Unix from the command line

Cambridge University Engineering Department
Teaching Workstations

Richard Prager, Tim Love
August 22, 2005

This document describes how to use the teaching system from a network terminal by issuing Unix shell commands. If an X-windows terminal is available new users should work through the "Introduction for new users" before starting this document.

Contents

1 Shell Commands

This is a guide for more advanced users on a X-windows workstation, or users who are working on a non-graphics terminal. If you are using a X-windows workstation (as in the DPO) you should work through the "Introduction for new users" before starting this document.

The "Introduction for new users" document describes how to log on to the teaching system and perform simple file manipulation using the *File Manager* graphical environment. This document provides an overview of the facilities provided by the command line and then describes some of the more useful utilities on the system.

1.1 Typing Commands to the Shell

This document is concerned with using the Unix operating system through a command line interpreter - a *shell*. Several shells are available. This document

covers the most common ones (`bash`, the Posix shell, etc).

You'll need a "terminal window" to type commands into. On our machines you'll find a "terminal window" program on the "Start" menu.

Firstly, type the command **date**, and press the `<Return>` key. This prints out the time and day. (Typing the name of a program executes that program just the same as double clicking on its icon in the *File Manager*).

Notice that you can use the `<Back space>` key to correct typing errors.

The interrupt character is `C-c` (*i.e.* hold the `CTRL` key down and press `c`). This is useful if you give a command or start a program which is either broken or is taking too long to finish. To abort you just type `C-c`. To try this, type the command

sleep 1000

This will make the computer wait for 1000 seconds before asking you for the next command. Let's assume you don't want to wait for a quarter of an hour so type `C-c`. Shortly the prompt will return. The **sleep** command has been aborted.

Throughout this guide the notation '`C-`' will be used to indicate use of the `CTRL` key (*e.g.* `C-c` for interrupt as above).

If you are on a non-graphics terminal you should now read section 2.1 describing how to use the editor with control keys. Then return and continue from here. If you are on a X-windows workstation you can work straight on.

1.2   File Manipulation

Create a file in your home directory called `flowers` with the names of some flowers in it. If you have not got the editor Emacs running you should start it up now to edit the file `flowers`. When you have typed in the names of the flowers save the file `flowers` to disk. Use the **ls** command in the terminal window to examine the files in your current working directory (**ls** is short for 'list files'). You can obtain a more detailed listing using **ls -l**.

Copying Files:- Use the **cp** command to make a copy of the file `flowers`. Call your second copy `myplants`. You should type:

cp flowers myplants

Use **ls** again to see that the new file has appeared

Leave the `flowers` file alone but edit `myplants` and add the names of some plants which are not flowers. When you have finished adding to the `myplants` file, save it to disk.

3

Moving Files:- The name `myplants` is rather long so let's rename this file. Use the command:

```
mv myplants plants
```

to rename the file `plants`. If you already have a file of that name, **mv** will prompt for confirmation. The mnemonic **mv** is short for 'move.' Do an **ls** again to see what you have done.

Viewing and printing Files:- To look at the contents of files without editing them you use the command **more**. Try

```
more flowers
```

The command to print out a copy of one of your files on the line printer is **lp**. For example, **lp flowers** would print out the file `flowers`. Please don't do this now or there will be rather a lot of wasted paper.

1.3   Directories

It would be nice to have somewhere to put the two files you have created so they can be together. Let's create a new directory. Type

```
mkdir garden
```

If you do an **ls** now you will see that `garden` has appeared just like `flowers` and `plants`. If you type **ls -l**, instead of the normal file pattern of `-rw-------` you will see `drwx------`. The 'd' in the first column tells you this is a directory not just an ordinary file. It is a place where other files (and indeed directories!) live, it does not contain any user data itself, so it's wrong to try to edit a directory. (If you want to know what the `rwx------` means, type **man chmod**. However, we are not concerned with this at present.)

Now we have an appropriate place to put `flowers` and `plants` we need to move them there. The command to do this is **mv**.

```
mv plants garden/plants
mv flowers garden/flowers
```

Now look at these files in their new home by typing **ls garden**. The command **ls** on its own will confirm that they are no longer in your home directory.

The directories form a tree structure. Each directory contains entries for the files and sub-directories it contains as well as a link back to itself called '.'

4

and a link to its parent called '..'. You won't see these when you type **ls** because **ls** normally does not display any directory entry which begins with a dot. Use **ls -al** to see absolutely all the files in your current directory.

It is also possible to change your current working directory to garden. Before we change anything let's establish where we are to start with. The command for this is **pwd** which stands for "print working directory". Try it now.

Now try the following commands and see how the response of **pwd** changes as you move around the file system.

```
cd garden
pwd
ls
ls ..
```

By using the **cd** command to change your working directory your perspective on the file system has changed. You can get back to the directory you came from (which in this case is your home directory) by using the **cd ..** command. You can get back to your home directory from anywhere by typing **cd** on its own.

Deleting Files & Directories:- Make a copy of flowers called doomed. This is a file for us to practice deleting. The delete command is **rm** (short for remove), but it is very easy to destroy valuable work irrecoverably with this, so use it with great care. Delete doomed with the command

```
rm doomed
```

rm will ask for confirmation that you really want to delete the file. Just like the **mv** command mentioned above, its default action has been adjusted to make it a safer command. Make another directory using **mkdir**. When a directory is empty you can delete it using **rmdir**. Try this on the directory you have just created. If you want to delete a directory with files in you have to use **ls -al** to see all the files, and then remove them (with **rm**) before you can get rid of the directory.

1.4 Processes and Job Control

To run a command you just type its name, *e.g.* **date**.

If you start the command this way then you cannot use the terminal for anything else until it has finished. This is satisfactory provided your command is not going to take very long.

Another alternative is to start the command 'in the background.' This sets the command going then prompts you again in case there is anything you want to do while the command is running. To start a command in the background you type an '&' at the end of the command. Try

```
sleep 1000 &
```

sleep is a command that does nothing for the number of seconds you specify, so what we have started is a command which sleeps in the background for 1000 seconds. This is not a very useful command but it could equally well be a long computation job.

You can find out what background jobs you have running using the **jobs** command. Try it now.

Suppose you want to bring your **sleep** job into the foreground now. If it was a real program it might need some input for instance. The way to do this is to type **fg**.

Notice that the prompt does not come back. Now that the job is in the foreground you can either kill it (forever) with C-c, or stop it with C-z (hold down CTRL and type z).

Stop the sleep command by typing C-z. You can now return it to running in the background with the command **bg**.

If you want to get rid of the **sleep** command now, bring it into the foreground using **fg** and then kill it with C-c.

Note that all your processes, whether in foreground or background, are likely to be killed when you log off. If you need to run long jobs then consult the online manual page for **batch**.

1.5   The Command History & Editing Commands

The shell keeps a record of the commands you use to help you avoid retyping things when the command you want to execute is similar to one you have used before. This record of the last few commands you have used is called your *command history*. Different shells offer different facilities for re-using old command lines.

You can always access the history using control characters or the arrow keys. To look at the last command in your history, press the up-arrow key. Doing this again takes you back over older commands. The down-arrow key displays more recent commands.

To execute an old command again use the arrow keys to display it then just press the <Return> key. If you decide that you don't want to repeat any of your old commands you can erase the command line with C-x.

Try a few simple commands. Type

```
date
sleep 1
hostname
```

Now repeat them in a different order using up-arrow to go back and select previous commands. The command **hostname** prints the name of the computer you are working on.

It is possible to modify old commands if you want to execute something similar to one of them. Pressing the left-arrow key moves the cursor backwards across the command, and the right-arrow key moves forward.

To add to a command you position the cursor at the point in the command you wish to alter using the left and right arrow keys, and type in the additional text.

To delete from a command you position the cursor in the same way and then use C-d to delete the character under the cursor. Furthermore C-k erases to the end of the current command line. You can also use the <Back Space> key on the keyboard to delete backwards from the cursor.

Pressing C-r produces the prompt ' ^R ', if you now type a small part of a command you would like to repeat and press <Return> the shell will search back through your history for a command which contains the character string you supplied. If it finds the old command you wanted you can then edit it and/or execute it as described above. To summarise :-

up-arrow - Moves up the history to the previous command.

down-arrow - Moves down the history to the next most recent command.

left-arrow - Moves cursor back to the left over a command.

right-arrow - Moves cursor forward to the right over a command.

C-d - Deletes the character under the cursor.

C-k - Kills all the characters from the current cursor position to the end of the command line.

C-r *string -* Display the most recent command which contained *string*.

1.6   Filename Completion & Wildcards

The program **wc** counts the number of lines, words and letters in any files it is given. Suppose we have two files averylongname, hissillyname and we

wish to run **wc** with these names as arguments

 wc averylongname hissillyname

It is tedious to type out long filenames like this in full. Instead, we can use *filename completion*. To use this you type enough of the file name to specify the file uniquely and then press the escape key twice `<ESC><ESC>`. The shell will then try and type in the rest of the file name for you if it can work out which file you want. So we might type

 wc aver`<TAB><TAB>`

and the shell would complete the name to

 wc averylongname

We then add the characters `hissi<TAB><TAB>` and the shell completes the command. This technique is useful, but it is sometimes awkward to specify filenames uniquely with their first few characters. In these circumstances it is sometimes convenient to use *wildcard characters*. Suppose we have three files called `FileJim`, `FileJames` and `FileJoyce`. The first characters of each filename are all rather similar and we want to type

 wc FileJim FileJames FileJoyce

Instead of all this typing we could use the wild card character '`*`'. This can be used to replace *any* string of characters (except an initial full stop character). So an alternative command would be **wc *** which would pass the names of all the files in the current directory to **wc**. However this would produce undesirable results if there were other files apart from `FileJim`, `FileJames` and `FileJoyce` in the current directory. Alternatively:

 wc File*

would only use files whose names began 'File.' The character '`?`' is also a wild card, but it can only represent a single character. For example:

 rm F*J????

will remove `FileJames` and `FileJoyce` because they both have a `F` followed by any other characters, then `J` followed by four other characters. The

file `FverysillylongfilenameJ1234` would also be deleted.

1.7   Redirection and Pipes

Processes have a standard input channel (*stdin*), a standard output channel (*stdout*) and a standard error channel (*stderr*). By default the keyboard is standard input, and output plus errors go to the screen, but it is one of the strengths of Unix that a process needn't know where its input came from and all the channels can easily be redirected to files. Type **date** and the date will be shown on screen, but type

date > **out**

and the output goes into the file called `out`. You can check this using the command **more out** which will write the contents of `out` onto the screen. **who** is a command which prints out who is logged on to the workstations near to you.

who >> **out**

(note the double '>' signs) will append the output of **who** to the file `out`. Check this again using **more** (**more** will display the first page of the file; to see successive pages press the `<space>` bar). You can also redirect output into another process's input. This is done using the ' | ' character (found above the `<Return>` key). Type

date | wc

and the output from date will be 'piped' straight into a program that counts the number of lines, words and characters in its input. Another pipe example is

cat text | spell

which would send the contents of `text` not to the screen, but through a spell checker. The spelling checker will then print out those words in the file text which are not in the online dictionary.

1.8   Summary of Commands

(For any commands not listed below try typing "*man*" or "*man -k*" followed by the command name) **File and Directory handling**

- **cat** stands for concatenate and takes any number of file names as arguments. The contents of the files are printed to the standard output (usually the screen) one after another.

- **more** takes any number of files as arguments and displays them one after another on the screen, pausing between screenfulls. Press the <space> bar to continue.

- **tail** takes a file as its argument and prints out the last 10 lines of the file.

- **cp** copies files, from its first argument to its second.

- **mv** renames files, from its first argument to its second.

- **chmod** changes the permissions on a file, *i.e.* who can read, write and execute it. The first arguments determine how the permissions are changed and all successive arguments are files to operate on. (Type **man chmod** for details).

- **rm** deletes files. Once removed, files can never be recovered so use this command with great care.

- **mkdir** takes an argument and makes a new directory with that name.

- **rmdir** takes a directory name as argument and deletes that directory. This only works if the directory is empty.

- **cd** changes the directory you are working in. If called without any arguments it takes you back to your home directory.

- **pwd** prints your current working directory.

- **ls** lists the files in the current working directory. If you give a different directory as an argument then the files in that directory are listed instead. If you use **ls -l** a long format listing is produced. Use **ls -al** to see the 'dot files' as well.

Utilities

- **passwd** lets you change your password.

- **date** prints the date and time.

- **hostname** prints out the name of the machine you are working on.

- **who** shows who is logged into the workstations near to you.

- **man** takes a unix topic as argument and prints its manual entry. The command **man -k** *string* supplies unix topics related to a given string.

- **lp** takes file name arguments and prints these files on the line printer.

- **sleep** takes a numeric argument and does nothing for that number of seconds.

- **bc** is a desk calculator. You type **bc** on a line by itself and then type in expressions for it to evaluate. Gives the answer to things like 3+4. For real numbers you will have to say **scale=10**. Finish with C-d.

- **wc** takes filenames as arguments and counts the number of characters, words and lines in each.

Job Control

- **fg** brings the current background job into the foreground.

- **bg** runs the current stopped job (if any) in the background.

- C-z suspends the current foreground job.

- The interrupt character is C-c (*i.e.* hold the control key down and press c).

2   Utilities

2.1   Emacs using Control Keys

This section describes how to use **emacs** using the control keys. Although learning to operate **emacs** in this way may seem difficult at first, much of the rich functionality of the editor is only available once you have learnt to drive it with the control keys. If you need to use Emacs from a dumb terminal the control key method might be the only one available. In what follows C- means use the control key and M- means use the meta key. On some keyboards the meta key is labelled 'Extend char'.

Some terminals will have their labelled function keys bound to appropriate things (*e.g.* 'search' might search-forward).

The DISPLAY Environment Variable:- *Miss out this subsection unless you are a fairly advanced user, or you want to use **emacs** on a dumb terminal (not running X-windows).*

When you invoke **emacs** on the shell command line, **emacs** determines whether to use the terminal window or start up its own X-window. To do this **emacs** looks at the 'DISPLAY' environment variable. If the DISPLAY variable is set then **emacs** tries to start up an X-window on that display otherwise it uses the terminal.

If you are using a dumb terminal (*i.e.* not X-windows) your DISPLAY variable should not be set. If you are on a X-windows workstation then it should be set.

You can check that the DISPLAY variable is in the correct state with the command:

```
print $DISPLAY
```

If you get a blank reply then the variable is not set. If you get something like `tw101.eng.cam.ac.uk:0` then it is set. If the variable is in the wrong state then change it as follows.

You can unset the `DISPLAY` variable by typing

unset DISPLAY

on a line by itself. To set the variable to the display of the machine you are sat at, look for the label on the machine for the machine's name (maybe `tw305`) then type

export DISPLAY=tw305:0

Starting Emacs from the Command Line:- To start the editor from the command line you type its name – **emacs**.

If you are using a X-windows workstation (and have set the `DISPLAY` variable) then you should add an ampersand. This will enable you to continue to use the terminal window while the editor is executing by running the editor "in the background". The command is thus **emacs &**.

The Emacs Tutorial:- Once **emacs** is running you can obtain an online tutorial by typing `C-h t` (hold the control key down and type 'h' then let go of the control key and type 't'). This is a very good way to start to learn control key use of the editor. It is recommended that you do at least part of this tutorial now.

Emacs Help & the Info System:- Typing `C-h` gets you into the Emacs help system. This is useful for finding out what a given control sequence does or whether emacs has a command to do something in particular.

Type `C-h k` and then any of the control sequences in the previous section and emacs will explain what it does. You will have to type `C-x 1` to get rid of the help when you have finished reading it.

Type `C-h a` and you will be prompted for a string. Type in `spell` to find out what spelling checking emacs provides. A number of commands are provided, only one of which (`spell-word`) is bound to control keys. To access a command which is not bound to a control sequence you type `M-x` (or `C-[ x`) and then type in the command name.

To find out if a command is bound to a control key sequence you can use `C-h w`. When the prompt appears specify `spell-word` and emacs will tell you which key sequence it is bound to.

More descriptive online documentation for emacs – and several other topics – is provided by the *info system*. This is entered using the command `C-h i`. Once you are in the info system, typing h will give you a primer for first time users. A list of simple commands is given below.

`d`	Go to top directory node.
`<space>`	Go to next page down in node.
`<Back space>`	Go to previous page in node.
`n`	Go to next node.
`m`	Select menu item.
`l`	Return to last visited node.
`q`	quit.

Summary of basic Emacs commands :-

`C-p`	Moves the cursor up to the previous line in the file.
`C-n`	Moves the cursor down to the next line in the file.
`C-b`	Moves the cursor one position back to the left.
`C-f`	Moves the cursor forward one position to the right.
`C-d`	Deletes the character under the cursor.
`C-k`	Kills all the characters from the current cursor position to the end of the line.
`C-space`	Marks a position.
`C-w`	Kills text from text pointer to marked position
`M-w`	Copies text from text pointer to marked position
`C-y`	Yanks back the most recent kill.
`C-r`*string*	Incrementally searches backwards for the *string* specified.
`C-s`*string*	Incrementally searches forwards for the *string* specified. *Beware of using `C-s` on dumb terminals as it might lock up the terminal and you will have to type `C-q` to fix it.*
C-v	Displays the next page down in the file.
M-v	Displays the next page up in the file. (If your terminal does not have a meta key you can use `C-[` or `ESC`.)
C-g	Interrupts the current command.
C-x C-f	Prompts you to select a new file to edit.
C-x s	Saves files currently being edited.
C-l	Redraws the screen with the cursor in the middle.
C-x C-c	Offers to save all files to disk then exits the editor.

2.2 Compilers

The Fortran 90, C and C++ compilers work in very similar ways. Creating a program you can run from program text files involves two main steps; compiling the source files to produce object code files, and linking the object files

to form an executable file.

Fortran 90 source files usually have the suffix `.f90`, C source files have `.c`, and C++ source files have `.cc`.

Compiling a Single Source File:- If the program is contained in only one file then the compilation and linking together can be performed in one step. The commands for each of the three languages are as follows:

```
gcc myprog.c -o myprog

g++ myprog.cc -o myprog

g77 myprog.f -o myprog
```

The `-o myprog` instructs it to name the executable program `myprog` in each case. If the `-o` flag and its argument are not present the name of the executable is `a.out`.

Compiling Multiple Source Files:- If the program is composed of several source files then these must be compiled individually to produce object files with the suffix `.o`. These `.o` files are then linked to generate the program `complete_prog`.

```
g++ -c firstprog.cc

g++ -c secprog.cc

g++ firstprog.o secprog.o -o complete_prog
```

The `-c` flag makes the compiler generate `.o` files rather than attempting to link the program to form a complete executable program.

2.3   Mail

The mail system enables you to send a message to another user once you know their user identifier. Type the command **pine**. This gives you **pine**'s main menu of commands. For simple use you need **C** which lets you compose and send a message, and **I** that gives you an index of the mail you have received for you to read.

At the bottom of the window **pine** gives a summary of what commands are available: as you move to different parts of **pine** the summary changes to give you appropriate help.

First try mailing yourself - type C. You will be presented with a screen to fill in

```
To      :
Cc      :
Attchmnt:
Subject :
----- Message Text -----
```

Put your *id* on the `To` line, leave the `Cc` and `Attchmnt` lines blank and make up a `Subject` line for your message.

Then move down under the Message Text line and write your message. You are now in **pine**'s simple builtin editor - you can use the cursor keys to move around and the help at the bottom of the window tell you of other commands you can use. When you've completed your message, send it off by pressing the `CTRL` key and pressing the `X` key (i.e. **C-x**.

Wait a few seconds, and if you are still in the main menu, type **I** to see if your mail has arrived. If you are already in your mail index, **pine** will check for new mail and automatically add it to your index every couple of minutes - type **C-l** to get it to check now. When in the index you can read mail by typing the **Return** key, and then **I** to get back to the index once you have read it. When you have finished with a piece of mail you can delete it with the **D** command.

If you've got a friend nearby, try sending them an email message by putting their *userid* in the **To:** field, and get them to send you a message too. You can then use the **R** command to reply to their email. Wait a few seconds, make sure you're in the main menu (type `M` if you're not) and type `I` to see if your mail's arrived.

Typing `Q` will terminate the mailer.

2.4   Text Processing

TEX is a powerful text processing language used throughout the academic community. It provides a fairly low level support for document writing. You have complete control over the style and layout and it is possible to work with a minimal number of macros. TEX provides an unrivaled facility for elegant typesetting of mathematics.

New commands (or *macros*) can easily be added to TEX. LATEX is a popular macro package which sits on top of TEX and takes the drudgery out of producing simpler documents as well as providing the structuring facilities to help with writing large documents. Automated chapter and section macros are provided, together with cross referencing and bibliography macros. Except for specialised work, you should use LATEX; all the benefits of plain TEX are still present when it comes to doing maths.

In this document we will present an outline of some of the simple features of LaTeX. Full LaTeX documentation can be found in *"A Document Preparation System: LaTeX User's Guide and Reference Manual."* by *Leslie Lamport.* Addison-Wesley 1994. Many handouts are also available.

TeX itself is fully documented in *"The TeXbook."* by *Donald E. Knuth.* American Mathematical Society and Addison-Wesley 1984.

2.5 LaTeX

The best way to get started with LaTeX is to look at a simple example. A short document is reproduced below, a file with a similar structure can be found in /export/Examples/LaTeX/demo0.tex

```
% EVERYTHING TO THE RIGHT OF A % IS A COMMENT
% THE FOLLOWING 10 CHARACTERS HAVE SPECIAL MEANINGS
%        &  $  #  %  _  {  }  ^  ~  \

\documentclass{article}
\usepackage{a4}
\begin{document}

\section{Experimental Results} % This command makes a section title.

Text is just entered in a fairly straightforward way, using a
blank line to separate paragraphs.  Formulae are surrounded by
brackets \emph{e.g.} \( x-3y=7 \) which will produce an equation
in the midst  of the text.

The \emph{e.g.} told latex to set `e.g.' in italic type.  You can
also get bold text using \textbf{this is bold}.

Subscripts are written \( x_{2y} \) and superscripts are written
\( x^{2y} \).  These are both in-line formulae again.

To get displayed equations you say something like
\[ y = ax^{2} + bx + c \]
Notice that square brackets are used rather than round ones.


\subsection{Conclusions} % This command makes a subsection title.

Double quotes are typed like this: ``quoted text''.
Single quotes are typed like this: `single-quoted text'.
Long dashes are typed as three dash characters---like this.
```

```
\end{document}      % The input file ends with this command.
```

Once you have created a LaTeX source file (called `myfile.tex`, say) it must be processed by LaTeX before it can be printed out. Use the command **latex myfile.tex** which will produce `myfile.log`, `myfile.aux` and `myfile.dvi`. If you are using various sorts of cross referencing then you may have to run LaTeX more than once. If you want an automated bibliography you will also have to run **bibtex** (see the 'LaTeX local guide' for details).

When this procedure is complete you will have a `myfile.dvi` to print out. This is a device independent representation of your document which can be displayed on an X-windows screen using the **xdvi** program or printed on a laser printer using an appropriate driving program. On the teaching system you can type **dvilp myfile**.

2.6   Summary of Commands

(NB many of these commands have friendlier graphical versions. If you're at a graphics terminal, browse through the menus)

- **g++** is a C++ compiler. It takes many flags and the name of at least one file. When it's given object code files, linking is performed to produce an executable.

- **gcc** is a C compiler.

- **g77** is a Fortran compiler.

- **pine** lets you read and write mail messages.

- If you have not got **emacs** running and you wish to edit a file you can start **emacs** from a shell. In a window environment use

  emacs *filename* **&**

  Omit the **&** on a dumb terminal.

- Once **emacs** is running type `C-h t` for the online tutorial or `C-h i` to enter the info system.

- **latex** takes a LaTeX file name as an argument and generates a `.dvi` file for the document.

18

3 Getting help

help Typing `help` will start up a menu-driven program that gives you access to all the locally written documentation and much else besides. It's the first thing to try if you want help. This information is part of the department's web info and can be accessed through the department's World Wide Web home page.

Online Manual:- The unix online manual gives detailed information on every program on the system. Although it is sometimes quite hard to understand it does cover a vast quantity of material.

If you are not sure what command to use for something, a useful way to find out is to try

`man -k` *topic*

which will print out a single line for all the unix manual entries which contain the word *topic* in their title. You can then find out more using `man` *ManualEntry*.

Apart from the online manuals there are a number of useful printed guides which can be borrowed from the machine room at the far end of the Design and Project office. There is a program called `manuals` which tells you which manuals can be borrowed.

lynx gives access to information that's on the World Wide Web. It's the text-only counterpart to `Netscape`. Most of the world's available on-line information can be reached, so the sooner you start browsing the better.

trn and xrn give you access to many thousands of international bulletin boards as well as many local to Cambridge University. `xrn` can only be used on X-terminals - `trn` can be used from the command line on any terminal.

There are a couple of engineering department newsgroups of particular interest to students: `ucam.eng.suggest` can be used to make suggestions about the teaching system, and `ucam.eng.students` is for engineering students to discuss anything they like which is at least vaguely related to the department, engineering or the course. It's a good idea to read a newsgroup for a couple of weeks before starting to contribute to avoid getting "flamed" for bad "netiquette" - it's also a good idea to look at the newsgroup `news.announce.newusers` which tells you about the conventions used on newsgroups.

sysnews is used to read system announcements - you should read these since they contain important information about system changes, scheduled downtime etc which may affect your work. Type `sysnews` to read all outstanding system news. `man sysnews` will tell you how

to read articles you have already read if you want to check on an old announcement.

Commands giving their own help:- Many of the graphics commands now accept `-help` as a flag. *E.g.*, **`xterm -help`** gives a lot of assistance.

Handouts:- These are kept in an office by the entrance to the DPO machine room. The information in the handouts is duplicated in the much more comprehensive help system, so look there first.

mail:- You can use electronic mail to report malfunctions and get help. Here's a list of people to mail –

`operators`	(the operators). General problems
`help`	More advanced problems, general advice
`postmaster`	Electronic mail problems
`newsmaster`	News problems
`webadmin`	WWW problems
`sysman`	Serious system problems
`bugs`	Bug reports
`comments`	General comments

People:- Near the DPO are programming advisors. Contact Tim Love on 32746 in case of difficulties (`e-mail tpl`)