A Quick Guide Around Linux

10th August 2005

Contents

Chapter 1

Getting Started

1.1 Logging In

Logging in to Linux machines is relatively painless - assuming that you have a Linux account. Depending on which machines you are using, there are two possibilities. If you are working on one of the computers on the third or fourth floor of the maths building, you will need a maths account - generally, if you are enrolled in Honours, the computer technicians would have set up an account for you - see them about your user name and password. If you are using one of the public computers, there's two ways of doing things. The first way you can only use if you have a general University Linux account - use your normal Authcate username and password to log in. The other way is to use an ssh client to do a remote login to one of the maths machines - more on this later.

1.2 Opening a Terminal

While most new versions of Linux will come with a Windows Explorer-type interface (eg. Konqurer), there's a lot of stuff which is still most easily done by running basic commands. To do this, you'll need a terminal window. Depending on the version of Linux you're using, this is either as easy as clicking on the computer monitor like button on the panel at the bottom of the screen, or right-clicking somewhere on the desktop and choosing "new terminal".

1.3 Basic Commands

The first (and most basic) command in Linux is `ls` - the list command. This gives you a list of the files and directories in your current location. When you log in for the first time, there's probably nothing in your home directory, so it'll look something like this:

```
duckman:~/>ls
duckman:~/>
```

Next most important is the `cd` command. For those who remember the old MS-DOS operating system, this command does exactly the same thing in Linux. For those who have only used Windows in their computing, this command allows you to move from one directory to another, similar to moving from one folder to another in Windows. Directories in Linux are usually pretty easy to spot - they're either highlighted in blue, or have a forward slash (/) at the end of them. So say we had a subdirectory in our home directory called temp. If we used the `ls` command, we might see something like:

```
duckman:~/>ls
temp/
duckman:~/>
```

To use the `cd` command, we type `cd <directory>`, and if we want to move up a level, we type `cd ..` (note the space). For example, if we want to move into our temp directory and back out, we'd go something like this:

```
duckman:~/>cd temp
duckman:~/temp/>cd ..
duckman:~/>
```

The `ls` command also has a few extra features which can be accessed by typing in letters after the minus (-) key. The main ones to use are `-t`, `-a` and `-l`. The first will show the contents of the directory in order of modification time, `ls -a` will show all the contests of a directory (including hidden files and those starting with a dot), and the last is the long listing - it will show everything about the file; who owns it, its size, the date it was modified and who can access it.

1.3.1 An Aside: The Linux Directory Structure

The way Linux organises its files can be somewhat complicated - its not quite as simple as the drive-and-folder organisation that is in Windows. Instead, it organises its files into specialist directories, each of which has holds files which perform a particular set of tasks. The actual functions of these directories are a bit complex, but the end result is that the default directory that you put your files into isn't the base directory of the computer (ie. `c:`\ in Windows, or `/` in Linux). Instead, they're put into what's known as the *home* directory, which is can be referred to as the `/home/<username>/` directory, or more convieniently, $\tilde{/}$.

1.3.2 Moving and copying

The three commands to move, copy, and delete files in Linux are `mv` and `cp` respectively. Both of the commands have the same format, being:

```
<command>␣<options>␣<source_file>␣<destination>
```

That is, you first type in the action you would like to perform (move or copy), then any options you would like to specify, followed by the file you want to copy/move, then the place you want it to be copied/moved to. For example, if we had a file called `foo.bar` in our current directory, and we wanted to copy it to the `temp` directory. We might type in the following commands:

```
duckman:~/>ls
foo.bar      temp\
duckman:~/>cp foo.bar temp
duckman:~/>ls
foo.bar      temp\
duckman:~/>cd temp
duckman:~/>ls
foo.bar
duckman:~/>
```

Note that using `cp` by itself will only copy individual files - to copy directory, you need to use the option `-R`, ie `cp -R <source directory> <destination directory>`. Other options you might find useful are `-i`, which makes the computer ask for confirmation if it needs to overwrite another file, and `-v`, which makes the computer tell you everything that its doing.

1.3.3 Deleting/Removing Files and Directories

Deleting files and directories is very similar to copying or moving them, although in this case, there's no destination, and the command is `rm`. Thus, if you want to remove the file `foo.bar` from your current directory, you type:

```
duckman:~/>rm foo.bar
```

The most common options for the `rm` command are `-i, -v, -r` and `-f`. The `-i` and `-v` options work in exactly the same way as they do in `cp` and `mv`, while the `-f` option will override the `-i` command. The `-r` option will remove directories. For example, if we had a directory called `foo` and a file called `foo.bar` in our current location, the command `rm foo*` would only remove the the file `foo.bar`. However, if you typed in `rm -r foo*`, both file and directory would be deleted.

1.3.3.1 Wildcards

If you've ever used MS-DOS before, you'd be familiar with the wildcard commands `*` and `?`. Both are available and have the same function in Linux. For the uninitiated, the `*` character is a "don't care" command - anything and everything is included, while the `?` command works for one character only. A few examples:

The command

```
duckman:~/> cp t* temp
```

will copy every file starting with t (lower case only - case is important in Linux) into the temp directory, while

```
duckman:~/> cp tes*.txt temp
```

will only copy the `.txt`files that start with `tes`. The command:

```
duckman:~/> cp tes?.txt temp
```

will copy the `.txt` files that begin with `tes` *and* only have four letters.

Linux also has a nifty tab completion feature - if you type the first few letters of a file, then hit <Tab>, the computer will complete the filename as best as it can, ie, up until a character is different across two files with the same heading.

Chapter 2

Running Programs

Running programs in Linux is pretty straight forward - as long as the file is an executable, then all you need to do is type in the name of the file from the directory it is in. Depending on how your version of Linux is set up, you might need to type in a `./` before the name, just to signify the fact that its in the local directory. Note that this also means that you can run things in another directory by typing in its full path. So say I'm in my `/temp` directory, and I want to run something in my `/programs` directory, I would type:

```
duckman:~/temp>../programs/myprogram
```

The `..` at the beginning tells the computer that I want to move up a directory from the one I'm in, then the rest tells it where to go from there. Note that we can also type `/programs/myprogram`, as the `/` symbolises the home directory.

Chapter 3

Stuck?

If you get stuck, or just want to know what a command does, Linux has the `man` utility. By typing in `man <command>`, Linux will display instructions on what the command actually does, as well as some general information on other options for running it.

Chapter 4

Glossary

- *Home directory* The home directory is the place where you automatically go when you first log in to a computer. Technically, it is in the `/home/<username>` directory of the computer's directory organisation