Hints and tips

Long running jobs

If you have a job that will run for a long time (more than a day) you should first run a test job on a data set and with parameters that are representative of the final problem, but that will complete much quicker. This way you can confirm that your code is syntactically correct and produces results that appear to be correct before you waste weeks or months of CPU time.

In addition, you should make sure that your job writes results in a form that allows the program to be restarted. This is because the longer your job runs for, the greater the chance that the cluster will suffer from a fault and cause your job to be killed. If your job doesn't save it's state then it will have to restart from the beginning. How often to save the program state depends on how much data you has to be written and what impact it has on the speed of the calculation.

Source code management

We recommended that you use a source code management system such as Subversion (SVN), Mercurial (Hg) or Git to capture the changes you make to your code. This has two benefits:

1. You will never lose any changes to your code and makes it much easier to have multiple versions of your code available - if you want to test a new approach you don't have to have multiple versions of the code in multiple directories and then remember which is which
2. If you modify your code to print the version of the source code when it reports the results, you will always know which code produced which results

Windows files compared to UNIX

With Microsoft Windows, the representation of the Enter key is different to on UNIX (i.e. Linux and Mac OS X) - Windows uses a carriage return (CR) and a line feed (LF) character, whereas UNIX just uses a line feed. This extra character causes problems with some programs that don't realize the CR is part of the new line. You can detect the problem files with the file command:

```
$ file windows.txt unix.txt
windows.txt: ASCII text, with CRLF line terminators
unix.txt: ASCII text
```

and then fix it with the dos2unix command:

```
$ dos2unix windows.txt
dos2unix: converting file test.txt to UNIX format
$ file windows.txt
windows.txt: ASCII text
```

If you need to go the other way, there is the inverse tool *unix2dos*.

Editing files

There are three text editors installed on the cluster:

- nano
- vi or vim
- emacs

If you don't already know vi or emacs, I recommend that you use nano. This is a very simple editor which allows you to change files without copying them to and from your workstation. To use nano, simply type nano or nano *filename* - help is provided on the screen and there is an online manual.