Introduction to Linux Tutorial

The aim of this tutorial is to familiarise you with the Linux operating system. It is very basic and is aimed at people who have no experience using Linux at all. The bulk of this tutorial is based upon modules 1, 2 and partly 3 at http://www.linuxsurvival.com . After you have finished going through this document, you might want to visit their site.

Structure of Tutorial

Linux Tutorial

The Linux and Windows Directory Trees
Using "ls"
Using "more" to view the contents of files
Organising files and directories - using "more", "mv", "pwd", and "cd"
Using Pathnames
Using "cp" to copy files
Using "rmdir"
Security
Changing the Security Permissions – "chmod"
Time Savers
Wildcards
Auto-complete
Accessing previously typed commands
Home Directories or Top Level or ~username
Using the Printer

Making PDF's – Acrobat Documents

Compiling and Executing C Programs Under Linux

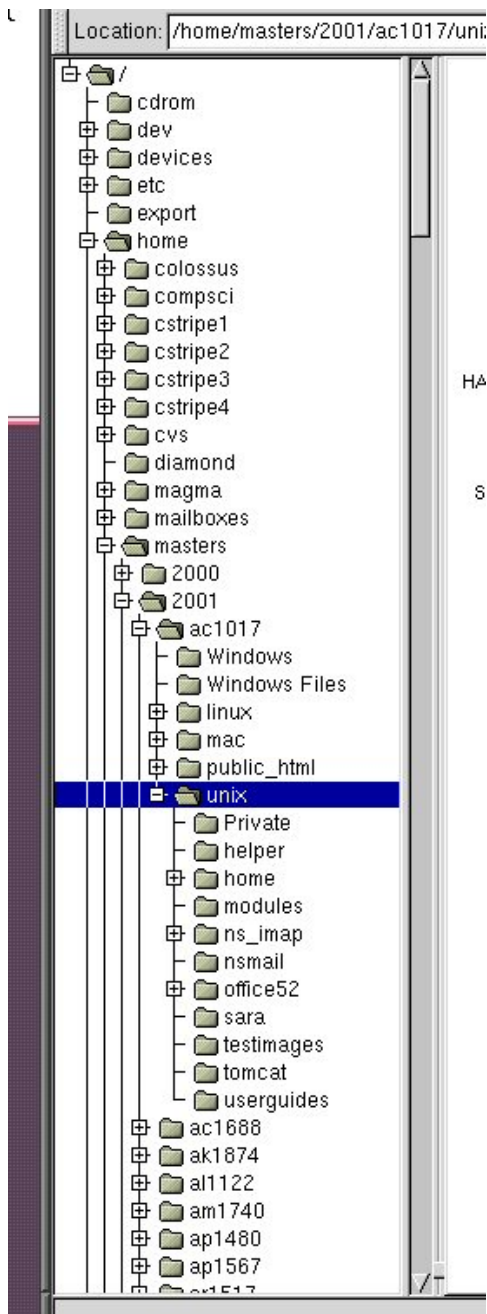## Other Useful Linux Programs and Tips

Note that hannuna and sh1670 are used interchangeably in this tutorial as my username was recently changed from hannuna to sh1670.
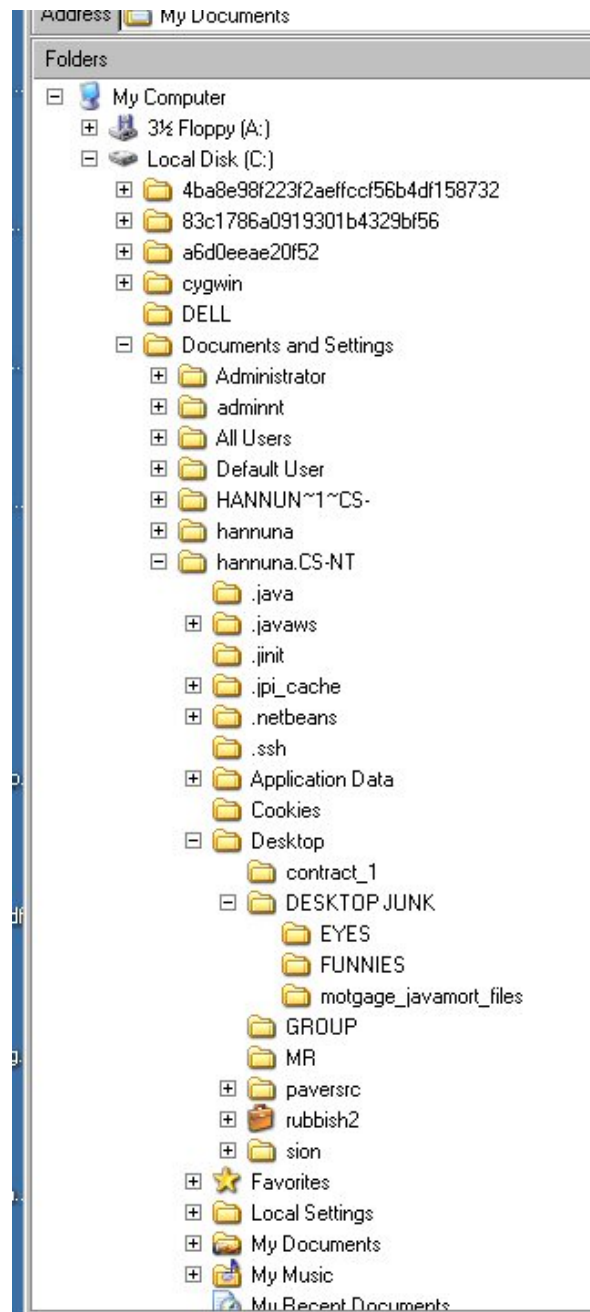
The Linux and Windows Directory Trees

Before we explore the commands used to manipulate the Linux environment, we should take a quick look at the structure of the environment itself.

Microsoft Windows users will find the Linux file system to be a familiar structure because it is basically the same. You can think of a Linux file system as a tree. See the diagrams below.

Linux Directory Tree

Location: /home/masters/2001/ac1017/uni

```
/
  cdrom
  dev
  devices
  etc
  export
  home
    colossus
    compsci
    cstripe1
    cstripe2
    cstripe3
    cstripe4
    cvs
    diamond
    magma
    mailboxes
    masters
      2000
      2001
        ac1017
          Windows
          Windows Files
          linux
          mac
          public_html
          unix
            Private
            helper
            home
            modules
            ns_imap
            nsmail
            office52
            sara
            testimages
            tomcat
            userguides
      ac1688
      ak1874
      al1122
      am1740
      ap1480
      ap1567
```

Windows Directory Tree

Address My Documents

Folders

```
My Computer
  3½ Floppy (A:)
  Local Disk (C:)
    4ba8e98f223f2aeffccf56b4df158732
    83c1786a0919301b4329bf56
    a6d0eeae20f52
    cygwin
    DELL
    Documents and Settings
      Administrator
      adminnt
      All Users
      Default User
      HANNUN~1~CS-
      hannuna
      hannuna.CS-NT
        .java
        .javaws
        .jinit
        .jpi_cache
        .netbeans
        .ssh
        Application Data
        Cookies
        Desktop
          contract_1
          DESKTOP JUNK
            EYES
            FUNNIES
            motgage_javamort_files
          GROUP
          MR
          paversrc
          rubbish2
          sion
        Favorites
        Local Settings
        My Documents
        My Music
        My Recent Documents
```
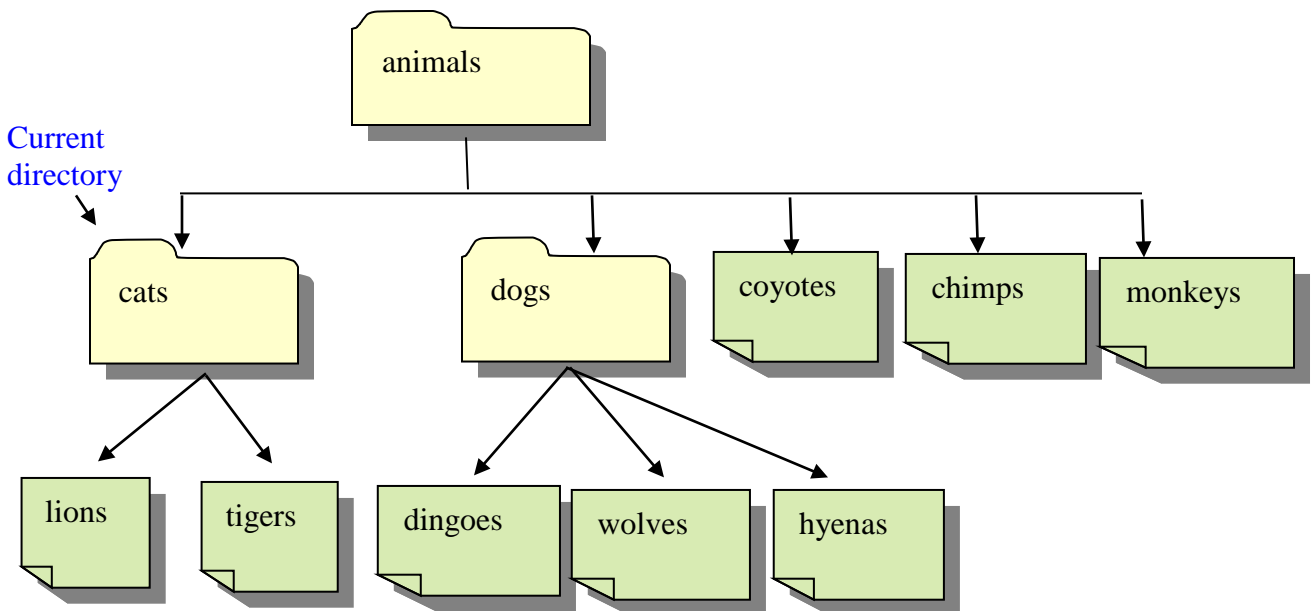
A Simpler Directory Structure

The previous structure was very complex. In order to practice on a simpler structure, open a terminal (right click on desktop and select new terminal) and then type the following:

```
cd ~
cp /home/cosc/sh1670/SHARE/make_tut ./
make_tut
cd animals/cats
```

This should add the following structure to your "top level" and place you in the "cats" folder:

Using "ls"

Probably the most often used command in Linux is the "ls" command. It is used to list the contents of a directory.

For example, your current directory is "cats". To see what it contained, you would type "ls". The output should be the following:

> lions   tigers

Unlike many other operating systems, Linux is case-sensitive. In other words, if you type "LS" instead of "ls", Linux will not recognize the command. This applies to filenames, like "lions" and "tigers", as well.

Using "more" to view the contents of files

Before we start moving the files around, let's take a look in one of them to see what it contains. We'll look at the lions file to make sure it has the proper information in it.

The "more" command is used to view the contents of a file. For example, to see the contents of the "lions" file, you would type

```
more lions
```

It is called "more" because after it has displayed a page of text, it pauses and puts "-- More --" at the bottom of the screen to let you know that there is more text yet to be shown. To see the next page of text, you just hit the spacebar.

Now type the command to view the contents of the "tigers" file.

Organising files and directories - using "more", "mv", "pwd", and "cd"

Okay, it's time to start organizing our animal files. **First, we'll create a directory under "animals" called "primates"**. The command to create a directory is "mkdir" which is short for "make directory".

For example, to make a directory named domestic_breeds, you would type

```
mkdir domestic_breeds
```

This will create a directory called domestic_breeds, within the current directory.

First we need to get into animals directory (there are shortcuts). To change directories, use the "cd" command, which stands for -- you guessed it -- "change directory".

For example, to change to directory "dogs", you would type

```
cd dogs
```

To move up one level, to the "parent directory", you would type

```
cd ..
```

To find out where you are, use the "pwd" command, which stands for "print working directory".

Type it at the command prompt to verify your current location. Then, type the command which lists the contents of the current directory.

Finally use the "mkdir" command to make a directory called primates

```
You work it out
```

Now, let's move all of the primate files into our newly created directory.

To move a file, you just use the "mv" command. For example, to move a file called "wolves" into directory "dogs", you would type

```
mv wolves dogs
```

Renaming files is simply a case of "moving" a file from one name to another. For example, to rename file "wolves" to "mean_dogs ", you would type

```
mv wolves mean_dogs
```

Let's start by moving "chimps" into the "primates" directory. Type the command to move file "chimps" into directory "primates". Do the same for the "monkeys". Finally put the "coyotes" and "kittens" (ahh bless) in the most appropriate folder.

To Summarize:

ls - list current directory contents
more - show the contents of a file (text file)
mv - move or rename file or directory
cd - change directory
pwd - present directory
mkdir - make a new directory

Using Pathnames

Now we have to take a time-out to explain "pathnames".

So far we have only been manipulating files which are in our current directory. But sometimes you might want to manipulate files which are not in your current directory. For example, you may be doing a lot of work in the "dogs" directory, but you remember that you wanted to rename "tigers" to "siberians". You could accomplish this by using these commands:

```
cd ..
cd cats
mv tigers siberians
cd ..
cd dogs
(continue working in "dogs")
```

That's an awful lot of work just to perform a simple task. Fortunately, there's a way to accomplish this task with only one command instead of five.

You can refer to a filename by its "pathname". This means that you specify what "path" to follow down the tree to find the file. The path to "tigers" is represented like this:

```
~/animals/cats/tigers
```

We built this pathname by starting at the top of the tree (your top level – represented by '~') and adding "animals" to it. Then we added a "slash" ('/') every time we moved down the tree another level.

We can use this pathname to accomplish our task with only one command:

```
mv ~/animals/cats/tigers ~/animals/cats/siberians
```

Typing in a full pathname can be tedious, though, especially if you are a long way down the tree. That's why it is often easier to specify a "relative pathname". It is called "relative" because it specifies a filename relative to your current location, rather than from the root directory. If you don't put '~/' or '/' at the beginning of a pathname, then Linux knows that you are using a relative pathname. For example, if you were in the "animals" directory and you wanted to rename "tigers" to "siberians", you could do so with this command:

```
mv cats/tigers cats/siberians
```

In this case, you specified your pathnames relative to the "animals" directory (your current directory), so you didn't need to include "animals" in them.

So far we haven't shown you how to use relative pathnames from the "dogs" directory, which was our current directory in the original example. There is a way to do it by using something we learned a few pages ago.

```
mv ../cats/tigers ../cats/siberians
```

Recall that ".." refers to the directory above your current directory.

Now try the same using the full pathname from the root: `/home/msc18/` you do the rest

Using "cp" to copy files

All right, enough theory -- it's time for practice. We need to move "hyenas" from the "dogs" directory to the "animals" directory, but out of paranoia, we're going to copy it and then remove the original file after we're sure that it was copied correctly. The copy command is "cp" and it has the same syntax as the "mv" command. For example:

```
cp apples apples2
```

We're not in the "dogs" directory, so you'll have to use a relative pathname in your copy command. Type the command to copy "hyenas" from the "dogs" directory to the "animals" directory. Note: Use './' to represent current directory.

Check the copy of "hyenas" which is in the "animals" directory using "more". If everything is okay, we should now remove the original "hyenas" file in the "dogs" directory. The remove command is "rm". For example, to remove file "platypus", you would type

```
rm platypus
```

Type the command to remove the "hyenas" file in the "dogs" directory.

Using "rmdir"

The "remove directory" command is "rmdir". For example, to remove directory "fish", you would type

```
rmdir fish
```

Now use `mkdir` and this command to create and remove a new directory in your unix folder.

Security

Now it's time to talk about security. Many PC users have little experience with file security because Windows does not have any. It doesn't need security because it is a single-user operating system. Linux is a multi-user operating system, and your files are stored on a server, so it has security to prevent people from accessing each other's confidential files. Unfortunately, the security system in Linux is not very flexible, but we'll talk about that later.

Say, we don't want anyone to modify the dog files except for a particular group. It will take quite a bit of explanation before we can show you how to arrange this sort of security.

When you execute an "ls" command, you are not given any information about the security of the files, because by default "ls" only lists the names of files. You can get more information by using an "option" with the "ls" command. All options start with a '-'. For example, to execute "ls" with the "long listing" option, you would type

        ls -l

When you do so, each file will be listed on a separate line in long format.

There are lots of other options you can use with the ls command, but we won't need them to accomplish our current goals.

Now lets try the using the 'ls' command combined with a smart pathname to list the contents of the **'dogs'** directory':

        ls -l ~/animals/dogs

type   security   owner   group   size (bytes)   last modified

There's a lot of information in those lines. The first character will almost always be either a '-', which means it's a file, or a 'd', which means it's a directory. In our example, all three of them are files.

The next nine characters (rw-r--r--) show the security; we'll talk about them on the next page.

Security Continued

Deciphering the security characters will take a bit more work.

First, you must think of those nine characters as three sets of three characters (see below). Each of the three "rwx" characters refers to a different operation you can perform on the file.

```
-rw-r--r--    1 hannuna group
```

    user   group other   Now sh1670
  (owner)   (world)

The 'r' means you can "read" the file's contents.
The 'w' means you can "write", or modify, the file's contents.
The 'x' means you can "execute" the file. This permission is given only if the file is a program.
If any of the "rwx" characters is replaced by a '-', then that permission has been revoked. For example, the owner's permissions for our three dog files are "rw-". This means that the owner of the file ("hannuna (now sh1670)", i.e. me) can "read" it (look at its contents) and "write" it (modify its contents). You cannot execute it because it is not a program; it is a text file.

Members of the group "mkgroup" can only read the files ("r--"). We want to change the second permission character from '-' to 'w' so that those people can modify the contents of these files as well.

The final three characters show the permissions allowed to anyone who has a UserID on this Linux system. We prefer to refer to this set as "world". Our three dog files are "world-readable", that is, anyone in our Linux world can read their contents, but they cannot modify the contents of the files.

Lazy Security

If this is all a bit daunting at the moment, you can always store your files and folders securely in your ~/Private folder (and its subfolders).

Changing the Security Permissions – "chmod"

The command you use to change the security permissions on files has a horribly cryptic name. It's called "chmod", which stands for "change mode", because the nine security characters are collectively called the security "mode" of the file.

Now it will become clear why we named the three "rwx" sets "user", "group", and "other". The first argument you give to the "chmod" command is 'u', 'g', 'o', or a combination of them which specifies which of the three "rwx" sets you want to modify. For example, if you want to give "execute" permission to the world ("other") for file "coyotes", you would start by typing

```
chmod o
```

Now you would type a '+' to say that you are "adding" a permission.

```
chmod o+
```

Then you would type an 'x' to say that you are adding "execute" permission.

```
chmod o+x
```

Finally, specify which file you are changing.

```
 chmod o+x coyotes
```

You can also change multiple permissions at once. For example, if you want to take all permissions away from everyone, you would type

```
chmod ugo-rwx coyotes
```

Okay, time to get back into action. Type the command to give "write" permission to the "world" (other), for the file "lions"

Then type the command to give a long listing of the files so you can check your handywork (try and use pathnames to achieve this).

Time Savers

Wildcards

A wildcard allows you to specify more than one file at the same time. The '*' matches any number of characters. For example, if you want to execute a command on all files in the current directory, you would specify '*' as the filename. If you want to be more selective and match only files which end in "ing", you would use "*ing". Note that the '*' can even match zero characters, so "*ing" would match "ing" as well as "sing".

The other wildcard, '?', is not used very often, but it can be useful. It matches exactly one character. For example, if you want to match "sport", but not "spat", you would use "sp??t". The first '?' matches the 'a' in "spat", but the second '?' can't match anything, so "spat" fails.

So to list all the html files in my public_html directory you would type:

```
ls /home/cosc/sh1670/public_html/*.html
```

A word of warning: It's not a good idea to use wildcards when deleting files. Linux stores previously typed commands. The can be accessed by pressing the up arrow. A common way of losing a lot of work is to type 'rm *.*' in one folder and then to input the same command in another folder by mistake.

Auto-complete

Pressing the 'TAB' button will execute Linux's auto-complete functionality.

Accessing previously typed commands

Pressing the 'up' and 'down' arrows at the command prompt will bring up previously typed commands. These are stored in the '.bash_history' file in your 'top level'. You can see this file by typing 'ls –las'.

Home Directories or Top Level or ~username

In a multi-user system, it is convenient for each user to have his own private place to store files. Linux calls this private place your "home directory". The location of home directories varies greatly between systems.

When you login to your Linux system, you are automatically placed in your home directory. So, if you want to display its pathname, simply login, then type "pwd". In our system, the output would be "/home/msc18/your_username".

Because most people need to refer to their home directories on a regular basis, Linux makes them easy to specify. You can use the tilde everywhere that you would normally use "/home/ msc18/your_username".

For example, if you wanted to copy a file called "jokes" from your home directory to the "/tmp" directory, you could just type

```
cp ~/jokes/tmp
```

rather than

```
cp /home/keeper/jokes/tmp
```

You can even use the tilde to specify another user's home directory. If '~' is immediately followed by a User ID, then it no longer refers to your own home directory; it refer's to User ID's home directory. For example, if you wanted to go into hannuna (now sh1670)'s home directory, you could just type

```
cd ~sh1670
```

rather than

```
cd /home/cosc/sh1670/linux
```

In fact, '~' by itself is just a short form of "~your_username".

Remember that if you put anything after '~' other than '/', Linux will assume that you are referring to another user's home directory. So, in our "cp ~/jokes /tmp" example above, if we had typed

```
cp ~jokes/tmp
```

instead, we would have received an error message because there is no User ID called "jokes".

One of the nicest things about using the tilde is that you don't have to know where anyone's home directory is located. Sometimes system administrators will move home directories to new locations to alleviate disk space problems. If you always use the tilde, you might not even notice such moves.

Using the Printer – Be patient this changes year to year !!

The easiest way to print stuff is from the GUI of an application; just as you would in Windows. However you can also print stuff from the command line.

Here are the basic Linux print commands:

    lpr   send to printer
    lpq   display print queue
    lprm   remove from print queue

The "lpr" command stands for "line printer", because when the command was developed, it was assumed that you would be using that type of printer. This, of course, is no longer the case for the majority of Linux users. Now, "lpr" can be used to send jobs to PostScript printers, plotters, and any other print devices you can think of.

For example, to send a file named "thoughts" to the default printer, you would type

```
lpr thoughts
```

To send your file to a printer called "cs211" rather than the default one, you would type

```
lpr -P cs211 thoughts
```

The '-P' stands for "printer". The printer can have any name that your system administrator chooses, even something like "tower-of-babel", if she feels like it.

Note that "lpr" does not produce any output if the print request was successful.

Checking the Print Queue

It is common for you to want to check on the status of your print job, in case there is a problem with the printer or there are a large numbers of print jobs ahead of your own.

If you type "lpq" without any arguments, you will be shown the status of all print jobs in the default print queue.

To display all of the print jobs in a different queue, use the '-P' option, just as you would for the "lpr" command. For example, to check on your print job in the " cs211" queue, you would type

```
          lpq -P cs211
```

Possible output for such a printer is shown on the next page.
Style of Output for 'lpq' command:

```
hp14 is ready and printing
Rank    Owner       Job  Files        Total Size
active keeper       154  thoughts     685 bytes
```

(yes I know we asked for the queue on cs211)

Removing Jobs from the Print Queue

The command to remove print jobs is called "lprm", which stands for "line printer remove". For example, to remove print job 148 from print queue " cs211", you would type

```
          lprm -P cs211 148
```

Making PDF's – Acrobat Documents

The majority of reports you will submit during your MSc are required to be in a PDF format (so they can be read by Acrobat Reader).

The Easiest Way

By far the easiest was to do this is to use linux's version of word - the Open Office word processor. This program may be started as follows:

Clicking on the start / applications icon (bottom left / top left)->Office->Word Processor / Oowriter/ or something similar

OR

Type "oowriter &" in a terminal.

Write your document just as you would for Word. To save it as a PDF, go to file->Export as PDF.

If your working at home (assuming you use Windows) there is a nice bit of freeware called PDF995. This is a really easy way to do PDF's on Windows.

Harder Methods

LaTeX - this is the software most people use to write scientific papers.

ps2pdf - If you print to file (assuming the printer you specify has a postscript printer driver), the output will be saved as a postscript (.ps) file. You can then use ps2pdf to convert it into a pdf.

For example to convert the text file 'sion.txt' into a pdf you could type:

```
a2ps sion.txt -o sion.ps
ps2pdf sion.ps sion.pdf
```

All in one line:

```
a2ps sion.txt -o sion.ps && ps2pdf sion.ps sion.pdf
```

a2ps is useful for printing and has lots of options. Type man a2ps to see them all.

Compiling and Executing C Programs Under Linux

Creating your file

Say you want to create a file called my_test.c. At a prompt type :

```
gedit my_test.c &
```

Or if you only have the terminal window (when using putty.exe, say)

```
pico my_test.c
```

If you already have a file called my_test.c in your current directory, gedit will open that file.

You can now type text, which will be inserted at the cursor. The mouse allows you to move the cursor around the screen, whilst the backspace and return keys all do what you would expect.

To save a file i.e. store it in your Linux file-space with a given name, click on the save icon. This saves the file in your top-level, with the name my_test.c

If you edit a file with a .c file suffix, then the editor recognises this as a C program and will switch on features related to the programming language. For example, the editor will calculate the proper indentation for each line. Just hit the tab key once when typing in a line, and the line will be indented the right distance.

What to Type !!

```
#include <stdio.h>

int returnSum(int iNum1, int iNum2);

int main(void)
{
     int i,j,k;

     j = 2; k = 3;

     i = returnSum(j,k);

     printf("The sum of j and k is %d\n", i);

return 0;
}

int returnSum(int iNum1, int iNum2)
{
     int i;
```

```
        i = iNum1 + iNum2;

        return i;
}
```
Compiling and Executing Your Program

To compile your program type:

```
gcc my_test.c -o my_test
```

To execute your program, type the name of the executable at the command prompt:

```
my_test
```

Be careful not to name your executable after a unix command ('test' is the usual culprit)!!

Other Useful Linux Programs and Tips

Vital Stuff When Using Command Prompt

Ctrl+C: This command halts a running process. Use this to quickly exit from any program that you are running.

Ctrl+Z: Sends a current process into the background. Also if your terminal is messed up because you 'cat' a binary file 'ctrl + z' will clear up the screen for you and give you a clean prompt.

Tab: One of the most used keys. Pressing the 'tab' key while typing the path to any directory or a filename is very helpful. Write the first few characters of the file or directory and press the tab key to complete the name or give you a list of possibilities. When pressed during an incomplete command, the 'tab' key completes the command for you.

UP Arrow: In the terminal, it cycles through the list of commands that you have executed.

MiddleMouseButton: Just select the text you would want to copy using your mouse. Do as you would under Windows. Press and hold the right mouse button and then drag to select the text. Then switch to the terminal you want to copy to and click the middle mouse button. This will paste the text at the current cursor location.

Manual: If you want to know more about a particular program – "ls" for example type "man ls" at the command prompt.

Useful Programs

Web Browser: Either type "firefox &" at the command prompt or click on the globe with a mouse attached to it.

Word Processor: Clicking on the CentOS icon (bottom left)->Office->Open Office.org Writer **OR** Type "oowriter &" in a terminal.

Acrobat Viewer: Type "kpdf &" or "acroread &" at the command prompt.

Viewing Postrscript: Type "kghostview &" at the command prompt. Alternatively, convert the file to a PDF using ps2pdf and use "acroread" to view it.

File Explorer – Like Windows explorer: Type "nautilus &" or click on your home directory on the desktop.

Image Viewer: Type "xv &" at the command prompt.

Launch Default Application for File type: gnome-open <your filename> &

Other Useful Stuff (John McGonigle's favourites):

```
useful commands:

[command] && [another command] //run commands one after the other

du -h    //how much space is being used
df -h    //how much on filesystems
rm -rf  //delete directories and file

ls -l -a //show files with permissions and show hidden files
chmod -R 700 [folder/file] //make file private
chmod -R 755 [folder/file] //other people can read it
chmod -R 666 [folder/file] //anyone can do anything

ssh csjmg@bluecrystal.acrc.bris.ac.uk //pass: main password
ssh ice -X //ssh with windows

scp -p -r [path to folder to copy/*] [username]@[other machine]:[path
to dir on other machine]

passwd //change password (usually on server)

rdesktop -a 16 -f atlas -g 1270x945 -u [username]//log into windows,
word 2007 etc.

convert -crop [finalxsize]x[finalysize]+[cutherex]+[cutherey]
Screenshot-*.png cropss*.png
//cuts out chunk of image
convert -delay 10 -loop 0 cropss*.png ani.gif
//make these into an animated gif loop

emacs -nw

matlab -nodisplay

anyprogram | grep [string to find in output]
anyprogram | more  //read the output of a program one screen at a time
cat /proc/meminfo | more //RAM
cat /proc/cpuinfo | more //CPU

cat > [new filename]
[write notes etc here]
[ctrl-d]

anyprogram > [output file]
anyprogram 2> [error output file]
anyprogram >> [add to output file]

pwd // what directory am i in (what's the path)
whoami // what's my user name



//compress stuff
zip stuff.zip *
```