Acknowledgements:
This handout was prepared by Dr. Paul Alexander, and has been updated by the previous lecturer, Dr Peter Haynes of the TCM group.
1. Logging into PWF Linux and the Linux Desktop

This document explains how to use Linux (a widely available free version of Unix) on the Physics Public Workstation Facility (PWF). The local term refers loosely to a Personal Computer together with its connection to a network. At first mention in this document, any technical term is written in italics.

You may be able to skip some aspects of the guide if you are familiar with either Linux or some other version of Unix. If you have never used Linux or Unix before then you should aim to work through to at least section 3.5 before moving onto the next self-study guide. The material in sections 3.6 and later can be returned to as needed later on.

We will often use the terms Linux and Unix interchangeably, but you may assume that whenever we say Unix then that statement refers to all versions of Unix, including Linux. It is assumed that you are have a PWF account. If you have forgotten your password on the PWF then it can be reset by the classes technician; please speak to him or your head of class.

1.1 The Workstation

- Sit down at a PC workstation. The main components are a screen (or monitor), a computer, a keyboard and a mouse.

- Certain keys on the keyboard are engraved with cryptic symbols. Note particularly:
  1. The SHIFT-keys each of which is engraved with an open up-arrow, \( ⇧ \).
  2. The TAB-key which is engraved with a pair of arrows, \( ⇌ \).
  3. The ENTER-key is engraved with a bent left-arrow, \( ← \).
  4. The BACKSPACE-key which is engraved with a straight left-arrow, \( ← \).

- The ENTER-key is normally pressed at the end of each line of input. It must not be confused with the Enter-key which is at the extreme right-hand end of the keyboard. This latter Enter-key will never be used in this course.

- The BACKSPACE-key is used for correcting mistakes; it cancels the character most recently keyed in.

1.2 Logging in to PWF Linux

Each Physics PWF computer can either run the Windows Operating System (OS) or the Linux OS.

- Everyone has a different personal identifier (CRSid) and password and this is the same whether you are using Windows or Linux on the PWF.
• If the computer is not switched on switch it on at the wall sockets and on the workstation and screen as well if necessary.

• If the computer is currently running Windows then you need to logout and restart the computer; select “Shut Down…” from the Start menu, then “Shut down and Restart” and when prompted select “OK”.

• When the computer starts (or restarts if previously running Windows) you will be offered a “OS Loader” menu which looks something like:

```
PWF Microsoft Windows
PWF Linux
```

You have 30 seconds to select the operating system you wish to run using the cursor keys (↑, ↓). Select PWF Linux.

• After a short while another menu appears. PWF Linux is now the default and will be automatically selected after ten seconds or when you press ENTER.

• After about two minutes a message window will appear asking you to accept the conditions of use of this machine – click on “OK”.

• Finally you are presented with a Login screen in which you should enter your CRSid and password. It is very important to realise that Unix is case sensitive and therefore the CRSid’s spqr1 and SPQR1 are different.
1.3 The Linux Desktop

Once logged on the Linux Desktop will appear as in the following figure:

Under Unix the program which provides the Graphical User Interface (GUI) that you are now using is called “X”, and therefore the windows which appear on the Desktop are often referred to as X-windows. The Linux Desktop is in many ways similar to the Microsoft Windows desktop, but there are some differences. In particular under Linux there are many different window managers – PWF Linux runs GNOME.

At the top of the screen is a menu bar:

- launch an application
- start a WWW browser
- date and time

At the bottom of the screen you see a task bar similar to the task bar under Windows:

- list of running applications
- the virtual desktop

The Applications menu (top left) is much like the Start menu under Windows and gives a menu of applications to run.
1.4 Launching a terminal window and other applications

The X terminal or xterm application is particularly important and may be launched from the Applications menu:

- When you select the xterm item by the icon a new window appears.
- The xterm provides a way of giving commands – unlike Windows we shall interact with Linux mainly by means of commands typed into an xterm. However, before we do this we shall first see how to move and manage multiple windows on the desktop.
1.5 Manipulating windows

Each window has a title bar with a set of buttons as shown below:

- Try minimising the `xterm` window by clicking on the left button. To get the window back you should click on the `xterm` icon in the list of current applications (bottom left) or you can click on the window list icon in the top right of the desktop and select from a list.

- Try maximising the window by clicking on the middle button; the same button will return the `xterm` to its original size.

- Now dismiss the `xterm` by clicking on the close button; you can launch another `xterm` from the Applications menu.

Windows can be moved by clicking with the left button on the title bar, then while holding down the left mouse button the window will move with the mouse pointer. Windows can overlap each other: to bring a window to the front so that it is completely visible (raise the window) simply click on the title bar.

- Try moving and raising the windows you have; you may like to create a second `xterm` window.
When you are using the mouse it is clear which window you are interacting with – input from the terminal is directed to the window which has “focus” – this is the window which has a blue title bar rather than a white one.

- Move the mouse pointer away from the `xterm` and click on the empty desktop – the title bar becomes white and the text cursor is a hollow rectangle. Text cannot be typed; try doing so.

- Now move the mouse pointer into the `xterm` window and click. The text cursor changes to a solid black box and the window is highlighted by a blue border and text may now be typed. Any text that is typed will appear at the text cursor and not at the mouse pointer.

Changing the size of the window is done by moving the mouse pointer over any part edge or corner of the window; note how the mouse pointer changes shape. Once positioned over the edge of the window click and hold down the left mouse button, and now while still holding down the left mouse button “drag out” the new size of the window. Some windows are of fixed size and cannot be changed in this way.

- Experiment by changing the size of the `xterm` window.

### 1.6 Two Unix commands

Unix is a very powerful system, but most of the power is available only via commands typed into the system. The `xterm` provides a **Command Line Interface** (CLI) to Unix. Just as with desktops so Unix supports many different CLI’s, some general purpose, some specialised; the Unix term for a program which provides you with a CLI is called a **shell**. The shell you will use is called `bash` (you will only need to know this if you try to look up information about the shell in the system documentation).

We will introduce this idea here and return to consider in more detail this important aspect in the next section.

The shell is ready to accept commands when the “prompt” is displayed. This will be something like

```
spqr1@pcphy117:~>
```

The prompt consists of your user name, the name of the workstation you are using, followed by the current **directory** (more about this later, but a directory is equivalent to a **folder** in Windows), and is terminated by a greater than sign.
• Make sure an **xterm** has the focus, then key in the following *Unix command* and press ENTER:

```
  date
```

- Unix is case sensitive, so it is important to type *date* and not **DATE** or *Date*.
- Any typing mistake which is noticed immediately can be corrected using BACKSPACE. It is almost always necessary to press ENTER after keying in a command. In the present case, the date and time are given.

• Try another command:

```
  cal
```

- This displays this month’s calendar.

### 1.7 Finishing

To exit the desktop, select “Log Out” from the Desktop menu:

• You will be asked whether you are sure you want to log out. If so, click “OK”.
• It is extremely important that you exit of “log off” in this fashion when you have finished so that other people are not able to gain access to your account.
• The Desktop will now be closed down and you will be returned to the graphical screen we saw earlier with the message to agree to the rules for using the PWF.
• To shut down the computer click on “OK” and then select “Actions” and “Shut Down”. Then click “OK”.

8
2 Introducing Unix commands

Only two Unix commands were described in Section 1. Here in Section 2 some of the numerous facilities provided by Unix are introduced. In particular we will see how to start applications and how to manage files.

2.1 Log in to PWF Linux

- As discussed in Section 1, log into PWF Linux exactly as before.
- Launch an xterm window.

2.2 The date command again

- After making sure that the xterm window is in focus, give the date command:
  
  ```
  date
  ```

  There should be no surprises unless you failed to focus on the window or failed to press ENTER.

2.3 Creating a new window from the command line

We have already seen how we can have multiple windows on the desktop; here we shall see how we can start new windows from the command line.

- Being very careful to note and include the & give the following command:

  ```
  xterm &
  ```

  A new xterm window appears which may partially obscure the existing windows. You could have launched the xterm application from the Applications menu, but you have achieved the same effect from the command line.

  - In the window in which you gave the command a number is printed (it is called the process ID), and the prompt reappears.
  - This new xterm has a command prompt (as does the original) and either can be used for command input.

2.4 Removing the xterm window

- The xterm window can be closed and removed from the screen by clicking on the close icon in the top right hand corner of the window.

- Alternatively we can issue a command to do the same thing; we can give the exit command within the xterm window which is a command to tell the shell to exit. This causes the command interface and hence the particular window to close.
2.5 More about the & - background and foreground tasks

- It is instructive to see what happens if you forget the & when giving the command “xterm &” to create an xterm window. Focus on the xterm window and deliberately omit the & when giving the command this time:
  ```
  xterm
  ```
  > A new xterm window is created but no process ID or prompt appears in the original xterm window.

- Focus on the new xterm window and give the date command:
  ```
  date
  ```
  There should be no problems. Now focus on the xterm window and give the date command again:
  ```
  date
  ```
  > This time nothing happens. When & was used to create a new xterm window, the system arranged for the window to be run as a background task. This means one can still focus on the xterm window and give Unix commands even though the new xterm window continues to run. Without the &, the xterm window is a foreground task: no useful work can be carried out in the xterm window without first destroying the xterm window.

- Now remove the new xterm window by one of the methods discussed in Section 2.4.

2.6 Files and directories

Any computer must store information; under Unix any information (text, data, programs or applications) is stored in a file. A Unix file is much the same as a “document” is under Windows. Files are organised in directories (again much like Windows folders) as discussed below. The programs you write and any data they need or results they generate will be stored as files. You can construct file names so that you can remember the contents easily; they usually contain only letters (a-z, A-Z), digits (0-9), underscore(_), period(.) and dash(-). It is very important to remember again that Unix is case sensitive so files with names such as

```
myfile  and  myFile
```

are different. We shall refer to the collection of all your files as your file space.

It is safe practice always to start your file name with an alphabetical character, certainly never with a ‘-‘. File names often have two components, separated by a ‘.’, the second component or extension indicating some property of the file and the first being at the disposal of the user. When you see file names containing ‘/’ these are separating the names of directories or subdirectories (described in the next section), such as the file /home/spqrl/myprog.f90.
Some commands expect file names with a particular second component after a ‘.’. For instance Fortran files should be terminated with .f90 or .f95. The extensions .o and .a are also used in the context of Fortran and should therefore be avoided unless for that specific purpose. We will meet these ideas in more detail later.

A special type of file is one whose name starts with a ‘.’; these “dot” or hidden files are used for special purposes, for example to hold preferences for the desktop.

Note also that your file space is physically limited, and you have limited allocation, so you should keep the number and size of files to a reasonable minimum, and delete them when you have finished with them.

Some of the more useful file handling commands are:

- `ls` list your files
- `mv xxx yyy` renames (or moves) the file `xxx` as `yyy`. This command can be used to not only rename files, but also move them around the file system.
- `cp xxx yyy` copies the file `xxx` to `yyy`.
- `rm xxx` removes the file `xxx` (deletes it); you may list several files separated by spaces. Note that once you delete a file it is gone forever.
- `more` To look at the contents of a text file (e.g. a Fortran program). A screen full of text is displayed at a time and you can display the next screen by pressing the SPACEBAR. Pressing `q` will quit this command.
- `quota` causes the current status of the user’s files pace to be displayed

- Let’s investigate these commands. We start by taking a copy of your login options file. Enter the following
  
  `cp $PHYTEACH/part_2/file.txt a.txt`

  This has made a copy of the original file called `a.txt`, we use a ‘.txt’ extension to suggest the type of data in the file (pure text).

- Now let’s rename the file to `b.txt`
  
  `mv a.txt b.txt`

- We can check the contents of the file using `more`
  
  `more b.txt`

- Finally let’s delete the file
  
  `rm b.txt`

Some degree of file protection is afforded by use of the `chmod` command. This can be used to change the group of users who can read your files, modify your files or modify the way in which the files are accessed. For instance to prevent anyone other than yourself as the file owner reading, writing or executing the file `xxx`, type

  `chmod go-rwx xxx`
2.7 Directories

Every file resides in a directory: the user spqr1 will have a default directory of 
/home/spqr1. This is where the files we have so far examined exist. The file 
b.txt can then be referred to as b.txt or as /home/spqr1/b.txt. The user 
can create subdirectories to organise data programs etc. in a logical way.

- Start by re-creating the file b.txt
  
```
  cp $PHYTEACH/part_2/file.txt b.txt
  ```

- To create a directory we use the `mkdir` command. Create a temporary directory 
called test in you home file space:
  
```
  mkdir test
  ```

- Now copy a file into this directory:
  
```
  cp b.txt test/a.txt
  ```
  
  - This has made a copy of the original file called a.txt, but this time in the 
directory test. Now use the `ls` command to list the files in the directory 
test.
    
```
  ls test
  ```
  
  - We can use the `mv` command not only to rename files, but also to move them 
around, try the following:
    
```
  mv test/a.txt .
  ```
  
  - The “.” is a shorthand for the current directory. Use `ls` to list the files in the 
current directory (your home directory) and test and see the results.
    
```
  ls test
  ```
  
```
  ls
  ```
  
  - Finally move the file a.txt back into the test directory
    
```
  mv a.txt test
  ```
  
  - Notice we just used the name of the target directory into which to move the 
file. We could have got the same result by typing
    
```
  mv a.txt test/a.txt
  ```

- We will often want to use files from a given directory without specifying the 
directory name. Unix has the concept of a *current working directory* and we can 
refer to files in this directory without specifying the directory name. We can 
change the current directory using the `cd` command.
  
```
  cd test
  ```
  
  - Now use the `ls` command without a directory name and see what happens
    
```
  ls
  ```
  
  - You have listed the files in the directory test. Changing directory is very 
similar to opening a folder on Windows. To see which directory you are 
currently in use the `pwd` command:
    
```
  pwd
  ```

12
- It is important to realise that each application (including an xterm window) has its own current directory. Start a new xterm
  xterm &

  ➢ Now change directory by typing the following in the new xterm
    cd ..

  ➢ The “..” is shorthand for the parent directory. Use pwd to see the current
directory
    pwd

  ➢ Now return to the original xterm window and type pwd here, note the
difference.
    pwd

Let’s recap these commands and a few others that will be useful:

mkdir mmmakes a new subdirectory mmm.
rmmdir mmmdeletes a subdirectory (which must be empty) mmm.
cd ddd makes ddd the current directory (change directory to ddd).
cd without an argument takes you to your home directory.
ls ddd lists the contents of the directory ddd. If ddd is omitted the
current directory is assumed.
pwd show current (print working) directory

A useful feature of Unix is the ‘wild-card’ use of “*”. This enables us to use ls to
list files whose names contain a certain character or characters, e.g.
ls ddd/*.f90 lists all files in directory ddd which terminate with ‘.f90’

Some options to ls are particularly useful
ls -a lists all files including hidden files
ls -l gives a more detailed an complete description of each file in
the current directory

- Enter the following command to get full details of the b.txt file:
  ls -l b.txt
  ➢ Note that it is “ls minus ell” not “ls minus one”
  ➢ This results in output like:
    -rw-r--r-- 1 spqr1 pwfread 21 2005-10-01 14:26 b.txt
    Which includes date and time the file was last modified and its size:
    21 bytes.

2.8 Local line editing

There are some very useful facilities when entering commands on the command line
which make life much easier.

- Enter the following command to display the contents of the file b.txt
  more b.txt
• Now see what happens if you press the ↑ key. Instead of the text cursor moving up a line, the result is that the most recent command is reproduced:
  more b.txt
  - The text cursor hovers at the end of the line. Use BACKSPACE to delete just the txt and then key in dat as a substitute. Now press ENTER, thereby giving the command:
  more b.dat
  - When you press ENTER this new command is executed (you will get an error message). It is easy to make a more substantial change to the most recent command. First, press the ↑ key twice, resulting in:
  more b.txt
  - Now use the ← key to position the text cursor just after the ‘b’ and use BACKSPACE (these are two different keys) to delete ‘b’ and then key in test/a the command line should now read
  more test/a.txt
  - Using the cursor keys and BACKSPACE to make changes to the line being typed is sometimes called *local line editing*.

• By using the ↑ and ↓ keys multiple times we can move recover commands we have used in the past and each of these commands can be edited as we have just seen.

### 2.9 Getting help and information online

• There are a number of commands which provide information. The most universal is:
  man ccc

  which will provide information on the command ccc. Most output from the man command will occupy more than one screen: to proceed to the next screenful you press the SPACEBAR, or a q to stop the output entirely. Try the following
  man ls
  - Get used to using the SPACEBAR and q option. This is very much like using the more command which we have already met.

• A related command is “apropos uuu” which returns a list of Unix commands which relate to uuu – again this list may be excessively long or unhelpfully short, but can be used to find the names of forgotten commands. Try the following:
  apropos f95
  - You will notice that not only commands, but also other information is returned. In this case it should be something like
    f95 (1)   - Fortran 95 compiler …
    fpp (1)   - the Fortran language …
    nag_modules (3)  - Overview of …
  - You will also notice that associated with each entry is a number, this is the section of the “manual” in which the information is stored. This is useful in one particularly useful application, xman.
• The xman application provides a graphical interface to the manual pages. Try the following and explore the application
  xman &

• Most Unix commands have a series of options (usually prefixed by a ‘-’) which can be invoked. These are many, and change with the flavour of Unix being used. Their order is usually significant, so that indicated by man should be followed. In this manual we will merely indicate some of the useful options on PWF Linux. More details are given in the manual pages.

• Of course there is a great deal of help information available on the WWW: much provided locally and directly relevant to Linux, Fortran and this course. The WWW browser available on Linux is firefox and this can be started either by clicking the icon just to the right of the Desktop menu at the top of the screen, or by typing the command
  firefox &

The computing service home page and the home page for this course are good places to find links to relevant information and documentation:

  www.cam.ac.uk/cs/
  www.tcm.phy.cam.ac.uk/teaching/compphys/
3 Editing

In this section we introduce the *emacs* editor. You will no doubt have met the idea of editing text on a computer before, most likely in the context of using a word processor such as Microsoft Word. A text editor like *emacs* is in some ways similar, but it is designed to edit plain text files such as the ones we will use to hold a program and perhaps data or results. There are many other editors available under Linux, but *emacs* is very much a standard in the Unix world and therefore worth learning.

All operations in *emacs* can be performed by giving commands; a subset of operations are available via the *emacs* menu. We will describe the commands here, you can explore the menu options for yourself. The *emacs* window looks like the following:
3.1 Setting up a text file – emacs

- The next task is to set up a Unix file whose file name is jobletter798 and whose contents are to be a rather silly letter. For this we will use emacs. Key in the command:
  emacs &
  ➢ The main part of the emacs window contains a welcome message.

- Click anywhere in the main part of the window and the welcome message is replaced by some instructions for creating a new file. The cryptic C-x C-f stands for Ctrl-x followed by Ctrl-f.
  ➢ emacs is a command driven editor and the keystrokes Ctrl-x Ctrl-f refer to the emacs find-file command. The x refers to an extended list of emacs commands and the f stands for find-file. The idea is to attempt to find a file called jobletter798 and hope to fail! [If you succeed in finding the file it would be an old file but you want to create a new file. You would therefore want to choose a different name.]
  ➢ Near the bottom of the emacs window is a highlighted mode line.
  ➢ The mode line begins --:-- *scratch* which implies that you haven’t yet specified a file name. (There may be a ‘u’ rather than ‘-‘ before the colon).
  ➢ The word All simply indicates that the entire text fits in the window.

- With the emacs window still in focus, key in Ctrl-x and notice that after a couple of seconds C-x- appears in the echo line, the line below the mode line. This is to remind you that you have keyed in Ctrl-x.

- It is useful to know what to do if you key in Ctrl-x (or any other emacs command) accidentally. Simply key in Ctrl-g. Try that now. The C-x- disappears. Now key in:
  Ctrl-x Ctrl-f
  ➢ A message which begins Find file appears in the echo line. The cursor jumps to the end of that line where you should key in:
  jobletter798
  ➢ Press ENTER. The *scratch* in the mode line changes to jobletter798 and the echo line confirms that this is a New file so the file name has not already been used. The main part of the window clears and the cursor jumps to the top left-hand corner ready for you to key in the contents of this new file.

We could have achieved the same by selecting the File menu and then Open File. You will probably want to use the menu options as you learn how to use emacs, however just as with the rest of Unix the power of emacs is only really available when you use the commands together with the menu options.
Type in the following, including the two spelling mistakes, and press ENTER at the end of every line:

Dear sir,

I am sorry I did not turn up for the job interview last week, but there was a good horror film on the television and I did not want to miss it.

I am willing to come for a new interview on Wednesday afternoon next week, though I must leave by 4:30 to go to the football game.

I am an expert in Unix, particularly the game playing programs, so I would be a great asset to your company.

yours sincerely,

Immediately you begin typing, the --:-- at the beginning of the mode line changes to --:** where the asterisks indicate that the text has changed.

One imagines that this is a job application letter written by someone who wants to work as a Unix programmer. There are two deliberate mistakes and you may have made some others yourself!

As usual, any typing mistake which is noticed immediately can be corrected by using BACKSPACE. To correct a typing mistake made on a previous line, use the text cursor keys (those marked with ↑, ←, ↓ and →) to move the text cursor to just after the position of the error and delete from there before retyping.

Try some experiments with the text cursor keys. See what happens when the ← or → key is used to move the text cursor off the left- or right-hand end of a line. Notice that if one of these keys (or indeed any key) is held down for more than a fraction of a second its effect is repeated, a feature known as auto-repeat. Double-check the text after finishing the experiments. Leave in the two deliberate mistakes.

The next step is to save this letter as the contents of a file jobletter798 (this applicant has had numerous previous unsuccessful attempts).

Saving requires another extended emacs command, the save-buffer command. The associated keystrokes are Ctrl-x Ctrl-s so key in:

Ctrl-x Ctrl-s

The echo line displays a message with begins Wrote and which ends with the file name jobletter798 to indicate that the file has been saved. The --:** in the mode line changes back to --:-- to signify that the displayed text is the same as that in the file.

Again you could have achieved the same result by selecting Save from the File menu.
3.2 Making use of multi-tasking

Unix is what is called a *multi-tasking* operating system; that means that more than one thing can be going on at once. Make use of this by returning the focus to the xterm window and using `ls` to list your files (which should now include `jobletter798`).

- Give focus to the *xterm* window (you may need to raise it to see it all) and type:
  ```
  ls
  ```

- You can now look at the contents of your new file using the `more` command
  ```
  more jobletter798
  ```

3.3 Leaving *emacs*

- Raise the *emacs* window prior to leaving *emacs*.

- Leaving *emacs* requires the `save-buffers-kill-emacs` command. The associated keystrokes are `Ctrl-x Ctrl-c` and it is probably easiest just to think of the `c` as meaning cancel *emacs*. Key in:
  ```
  Ctrl-x Ctrl-c
  ```

- A message *(No files need saving)* appears momentarily in the echo line and the window disappears. If you *had* forgotten the `&` earlier and results of the `ls` command would appear now.

3.4 The *spell* command

A separate command is used under Unix to look for spelling errors in a file, it is called *spell*.

- Focus on the *xterm* window and, *carefully noting the < character*, give the command:
  ```
  spell <jobletter798
  ```

- The *spell* command gives a list of spelling mistakes in the file heralded by `< which may be pronounced ‘from’. The list should include *expurt* but not *grate*. The `< character is important as it says that the input to the *spell* command should come from the named file; without it *spell* would expect you to type your text into the *xterm* window terminated by `Ctrl-d`.}

19
3.5 Correcting the mistakes

- The obvious way to correct mistakes is to use emacs. Enter emacs again (and don’t forget the &):
  ```
  emacs &
  ```

- To open an existing file, use the emacs find-file command exactly as before. The difference is that you are not proposing to set up a new file this time so you want to find the file.
  - Key in Ctrl-x Ctrl-f again.
  - The message which begins Find file appears in the echo line and the cursor jumps to the end of that line where you should again key in:
    ```
    jobletter798
    ```
  - Press ENTER. The letter reappears in the main part of the emacs window and the *scratch* again changes to jobletter798 but there is no mention of New file in the echo line.

- Other things to notice are that the cursor jumps to the beginning of the letter and there are no asterisks in the mode line. Also Lisp Interaction changes to Fundamental; emacs has decided that the file contains ordinary text.

- Correct the spelling mistakes. The asterisks reappear. You should next save the revised version in exactly the way you saved the earlier version by using the emacs save-buffer command, so key in:
  ```
  Ctrl-x Ctrl-s
  ```
  - The echo line confirms that emacs duly wrote the text to the file.

- Leaving emacs before giving the spell command and then immediately re-entering emacs was unnecessary! The reason for doing so here was to show how to open an existing file on re-entry into emacs.

3.6 Some emacs commands

In all emacs documentation the abbreviation C- is used for Ctrl- and this is what is used in this summary of the emacs commands that have been mentioned so far:

- **C-g** keyboard-quit Go away, I didn’t mean it
- **C-x c-f** find-file Find (open) an existing file
- **C-x C-s** save-buffer Save an existing file
- **C-x C-c** save-buffers-kill-emacs Cancel (leave) emacs

When used in a simple way, emacs can be menu driven. With the exception of C-g, the commands listed in the summary can be found as entries in the File menu. Against each entry, the associated keystrokes are shown. Whether to use menus or keystrokes is largely a matter of taste but note that by no means all emacs commands can be found as entries in menus.
3.7 Further use of *emacs*

- *emacs* has many hundreds of features and it is worth trying out just a few more in this introductory document. First make some copies of *jobletter798*:
  
  ```
  cp jobletter798 jobletter800
  cp jobletter798 jb.txt
  ```

- Start up *emacs* if it is not already running. Begin with:
  
  ```
  C-x C-f
  ```
  
  This invokes the *find-file* command. The cursor jumps to the end of the echo line where you could key in *jobletter798* but it is unnecessary to type this in full. *Without pressing ENTER* just key in the first letter:
  
  ```
  j
  ```
  
  Now press the SPACEBAR. The window splits and all files which begin with *j* are listed in the bottom half; these are the possible *completions*. As well as *jb.txt* there are our file names which begin *jobletter*. Now just key in the single letter *o* and press the SPACEBAR again.
  
  - *emacs* now knows enough to complete as far as *jobletter* but it still doesn’t know exactly what you want. You have to key in:
    
    ```
    798
    ```
    
  - Press ENTER and the file’s contents duly appear.

- Click the first *y* of *yours sincerely*. The text cursor jumps to the position of the mouse pointer. This is an alternative to using the cursor keys.

- If you press BACKSPACE you will, of course, delete the character to the left of the *y*, probably a space. To delete the *y* without first moving the cursor, key in:
  
  ```
  C-d
  ```

  This is a *delete-char* command and deletes the character the cursor is on rather than the one to its left. Now key in an upper-case *Y* instead:

  ```
  Y
  ```

  - In a similar way change the *s* of *sir* to an upper-case *S* (or make it lower case if it already is upper case!).

When you edit a file and save the edited version *emacs* creates a backup of the original file before the edits – this file has the same name as before but has a ‘~’ appended to the name.
3.8 The emacs undo command

- A mistake can be undone using the emacs undo command. emacs documentation gives C-_ for this but notice that as well as the usual use of Ctrl you will need to use SHIFT for the underscore character. That said, key in C-_ four times very slowly to see the effect:

  C-_ C-_ C-_ C-_  

- Not only does this negate the earlier changes but, assuming those were the only changes, the two asterisks in the mode line disappear to indicate that the text has not been changed since the file was opened.

- You can also use the Undo option under the Edit menu.

- If you have not already done so, now would be a good time to investigate the various functions provided by the emacs menus.

3.9 emacs: deleting lines, cutting and pasting

- Click the D of Dear and key in:

  C-k

  This is the kill-line command. The Dear sir, goes blank. To get rid of this blank line, give another:

  C-k

  Killed text goes to the kill-buffer and can be brought back. Try:

  C-y

  This is the yank command. Anything in the kill-buffer is yanked back.

- Kill and yank can be used as cut and paste in a word processor. Click the I at the beginning of the final paragraph and then give the kill-line command five times:

  C-k C-k C-k C-k C-k

  The first four kill the two lines of the paragraph and the last kills the blank line before yours sincerely, Next click the I at the beginning of what has just become the final paragraph and then yank back the killed text:

  C-y
### Appendix: emacs commands

- In the following summary, the emacs commands which have been mentioned in this document are shown in what is likely to be, very roughly, decreasing order of frequency of use. Since you are advised not to leave emacs, C-x C-c is last in the list.

- The three columns show key bindings (where available), the full formal names of the commands and an informal description:

<table>
<thead>
<tr>
<th>Key Binding</th>
<th>Formal Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>C-x C-f</td>
<td>find-file</td>
<td>Find (open) an existing file</td>
</tr>
<tr>
<td>C-x C-s</td>
<td>save-buffer</td>
<td>Save an existing file</td>
</tr>
<tr>
<td>C-x C-w</td>
<td>write-file</td>
<td>Write a new file</td>
</tr>
<tr>
<td>C-_</td>
<td>undo</td>
<td>Undo a command</td>
</tr>
<tr>
<td>C-g</td>
<td>keyboard-quit</td>
<td>Go away, I didn’t mean it</td>
</tr>
<tr>
<td>C-d</td>
<td>delete-char</td>
<td>Delete a character</td>
</tr>
<tr>
<td>C-k</td>
<td>kill-line</td>
<td>Delete from the cursor rightwards</td>
</tr>
<tr>
<td>C-y</td>
<td>yank</td>
<td>Bring back killed text</td>
</tr>
<tr>
<td>M-q</td>
<td>fill-paragraph</td>
<td>Tidy up a paragraph</td>
</tr>
<tr>
<td>M-x</td>
<td>execute-extended-command</td>
<td>Give a command the long way</td>
</tr>
<tr>
<td>C-x C-b</td>
<td>list-buffers</td>
<td>List the buffers</td>
</tr>
<tr>
<td>C-x l</td>
<td>delete-other-windows</td>
<td>Get rid of other emacs windows</td>
</tr>
<tr>
<td>C-x b</td>
<td>switch-to-buffer</td>
<td>Change to another buffer</td>
</tr>
<tr>
<td>C-x C-c</td>
<td>save-buffers-kill-emacs</td>
<td>Cancel (leave) emacs</td>
</tr>
</tbody>
</table>

- M- stands for the META key (as C- stands for Ctrl) which is equivalent to holding down the left Alt key. A command like M-x may also be entered in emacs by pressing Esc followed by x.
4. Further use of Unix and editing with X

In this section we shall look at some further aspects of Unix and editing within X windows.

4.1 Stopping a running program

Occasionally you may need to stop a running program or Unix command; there are a number of possible ways of doing so, depending on the context. We shall often refer in this context to “killing” a running program.

In the window in which the command is running you can interrupt a Unix command or your own program by typing \texttt{CTRL-c}\ (this means press and hold down the key marked “Ctrl” (the control key) and press the letter “c” on the keyboard).

4.2 Sending output to a file

- From the \texttt{xterm} window, press ENTER and then give the following version of command which produces a calendar for the specified year
  \begin{verbatim}
  cal 1900 >past
  \end{verbatim}
  The calendar for 1900 goes to the file \texttt{past}. The > character (‘to’) redirects what would otherwise appear on the screen to a file (just as < takes input from a file rather than the keyboard).
  This can then be read with the more command
  \begin{verbatim}
  more past
  \end{verbatim}
  Only as many lines appear as will fit and there is a note of what percentage of the text this is. Press ENTER repeatedly to advance through the text one line at a time. To advance through the text one window full at a time, repeatedly press the SPACEBAR.

4.3 Pipes

- A pipe is a sequence of commands which are strung together so that the output of one command becomes the input of the next. The commands are separated by vertical bars (obtained by pressing Shift-\). An example of a pipe is:
  \begin{verbatim}
  cal 1900 | more
  \end{verbatim}
  The output from the command to the left of the | becomes the input of the command to the right of the |. This example of a pipe is quicker than typing \begin{verbatim}
  cal 1900 >past followed by more past
  \end{verbatim} which not only involves keying in two lines but introduces the file \texttt{past} which isn’t needed when a pipe is used.
  The more command again causes the display to pause after a window full but notice that no percentage is quoted because more doesn’t know how much more there is to come. As before, press the SPACEBAR to see the rest of the text.
4.4 Highlighting and copying text

- Suppose you wish to set up a file containing just the last three months of 1900. Clearly you could edit the file past but it is quicker to use the highlighting feature of X.

- Start up emacs if it is not already running and open a new file.

- In the xterm window give the cal command again:
  
  \texttt{cal 1900}

- Move the mouse pointer to the beginning of the line on which \texttt{October} is written. Press the \textit{left-hand} button. Then, move the mouse pointer to the beginning of the final blank line and press the \textit{right-hand} button.
  - The use of the left- and right-hand buttons in this way highlights a \textit{region} of text. Highlighted regions of text can be copied to the place marked by the text cursor and just now the text cursor is at the bottom of the window.
  - Now shift focus to the \texttt{emacs} window.
  - To copy the highlighted text, simply press the \textit{middle} button. The copy appears in the \texttt{emacs} window.
  - To turn the highlighting off, click the mouse anywhere in the \texttt{xterm} window.
  - Note that highlighted regions do not have to be large chunks of text. Single sentences or even single words can be highlighted and copied.

Summary

- To highlight a region of text:
  - Press the left mouse button at the beginning of the region
  - Press the right mouse button at the end of the region

- To copy a highlighted region of text to the position marked by the text cursor:
  - Press the middle mouse button

4.5 Printing

Printing to both CS printers and the printers in the Physics PWF is charged for. Only two types of file should be printed: plain text files or PostScript files. For obtaining credit on the Physics PWF printers see the notices in the Physics PWF room.

- Before attempting to print a file you should be sure it is a text file or a PostScript file. You can do this by using the \texttt{file} command. If you type
  \begin{verbatim}
  file name
  \end{verbatim}
  - after which, if the file is a standard text file, you will see a response such as
    \begin{verbatim}
    name: ASCII text
    \end{verbatim}
  - examples of files you should not try to print are executable files (programs) such as those produced after compiling a Fortran program (see later)
• To print use the \texttt{lpr} command:
  \begin{verbatim}
  lpr  -P\textit{name} \textit{file.f}
  \end{verbatim}

• This will print a copy of \textit{file.f} at the printer called \textit{name}. Printers are available in some colleges and centrally. Their names would replace \textit{name} above. If you leave out the \texttt{-Pname} part then the default printer is used. For example to print a plain text file (example.txt) on the standard Physics PWF printer just type:
  \begin{verbatim}
  lpr example.txt
  \end{verbatim}

• To print a colour PostScript file to the colour laser printer in the Physics PWF use:
  \begin{verbatim}
  lpr -P\textit{phy_bragg_clj} colour_fig.ps
  \end{verbatim}

• Finally other commands exist to format plain text files before printing; you may well wish to use the command \texttt{a2ps} to convert a plain text file into a PostScript file before printing. It will be nicely formatted if, say, it contains a Fortran program and two pages of plain text will be printed per sheet thus saving you money!