

Computational Astrophysics

Unix/Linux notes

1 Connecting to a Linux or Unix machine

The first step that you need to complete before working on this lab is to login to a computer. As of February 2006 there are several ways to do this:

Linux machines These are the most recent machines. To log on, simply enter your ITS username and password, and you will be logged in to the K Desktop Environment (KDE).

Sun workstations These are the oldest machines in the lab, and run the UNIX operating system. To log on, first click on **Options** and choose **Remote login > Enter Host name**. Try entering **benetnasch**, and if this does not work, try **capella**. Then, enter your ITS username and password. Be patient and it should log you in.

Windows PCs Go to **Start > Astronomy Apps > Exceed (Astronomy)**. Then choose **pesto** from the list which comes up. Then, enter your ITS username and password (use the tabulation key to go from username to password). Then, be patient and a few windows should appear.

2 The command-line

2.1 Opening a terminal

This may be the first time you use a Linux or Unix system. Instead of performing tasks graphically, you will now spend a lot of your time performing simple tasks (such as copying, deleting, and moving files) using the what we call a *command-line*, i.e. typing in commands rather than using the mouse. Although this may seem like a step backwards technologically, don't be fooled! - it is actually more powerful because commands are much more flexible. For example I can easily delete all files which end in **.dat**, start with the letter **p**, and contain the number **8** somewhere in the middle. This is not something you can easily do graphically, especially if you are in a folder with thousands of files!

To open a new *terminal* (a window where you type in commands), follow these steps:

Linux machines Click on the icon on the bar at the bottom of the screen which looks like a little dark screen with a **'>'** symbol.

Sun workstations Right click on the desktop and choose **Hosts > Terminal Console**.

Windows machines You should see a little window waiting for you to type a command. Type **xterm &** to open a terminal.

2.2 Commands

You are now ready to use the command line! You should see something like `[username@machine]$` or `[username@machine]%` in the terminal at the beginning of the line. We call this a *prompt* because the computer is waiting for you to type in a command. Once you have typed a command, press the `enter/return` key to send the command to the computer.

For the purposes of this lab you will need access to a Fortran compiler and other tools. You will need to type the following commands in order to have access to them (Note: do *not* type in `[username@machine]$,` but only what comes after)

```
[username@machine]$ source /star/etc/login
[username@machine]$ source /star/etc/cshrc
```

Although command-line may be new to you, the way files are stored is more or less identical to that of a normal Mac or PC, in that you can have folders, and files or folders within these folders. In this section I will list a few useful commands that you will need to start with.

Commands are usually case sensitive. For example the `LS` command may have a different effect to the lower case `ls`. And always be very careful when using the command line, if you misuse the `rm` command for example you might end up deleting all your files!

The basic commands you will need are:

- `ls` lists all the files and folders in the current folder (list contents). Type it, and you should see something like this:

```
[username@machine]$ ls
document.txt  mail  work
```

As with any UNIX command, you can add ‘options’ to commands. For example `ls -l` lists all the files and folders, but with some extra info (e.g. size, date modified, etc...).

```
[username@machine]$ ls -l
total 0
-rw-r--r--  1 tom      tom      2399 Feb  9 11:55 document.txt
drwxr-xr-x  2 tom      tom        68 Feb  9 11:55 mail
drwxr-xr-x  2 tom      tom        68 Feb  9 11:55 work
```

- `man` is a life-saving command which gives you a help page about any command you may wish to use, and the options you can use with it. For example, you could type `man ls` to see some information on how to use the `ls` command, and what the different options do (for example `-l` which we used above). Use the arrow keys to move up and down, and press `q` to quit. Don’t hesitate to use the `man` command!
- `pwd` tells you what directory you are in at the moment (print working directory)
- `mkdir folder` creates a folder (make directory)
- `cd folder` moves you to the folder you want to go to (change directory). There are several ways you can use this. You can either type in a ‘relative’ path, which is the folder you want to go to relative to where you are at the moment. Or you can enter an absolute path, which is the path from the ‘root’ of the disk, denoted by a `/`, which is the folder which contains *everything* on the computer. See Section 2.3 for an example of this. Typing `cd` on its own returns you to your home directory.

2.3 A short example

Examples speak better than words, so consider the following example which uses what we have just learned. (anything after the # is just my comments). Don't hesitate to type these commands yourself at the same time - I have not included any 'dangerous' commands.

```
[username@machine]$ pwd          # What directory are we in?
/home/purds/username/          # This is the absolute path to this directory
[username@machine]$ ls          # What is in this directory?
document.txt mail work         # There are three files/folders
[username@machine]$ ls -l       # I need more information
total 0
-rw-r--r--  1 tom      tom      100 Feb  9 11:55 document.txt
drwxr-xr-x  2 tom      tom        68 Feb  9 11:55 mail
drwxr-xr-x  2 tom      tom        68 Feb  9 11:55 work
# document.txt is a file and the mail and work are directories
# (note the 'd' at the beginning of the line which indicates directories)
```

```
[username@machine]$ mkdir test   # Create a new directory called test
[username@machine]$ ls -l        # I can check that it is there
total 0
-rw-r--r--  1 tom      tom      100 Feb  9 11:55 document.txt
drwxr-xr-x  2 tom      tom        68 Feb  9 11:55 mail
drwxr-xr-x  2 tom      tom        68 Feb  9 12:06 test  # Here it is!
drwxr-xr-x  2 tom      tom        68 Feb  9 11:55 work
```

I now want to move to this directory ('moving down'). I can type it two ways:

```
[username@machine]$ cd test      # relative path
[username@machine]$ cd /home/purds/username/test  # absolute path
```

```
[username@machine]$ pwd          # check what directory I am in
/home/purds/username/test
```

I now want to move out of this directory ('moving up'). I can type it two ways:

```
[username@machine]$ cd ../       # relative path
[username@machine]$ cd /home/purds/username/  # absolute path
# In Unix systems .. and . are two directories which are present in all directories
# . refers to the current directory. So cd . doesn't get you very far!
# .. refers to the directory one level up
```

If I had wanted to move up and into the work directory I could have typed

```
[username@machine]$ cd ../work/   # relative path
[username@machine]$ cd /home/purds/username/work/  # absolute path
```

Now I want to go back to my home directory. I can just type

```
[username@machine]$ cd
[username@machine]$ pwd          # just to make sure!
/home/purds/username
```

2.4 More commands

- `du -sh folder` will give you the size of the directory `folder` and `du -sh .` will give you the size of the current folder you are in.
- `mv file1 file2` will move `file1` to `file2`. You can use this with directories, for example `mv file work/file` will simply move the file to the work directory without renaming it.
- `rm filename` removes files. Unlike on MacOS or Windows, there is no such thing as a 'Trash' bin. Files are deleted immediately! If you want to avoid any blunders, use `rm -i`

which always asks for confirmation before deleting files. To delete a directory, use `rm -r directory`, although we now enter the realm of the really dangerous commands! Use this badly and you could lose all your files! To delete the `test` directory we created in Section 2.3, go to your home directory and type `rm -r test`

- `exit` This closes the current terminal you are using.

2.5 Batch processing

What if we want to list only fortran source code files, i.e. files ending with `.f90`? Or what if we want to delete all files ending in `.out`? The unix command-line makes this very easy. Two important characters which you will need for this type of commands are `*` and `?`. The asterisk symbol `*` has the meaning ‘any string of characters’. So if you type `*.f90` in a command, this means ‘all files and directories whose name end in `.f90`’. The question mark `?` has the meaning ‘any character’. Therefore `???.f90` has the meaning ‘all files or directories whose name is made up of any three characters followed by `.f90`’. This can be used with any command. Here are a few examples:

```
ls *.f90      # List all files whose name ends in .f90
rm *.out      # Remove all files ending in .out
du -sh *      # Show the size of all the files and directories
mv *.f90 programs/ # Move all files whose name ends in .f90 into programs/
ls p????f90   # list all files whose name is p+any 4 characters+.f90
ls ?.out      # list all files whose name is any one character followed by .out
```

WARNING! these special characters have to be treated with extreme caution. For example you will probably never want to type `rm -r *` since this would simply erase all files and folders in the current folders. It is easier than one might think to type this command inadvertently. For example, if you wish to type `rm -r output*` to delete all folders beginning with the word `output`, you might make a mistake and include a space, which would give `rm -r output *`, which means delete the `output` directory, and `*`, i.e. everything!

3 A note about where files are kept

When you started at this University, you were assigned a username and password, as well as some disk space (which is usually referred to as your *home directory*) on a server (e.g. `maths`, `purds`, referred to as your *home server*). Whatever machine you connect to for this lab, you should have access to your home directory.

However, you cannot hold more than 20MB in your home directory. This can be a problem once you run programs which write out large files. Therefore, a directory has been dedicated to this module, and sits on a 20GB disk (which does not mean there is 20GB free at all times!). The path to this directory is:

```
/net/benetnasch/honsastro/AS3013/
```

Inside this directory there should be a directory for each user. If this is not the case, ask the lab demonstrator.

If you do not wish to have to type `cd /net/benetnasch/honsastro/AS3013/` each time you log in or open a new terminal, you can create a shortcut, or *symbolic link* to this directory:

```
[username@machine]$ cd          # To make sure you are in your home directory
[username@machine]$ ln -s /net/benetnasch/honsastro/AS3013/username as3013

# Replace username by your ITS username,
# and as3013 by whatever you want to call the shortcut

# Now if you want to access your directory on /net/benetnasch, you can type
[username@machine]$ cd as3013
# whenever you are in your home directory.
```

4 Editing files/programs

4.1 pico

When writing a fortran program you can use any text editor you want, but there are reasons why you might want to use one over another. One of the simplest and easiest text editors is `pico`. To edit (or create) a file named `myfile` for example, simply type `pico myfile`. At the bottom of the window, a list of useful shortcuts is listed. For example `Ctrl-X` saves and exits the document. Just for practice, try and create a file named `myfile`, and just write anything in it. Then use `Ctrl-X` to save it and exit. You can then use the `cat myfile` command to view the contents of the file.

4.2 xemacs and kate

As your programs grow in length and complexity, you will feel limited by such a basic editor. Instead of using `pico`, you should probably turn from the start to editors such as `xemacs` or `kate` (the latter being only available on the linux computers). These two are graphical editors, which means the mouse will be useful once again, and it will be much more intuitive to perform basic tasks such as opening and saving files.

To start `xemacs`, type `xemacs &`, or to start `kate`, type `kate &`. The ampersand `&` which can be used after any command means that once the graphical program is open the terminal returns to the prompt, so that you can continue to type commands.

To create a new file in `xemacs`, click on `open`, and simply type the name of the new file you want to create. Click on `OK`, and if the file does not exist it will be created. In `kate`, simply start writing your program, then save, and you will be asked for a filename. Both of these editors have advanced features such as syntax coloring and auto-indenting - which of the two you use is a matter of preference - try the two, and then you can decide for yourself!

To activate syntax coloring in `xemacs`, go to the `options` menu, and click on `syntax coloring`, and choose `in this buffer`. Then go to the `options` menu again, and choose `save options to init file`. Then click on `yes`. This now means that syntax coloring will always be on when you open a Fortran 90 program. If you want to automatically indent all or part of your program, select the region you want to indent, go to the `F90` menu, and choose `indent region`. Note that whichever editor you decide to use, the filename of the program has to end in `.f90` in order to benefit from the above features. The Fortran 90 compiler might also get confused if you forget the `.f90` extension

Other commands you will find useful in future and which you should find out more about using `man` include `cat`, `more`, `head`, `tail`.

5 Setting up a `.cshrc.extras` file

Each time you open up a terminal, the computer executes the contents of a file named `.cshrc` in your home directory. Its name begins with a period, so it is invisible to a normal `ls`. However, typing `ls -a` should reveal all files including hidden ones.

The `.cshrc` file should not be modified. However, if you want to add some commands which will be run each time you open a terminal, it is possible to do so, by creating a file called `.cshrc.extras`

To do this, go to your home directory (can be done by just typing `cd` on its own). Then, type the following

```
pico .cshrc.extras
```

to create and open this file. You will probably want to write in this file the following commands, so that you don't ever have to type them again:

```
source /star/etc/login  
source /star/etc/cshrc
```

Then simply press `Ctrl-X` to save and close `pico`.

Now every time you open a new terminal, commands such as `f90` or `g95` should work straight away!

Thomas Robitaille
12 February 2006