



The University of Sheffield

CiCS

Introduction to...

UNIX

**Bob Booth
December 2004
AP-UNIX2**

© University of Sheffield

Contents

1.	INTRODUCTION.....	3
1.1	THE SHELL	3
1.2	FORMAT OF COMMANDS	4
1.3	ENTERING COMMANDS	4
2.	ACCESSING UNIX MACHINES.....	5
2.1	TERMINAL EMULATION SOFTWARE.....	5
2.2	LOGGING IN.....	6
2.3	LOGGING OUT	6
2.4	CHANGING YOUR PASSWORD.....	8
3.	BASIC UNIX COMMANDS.....	9
3.1	FILENAMES.....	9
3.2	LISTING YOUR FILES	10
3.3	DISPLAYING THE CONTENTS OF A FILE.....	10
3.4	COPY, RENAME AND REMOVE.....	11
3.5	SEARCHING FILES.....	11
4.	REDIRECTION AND PIPING	12
5.	GETTING HELP	13
6.	WORKING WITH DIRECTORIES.....	14
6.1	WORKING DIRECTORY	14
6.2	CHANGING DIRECTORY	14
6.3	MAKING AND DELETING DIRECTORIES.....	15
6.4	MANAGING YOUR FILESTORE	15
7.	RUNNING PROGRAMS	16
7.1	RUNNING PROGRAMS	16
7.2	CONTROLLING PROGRAMS	16
7.3	RUNNING FORTRAN PROGRAMS	18
8.	CREATING AND EDITING FILES: THE vi EDITOR	19
8.1	THE THREE MODES	19
8.2	SIMPLE EDITING	20
8.3	COMMAND SUMMARY	21
9.	SENDING A FILE TO THE PRINTER.....	22
10.	T-SHELL FEATURES	23
10.1	FILENAME COMPLETION	23
10.2	REPEATING PREVIOUS COMMANDS	24
10.3	COMMAND LINE EDITING	25
10.4	SHELL/ENVIRONMENT VARIABLES.....	25
10.5	COMMAND ALIASING	26
11.	APPENDIX: SUMMARY OF UNIX COMMANDS	27

1. Introduction

The UNIX operating system was developed in the early 1970s by a group of enthusiasts at Bell Laboratories in the USA. Since then it has been modified by a number of different groups and has evolved into many similar but not identical flavours. In addition, UNIX is structured so that the user works within a 'shell' that can be configured by the UNIX administrators, working behind the scenes.

UNIX is a multi-user, multi-tasking operating system, which has the following features:

- Hierarchical File System
- Process Management
- Command Interpreter (Shell)
- Commands:
 - File Management
 - Editors
 - ...

As it is written in the language 'C' it is possible to run UNIX on any machine that has had a 'C' compiler written for it. As a result UNIX is available on many types of machines from super computers to PCs.

1.1 The Shell

The user interface of UNIX is provided by a command language called the *shell*. There are several shells available differing in the facilities they provide. The most common shells are listed below.

- Bourne Shell
- C Shell
- T Shell
- Korn Shell
- Bash (Bourne Again Shell)

At Sheffield you will use the T shell by default.

The T shell running on the UNIX machines at Sheffield is a very *economical* system in which you can perform amazing feats with only a handful of carefully chosen characters. However, the brevity of commands can make them difficult to remember, and it is not always easy to tell from a sequence of commands exactly what is happening.

1.2 Format of Commands

UNIX distinguishes upper case letters from lower case, and *insists* that many commands are written in lower case. Typing a command in upper case will probably generate the response

```
Command not found
```

A typical UNIX command consists of a general command word, which may be followed by optional parameters that specify more precisely what you want the command to do. Many of these options consist of a single letter, making the command brief but not altogether easy to remember. If a command operates on files then the filenames must come after the options.

```
command [option ...] [filename ...]
```

1.3 Entering commands

UNIX is a case sensitive operating system and it insists that (almost) all commands be typed in lowercase. For this reason you should make sure that the **Caps Lock** key on your keyboard is not activated.

If you make a mistake when typing a command you can correct it using the **Backspace** or **Del** key. Within the T-shell you can also use the arrow keys to move around the command line. If you want to clear the entire command line you can hold down the **Ctrl** key and press **u (Ctrl u)**.

So to summarise:

- UNIX is case sensitive
- Most commands are in lower case
- **Backspace** key to correct typing errors
- **Ctrl + u** to clear the command line
- **Ctrl + c** to abort a program or command
- **Enter/Return** sends a typed command to the shell
- Arrow keys may be used within the T-Shell

2. Accessing UNIX Machines

As with all multi-user computers you access UNIX machines via a terminal. Initially terminals were very simple machines consisting of a monitor, a keyboard, and minimal intelligence. You typed commands at the keyboard, the command was sent to the computer, and any response was sent from the computer to the monitor.

Now people want to use their own desktop computers to access these multi-user computers, so they run software that enables their machine to act like a terminal. This process is called *terminal emulation*, and all CICS networked PCs and Macintoshes have terminal emulation software.

2.1 Terminal Emulation Software

Switch on your PC or Macintosh and log on with your usual username and password. Depending on your circumstances use one of the following terminal emulation methods

Managed NT PC Using Exceed

From **Start, Applications, Internet, Exceed** select **XSession**.

In the dialog box that appears select the profile for the machine and click **Run** in the menu bar. In the next dialog box supply your username and your password then click on the **OK** button.

An Xterm window will open and you will be logged in automatically.

Personal PC Using Exceed

Exceed can be obtained from the Computer Centre, and installed on your own machine. You can then set up your own machine profiles following guidelines in the document "Graphical Unix using Exceed".

DOS-Based Telnet

From the DOS prompt, you can access a Unix host by using the telnet command in the form:

```
telnet host
```

substituting the name of the host you wish to access.

Networked Macintosh Using Telnet

Select **Telnet** from the Apple menu or double-click the icon in the MacApplics folder. Then pull down the **File** menu and select **Open Connection**.

In the **Session Name** box type the machine name then click on the **OK** button. A terminal window will open and you can log in as usual.

Windows 95/98/NT Machine

Depending on which software you have installed, you can run Telnet from the **Start** menu, or X Windows from the Exceed software available from the **Start** menu. The procedure is much the same as above.

Web Browsers

On any machine that has a Telnet client you can use any modern Web Browser to access a UNIX host. Type in an address of the form **telnet://titania.shef.ac.uk** and the telnet session will start either in a separate Telnet window, or within the Web Browser itself. You can then save this address in your bookmarks.

2.2 Logging In

Whichever terminal emulation method you use, you will be asked to login the your specified host. This document uses the machine titania in its examples, but the procedure is the same for other CICS UNIX hosts.

```
SunOS 5.8
login:
```

Enter your UNIX username in lower case, then press Enter. You will be prompted

```
Password:
```

to which you should respond with your password (in the correct caSe). If you have logged in successfully you will see a few messages followed by the prompt:

```
titania{username}41:
```

This will always show your username. When you see this prompt titania is waiting for you to type in a command.

2.3 Logging Out

You logout with the

```
logout
```

command. Never leave your machine logged into UNIX after you have completed your work.

2.4 Changing Your Password

To change your password, use the web form at <http://www.shef.ac.uk/cics/password>

It is **vitaly important to choose a secure password**, not a dictionary word or your date of birth, for example. The password may be as long as you like, but only the first eight characters are significant. Your password should be at least 6 characters long, and should be a mixture of upper and lower caSe letters and numbers. It will not allow any non alphanumeric character. This password format is not a Unix restriction, but is a format that will work consistently on all CICS systems.

Examples (which you should not now use!) of good passwords are:

```
HondaV12
Oneplus2
39steps
```

Examples of 'bad' passwords are:

```
Mozart      (too well known)
Jane        (familiar female name)
holiday     (in any dictionary)
titania     (very silly!)
```

or indeed any other word that might appear in a comprehensive English dictionary.

3. Basic UNIX Commands

Basic UNIX commands consist of a lower case command word that uses the minimum number of characters to signify its function. Command words may be followed by one or more optional parameters, often consisting of a single letter preceded by a hyphen. For example:

```
ls          list directory.
ls -l       list directory in long format.
ls -al      list directory in long format, listing all entries.
```

Format of UNIX commands

```
command [option ...] [filename ...]
```

3.1 Filenames

When you work on a computer you create files. Files hold information such as programs, data sets and text passages. Each file is distinguished by a unique name.

Filenames in UNIX may include any number of letters and digits. You can use full stops, hyphens, and underscores (. - _) to divide the name into sections for clarity. There must be no spaces within a filename. For example a name of a report for 11 March 1991 might be.

```
rep.11-3-91
```

UNIX distinguishes between upper and lower case, so the two filenames MYFILE and myfile are different.

The asterisk * has a special meaning, it is called a wildcard character. We use the * to stand for any character or sequence of characters.

3.2 Listing Your Files

The first thing you need to be able to do is produce a list of your files. To do this you use the `ls` command with the options `-l` (long listing) and `-a` (all entries). So you would type in

```
ls -al
```

and you will receive a response similar to:

```
titania{course01}41: ls -al
total 9
drwxr-xr-x  2 course01      512 Oct 18 14:31 .
drwxr-xr-x 27 root         512 Oct 17 09:52 ..
-rw-r--r--  1 course01     715 Mar 18 1991 .cshrc
-rw-----  1 course01      27 Oct 18 14:31 .history
-rw-----  1 course01     708 Jan 25 1993 .login
-rw-----  1 course01     402 Feb  5 1991 .logout
-rw-----  1 course01    1891 Oct 18 11:09 UNIX.intro
-rw-----  1 course01       0 Oct 18 11:08 empty_file
-rw-----  1 course01      57 Oct 18 11:09 hello.c
titania{ course01}42:
```

From left to right this list details:

- File type and access permissions
- Number of links
- Name of the owner
- Number of bytes (characters) in the file
- Date and time the file or directory was last updated
- File or directory name

3.3 Displaying the Contents of a File

You can look at your files on the screen using the `more` command:

```
more filename
```

The contents of the file will be displayed and, at the bottom of the screen will be the prompt:

```
--More--(xx%)
```

To continue you can press:

- **Spacebar** next screenful
- **Enter** next line
- **q** quit listing
- **?** list commands

The `more` command has many subcommands for finding strings and moving up and down the file. You can find out about these by consulting the on-line help described in section 5.

3.4 Copy, Rename and Remove

The command to copy a file is `cp` and the syntax is:

```
cp oldfile newfile
```

To copy a file called `empty_file` to a file called `copyfile` use the command:

```
cp empty_file copyfile
```

To rename and move files we use the `mv` (move) command. For example, to rename the file called `copyfile` to `newfile` we would use

```
mv copyfile newfile
```

If you attempt to to overwrite an existing file at Sheffield, then the T-shell will generate a warning. This behaviour is not the default behaviour (which enables overwriting). This functionality has been enabled on CICS UNIX machines.

Lists of one or more files can be deleted with or without confirmation using the `rm` command. The command was designed to delete files without confirmation, unless otherwise stated. However at CICS we have re-configured our machines so that the command asks you to confirm each delete.

To delete files with confirmation before each one, type

```
rm file1 file2 ...
```

Or you can bypass our configuration by preceding the command with the `\` character. So to delete files without confirmation the command is

```
\rm file1 file2 ...
```

Be careful when using this command not to delete your entire filestore.

3.5 Searching Files

To search for a particular character string in a file, use the `grep` command. The basic syntax of the `grep` command is:

```
grep string file
```

Where *string* is the word or phrase you want to find, and *file* is the file or files to be searched. The following example will list all the lines which contain the string `UNIX` in a file called `tutorial`:

```
grep UNIX tutorial
```

4. Redirection and Piping

The results from a command are normally displayed on the screen. There are special symbols which redirect the output of a command. For example, you may want to save the output to a file rather than display it on the screen.

For example, to save a directory listing in a file called `dir_list` you would use:

```
ls -l > dir_list
```

The output of one command can become the input to another command. Commands strung together in this fashion are called a pipeline. The symbol for this kind of redirection is a vertical bar `|` called a pipe.

```
who | wc -l
```

The command `wc` with the `-l` option counts lines. So the pipeline is counting the number of users who are logged in.

Another example is to use the `more` command to scan through a long directory listing

```
ls -al | more
```

Using the pipe symbol with the `grep` command, which searches for strings, we can set up a filter. A filter allows you to select only the useful information from the output of a command. To use `grep` as a filter you must pipe the output of the command through `grep`.

The following example displays files that were created in November.

```
ls -la | grep Nov
```

5. Getting Help

Comprehensive on-line help is available for all UNIX commands. The information is stored in files called reference manual pages and these can be viewed using the `man` command. To get a description of any command, type

```
man command
```

So to view the manual pages on, for example, the `ls` command you would type

```
man ls
```

The pages would then be displayed via the `more` command, so you can use the Spacebar to advance the manual page and the `q` key to quit the listing. It is well worth looking at the manual pages on the `more` command simply to help you view the manual pages! In addition you can get a full description of the `man` help system by typing

```
man man
```

UNIX has a rich variety of commands for doing many things. The manual pages give you full information on each command and, usefully, the last few lines of each entry give cross references to other related commands.

If you are having trouble locating information on a particular topic you can type

```
man -k topic | more
```

and this will list possible sources.

If you want to know what software is available on the machine you are using you can type:

```
man software
```

This will take you to a manual page specific to Sheffield, because like all things UNIX the manual pages can be configured and added to.

Finally, to get a list of UNIX commands type

```
man -s 1 Intro
```

This only applies to the CICS UNIX machines at Sheffield.

6. Working with Directories

When you start to work on your UNIX machine you will quickly accumulate files and so will need some means to manage them before they multiply out of control.

To do this you can create a directory structure. Directories are like divisions that can contain files. By using a directory structure you can keep related files together, but separate from other files. In addition any directory can contain further divisions called subdirectories in case you need to sub-divide your groups of files further.

6.1 Working Directory

At the very top of UNIX's directory structure is the *root* directory. This contains a number of directories, each of which contain their own subdirectories. When you log on to UNIX you are put into your own private directory, called your *home* directory. To get an idea where you lie in the directory structure give the command

```
pwd
```

This stands for 'print working directory', and is useful as you navigate the directory structure. You will get a response similar to:

```
/home1/cs/cs4un1
```

Note that the character that divides the directories and filename in a path is a forward slash '/', whereas on a PC, a backslash '\' is used.

6.2 Changing Directory

Once you know where you are you can start to look around using various forms of the change directory command, `cd`.

```
cd ..      Travel up the structure, towards the root, one step at a
           time.
cd /       Jump straight to the root.
cd         Return to your home directory.
cd mydir   Go to a subdirectory called, in this case, mydir.
```

There are many other ways to use this command, but these four will get you wherever you're going.

6.3 Making and Deleting Directories

You can create and remove your own subdirectories using the command:

<code>mkdir mydir2</code>	To make a directory called, in this case, <code>mydir2</code> .
<code>rmdir mydir2</code>	To remove it again.

6.4 Managing Your Filestore

As was mentioned earlier, when you register you are given an allocation of space called your filestore, in which you can store your files etc. If, however, you allow your files to mount up unchecked, you will eventually exceed your filestore limit, and run the risk of corrupting your files.

We try to prevent this from happening by displaying a warning message as soon as you exceed your quota. If you exceed your filestore limit you will be frozen. If your account has been frozen you should delete files to bring your filestore within its quota, after which your account will subsequently be unfrozen.

In order to keep track of your filestore you can use the command

```
quota -v
```

Which will give a response similar to the following:

```
Disk quotas for username (uid 1212):
Filesystem  usage  quota  limit  timeleft  files  quota  limit  timeleft
/home       69    5000  10000      30    9000  10000
/scratch    1 200000 1000000      1    9000  10000
```

Where `files` is the number of files contained in your directory and `usage` is the total number of kilobytes that they occupy. You must keep both these numbers below their corresponding `quota`. If you exceed either `quota` you will be given seven days in which to bring your total below it again. If you exceed the `limit` figure you will immediately be prevented from using any more space.

7. Running Programs

Programs run on Unix will operate either in the **foreground** or the **background**. Programs run in the foreground are connected to the terminal session, and will occupy that session until the program terminates. Programs run in the background are independent of the terminal session; whilst the program is running you can execute other Unix commands and even log out from Unix leaving the program to complete in its own time.

7.1 Running Programs

To run a program in the foreground type the program name and press the **Enter** or **Return** key. To run `ansys` in the foreground you would type:

```
ansys
```

To run a program in the background type the command then add the `&` character and press the **Enter** or **Return** key. To run `clock` in the background you would type:

```
clock &
```

If a background program produces output then you must direct this to a file

```
myprog >& outputfile &
```

This command runs the program `myprog` in the background. Output (and errors `>&`) are directed to the file `outputfile`.

7.2 Controlling Programs

For a program running in the foreground of the current session:

Ctrl C will completely kill the program.

Ctrl Z will stop the program, but the program will still exist in the system.

To control background programs in the current session the following commands are available:

<code>jobs</code>	List programs (jobs).
<code>bg %job_id</code>	Places a job in the background.
<code>fg %job_id</code>	Returns a job to the foreground.
<code>stop %job_id</code>	Stops a background job.
<code>kill %jobs_id</code>	Kill a job.

Where `jobs_id` represents the number or a job returned by the `jobs` command.

Example

If you had the program `time.sh > out` running you could list it by typing:

```
jobs
```

This would produce the response

```
[1] + Running                time.sh > out
```

To stop this job (number 1) you would type:

```
stop %1
```

and get the response

```
[1] + Stopped (signal)      time.sh > out
```

To run this job in the background:

```
bg %1
```

```
[1] + time.sh > out &
```

Then to kill this job:

```
kill %1
```

```
Terminated
```

Program Control Using the ps Command

To check all programs running under your username type

```
ps -f -u username
```

Substituting your username for *username*. This would produce:

```
UID      PID    PPID  C    STIME TTY      TIME CMD
username 24816 24585 0 16:36:04 pts/50  0:00 sleep 2
username 20169 19956 0   Jan 05 pts/50  0:01 -csh
username 24585 20169 0 16:35:07 pts/50  0:00 /bin/sh time.sh
```

You could then use the PID (Process ID) of any program to kill it

```
kill PID
```

If this does not work try

```
kill -KILL PID
```

But always try `kill` on its own first to allow the process to terminate cleanly.

Program Control Using the top Command

Top is not part of the operating system, but is a useful utility. Type

```
top -Uusername
```

to list your programs, then use any of the following commands

```
k          kill program (process)          k PID, k -9 PID
h          help
q          quit top
```

7.3 Running Fortran Programs

To create a Fortran program on a UNIX machine, enter a text editor such as `nedit`, described in the document “Graphical Unix via Exceed”. Give the program name, which must have the extension `.f`, as follows:

```
nedit prog.f
```

Type in your Fortran program; starting, as usual, in column seven.

When the program is complete save the file using **Ctrl+O** then leave the editor using **Ctrl+X**

You can then compile and link your Fortran program using the command:

```
f95 -o prog prog.f
```

Or if your Fortran source code is in several files, say `prog1.f`, `prog2.f` and `prog3.f` then you should type :

```
f95 -o prog prog1.f prog2.f prog3.f
```

If your program uses a library such as the NAG Library then you should use the compiler option `-lnag` as follows

```
f95 -o prog prog.f -lnag
```

Whichever way you compile and link your program you will produce an executable file whos name is specified by `prog`. To run your program simply type

```
prog
```

and press **Enter**.

For an introduction to the fortran compiler `f95` type

```
man fortran
```

Or you can get a full description of `f95` by typing

```
man f95
```

8. Creating and Editing Files: The vi Editor

Many editors are available for different versions of UNIX. However the one editor available with all versions of UNIX is `vi`. This is extremely powerful but has no built-in help. In more recent releases of UNIX, the `vi` manual page explains in detail how to use `vi`.

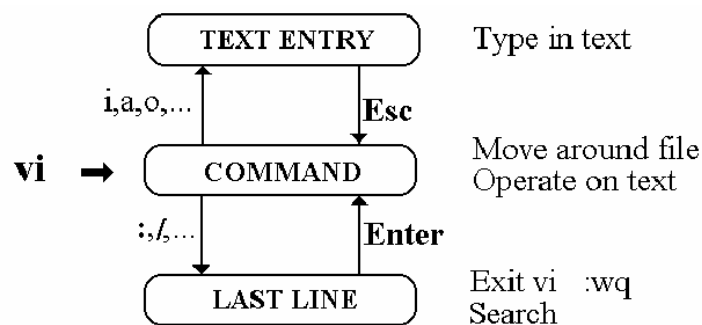
Although `vi` is explained in the following sections, you might prefer to use the graphical text editors described in the document “Graphical Unix via Exceed”.

8.1 The Three Modes

The editor has three modes:

Command mode:	Navigate around text using arrow keys Issue commands that operate on text Can NOT type text in
Text entry mode:	Type text into the file Use Backspace on current line only Can NOT use arrow keys Use Esc key to return to command mode
Last line mode:	Issue commands that operate on files Perform complex searches

You move between the three modes using various keystrokes detailed in section 7.3. The most common ones are displayed on the following map of the `vi` editor.



If you are not sure what state you are in (some keys can take you into text entry mode without you realising it), type `Esc` to ensure you are in command mode. If you press `Esc` while you are already in command mode, the system beeps and the screen will flash, but no harm will be done.

The editor `vi` has a history that may help explain some of these idiosyncrasies. It is an extended version of `ex` (itself an extended version of `ed`) which is a line editor. The `ex` line editor commands are available in the last line mode.

8.2 Simple Editing

To create a new file you must invoke the `vi` editor by typing

```
vi filename
```

where *filename* is the name to be given to your new file; it must not already exist. The screen will clear, leaving a column of tildes (~) on the left-hand side and displaying the name of your file at the bottom.

Command mode

You are now in command mode. To type text into your file you must go into text entry mode. There are many, many ways to go into text entry mode, the usual way when you are creating a new file is to type the character `i` (without pressing **Enter**). The words `INSERT MODE` will appear in the bottom right corner, text is displayed here whenever you are in text entry mode.

Text Entry mode

Once in text entry mode you can type in your text. Use the **Backspace** key to correct mistakes on the current line and start new lines by pressing the **Enter** key. If you spot mistakes on previous lines you will have to leave them for later. Once the body of your text has been typed in you should return to command mode by pressing the **Esc** key.

Command mode

Now you can correct any mistakes. Use the arrow keys to position the cursor by the offending text and issue the appropriate command to make amends. Possible commands are listed in the next section. Correcting mistakes may require you to go into text entry mode again, once you have corrected all mistakes ensure you are back in command mode by pressing the **Esc** key.

Command mode

Having entered your text and corrected all mistakes your file is now complete. You can save your file and exit the editor. To save your file you must enter last line mode. To do this, in this situation, type in a colon (:) character without pressing **Enter**.

Last Line mode

The colon will appear on the bottom line of the screen and you can type in any last line command, useful ones are listed in the next section. In this case we want to save the file and exit the editor which we do by typing

```
wq
```

after the colon, then pressing the **Enter** key. The file will be saved and you will exit `vi`, returning the shell command prompt.

8.3 Command Summary

Cursor movement:

arrow keys	these keys work as expected
Enter	move to the start of the next line
\$	move to end of current line
Ctrl b	move back one screen
Ctrl f	move forward one screen
Ctrl d	down half a screen
Ctrl u	up half a screen
w	move forward one word
b	move backward one word
1G	go to top of file
G	go to bottom of file
221G	go to line 221

Going into text entry mode:

a	<i>append</i> text after the cursor
A	go to end of current line and <i>append</i> text
i	<i>insert</i> text at the cursor
I	<i>insert</i> text at the start of current line
r	<i>overwrite</i> single character at cursor
R	<i>overwrite</i> text at the cursor
o	<i>insert</i> text as new line(s) below cursor
O	<i>insert</i> text as new line(s) above cursor

Deleting/undeleting text:

x	delete single character at cursor
dd	delete line
D	delete line to right of cursor
p	put deleted text below current line
P	put deleted text above current line

Other useful commands:

J	join this line with the line below
u	undo previous command
Ctrl L	redraw screen

Last line commands:

:wq	write the file and quit
:q!	quit and discard changes
:r <i>filename</i>	read (insert) contents of file after cursor
:w <i>filename</i>	write (save) to file
/ <i>string</i>	search for <i>string</i>

9. Sending a File to the Printer

To print one, or more, files on the printer use the command

```
lpr file1 file2 ...
```

The files will be output on the line printer at the IT Centre, listed one after the other with a blank line between them.

For a list of the printers available to you, type:

```
man lpr
```

Press the **Spacebar** to scroll down the listing to the section headed **OPTIONS**, and look at the subsection headed

```
At Sheffield, the following printers are available
```

Bear in mind that, although it is very simple to send a file to the laser printers, these printers will only print files that are already in Postscript format.

To print text files on a the postscript laser printer at the IT Centre you would type

```
mp filename | lpr -Pitps
```

The `mp` command translates text to postscript, which is piped to the `lpr` command.

10. T-Shell Features

10.1 Filename Completion

CICS have implemented "filename Completion" on all their Unix machines. If you are typing a filename in a command, for example

```
more dataset1
```

you need only type enough of the filename to specify it uniquely, then press **Tab**. For example, you could type

```
more da
```

then press **Tab**.

If you have only the one file whose name starts with the letters "da", the name will be completed for you, and all you have to do is press **Enter** to execute the command.

If you have several files with names beginning "da", then when you press **Tab**, your filename will be completed up to the point where ambiguity starts, and you will get a bleep. For example if you have files called data1, data2 and data3, then when you type

```
more da
```

and press **Tab**, your command will become

```
more data
```

and you will get a bleep. Now you can press **Ctrl** and **d** to get a list of possible filenames, with your partially completed command displayed again. Type in as much more as you need to specify the file uniquely, then press **Tab** again to complete it.

This may sound complicated, but do try it out. You will find it a very useful facility, and extremely easy to use.

If you are using a departmental Unix host (i.e. one that is not managed by CICS), you may find this facility has not been set. To make it work for the current session, type

```
set filec
```

If you wish to make filename completion available for every time that you use your departmental Unix host then you should add the above command to your `.cshrc` file in your home directory.

It is worth pointing out that although the **Tab** key is used for filename completion in the T-shell, if you were to work on a machine running the C-shell you would use the **Esc** key for filename completion.

10.2 Repeating Previous Commands

The system can keep a history of the commands you type and is able to repeat previous commands. In the T-shell you can scroll through your previous commands using the up and down arrow keys. To execute a displayed command press Enter.

In addition, the `history` command will list your previous commands:

```
history

1 ls
2 ls -a
3 ls -la
4 more UNIX.intro
5 cp empty_file copyfile
6 ls *file
7 mv copyfile myfile
8 rm myfile
9 pwd
```

To re-issue a command type `!n`, where *n* is the number of the command from the history list.

```
!4
```

Alternatively you could type

```
!string
```

to execute the most recent command starting with the specified *string*. For example if your most recent history is

```
75 vi test.f
76 f77 test.f
77 a.out
```

you can re-issue the `vi test.f` command by typing

```
!75
```

or

```
!v
```

To re-issue your previous command, you need only type

```
!!
```

The `history` command, like filename completion, is a feature of the T-shell that has to be enabled. For CICS UNIX machines these features are enabled; for departmental UNIX machines add the following lines to your `.cshrc` file.

```
set history=40
set savehistory=40
```

10.3 Command Line Editing

If you mistype a command, the t-shell allows you to edit the current command line before pressing **Enter**.

Use the right and left arrow keys to move to the correct position and type in the correct characters. You can delete incorrect characters using the **Backspace** or **Del** key.

At any time you can jump to the beginning of the command line using **Ctrl b** and you can jump to the end of the command line using **Ctrl e**

As well as editing the current command line, you can edit any command that you have displayed from history. This can be used to process a series of files, each time using the same command from history, but editing the filename.

10.4 Shell/Environment Variables

Shell or environment, variables are used as abbreviations for frequently used strings such as path names. The convention for naming a shell variable is

- Start with a letter (upper or lower case)
- Remaining characters can be letter, digits and the underscore "_"

To define a shell variable use the `setenv` command as follows:

```
setenv variable string
```

so to use the word "docs" to refer to a specific directory in the scratch area you might type:

```
setenv docs /scratch/cslibo/shef
```

To use a variable precede it with the dollar sign \$, for example

```
cd $docs
```

If the variable is to be followed by a letter or digit then curl braces are required when using the variable, so the command

```
cd ${docs}field
```

would be equivalent to

```
cd /scratch/cslibo/sheffield
```

Shell variables can be created dynamically at the shell prompt but they only exist for the duration of the shell. To create a permanent variable, the variable definition must be added to your `.cshrc` file in your home directory.

10.5 Command Aliasing

If you frequently type a long command or a long sequence of piped commands, you can invent an alias, and then use that instead. The alias definition is written

```
alias name 'command'
```

where the command must be enclosed in single or double quotes. For example, if you type

```
alias hm 'history | more'
```

you then only have to type `hm` to get your `history` listing piped into `more`.

To cancel an alias definition, type

```
unalias name
```

For aliases to take effect for every session they must be placed in the users `.cshrc` file.

To get a list of all current aliases, including all aliases created by Corporate Information & Computing Services, type

```
alias
```

Within the T-shell, aliases can include previously defined shell variables.

11. Appendix: Summary of UNIX Commands

The following is a list of the UNIX commands included in this document. The numbers on the right indicate the page on which they are described.

<i>alias name command</i>	Define a command shortcut.	25
<i>bg option</i>	Run a program in the background	15
<i>cd options</i>	Change to new directory.	13
<i>cp file1 file2</i>	Make a copy of an existing file.	10
<i>f95</i>	Run the Fortran compiler.	17
<i>fg option</i>	Run a program in the foreground	15
<i>grep string file</i>	Search a file for specified text string.	10
<i>history</i>	List previous commands used.	23
<i>jobs</i>	List all jobs (programs) running	15
<i>kill option</i>	Kill a specific program	15
<i>logout</i>	Log off the current UNIX host.	6
<i>lpr options file1</i>	Print the specified files on the printer.	21
<i>ls options</i>	Display contents of current directory.	9
<i>man options command</i>	Display help manual for command.	12
<i>mkdir directory</i>	Create a new directory.	14
<i>more file1 ...</i>	Display specified files on screen.	9
<i>mp files</i>	Converts text files to postscript.	21
<i>mv oldfile newfile</i>	Rename (move) files.	10
<i>passwd</i>	Change the login password.	7
<i>ps -options</i>	List all jobs (programs) running	15
<i>pwd</i>	Show position in directory structure.	13
<i>quota -v</i>	Display filestore details.	14
<i>rm file1 file2...</i>	Delete the specified files.	10
<i>rmdir directory</i>	Remove (delete) directory.	14
<i>set</i>	Add new T shell features.	23
<i>setenv</i>	Define an environment variable	24
<i>stop option</i>	Stop a specific program	15
<i>top -option</i>	List all jobs (programs) running	16
<i>unalias</i>	Remove a command shortcut.	25
<i>vi file1</i>	Edit files in vi editor.	18
<i>wc file1</i>	Count lines in file.	11
<i>who</i>	List logged-in users.	11