An Introductory Tutorial on UNIX

Kevin Keay

February 6 2009

Introduction

The purpose of this document is to guide you through the sequence of:

1. Describing a quick method of connecting to a remote UNIX machine from a PC.
2. Providing a basic UNIX tutorial.

Some useful references are:

Anderson, G. and P. Anderson (1986): *The UNIX C Shell Field Guide*. Prentice-Hall.

There is a copy in the Visualisation Lab (VisLab) (Room 348). The UNIX Lab is Room 347. *Please* don't take this away except for photocopying purposes.

Gilly, D (1998): *Unix in a Nutshell*, O'Reilly and Associates.

See: Engineering Library: 005.43 GILL.

There are some useful notes for our network in: *Introduction to the School of Earth Sciences UNIX computer network*. This is available as a PDF file at:

http://www.earthsci.unimelb.edu.au/~kevin/UNIX_Course/Intro_to_SES_UNIX_computer_network.pdf

My lecture is available as a Powerpoint presentation at:

http://www.earthsci.unimelb.edu.au/~kevin/UNIX_Course/Intro_Lecture_Feb-2009.ppt

A useful online resource from the University of Surrey, *UNIX Tutorial for Beginners,* is available at:

http://www.ee.surrey.ac.uk/Teaching/Unix/

The above webpage includes a link to download a zip file of the tutorial to your PC.

A quick method of connecting to a remote UNIX machine from a PC

Note: The following instructions are intended for starting UNIX sessions from the PC Lab.

1. Log on to the PC with your University username and password. Click on the **Cygwin desktop icon**. If there is no desktop icon then from the Start button:
- All Programs – Cygwin – Cygwin Bash Shell; Right-click – Create Shortcut; Yes to place shortcut on desktop. Then click on the new Cygwin icon.
The text window which appears is controlled by the *Bash* shell. We will represent the Bash shell prompt by `$`. For your information you may activate the *C-shell* by typing: `csh`. The C-shell prompt is represented by `%`.
The following steps only require the Bash shell but *should* still work in the C-shell.

Note: You *don't* type $ or % in the following commands.

2. `$ startx`
This activates the Cygwin X server. You should get a lot of messages in the Cygwin Bash window. The X icon will appear in your PC System Tray, then a new window (white background with the X symbol at upper left) will appear. This window is called a `xterm`.
The prompt in a `xterm` windows should start with `$` but may contain other 'junk'. You may use this window or create an additional one with: `xterm &` - click inside the additional window to use it.

Note: This set of windows is actually running on the PC, not on a remote UNIX machine. You need to connect to a UNIX machine to access remote applications.

3. We need to connect to a *remote* UNIX machine. From one of the `xterm` windows:

```
$ ssh -X atlas
```

Note: If your UNIX username is different from your University username then you will have to include the former e.g. `ssh -X kevin@atlas` where the UNIX username is `kevin`.
The `-X` option allows the graphical output of X applications on the remote UNIX machine to be displayed on the PC.
For other UNIX machines you may need the *full* name e.g. for cove,

```
$ ssh -X cove.earthsci.unimelb.edu.au
```

if:

```
$ ssh -X cove
```

does not work.
Note: For 'machine' choose from:
atlas,orthus,cove,gyre,tide,wave,vislab01,vislab02,vislab03

The *first* time that you attempt a connection from the current PC to the remote UNIX machine e.g. atlas, you will get a `ssh` warning message – respond with the full answer: `yes`.
You will then be prompted for your UNIX password (see Doug Morrison to set up this but it is normally done prior to this tutorial). If successful, you are now logged on to the remote machine. It is equivalent to sitting at the console of the UNIX machine, in the next room or on the other side of the planet.
On the remote UNIX machine you will be logged on into a C-shell window (`%` prompt).

Important: Since our network has a mixture of UNIX operating systems e.g. SUN Solaris, PC Red Hat Linux, you need to accommodate these with suitable *initialisation scripts*. Note that this procedure has to be done only *once*: for future sessions the initialisation will be performed *automatically*.

To carry out this initialisation:

```
% cp /home/kevin/UNIX_Course/zunix_setup.zip   .
```

- **note the dot**

```
% unzip zunix_setup.zip
% ls -l .cshrc.*
```

(you will see a set of files named .cshrc*)

Now exit the session and log in again e.g.

```
% exit
```

(this returns you to the Cygwin Bash shell window)

```
$ ssh -X cove.earthsci.unimelb.edu.au
```

You are now logged on into a C-shell window (`%` prompt). You can now do any text or graphics based work that you desire.

4. When finished it is a good idea to logout (exit) from the window(s) connected to atlas (or cove etc.) i.e.

```
% exit
```

5**. Important: Close the X server by right-clicking on the X icon in the System Tray and then Exit. Owing to recent PC operating systems changes this step may not be required.**

6. You may also access a UNIX machine from a PC with SSH Secure Shell Client. There should be a desktop icon for this. If not, access the program via the Start button – All Programs. Note that this will open a *text-based* window only i.e. you can't run `nedit` or other X-Windows programs which produce graphical output. It is possible to run X-Windows in conjunction with X-Win32, a commercial product from: www.starnet.com. There is a student price available but it will work for free for 15 minutes at a time in 'demo' mode. It might be useful for access from a home PC (this is the method that I use) or you could install Cygwin for Windows on your home PC (see: www.cygwin.com). Alternatively, you may log in directly from a *console* of a Linux PC in the UNIX Lab or the Visualisation Lab.

Directly connecting to a remote UNIX machine from a console

Note: The following instructions are intended for starting UNIX sessions from the console of a machine in the UNIX Lab or the VisLab.

From the console of a machine in the UNIX Lab or the VisLab:

1. Log on with your UNIX username and password.

2. You should be in the KDE desktop. Left-click on the Shell-Konsole icon or Terminal Program icon in the toolbar at the bottom of the screen. This will activate a terminal window (`xterm`). You may now enter UNIX commands. Note: If you log on to orthus (Solaris UNIX) you will get a different desktop – simply use one of the available windows. To get an additional window type: `xterm &` from one of the current windows.

A basic UNIX tutorial

The purpose of this tutorial is to provide a *basic* introduction to UNIX. You may access a UNIX machine from a PC running X Windows under Cygwin or an iMac running X Darwin. On a PC you may also use Secure Shell Client. There should be desktop icons for Cygwin and Secure Shell Client – if not, access them via the Start button – All Programs. Alternatively, you may log in directly to a console of a Linux PC in the UNIX Lab.
In this document, `%` represents your prompt and user is your username e.g. `kevin`.

Note: You *don't* type the `%`.

Important: Owing to some recent upgrades to atlas we won't be able to run the tutorial in full on atlas at the present time. Since you will normally use a Linux machine for your work then choose from **cove, gulf, gyre, tide or wave**. Or pick one of the Visualisation Lab machines e.g. vislab01. You may have to give the complete machine name
e.g. `ssh -X cove.earthsci.unimelb.edu.au`

1. Log on to a remote UNIX machine e.g.:
   ```
   % ssh -X cove.earthsci.unimelb.edu.au
   ```

2. Create a folder (also called a directory) in your home directory `/home/user` where user is your username – by default, this is your current location. If you are unsure, go to your *home* directory with
   ```
   % cd ~
   ```
   Note: Regardless of which remote UNIX machine you connect to, your file structure is the *same* e.g. /home/user on atlas contains the same files and folders that you see on cove.
   In this example our new folder is called Tute – note: UNIX is *case-sensitive*.
   ```
   % mkdir Tute
   ```
   Go into this folder.
   ```
   % cd Tute
   ```
   You can check your location with `pwd`.
   ```
   % pwd
   ```
   Your location should be: `/home/user/Tute`

3. Copy the following file from Kevin Keay's account.
   ```
   % cp ~kevin/UNIX_Course/test.f .
   ```
   The dot at the end of the command means that the copied file will have the same name as he original. You can specify your own name if you wish.
   ```
   % cp ~kevin/UNIX_Course/test.f  mytest.f
   ```
   To see a listing of the files in the folder Tute use the `ls` command.

`% ls`	
`% ls -l`	Gives a long listing – more details
`% ls -l *`	All files but not those starting with a dot (dot files)
`% ls -la`	Includes dot files
`% ls -l t*`	Those starting with t
`% ls -l *.f`	Those ending with .f
`% ls -l *.?`	Those with a second-last dot, then a single character

4. The file test.f is a text file. If you use the `file` command, you are told it is ASCII text (normal text).

```
% file test.f
```

The extension .f indicates that this is a *Fortran source file*. This particular program prints a name (some text) and the output of a simple calculation on the screen. To convert this code into an *executable* (program or software) you need to *compile* it with `g77`.

```
% g77 -o test test.f
```

The argument after the –o option is the name of the executable, in this case test. After compilation you can run the program by typing its name.

`% test` or perhaps `./test` if the current folder is not in your path (check the PATH environment variable i.e. % echo $PATH)

You should see the following text printed on the screen.

My name is: Kevin
x= 2.000000 y= 9.200000

The name is 'hardwired' into the program. The value of y = x + 7.2 where x=2.0.

Note: On most Linux machines the Fortran 77 compiler `g77` is equivalent to (or an alias of) `f77`. On Solaris machines `f77` is different from `g77`. For the compilation of both Fortran 77 and Fortran 90 programs use `f90` (Solaris) or `pgf90` (Linux).

5. You can edit the Fortran source code to give a different response. The standard UNIX text editor is called `vi` (visual editor but some say 'virtually impossible'!) However there is an easier text editor called `nedit`, which is similar to Windows programs like notepad. Let us edit the file test.f.

```
% nedit test.f &
```

The ampersand (`&`) allows the current window to be used for other tasks i.e. `nedit` is run in the background. You can change the name Kevin to another one. In addition you could change the value of x or the number added to it. Save the file using the top menu: File – Save. Then redo step 4 and see how the output has changed. You could alter the program so that it prompts the user for a name or number. You will learn how to do this and many other tasks during the Fortran course later in the semester.

It may be instructive to go to `nedit` Preferences and check the box for Statistics Line (this shows the line and column numbers). For Fortran syntax highlighting you can go to Preferences – Language Mode – Fortran. This is useful for spotting typing errors in a Fortran program.

Note: Via Cygwin there is a bug with `nedit`. If you *select* (highlight) text and use *Copy* the `nedit` session will close and you will lose any unsaved changes. This does not occur from a console or X-Win32 (a proprietary software package). An alternative is to use `kedit`. With both `nedit` and `kedit` there are screen diagnostic messages that you can ignore. There are no problems using a console or X-Win32 on a PC.

6. If you list the files in the directory you should see something like this (there *might* be other files too).

```
% ls -l

total  82
```

```
-rwxr-xr-x    1 kevin    wheel       81595 Feb 17 14:34 test*
-rw-r--r--    1 kevin    wheel         206 Feb 17 13:44 test.f
```

The total value of 82 means that the files occupy 82 KB (kilobytes) of disk space. The owner is kevin and the group is called wheel . The group name is not usually important. However the file system is normally set up such that your fellow students and perhaps staff have read access to your files (except mail folders).
The file test is 81595 bytes in size while test.f is much smaller at just 206 bytes. Also test.f was created (or last modified) at 13:44 on Feb 17 of the current year (e.g. 2008). The asterisk (*) next to the filename test indicates it is an executable file i.e. it can be run as software.
The first column in the listing gives the permissions for the files. The first character (-) means it is not a directory (you will see d here for a directory or folder) followed by three sets of three characters. These sets represent the permissions for the user (you!), the group and other (anyone else). The codes are read (r), write (w) and execute (x). The symbol – means no access. They also have the numeric codes 4, 2 and 1 respectively, with 0 denoting no access.
For the text file test.f, the user can read and write to it and all others can just read it. For the executable file test, the user can read, write and execute it and all others can read and execute it.
You can change the file *access permissions* with `chmod`. For instance, to prohibit anyone but you from accessing your Fortran source code file you can use the command:

```
% chmod 600 test.f   or:
% chmod 600 *.f
```

where the user permission is rw- (6= 4 + 2 + 0) and the group and other permissions are both 0 (0 = 0 + 0 + 0). Executables like test (step 4) usually have permission 755.
There is another way of changing permissions. For instance `chmod 755 test` is equivalent to: `chmod u+rwx,g+rx,o+rx test` where `u` is user, `g` is group and o is others.
See: `% man chmod`

7. To move back to the home directory (/home/user) type:

```
% cd ~
```

To see a directory listing of your home directory type:

```
% ls -l
```

or:

```
% ls -la
```

The `-a` option shows the dot files (those starting with a dot or full-stop) e.g. .cshrc
You will also see the directory Tute (with a d as the first permission code):

```
drwxr-xr-x … Tute
```

Depending on whether you use mail programs on a UNIX system you might see a Mail or mail folder.

```
drwx------ … Mail
```

Note that only the user can look at files in this directory (the group and other permissions are both ---).
If you create a folder within a folder then you can use the `./` notation to move around the directory structure or tree. Assume that you have created two folders called examples and code in the folder called Tute.

```
% cd ~          (this only ensures that you are in your home directory)
```

```
% cd Tute
% mkdir examples
% mkdir code
% cd examples
```

To go up one level type:

```
% cd ..
```

A directory listing with ls would show that there are two folders, examples and code, in this current location (Tute). Repeating `cd ..` would take you up one level to your home directory. Assume you are in the folder examples. Then to go to code you would type:

```
% cd ../code
```

You can always type the complete path to a folder if it gets complicated.

```
% cd /home/user/Tute/code
```

Later in the semester you will be assigned a *work disk area*. The work areas have names like /work18/user. Hence you would issue commands like:

```
% cd /work18/user/data.dir
```

The folder data.dir (file and folder names may contain dots) might contain some pressure data files.

8. As a final exercise you can perform some typical UNIX tasks related to your Honours work. Assume that you need to produce a contour map of a pressure data file for June 1 1996 600 UTC. There is an appropriate file from the NCEP Reanalysis in Kevin Keay's area.

```
% cp ~kevin/UNIX_Course/pmsl.ncep.1996060106.cmp .
```

Note: If you are working on a Linux machine e.g. cove, you will need to convert the SUN binary file (big-endian) to a PC binary file (little-endian) using `binswap`:

```
% binswap  -c pmsl.ncep.1996060106.cmp{,}
```

The `{,}` uses the input filename as the output name. To use another output filename:

```
% binswap  -c pmsl.ncep.1996060106.cmp pmsl_new.cmp
```

and use this new filename from this point on.

To see the contents of the file:

```
% mapx -f pmsl.ncep.1996060106.cmp | more
% mapx -f pmsl.ncep.1996060106.cmp >! pmsl.txt
```

In the first form the output of the `mapx` command is piped to the more command so you can view it one page at a time. Without `| more` the output will quickly rush by on the screen. In the second form the output of the `mapx` command is *re-directed* (>) to a file called pmsl.txt. The `!` means to overwrite pmsl.txt if it already exists. You can view this file with:

```
% more pmsl.txt
% nedit pmsl.txt (followed by & to run in the background if you wish)
```

In this 'dump' of the contents of the pressure file you can see information about the latitudes and longitudes as well as the actual values at each (longitude, latitude) grid point.

To produce an image of the pressure data you can use a NCAR Graphics program called `conmap`.

```
% conmap -G  pmsl.ncep.1996060106.cmp
```

If you type:

```
% ls -l g*
```

You will see a file called gmeta. This is a graphics metafile. To view this file:

```
% ctrans gmeta &
```

A graphics window will appear with a pressure map. With the mouse over the window, type q to close the window.

To convert the file to Postscript for printing:

```
% g2ps gmeta
```

List the files starting with g with:

```
% ls -l g*
```

or Postscript files ending with .ps

```
% ls -l *.ps
```

and you will see a file called g.ps

You can view this file with `gv`:

```
% gv g.ps
```

The top menu allows you to reduce or enlarge the image on the screen. With the mouse over the window, type q to close the window.

You may print this file to the printer in the PC Lab:

```
% lpr -Ppclab g.ps
```

You can also use `convert` to translate the Postscript file to, for instance, a PNG file that may be pasted into a Word document:

```
% convert -trim -density 120 g.ps pmsl.png
```

9. This concludes the introduction. Later in the semester you will learn about scripts and more advanced features of UNIX. For now, you should have enough information to use the UNIX network for most common tasks. Please review the material in the main part of the notes notes *Introduction to the School of Earth Sciences UNIX computer network*. In addition, some help on UNIX commands is given by:

   ```
   % man unixcommand
   ```

   e.g.

   ```
   % man nedit
   ```

   Note: Sometimes the `man` pages are unavailable or in some weird system location. It may be easier to use Google to find information about the command.

10. Some useful keyboard tips for the C-shell:
    - *Ctrl-C* is used to stop execution of a command or program within a C-shell window
    - Use *Tab completion* to reduce typing i.e. type the first few characters of a file name and then press the Tab key to see the matches; add more characters as required
    - Use the up-arrow key ↑ to get previous commands; use the left and right arrow keys to move through the command to edit (the Insert and Backspace keys may help too)