a line of text (or indeed any number of lines), and \texttt{cat} sends this into the file;

➲ type \texttt{CTRL-D}

This tells the system to stop the \texttt{cat} command and return to the command prompt.

\textit{Note: if you specify an existing filename when using \texttt{>}, the file will be overwritten with no warning, unless the shell variable noclobber is set. For more details on shell variables see the EUCS document Shells and Shell Scripts – A guide to Tailoring Your UNIX Environment.}
7 Finding out what files you have

All the files you own are held together in a group known as a directory; within your directory the files may be grouped into further directories created as you need them. There can be many depths of directories within directories in your account. This is all explained in “Grouping your files”, in Chapter 11.

Use the command `ls` to see what files you have:

```
bash$ ls
Johnon  Johnstone  MacJohnston  bin  oldlogin
Johnson  Littlejohn  Mail  dl
Johnston  Mac  Johnson  News  edlan
bash$
```

7.1 Looking for related sets of filenames: wild cards

The `ls` command can be used in a number of ways: you can use these in order to make a selective examination of the list of your files. This makes use of a device called a wild card symbol.

A wild card is a symbol which can represent any character – rather like a joker in a pack of cards. The UNIX shell recognises two wild card symbols:

- `*` stands for zero or more characters
- `?` stands for exactly one character.

**Example**

The example above contains seven obviously related file names containing the word `john`. Different groups can be selected depending on the type of wild card symbols used and their location, as follows:

- **Remember that UNIX distinguishes between upper and lower case.**

```
*John* selects  Johnson  Johnson  Johnston  Johnstone
              MacJohnson  MacJohnston
John* selects  Johnson  Johnson  Johnston  Johnstone
*john* selects  Littlejohn
Mac* selects  MacJohnson  MacJohnston
*ston* selects  Johnston, Johnstone, MacJohnston
*ston? selects  Johnstone
```

If you want to check what files you have starting with “Mac”, type:

```
bash$ ls Mac*
MacJohnson  MacJohnston
bash$
```

If you try to use a wild card which doesn’t match any of your files you will get a message such as “No such file or directory” (the precise wording depends on the shell you are using).
7.2 Getting more information about your files

To obtain more information about your files you can use the `-l` (‘long format’) option with `ls`, thus:

```
bash$ ls -l
total 26
-rw------- 1 efgu02 biotech 7 Aug 20 14:00 Johnson
-rw------- 1 efgu02 biotech 10 Aug 20 14:01 Johnstone
-rw------- 1 efgu02 biotech 7 Aug 20 14:02 Johnstone
-rw------- 1 efgu02 biotech 7 Aug 20 14:02 Johnstone
-rw------- 1 efgu02 biotech 11 Aug 20 14:02 Littlejohn
-rw------- 1 efgu02 biotech 14 Aug 20 14:06 MacJohnso
-rw------- 1 efgu02 biotech 14 Aug 20 14:06 MacJohnston
drwxr-xr-x 2 efgu02 biotech 1024 Jul 24 09:50 Mail
drwxr-x--- 2 efgu02 biotech 1024 Aug 17 19:02 News
drwxr-x-x 2 efgu02 biotech 1024 Jun 5 13:35 bin
-rw------- 1 efgu02 biotech 514 Aug 28 14:30 dl
drwxr-x--- 2 efgu02 biotech 1024 Aug 30 18:24 edlan
-rwxr-x--- 1 efgu02 biotech 483 Jun 5 13:34 oldlogin
bash$
```

The first line printed (‘total 26’) tells you how much space your files occupy on the computer’s disks, in blocks, and each subsequent line gives details of one file. Each line of file information consists of several items:

- **10 characters showing the file’s type and modes.** A `d` in the first column denotes a directory. More explanation of these items is given in “File security” in Chapter 12.
- **A number which is the number of links to the file.** “Links” are not explained in this guide, but the number will normally be 1 for a file and 2 or more for a directory.
- **The username of the file’s owner.** This will normally be your own username.
- **The name of the group.** “Groups” are explained in Chapter 12.
- **The length of the file in bytes.**
- **The date and time when the file was last modified.**
- **The name of the file.**

Of these items the most useful are the name of the file, the date and time of its last modification, and its length (in bytes).

You can combine options and wild cards; for example:

```
bash$ ls -l Mac*
-rw------- 1 efgu02 biosup 14 Aug 20 14:06 MacJohnson
-rw------- 1 efgu02 biosup 14 Aug 20 14:06 MacJohnston
bash$
```
Finding out what files you have

7.3 Hidden files

When working with UNIX you may come across references to files whose names begin with a full stop: for instance files called `bashrc` and `bash_profile` contain details of how you want your shell to appear, and `emacsrc` stores your preferred options for use with the MicroEMACS editor. Such files contain ‘administrative’ information; they are usually created automatically for you by the system, and most of the time you don’t need to know about them.

`ls` normally hides any files whose names begin with a full stop, so they don’t distract you from the ‘regular’ files on which you are usually working. You can see all the files you have, including the hidden ones, by using `ls` with the `-a` (for ‘all’) option:

```
bash$ ls -a
.
..  .emacsrc  .reamrc  Littlejohn  dl
..  .inputrc  .rhosts  MacJohnson  dlan
.bash_history  .lastlogin  Johnson  MacJohnson
.bash_profile  .login  Johnson  Mail
.bashrc  .logout  Johnston  News
.cshrc  .profile  Johnstone  bin
bash$
```

If you want to know more details of all your files you can combine the `-a` and `-l` options and give the command:

```
bash$ ls -al
total 192
drwxr-x--- 2 efgu02  biosup 1024 Aug 28 18:24 edlan
-rwxr-x--- 1 efgu02  biosup  483 Jun  5 13:34 oldlogin
bash
```

Note: The `.` and `..` shown as the filenames in the first two lines of this output are not typing errors! They have a special significance, which is described in “Path names” in section 11.2.
8 Finding out what is in a file

8.1 Using the file command

A file is a unit of storage. In UNIX a file can contain anything: a program, executable object code, text and so on – all are just sequences of raw data until they are interpreted by the right program.

The `file` command can be used to discover what sort of data is in a file. This command can produce a variety of messages, but so long as the message includes one of the words “text” or “script” then it is probably safe to list the file to your screen or a printer. For example:

```
bash$ file Littlejohn
Littlejohn: ascii text
bash$
```

8.2 Inspecting a file

There are several methods for inspecting a file on the screen. However, some files – for instance executable programs – can give strange results if you try to display them on your screen or on a printer. You can use the file command to find out what's in a file before you:

- use the command `cat`: this lists the contents of a specified file to the screen (see the next section);
- use a pager: a pager is a command whose basic function is to output the contents of a specified file one screenful at a time. Most pagers have many other capabilities built in, such as searching for a particular piece of text. Common pager commands are `more` and `less` (see “Using a pager” in section 8.4);
- The `more` command is used in the examples in this Introductory Guide because it is the default pager on the EUCS central UNIX Services.
- use an editor: this is especially useful if you want to make changes to a file when inspecting it (see “Using an editor” in section 8.5);
- print the file: this is discussed in “Printing your files” in Chapter 9.

8.3 Using the cat command

The `cat` command has the simple function of listing the specified file or files to the standard output, in this case the screen:

```
bash$ cat filename
```

However, the more usual and reliable way to view a file like this is to use a pager, particularly if the file is larger than one screenful.
8.4 Using a pager

You can use the command `more` to inspect part or all of a file. With `more`, a new page is obtained on the screen by pressing the [Space] bar. To list the first screenful of a file (22 lines on a standard display) use the command:

```
bash$ more filename
```

This lists the first screenful, then gives the prompt —More—. When you have read all you want, you can use one of the following commands:

- [Space] Display next screenful of text.
- [Return] Display next line of text.
- q Exit from more: this can be done at any time.
- d Scroll forwards about half a screenful of text.
- b Skip backwards one screenful of text.
- h Display a list of commands (help).

8.5 Using an editor

You can use an `editor` to move around a file. An editor is a program that allows you to create and alter the contents of a file. There are lots of different UNIX editors, each with its own set of commands.

Chapter 14 briefly introduces MicroEMACS, one of the most popular UNIX editors.
9 Printing your files

One of the most convenient ways of seeing what is in a file is to have it printed on paper. Having information printed is also known as "getting hard copy".

9.1 Types of printer

EUCS provides access to different types of printer:

- **laser printers**: good quality A4 or A3 paper, very high print quality. This includes PostScript printers (see section 9.4). Many graphics and text formatting programs produce PostScript output which can be sent to a laser printer. See the EUCS charges document (available from EUCS Sales Points) for costs.
- **line printers**: fast, draft-quality, continuous paper, produce output quickly.

9.2 Printer names

EUCS printers and plotters have unique names recognised by most Edinburgh computers. The names have two parts:

- an abbreviation for the device type (such as 'lp' for Line Printer).
- a code or abbreviation to identify the location of a device.

**Examples**

- lp25  EUCS Appleton Tower line-printer
- pskbc2a  PostScript printer in KB Centre lab level 2
- csjcmb  A4 Colour Printer - EUCS job reception JCMB level 3

There are many more devices than shown here: see the *EUCS Printers and Plotters* Reference Card for more details. You should obtain permission before using a departmental printer.

9.3 Sending files to a printer

To send a file to any printer connected to the network, you will need to use the command `lpr`. Various options are available which allow you to specify which printer is to be used, whether the file contains format effectors, etc. All the options are described in the manual page for `lpr`; only a few are shown here.

**Note**: *Send your file to a printer which suits it: send a plain text file to a line printer; send PostScript files to a PostScript printer (see next section).*

You can access the manual page for further details by typing

```
bash$ man lpr
```
Examples of the use of lpr:

```
bash$ lpr file          prints file on your default printer
                       (see end of this section)
bash$ lpr -Pprinter file selects a different printer: note that the 'P'
                       before printer is in upper case
bash$ lpr -Gform file   sends file to form queue form
```

For example, to print a file called fred on your default printer you would use the command:

```
bash$ lpr fred
```

To print fred on ps15, on A3 paper, use the command:

```
bash$ lpr -Pps15 -Ga3 fred
```

To find out what your default printer is, use the command:

```
bash$ printenv PRINTER
```

If this does not produce any output then your default printer is the main one connected
to the system you are using. Details of how to change your default printer are given in
“Customising your Environment” in Chapter 13.

### 9.4 PostScript Files

PostScript is a ‘page description language’ which allows a computer to send both text
and graphics to a laser printer. Most word-processing and desktop publishing programs
produce PostScript output ready to send to a laser printer, bringing high-quality printing
within easy reach.

‘Plain’ text, such as an email message, can be sent to a PostScript printer, if it is
converted into simple PostScript first. Two straightforward utilities – ‘enscript’ and
‘a2ps’ – are available on most UNIX systems to do this, and are used in these ways:

```
bash$ enscript -Pprintername filename
bash$ a2ps filename | lpr -Pprintername
```

(Note the use of the pipe character | in the second example: see section 5.5.) ‘a2ps’
automatically numbers the lines in its output; to suppress this, use the form:

```
bash$ a2ps -nn filename | lpr -Pprintername
```
9.5 Checking if your file has been printed

There is often a queue of documents from all over the network waiting to be printed. To check if your documents are still waiting to be printed type

\$ \texttt{lpq}

which will show what files are queued for your usual printer, or

\$ \texttt{lpq \ -P\ \textit{printer}}

which will show you the queue for the named printer. If you used \texttt{lpr} with a \texttt{-P} option you should use \texttt{lpq} with the same option.

The central UNIX service will respond either with a message saying “no entries”, meaning that the queue for the printer is empty and that your document has already been printed, or it will display something like this:

\$ \texttt{lpq \ -P\ lp25}

\texttt{Printer\ lp25@poirot\ 'line\ printer\ Appleton\ Tower'}

<table>
<thead>
<tr>
<th>Rank</th>
<th>Owner</th>
<th>Pr</th>
<th>Opt</th>
<th>Job</th>
<th>Host</th>
<th>Files</th>
<th>Size</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>active</td>
<td>ercc99 Z</td>
<td>-</td>
<td>-</td>
<td>113</td>
<td>holyrood</td>
<td>fred</td>
<td>2812</td>
<td>Aug 5 10:57</td>
</tr>
<tr>
<td>2nd</td>
<td>ercn20 Z</td>
<td>-</td>
<td>-</td>
<td>114</td>
<td>holyrood</td>
<td>prog.1</td>
<td>4322</td>
<td>Aug 5 10:59</td>
</tr>
<tr>
<td>3rd</td>
<td>tony Z</td>
<td>-</td>
<td>-</td>
<td>115</td>
<td>holyrood</td>
<td>temp</td>
<td>124</td>
<td>Aug 5 11:01</td>
</tr>
<tr>
<td>4th</td>
<td>jamier Z</td>
<td>-</td>
<td>-</td>
<td>116</td>
<td>holyrood</td>
<td>atest</td>
<td>786</td>
<td>Aug 5 11:02</td>
</tr>
</tbody>
</table>

This shows that ercc99’s file \texttt{fred} is currently printing, and that ercn20’s file \texttt{prog.1} is the next item in the queue.

9.6 Canceling a print request

You can cancel a print request by removing one or more of your files from the printer queue, using the command \texttt{lprm}. This command has the form:

\texttt{bash\$ lprm \ -P\textit{printer}\ jobnumber}

Here, \texttt{-P\textit{printer}} has the same meaning as for \texttt{lpr} (section 9.3), and \texttt{jobnumber} is obtained from the output of \texttt{lpq}. Thus, continuing the example of the previous section, user \texttt{ercn20} could cancel the printing of file \texttt{prog.1} on line printer \texttt{lp23} by typing

\texttt{bash\$ lprm \ -P\ lp23 114}

and the system would respond with

\texttt{Printer\ ‘lp23’ (holyrood): removing\ prog.1\ holyrood,\ job\ 114\ owner\ ercn20}

\texttt{bash\$}
10 Manipulating your files

This section describes how you can:

- copy a file
- join files together
- rename a file
- destroy a file
- use a wild card symbol when selecting files to be manipulated.

10.1 Copying a file

To create an exact copy of a file use the `cp` command (short for 'copy'):

```
bash$ cp filename newfilename
```

where `filename` is the name of the file to be copied, and `newfilename` is the name of the file in which the copy is to be placed. If `newfilename` is the name of a directory, the file is copied into that directory with its existing name. See “Grouping your files” in Chapter 11 for a description of directories.

For example, to make a copy of `fred` called `fred2`, type:

```
bash$ cp fred fred2
```

Warning: if you choose a new `filename` that already exists, the original file will be deleted and replaced with the copy you are making.

10.2 Joining files together

You can use the `cat` command to join two or more files together into a new file. `cat` lists all the files given as arguments to the standard output. This output, which is normally sent to the screen, can be redirected to a file using the `>` operator:

```
bash$ cat file1 file2 file3 > newfile
```

The first three files are the files to be copied, and `newfile` is the name of the file in which the copy is to be placed. You will now have the original files and the new combined file. For example, to merge files `fred`, `peter` and `paul` into one file `classmates`, type:

```
bash$ cat fred peter paul > classmates
```

Note: if `classmates` already exists, this will overwrite it.

10.3 Renaming a file

To rename a file, use the command `mv` (short for 'move'):

```
bash$ mv originalname newname
```

Beware! If a file called ‘newname’ already exists, the `mv` command will overwrite it without warning, losing its entire contents.

As with `cp`, if the second argument is the name of a directory, the file will be moved into that directory. See “Grouping your files” in Chapter 11 for information on directories.
`mv` can be made to check if the proposed new filename is already used, by giving it the `-i` option. If there is a file with this name, `mv` will then prompt you with the name and a question mark. Typing anything beginning with 'y' (for “yes”) allows the move to take place; typing anything else will abort it.

For example, supposing you have two files, `myfile` and `newfile`, and you want to change the name of `myfile` to `newfile`:

```bash
bash$ ls *file
myfile newfile
bash$ mv -i myfile newfile
remove newfile? y
bash$ ls *file
newfile
bash$
```

### 10.4 Deleting a file

To delete a file use the command `rm` (for ‘remove’):

```bash
bash$ rm filename
```

*Warning!* you cannot reverse the effect of this command: once a file is destroyed it cannot be retrieved unless it has been backed up by the System Administrator.

`rm` does not delete directories. See “Deleting directories” in section 11.8 for more information about that.

### 10.5 Using wild card symbols

If you need to carry out the same command on several files, for example to delete a set of files with similar names, you can use the wild card facility. This is exactly as described for the `ls` command in section 7.1. For example, to destroy all files starting “Mac” (remember upper and lower case are significant), type:

```bash
bash$ rm Mac*
```

This is obviously a risky procedure: you may inadvertently delete more files than you intended. However, `rm`, like `mv`, can be made to ask whether you really want to delete each file matching a pattern. To do this use the `-i` option:

```bash
bash$ ls Mac*
MacJohnson  MacJohnston
bash$ rm -i Mac*
rm: remove MacJohnson? y
rm: remove MacJohnston? n
bash$ ls Mac*
MacJohnston
```

*Note:* Do not put a space between `Mac` and `*`: if you do, UNIX will delete two sets of files, one matching the pattern `Mac` (a single file), and any matching `*`, namely everything! (apart from hidden files)
11 Grouping your files

There are many different types of file on UNIX, for example, ‘ordinary’, ‘special’ and ‘directory’. A directory file holds information about other files, which may be of any type. On UNIX you are always working within a particular directory.

Every directory has two important properties:

Firstly, UNIX uses the directory to locate all the files in it, so this is a useful way to group related files. Just as there is no confusion between Mr J.Bloggs of “Arvena, Milton”, and Mr J.Bloggs of “Arvena, Overshot”, so there is no confusion between files with the same name but in different directories. You can create directories called ‘firstyear’ and ‘secondyear’ and put files with the same names in each.

The second property is that directories can contain other directories, which in turn can contain other directories and so on. UNIX uses this to organise its file system.

11.1 The UNIX file system

Directories containing other directories create a branching structure like an inverted tree, known as a ‘hierarchical file structure’. This analogy carries over into the naming of the topmost directory, which is called the root directory, and identified as / on all UNIX systems. Different UNIX systems call their lower directories by different names – as in this simplified diagram of the file structure on holyrood:

```
/  bin usr home tmp
    |     |     |
    local adnam eve
    |     |     |
    bin share file1 file2 firstyear secondyear
    |     |     |    |
    lectures tutorials oats barley
```

11.2 Path names

If you need to refer to a file in a directory other than the one you are in, you do so by describing its position in the file hierarchy and giving its name. This is called the file’s full path name or absolute path name, and is a list of the directories down the path from the root to the directory within which the file exists, followed by the name of the file itself. A forward slash / is used both to indicate the root directory of the file system and to separate the names of directories in the path name; the context always makes it clear which meaning is intended.
For example, the full path name of the file barley in the previous illustration is:

```
/home/eve/firstyear/lectures/barley
```

Any path name which starts with a / is a full path name, beginning at the root of the file system. Any other path name is a relative path name, and begins from the directory you are in. Thus the full path name as in the above example always refers to a unique file, regardless of which directory you are in, whereas a relative pathname might identify different files, depending on the starting point.

There are two symbols you can use in path names to refer to certain directories:

- . a single dot means your current working directory
- .. two dots means the next directory up the path towards the root. This is called the parent directory.

### 11.3 Important directories

**Root directory**

The topmost directory on the whole computer is called simply /, or the root directory.

**Home or login directory**

When you log on you are placed in what is known as your home directory, which is contained in the hierarchy of files and directories which make up the UNIX system.

The shells used by most Edinburgh users (except the Bourne shell) have a special character ~ (‘tilde’) which is used with a login name to expand to the pathname of the user’s home directory. For example the home directory of user eve might have a full pathname of /home/eve, but most of the time you can refer to this as ~eve.

Note: It is in fact better to use the shorthand method for a home directory (~eve rather than /home/eve) because the UNIX system administrators may move home directories around the file system – so changing the full pathname.

**Current working directory**

Once you have logged on you can move from your home directory into another directory, which becomes your current working directory. For example, user eve can move away from /home/eve and roam around the UNIX file system wherever the access permissions of the hierarchy allow.

You can find out the full path name of your current working directory by using the `pwd` (‘print working directory’) command:

```
bash$ pwd
/home/eve
```

### 11.4 Creating new directories

Directories are created with the `mkdir` (‘make directory’) command:

```
bash$ mkdir name
```
You can check that the new directory has been created by using the `ls` command.

Remember that this new directory is located in your current working directory to take its place in the overall hierarchy.

### 11.5 Moving to another directory

To move to another directory you use the command `cd` ('change directory'):

```
bash$ cd pathname
```

where `pathname` specifies the directory into which you want to move. For example, user `eve` can move from her home directory to `firstYear` (see previous illustration) by typing:

```
bash$ cd firstyear
```

From here there are two ways to move into `Secondyear` – either by moving back up a level then down another branch:

```
bash$ cd ../secondyear
```

or by using the full path name:

```
bash$ cd /home/eve/secondyear
```

You can move ‘up’ two levels by typing:

```
bash$ cd ../..
```

You can get directly to your home directory simply by typing:

```
bash$ cd
```

Whatever directory you are in, the `ls` command shows only the files and directories which are held in that directory.

### 11.6 Putting files into a directory

There are three ways of putting files into a directory: creation, movement, and duplication.

When you create a file it will go into your current working directory, unless you explicitly specify another destination by a path name. Thus from her home directory (shown in the previous illustration), `eve` could create the file `oats` using the MicroEMACS editor with these commands:

```
bash$ ue firstyear/lectures/oats
```

```
To move a file from one directory to another you can use the command `mv`. For example, if user `eve` were in directory `firstyear`, the file `oats` could be moved from directory `lectures` into directory `tutorials` using the command:

```
bash$ mv lectures/oats tutorials
```

Remembering that to UNIX a directory is no more than one type of file, you can move a directory into another directory by using `mv`. So the command

```
bash$ mv lectures tutorials
```

would result in the file structure shown here:

```
11.7 Listing directory contents
```

The `ls -l` command (described in Chapter 7) is used to list the contents of a directory: the first character on each line of the resulting list shows whether a particular entry in the list is a directory or a file. For example, in the file system illustrated above, if user `eve` were in her directory `tutorials` she would get the following:

```
bash$ ls -l
drwx------ 1 eve socsci 512 Aug 24 16:12 lectures
-rw------- 1 eve socsci 234 Aug 24 16:12 oats
```

This shows two files in the current directory, a directory called `lectures` and an ordinary file called `oats`
11.8 Deleting directories

There are two commands to delete a directory: `rmdir`, and `rm -r`.

The command `rmdir` ('remove directory') is the inverse of `mkdir`. It requires the directory to be empty, that is to have no files in it. This can be achieved with:

```
bash$ rm directory/*
bash$ rmdir directory
```

For example, `lectures` contains one file `barley`:

```
bash$ rm lectures
rm: lectures directory (can’t do that! it’s a directory)
bash$ rmdir lectures
rmdir: lectures not empty (and there are files in it)
bash$ ls lectures
barley (a file called barley)
bash$ rm lectures/* (remove the file)
bash$ rmdir lectures (...now it’s OK to remove the directory)
bash$
```

The second command to use is `rm` (the ordinary ‘remove’) with the `-r` option. This can be somewhat drastic in that it deletes every file in the named directory before deleting the directory itself.

For example, starting with `lectures`, and `Barley` again:

```
bash$ rmdir lectures
rmdir: lectures not empty (there are files in it)
bash$ ls lectures
barley (a file called barley)
bash$ rm lectures/* (everything deleted)
bash$
```

In both the above examples, `rm` can be used with the `-i` (‘interactive’) option to check whether you really want to delete each file before you actually do it.

Thus re-running the last example:

```
bash$ rm -ir lectures
rm: remove lectures/barley? y
rm: remove lectures? y
bash$
```
12 Exchanging files with another user

Users will frequently want to exchange files with each other.

It is not possible to give someone a file, or take, or copy a file from them, without their cooperation. This section therefore describes the actions that both sender and receiver have to take.

12.1 File security

Every file on your UNIX account can be protected from or made accessible to other users. This is done by changing the access permissions or access modes of the file.

Amongst the information given by the `ls` command with its `-l` option is the permission information for your files:

```
bash$ ls -l file1
-rwxr-x--x 1 adam biosup 34  march 28 15:12 file1
```

The block of ten characters at the beginning of the line contains the access permissions.

The first character is used to denote whether the file is an ordinary file (-) or a directory (d). The other nine characters are split into three sets of three characters. The sets refer to the owner of the file, the group to which the owner belongs, and all other users, respectively.

The three main symbols used are:

- `r` read (look at the file)
- `w` write (change the file)
- `x` execute (run that file as a program)

**Example**

Consider the previous example again:

```
bash$ ls -l file1
-rwxr-x--x 1 adam biosup 34  march 28 15:12 file1
```

Here the first character, -, shows that this is an ordinary file. The next three characters, rwx, show that the owner of the file – adam – has permission to read, write and execute the file. The next three, r-x, show that colleagues in the owner’s group – biosup – may read (copy) and execute the file, but not write to it. The final set, --x, shows that everyone may execute the file but neither write to it nor copy it.

When you create a file or directory on UNIX, the permissions are set to a default; for instance for a file on the central EUCS UNIX services these are:

```
-rw------- 1 adam biosup 34  march 28 15:12 file1
```

That is, the owner of the file has read and write permission, the group and others have no permissions at all. If you are compiling a program, the permission on the executable image file will automatically be set to `x` (execute) for the owner:

```
-rwx------ 1 adam biosup 4567  march 28 15:12 a.out
```
For a directory which you create on the central UNIX services, the default permissions are:

```
drwx------ 2 adam biosup 24 march 28 16:34 newdir
```

whilst those for your home directory on the central UNIX services are:

```
drwx--x--x 2 adam biosup 1024 march 28 16:34 adam
```

The x (execute or search) permission has a special significance for directories (which are not programs and cannot be executed, of course): execute permission allows access to any permitted files within a directory. Thus to give another user access to a file they must have appropriate access to the file itself and at least execute permission for the directory containing the file. The significance of the various permissions on files and directories is given in the following table:

<table>
<thead>
<tr>
<th>Action</th>
<th>File</th>
<th>Directory</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>r</td>
<td>w</td>
</tr>
<tr>
<td></td>
<td>x</td>
<td>r</td>
</tr>
<tr>
<td>See what files are in a directory</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Create file in directory</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Delete a file</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Read a file</td>
<td>•</td>
<td>-</td>
</tr>
<tr>
<td>Write to a file</td>
<td>-</td>
<td>•</td>
</tr>
<tr>
<td>Execute a file</td>
<td>-</td>
<td>•</td>
</tr>
<tr>
<td>Execute a shell script file</td>
<td>•</td>
<td>-</td>
</tr>
</tbody>
</table>

- permission required  
- permission not relevant

Note: if you grant write access to a directory, you allow anyone to rename or delete existing files and create new ones in that directory, regardless of individual file permissions.

**Groups**

On UNIX, every user is a member of one or more groups. The default group usually comprises members of a department or workgroup. You can find out which groups you belong to by using the command groups. For example:

```
bash$ groups
biotech support
```

This user is a member of two groups, ‘biotech’ and ‘support’.
Files are also accessible by a particular group as well as being owned by a user. Group permissions refer to the users who are members of the group which can access the file. You can find out which group can access the file by using the option `-g` with the command `ls`. For example, for your own files:

```
bash$ ls -g
-rw-rw-r-- 1 biotech 918 Aug 28 16:34 anemone
-rw-r----- 1 biotech 441 Aug 81 16:34 blanda
```

This shows that the files `anemone` and `blanda` are accessible to the group 'biotech'. The user has the usual `rw` permissions to both. The file `blanda` has `r` permission for group 'biotech', so any user in group 'biotech' can read it. The file `anemone` has `rw` permission for users in group 'biotech', and has `r` permission for the whole world.

### 12.2 Changing permissions: the chmod command

The `chmod` ('change mode') command is used to change the modes (access permissions) associated with a file. The change is specified in the form:

```
bash$ chmod modes file
```

A mode is a three-item sequence which represents `who`, `operation` and `permission`.

`who` is one of:

- `u` user — the owner of the file
- `g` group — the group to which the owner belongs
- `o` others — everyone else
- `a` all — user, group and others

`operation` is one of:

- `+` add the specified permission
- `-` subtract the specified permission
- `=` assign the specified permission, ignoring whatever may have been set before.

`permission` is specified by one or more of:

- `r` read
- `w` write
- `x` execute
In the following examples, the `chmod` command is given together with the old and new modes for the files concerned.

**Examples**

Given a file `file1` with the default access modes (read and write for the user), add read mode for the group:

```
bash$ chmod g+r file1
```

<table>
<thead>
<tr>
<th></th>
<th>user</th>
<th>group</th>
<th>others</th>
</tr>
</thead>
<tbody>
<tr>
<td>Before</td>
<td>rw-</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>After</td>
<td>rw-</td>
<td>r--</td>
<td>---</td>
</tr>
</tbody>
</table>

Remove read and write access for everyone else from file `file2` (showing that you can use multiple permissions):

```
bash$ chmod o-rw file2
```

<table>
<thead>
<tr>
<th></th>
<th>user</th>
<th>group</th>
<th>others</th>
</tr>
</thead>
<tbody>
<tr>
<td>Before</td>
<td>rwx</td>
<td>rw-</td>
<td>rw-</td>
</tr>
<tr>
<td>After</td>
<td>rwx</td>
<td>rw-</td>
<td>---</td>
</tr>
</tbody>
</table>

Give everyone (owner, group and others) read, write and execute permissions on file `file3`:

```
bash$ chmod a=rwx file3
```

<table>
<thead>
<tr>
<th></th>
<th>user</th>
<th>group</th>
<th>others</th>
</tr>
</thead>
<tbody>
<tr>
<td>Before</td>
<td>irrelevant</td>
<td></td>
<td></td>
</tr>
<tr>
<td>After</td>
<td>rwx</td>
<td>rwx</td>
<td>rwx</td>
</tr>
</tbody>
</table>

**12.3 Setting modes numerically**

There is a shorthand way of setting permissions by using “octal numbers”. ‘Read permission’ is given the value 4, ‘write permission’ the value 2 and ‘execute permission’ the value 1. These are added together for any one user category, so:

```
1 = execute only
2 = write only
3 = write and execute
4 = read only
5 = read and execute
6 = read and write
7 = read and write and execute
```
The mode for a file can therefore be expressed as three digits, the leftmost being the composite permission for the owner of the file, the middle for the group and the rightmost for everyone else. Below are some examples of changing access permissions using this method:

<table>
<thead>
<tr>
<th>Command</th>
<th>Mode</th>
<th>Owner</th>
<th>Group</th>
<th>Others</th>
</tr>
</thead>
<tbody>
<tr>
<td>bash$ chmod 640 file1</td>
<td>rwx</td>
<td>r-x</td>
<td>r--</td>
<td></td>
</tr>
<tr>
<td>bash$ chmod 754 file1</td>
<td>rwx</td>
<td>r-x</td>
<td>r--</td>
<td></td>
</tr>
<tr>
<td>bash$ chmod 664 file1</td>
<td>rwx</td>
<td>r-x</td>
<td>r--</td>
<td></td>
</tr>
</tbody>
</table>

### 12.4 Accessing files belonging to another user

If you want to use someone else's files, or want to give them access to your own files, there are three steps to go through:

**Where is the file?**

First of all you must obtain the exact details of where the file is on their account. Is it in the home directory or in another directory? You must know the pathname of the file (see Chapter 11 for details about directory structures and pathnames).

The shells used by most Edinburgh users (except the Bourne shell, 'sh'), use the special character ~ to expand a login name to the pathname of the corresponding home directory. Thus you can usually refer to the home directory of user eve as ~eve rather than /home/eve.

**Have the relevant access permissions been given?**

The owner of the file must set the access permissions to allow you to access the file to the level you have agreed on – for example, read only.

Also, all directories in the path containing the file must have the execute (search) permission set to the appropriate level (that is, for the group or for all users).

**How to do it**

For example if user eve has given you permission to copy a file called file1 in her home directory, this is what you would type:

```
bash$ cp ~eve/file1 newfile
```

If the file had been in a lower directory, say called fred, then you would use the pathname as if you were working from the other user's home directory:

```
bash$ cp ~eve/fred/file1 newfile
```
13 Customising your Environment

Your account on UNIX has a number of properties known as *environment variables*. These contain your logon name, specify the printer you normally use, the default search path and in many other ways tailor your UNIX account the way you want it.

To show the values of these variables, use the `printenv` command:

```
holyrood$ printenv
LOGNAME=helper
VISUAL=ue
MAIL=/home/helper/.Mail
PAGER=less
DELIVERY=JCMB Job Reception.
FOLDERS=/home/helper/Mail
TERM=xterm
HOSTTYPE=sparc
PATH=/home/helper/bin:/usr/bin/X11:/usr/local/bin:/usr/ccs/bin:/opt/SUNWspro/bin:/opt/epc/bin:/usr/bin:/usr/local/epc/bin:/usr/local/SUNWspro/bin:/usr/local/java/bin:/usr/bin:
PRINTER=lp
HOME=/home/adam
NNTPSERVER=newsread.ed.ac.uk
SHLVL=1
EDITOR=ue
_=/usr/ucb/printenv
holyrood$
```

You will see from this example that environment variables have names like `TERM`, `PRINTER`, etc.

To alter a variable, type:

```
bash$ PRINTER=ps15
```

This value will be used by your current shell. However, you need to use the `export` command to tell all other programs (editors, Mail programs etc.) that you have altered the value of this variable.

```
bash$ export PRINTER
```
The following examples show how to alter a couple of the more common environment variables:

**TERM**, your default terminal:

```
bash$ TERM=xterm
bash$ export TERM
bash$ printenv TERM
xterm
```

**DELIVERY**, where you'd like your output to be delivered:

```
bash$ DELIVERY="Appleton Tower"
bash$ export DELIVERY
bash$ printenv DELIVERY
Appleton Tower
```

Note that the command `printenv` on its own will list out the values of all your Environment Variables.

*Note: The settings you make only apply for the duration of the session: when you next log on, the old values will be back. Altering these variables permanently involves editing special ‘profile’ files in your account, called `.bashrc` and `.bash_profile`, and precise details of how to do this are not given in this guide.*

Another useful environment variable is **PATH** This tells the computer where to look to find the programs which need to be activated by the commands you give it. You need to modify **PATH** if you want to use new programs, such as an applications package. **PATH** simply gives the full path names of all the directories which are to be searched when the Shell is looking for the commands you enter. You do need to know the full path name of any directory you'd like to add. You can find the current setting with the command:

```
bash$ printenv PATH
```

The new directory is added by editing the setting of **PATH** in your `.bashrc`, `.bash_profile` or `.login` file. Before doing this you need to know more than is covered in this guide, so should check the other UNIX publications available from EUCS Sales Points.

One final command which you might find useful is **alias**, which can be used to change command names or to provide shorthand names for commands and options. For example the command:

```
bash$ alias ls='ls -F "$@"'
```

creates your own version of the `ls` command, and when you subsequently type `ls` it will be interpreted as `ls -F`. The `$@` means that any arguments which are included with the command are passed from your own version of the command to the one you are aliasing. Again this only applies until you log off, unless you insert the alias command(s) in your `.bashrc` or `.bash_profile` file.

*Note that there should be no spaces between your new command, the = and the alias.*

The command **alias** by itself prints a list of your current aliases.

For more details on customising your environment see *Shells and Shell Scripts – a Guide to Tailoring your UNIX Environment.*
14 MicroEMACS: a text editor

14.1 Introduction to MicroEMACS

MicroEMACS – usually written as ‘ue’ – is a text editor. A text editor is a very basic word processing system, which allows you to create and alter files which may contain text or numerical data. The more advanced facilities of a word processor – such as multiple fonts, emboldening characters and underlining words – are not available, but you can carry out a few basic kinds of text formatting.

MicroEMACS is very easy to get started with, yet contains sophisticated features to handle complex editing needs.

14.2 Starting the ue Editor

To start MicroEMACS from the shell prompt, use the command:

```
ue filename
```

where `filename` is the name of a file.

If the file already exists then `ue` will display its text ready for you to make some changes. If the file does not exist then `ue` will start with a blank screen ready for you to type.

Note that when you get close to the right margin, `ue` will automatically move or “wrap” your text to the next line. You only press the [Return] key when you want to start a new paragraph.

The [Delete] key removes characters: if you make a mistake, simply press the [Delete] key until the mistake is removed, then type in the correct version.

The arrow keys on your keyboard will generally allow you to move the cursor around in the file.

You could use `ue` just by typing and using [Delete], but knowing a few more commands makes text editing a lot easier.

Any ordinary text that you type is inserted directly into the file, so special characters are used to distinguish commands to the editor – with the [Control] and [Escape] keys. For example, the command to move the cursor forward by one character is `CTRL-F`, meaning “hold down the [Control] key, type F, then release the [Control] key”. In contrast, the command to move the cursor forward by one whole word is `[Escape]F`, meaning “press and release the [Escape] key, then press and release F”.

Note: holding down the [Escape] key in the same way as you hold down the [Control] or [Shift] keys is the most common initial mistake using `ue`. The usual result is some beeps and perhaps some funny characters on the screen that you’ll have to delete.
14.3 Moving the cursor around the file

within a line...

Wherever you are in a line, the commands `CTRL-A` and `CTRL-E` take the cursor to the start and end of the line, respectively.

You can use the left and right arrow keys to move the cursor backwards or forwards by one character; the commands `CTRL-B` and `CTRL-F` have the same effect. To move the cursor backwards or forwards by one whole word at a time, use `[Escape]B` and `[Escape]F`, respectively. These commands enable you to position the cursor to make specific changes within a line of text.

between lines...

The up and down arrow keys move the cursor up and down the lines in your text. Alternatively, you can use the commands `CTRL-P` to go to the previous line and `CTRL-N` to go to the next one.

around a file...

Files usually consist of more than one screenfull of text. You can move up by a whole screen with command `CTRL-Z`, and down a whole screen with `CTRL-V`.

You can jump to the very end of your file with the command `[Escape]>` (right angle bracket), and to the very beginning with `[Escape]<`.

In MicroEMACS the `[Escape]` key is known as the “Meta” key, and given the computer jargon written form ‘M’-’. So you will sometimes see the end-of-file command written as `M->` (this means press and release the `[Escape]` key then press and release the > key), and the start-of-file command as `M<-`.

14.4 Deleting characters in a line

To delete the character immediately to the left of the cursor, press the [Delete] key.

You can delete the character under the cursor (rather than to its left) by using command `CTRL-D`. This will drag the line text following the cursor back to fill the space. You can thus repeat the `CTRL-D` command several times to delete a word or phrase.

To delete the previous word or part of a word, use `[Escape][Delete]`. To delete the next word or part of word, use `[Escape]D`.

You can delete all the characters between the cursor and the end of a line with the command `CTRL-K`. To delete a whole line, go to the start of the line with `CTRL-A`, then use `CTRL-K`. Note in this case that `CTRL-K` deletes all the visible characters in the line, but not the “newline” character at the end, so you are left with a blank line: if you want to remove that too, just use the `CTRL-K` command again.
14.5 Quitting ue

The normal way to quit ue is to exit and save any changes you’ve made. Use the command M-Z to do this (that is, press the [Escape] key, then Z). If you’re planning a long complicated edit, it’s a good idea to save the state of the file every now and then, by using the command CTRL-XS (that is, hold down the [Control] key, type X then S, then release the [Control] key). If you’ve made a complete mess of the editing and you want to return to the way the file was before you started this edit session, then abandon it by using the command CTRL-XC (that is, hold down the [Control] key, type X then C, then release the [Control] key). When you do this, you’ll be asked if you really want to quit, with the message at the bottom of the screen:

Modified buffers exist. Leave anyway [y/n]

Press Y if you do want to exit and return to the previous state of the file, otherwise N. Note that all changes you made since you started the edit will be lost when you exit by this method.

14.6 Inserting another file

You can insert a previously prepared file at the cursor’s position by using the command CTRL-XI (that is, hold down the [Control] key, type X then I – for ‘Insert’ – then release the [Control] key). You will then be prompted for a filename. Type the name of the file you want to insert and then press the [Return] key.

14.7 Deleting chunks of text

You can delete a whole chunk of text quite easily. To do this you set a marker in the file, then move the cursor, and use a command which deletes all the text between the mark and the cursor. So the method is to place the mark at one end of the text to be deleted, then move the cursor to the other end. The order of them doesn’t matter.

Place a mark by using the command M-[Space] M-[Space] M-[Space] M-[Space] M-[Space] (that is, press the [Escape] key then the Spacebar). You won’t see a mark being made on the screen, but there will be a “mark set” message at the bottom of the ue window.

Note that whenever you place a mark, it replaces any previous mark you’ve set.

Once you have placed a mark and positioned the cursor, the command which actually deletes the text is CTRL-W.

A very useful feature of MicroEMACS is that it stores the last chunk of deleted text in a part of its internal memory called the ‘copy buffer’. You can “yank” whatever is in the copy buffer back into your text at the cursor point by using the command CTRL-Y. This is useful if you want to move a piece of text around in your file. Just delete it, move the cursor, and yank it back in. You can also repeat the same text by yanking it in more than once.

If you want to copy the original piece of text into the copy buffer without deleting it, use M-W instead of CTRL-W the copy is then available to be yanked in.
14.8 Searching

You will often want to search through your text for a particular word or phrase. To do this, use the command \texttt{CTRL-	extbackslash}. The following prompt will then appear at the foot of the screen:

\texttt{Search [\texttt{META}] :}

The square brackets are empty when you first search, but will subsequently contain the text of the previous search to make it easy to repeat a search. \texttt{\texttt{META}} reminds you that you initiate the search by pressing the [Escape] key – the ‘meta key’.

Type the word or phrase for which you are searching and press [Escape]. A common mistake here is to press [Return] instead of [Escape], but this simply includes the ‘newline’ character in the text being searched for (you can use this feature to find a word only at the start or end of a line, for example, or to find blank lines by searching for two newline characters in a row).

The cursor will move to the first occurrence of the text you specified. Note that some words can appear as a part of others. For example a search for “the” might find the word “then”, or “another”.

The search always commences from the current position of the cursor: if you want to search from the start of the file you should move the cursor there first by using the \texttt{M-<} command.

Sometimes you will find the wrong occurrence of the text you are looking for, and will want to continue the search with the same target. Simply repeat the \texttt{CTRL-	extbackslash} command, but instead of typing the text again, just press [Escape]: \texttt{ue} then uses the previous search target (shown in the square brackets in its prompt).

To search \textit{backwards} from the cursor, use \texttt{CTRL-R} instead of \texttt{CTRL-\textbackslash}.

14.9 Search and Replace

You can replace all occurrences of one word (or phrase) with another by using the command \texttt{M-R}. You are prompted for the target (exactly as when doing a plain search, described in the previous section); type this and press [Escape]. You are then prompted for the replacement text. Again, type this and press [Escape]. All occurrences of the target text will be altered to the replacement text.

Note the comment in the previous section, about some words being contained in others: they too will be changed by this method. If you think this might cause a problem, or if you don’t want to change every occurrence, there’s a useful variation of this command which asks you to confirm each replacement. To do this, use the command \texttt{M-CTRL-R} (that is, press and release [Escape], then use \texttt{CTRL-R}). For each occurrence of the target string found, you’ll be asked whether you want it replaced or not. Just press “y” (to replace it) or “n” (to leave it), then \texttt{ue} moves the cursor on to the next occurrence of the target.
14.10 Repeating a ue command many times

There is a handy short cut which allows you to repeat a ue command any number of times. For example, if you want to move the cursor forwards by 40 characters, you can press and release the [Escape] key, type 40 followed by the [Return] key, then press the right-arrow key (or \texttt{CTRL-F}).

The "[Escape]number" combination is a way to instruct ue to do the next command \texttt{number} times. This repeat-command instruction works with most ue commands.

14.11 Redrawing the Screen

Sometimes the display of text on your screen is messed up, for example by messages announcing an alert or the arrival of email. When this happens, simply use \texttt{CTRL-L} to redraw the screen.

14.12 Reference Card

All the commands described in this chapter are listed in the MicroEMACS Reference Card which is available in the Open Access Computing Labs and from EUCS Sales Points. For further information on the more advanced features of MicroEMACS, you can get the Introduction to MicroEMACS manual from the Sales Points.

14.13 Text processing: TeX and LaTeX

MicroEMACS is just an editor. It can be used to create and rearrange text, but it cannot produce formatted documents with, for example, bold headings, text in different fonts, diagrams etc. Its job is purely and simply to manipulate plain text.

The TeX system is available on central UNIX services for those who need text-processing; it is particularly good at handling mathematical equations and symbols. TeX is most commonly used through an interface called LaTeX, and its output is usually produced on laser printers for the best quality. Documents to help you use LaTeX are available from the EUCS Sales Points.
15 Communicating by Mail and News

15.1 Electronic Mail

Electronic mail – email – is a system with which people can send each other messages via computer networks. It is straightforward to send messages to people all over the world.

The email program supported on central EUCS Unix services is called pine. It is a screen-based full-feature email programs which allow users to send and receive messages, reply to and forward them, and keep them in folders.

Electronic Mail in general, and pine in particular, is described in various documents which are available from the EUCS Sales Points (see Appendix A).

15.2 Network News

The world-wide network news system called UseNet is available on the central UNIX services. The system is like a collection of electronic noticeboards, each devoted to a specific topic, and enables people all over the world to communicate with each other on a variety of subjects, only a minority of which are related to UNIX or computing.

The network news readers which are supported on the central UNIX services are called tin and nn.

Network news in general, and the tin and nn programs in particular, are described in various documents which are available from the EUCS Sales Points (see Appendix A).
Appendix A: Further Reading

The following documents are some of those available at the EUCS Reception points for EUCS documentation:

**UNIX operating system**

- *UNIX on the EUCS UNIX Services* (Task Note)
- *Exploiting UNIX (some bash facilities and utilities)* (Task Note)
- *UNIX 1 - Introduction to UNIX* (Workbook)
- *UNIX 2 - Enhancing your UNIX skills* (Workbook)
- *UNIX 3 - UNIX shell programming* (Workbook)

*Shells and Shell scripts – a Guide to Tailoring your UNIX Environment.*

**Editors**

- *MicroEMACS* (Reference Card)
- *MicroEMACS* (Introductory Guide)
- *MicroEMACS* (Reference Manual)

**Electronic Mail**

- *Pine for UNIX* (Workbook)

*pine* on the central UNIX servers (Reference Card)

**Network News**

- *Usenet overview*
- *Reading Usenet with tin*
- *Reading Usenet with nn*
Appendix B: UNIX commands described in this Guide

- **alias**: change command names or options
- **cat**: list or concatenate files
- **cd**: change working directory
- **chmod**: change modes (permissions) of a file
- **cp**: copy files
- **date**: print date and time
- **export**: put the value of a variable into an environment
- **file**: attempt to interpret contents of a file
- **groups**: list groups user belongs to
- **logout**: leave UNIX
- **lpq**: show which files are queued for printer
- **lpr**: print a file
- **lprm**: remove file from printer queue
- **ls**: list filenames
- **man**: display manual pages
- **mkdir**: make a new directory
- **more**: page through contents of file
- **mv**: move contents elsewhere (change name of file)
- **passwd**: change your login password
- **printenv**: print environment variables
- **pwd**: print working directory
- **rm**: remove (destroy) a file
- **rmdir**: remove (destroy) a directory
- **wc**: count words, lines and characters in a file
- **who**: give information about current logged on users
Appendix C: Logging on to a UNIX System from a Mac or PC

This Appendix describes how to log on to a UNIX system from a Mac or PC in one of the Open Access Computing Labs operated by EUCS. If you want to set up your own computer to do this you need to ask the EUCS Sales Points or your User Support Team for information about telnet (Mac) or Exceed(PC).

C.1 From a Mac

➲ If the Hard Disk window (Macintosh HD) is not already open, double-click on the Macintosh HD icon.

➲ Double-click on the Communications folder icon:
   The contents of the Communications folder will be displayed in a new window:

➲ Double-click on the Telnet folder:

➲ For a holyrood account, double click on the Holyrood icon

➲ For other systems:
   Double-click on the “Telnet” icon:

➲ Choose Open Connection from the File menu

➲ Type the system name in the “host/session name” box

➲ Click connect. A window will open connected to that system. You will be prompted for your “login” or userid on the selected host, and then for your password”. Passwords are not echoed on the screen for security reasons.

C.2 From a PC

➲ Log in to the computer

➲ For CompSci, ChemEng, ElecEng and Holyrood:
   Go to Start → Programs → comms → Exceed X server → choose option

➲ Otherwise, telnet to host:
   Go to Start → Programs → comms → Exceed → Host → Access → Telnet

➲ Type the system name in the “host/session name” box

➲ Click ok. A window will open connected to that system. You will be prompted for your “login” or userid on the selected host, and then for your “password”. Passwords are not echoed on the screen for security reasons.
Appendix D: Glossary of terms

**Account**  The area on the UNIX file system that is allocated to you.

**Argument**  Most commands require arguments which are the names of files or other information to which the command applies.

**Command**  What you type to make the UNIX system do something for you.

**Current working directory**  Your present location in the UNIX file system hierarchy.

**Directory**  All the files on a UNIX system are grouped in directories. For example, all the files in your account are held in a collection of directories.

**Editor**  An editor is a program with which you can alter the contents of a text file.

**File**  A file is the main unit of storage on UNIX. It can contain data, an executable program, text etc.

**Flag**  See option

**Full path name**  A list of directories and subdirectories which precisely define the location of a file in relation to the root directory. Any file’s full path name is the same regardless of where you are in the UNIX file system.

**Group**  On UNIX every user is a member of one or more groups. The default group usually comprises members of a department or workgroup. You can find out which groups you belong to by using the command `groups`.

**Home Directory**  The part of the file system that you enter automatically when you log in.

**Logging In**  See logging on.

**Logging off**  Leaving your UNIX account.

**Logging on**  Getting on to your UNIX account.

**Logging out**  See Logging off.

**Option**  Options (also known as flags) modify the way commands work. They usually consist of a single letter preceded by a hyphen.

**Password**  A secret code that you type in after your username.

**Pager**  A program with which you can control how a file is displayed on your screen.

**Path name**  Every file or directory on a UNIX system has an address known as a path name which defines its location in the UNIX file system hierarchy. This path name consists of the names of the directories you would need to go through to get to the file or directory you want, starting from the root directory. The slash (/) character is used to separate directories in a Path name. For example:

```
/usr/local/lib/jimmy/text
```

**Pipe**  Connects the output from one command into the input of another, whose output can in turn be piped to a further command, etc.
**PostScript**  This page description language has achieved almost universal use in word-processing and desktop publishing programs and in laser printers. It allows the text and graphics which are to be shown on a page to be described in very precise detail, enabling a computer to achieve the best possible results from any particular printer.

**Relative path name** The path name of a file or directory relative to the directory you are in.

**Root**  The UNIX file system is a hierarchy of directories. The top directory which contains all the others is known as the root, and is referred to as `/`.

**Script** A file containing UNIX shell commands. The file can be executed like a program.

**Shell**  The shell reads what you type and decides which programs to use. It is like a buffer between you and the UNIX system. There are a number of shells available. At Edinburgh University the Bash shell is recommended.

**Username**  A unique name used to identify you when logging in.

**Wild card**  A symbol which can represent any character in a filename - rather like a joker in a pack of cards. For example, the command `ls *ps` will list all files whose name ends in 'ps', `*` being the wild card.
Symbols
S (in system prompt) 2
* (wild card character) 15
-i (interactive option) 24, 29
. (current working directory) 26
.. (parent directory) 26
.bash_profile (profile file) 36
.bashrc (profile file) 36
.filenames 17
/ (pathname separator) 26
/ (root directory) 25
< (redirects input) 11
> (redirects output) 11, 14
>> (redirects output) 11
? (wild card character) 15
[Return] key 1
| (pipe operator) 11
~ (home directory locator) 26, 34

A
absolute path name 26
access permissions 30
account 3, 49
alias (UNIX command) 36
appending output to a file 11
Apple Macintosh 3, 45
arguments 9, 49

B
bash (Bourne again shell) 12
bash$ 2
Bourne shell 12

C
C shell 12
cancelling a print request 22
cat (UNIX command) 11, 13, 18, 23
to display file on screen 18
to inspect file contents 18
to join files together 23
with redirected input/output 11
cd (UNIX command) 27
change directory (cd) 27
changing permissions 32
changing shell (chsh) 12
chmod (UNIX command) 32
chsh (UNIX command) 12
cmd 49
commands 9
and arguments
optional 9
interrupting 14
options 10
terminated by [Return] 1
concatenating files 23
content of a file 18
copy permission 30, 33
copying files 23, 34
counting lines, words etc 10
cp (UNIX command) 23
to copy a file 23
to copy another user's file 34
creating a new directory 27
creating files 13
current working directory 26, 49
customising your Environment 35

D
date (UNIX command) 9
date and time 9, 16
deleting a directory 29
deleting a file 24
DELIVERY (environment variable) 36
departmental colleagues 31
destroying a directory 29
destroying a file 24
directories 15, 25
and access permissions 31
deleting 29
important 26
listing contents of 29
moving files into 28
switching between 27
directory 49
changing 27
current working 26
home 26
parent 26
root 26
document printing queues 22
documentation 50
documentation online 7

E
editor 49
editors 19
Electronic Mail 42, 50
Environment 35
Examples
alias 36
cat 13, 18, 23
cd 27
changing directory 27
chmod 33
<table>
<thead>
<tr>
<th>Command/Concept</th>
<th>Page(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>chsh</td>
<td>12</td>
</tr>
<tr>
<td>copying files</td>
<td>34</td>
</tr>
<tr>
<td>cp</td>
<td>23</td>
</tr>
<tr>
<td>date and time</td>
<td>9</td>
</tr>
<tr>
<td>deleting files</td>
<td>24</td>
</tr>
<tr>
<td>DELIVERY</td>
<td>36</td>
</tr>
<tr>
<td>file</td>
<td>18</td>
</tr>
<tr>
<td>groups</td>
<td>31</td>
</tr>
<tr>
<td>joining files with cat</td>
<td>23</td>
</tr>
<tr>
<td>lpq</td>
<td>22</td>
</tr>
<tr>
<td>lpr</td>
<td>10, 21</td>
</tr>
<tr>
<td>lprm</td>
<td>22</td>
</tr>
<tr>
<td>ls</td>
<td>9, 15</td>
</tr>
<tr>
<td>ls -a</td>
<td>17</td>
</tr>
<tr>
<td>ls -l</td>
<td>16, 28</td>
</tr>
<tr>
<td>man</td>
<td>7</td>
</tr>
<tr>
<td>moving files and directories</td>
<td>28</td>
</tr>
<tr>
<td>mv</td>
<td>28</td>
</tr>
<tr>
<td>pathname</td>
<td>26</td>
</tr>
<tr>
<td>piping</td>
<td>11</td>
</tr>
<tr>
<td>printenv</td>
<td>35</td>
</tr>
<tr>
<td>printing</td>
<td>20</td>
</tr>
<tr>
<td>pwd</td>
<td>27</td>
</tr>
<tr>
<td>redirection</td>
<td>11</td>
</tr>
<tr>
<td>rm</td>
<td>24</td>
</tr>
<tr>
<td>rm -i</td>
<td>29</td>
</tr>
<tr>
<td>rm -r</td>
<td>29</td>
</tr>
<tr>
<td>rmdir</td>
<td>29</td>
</tr>
<tr>
<td>setting access permissions</td>
<td>33</td>
</tr>
<tr>
<td>TERM</td>
<td>36</td>
</tr>
<tr>
<td>wc</td>
<td>10</td>
</tr>
<tr>
<td>who</td>
<td>9</td>
</tr>
<tr>
<td>wild cards</td>
<td>15</td>
</tr>
<tr>
<td>exchanging files</td>
<td>30</td>
</tr>
<tr>
<td>execute permission</td>
<td>30, 33</td>
</tr>
<tr>
<td>export (UNIX command)</td>
<td>35</td>
</tr>
<tr>
<td>F</td>
<td></td>
</tr>
<tr>
<td>file</td>
<td>49</td>
</tr>
<tr>
<td>file (UNIX command)</td>
<td>18</td>
</tr>
<tr>
<td>file security</td>
<td>30</td>
</tr>
<tr>
<td>file system</td>
<td>25</td>
</tr>
<tr>
<td>files</td>
<td>13</td>
</tr>
<tr>
<td>content</td>
<td>18</td>
</tr>
<tr>
<td>copying</td>
<td>23, 34</td>
</tr>
<tr>
<td>deleting</td>
<td>24</td>
</tr>
<tr>
<td>exchanging with other users</td>
<td>30</td>
</tr>
<tr>
<td>getting more information about</td>
<td>16</td>
</tr>
<tr>
<td>grouping together</td>
<td>25</td>
</tr>
<tr>
<td>hierarchical structure</td>
<td>25</td>
</tr>
<tr>
<td>joining</td>
<td>23</td>
</tr>
<tr>
<td>length</td>
<td>16</td>
</tr>
<tr>
<td>listing names</td>
<td>15</td>
</tr>
<tr>
<td>looking at</td>
<td>18</td>
</tr>
<tr>
<td>moving</td>
<td>28</td>
</tr>
<tr>
<td>names</td>
<td>13</td>
</tr>
<tr>
<td>permissions</td>
<td>30</td>
</tr>
<tr>
<td>printing</td>
<td>20</td>
</tr>
<tr>
<td>removing</td>
<td>24</td>
</tr>
<tr>
<td>renaming</td>
<td>23</td>
</tr>
<tr>
<td>security</td>
<td>30</td>
</tr>
<tr>
<td>sharing</td>
<td>30, 34</td>
</tr>
<tr>
<td>time last modified</td>
<td>16</td>
</tr>
<tr>
<td>type</td>
<td>16</td>
</tr>
<tr>
<td>flags</td>
<td>10, 49</td>
</tr>
<tr>
<td>full path name</td>
<td>26, 49</td>
</tr>
<tr>
<td>G</td>
<td></td>
</tr>
<tr>
<td>Groups</td>
<td>31, 49</td>
</tr>
<tr>
<td>H</td>
<td></td>
</tr>
<tr>
<td>hard copy</td>
<td>20</td>
</tr>
<tr>
<td>help</td>
<td>8</td>
</tr>
<tr>
<td>Hidden files</td>
<td>17</td>
</tr>
<tr>
<td>hierarchical file structure</td>
<td>25</td>
</tr>
<tr>
<td>home directory</td>
<td>26, 34, 49</td>
</tr>
<tr>
<td>I</td>
<td></td>
</tr>
<tr>
<td>IBM PC</td>
<td>45</td>
</tr>
<tr>
<td>information about files</td>
<td>16</td>
</tr>
<tr>
<td>inkjet printers</td>
<td>20</td>
</tr>
<tr>
<td>input to a command</td>
<td>11</td>
</tr>
<tr>
<td>inspecting a file</td>
<td>18</td>
</tr>
<tr>
<td>interrupting commands</td>
<td>14</td>
</tr>
<tr>
<td>J</td>
<td></td>
</tr>
<tr>
<td>J.Bloggs</td>
<td>25</td>
</tr>
<tr>
<td>joining files together</td>
<td>23</td>
</tr>
<tr>
<td>K</td>
<td></td>
</tr>
<tr>
<td>Korn shell</td>
<td>12</td>
</tr>
<tr>
<td>L</td>
<td></td>
</tr>
<tr>
<td>laser printers</td>
<td>20</td>
</tr>
<tr>
<td>length of file</td>
<td>16</td>
</tr>
<tr>
<td>line printers</td>
<td>20</td>
</tr>
<tr>
<td>listing a file</td>
<td>18, 20</td>
</tr>
<tr>
<td>logging off</td>
<td>49</td>
</tr>
<tr>
<td>logging on</td>
<td>49</td>
</tr>
<tr>
<td>logout (UNIX command)</td>
<td>6</td>
</tr>
<tr>
<td>looking at a file</td>
<td>18</td>
</tr>
<tr>
<td>lpq (UNIX command)</td>
<td>22</td>
</tr>
<tr>
<td>lpr (UNIX command)</td>
<td>20</td>
</tr>
<tr>
<td>lprm (UNIX command)</td>
<td>22</td>
</tr>
<tr>
<td>ls (UNIX command)</td>
<td>9, 15, 27</td>
</tr>
</tbody>
</table>
ls -a (UNIX command) 17
ls -g (UNIX command) 32
ls -l (UNIX command) 16, 28, 30

M
Macintosh 45
make new directory 27
man (manual command) 7
matrix printers 20
MicroEMACS documentation 50
MicroEMACS: a text editor 37
mkdir (UNIX command) 27
modes (access permissions) 32
more (UNIX command) 19
moving files 23, 28
moving files into a directory 28
moving to another directory 27
mv 24
mv (UNIX command) 23, 28
with interactive option 24

N
Network News 43, 50
nn (for network news) 43
numerical access modes 33

O
octal numbers for file permissions 33
On-line Help 7
options (command modifiers) 10, 49

P
 pagers 19, 49
parent directory 26
password 3, 49
PATH (environment variable) 36
path names 26, 49
abbreviating with ~ 26, 34
PC 45
permitting files 30
pine (for email) 50
Pipe 49
Pipelines 11
PostScript 21, 50
PostScript printers 20
print queues 22
print working directory 27
printenv (UNIX command) 21, 35
PRINTER (environment variable) 35
printer names 20
printing files 20
profile files 36
putting files into a directory 28
pwd (UNIX command) 27

Q
queue of documents for printing 22
queues
checking 22
removing print jobs from 22

R
read permission 30, 33
ream (for email) 50
redirecting input and output 11
reference cards for UNIX 50
relative path name 26, 50
removing a directory 29
removing a file 24
removing jobs from print queues 22
rename a file 23
renaming a file 23
rm (UNIX command) 24
with interactive option 24, 29
rm -r (UNIX command) 29
rmdir (UNIX command) 29
root directory 25, 26, 50

S
scripts 12, 50
security of files 30
sending files to a printer 20
setting your
delivery 36
printer 35
search PATH 36
terminal 36
setting your environment 35
sharing files 30, 34
shell scripts 12
shells 12, 50
standard input file 11
standard output file 11
swapping files 34
system manuals 7

T
TC shell 12
telnet 45
TERM (environment variable) 36
terminal 10, 50
text editing 19
text editors 37
tin (for network news) 43, 50
Index

U
UNIX
commands 9
documentation 7
Environment 35
file system 25
pipes 11
reference cards 50
shells 12
UNIXhelp 8
Usenet 43, 50
username 50

V
Visual Display Unit (VDU) 3

W
wc (UNIX command) 10
who (UNIX command) 9
wild cards 15, 24, 50
workgroup 31
World Wide Web 8
write permission 30, 33

X
X terminal 3
Xlogin 45
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:

EUCS Documentation Team,
Computing Services,
Edinburgh University,
Main Library,
George Square,
Edinburgh EH8 9LJ

or by email to: EUCS.Documentation.Team@ed.ac.uk

Document: UNIX Introductory Guide

Edition: Aug 2002

Your name and address:

Comments: