Software Modules

Default Environment

The user environment is setup using environment modules. By default, a number of modules are automatically loaded to configure the environment to allow running of applications and the submission of jobs to the cluster.

It is possible to change the environment which is loaded when logging in, by editing the shell initialisation file ~/.bashrc (or ~/.cshrc or ~/.tcsh if using csh/tcsh as your shell). If your shell initialisation file is modified it will affect all future login sessions, and all batch jobs not yet started. Since some modules are required for proper operation of the account, caution is needed before removing any automatically loaded modules. Changes to the shell initialisation file will take effect during the next login or at the creation of the next shell.

At any time you can check the currently loaded modules via:

```
module list
```

which possibly produce the following output:

```
Currently Loaded Modulefiles:
   1) default-manpath/1.2.1
   2) torque-oscar/4.2.9
   3) oscar-modules/1.0.5
```

Modules work by setting environment variables such as PATH and LD_LIBRARY_PATH. Therefore if you need to modify variables directly it is essential to retain the original values to avoid breaking loaded modules (and potentially rendering essential software "not found"), e.g. do

```
export PATH=$PATH:/home/abc123/custom_bin_directory
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/home/abc123/custom_lib_directory
```

and not

```
export PATH=/home/abc123/custom_bin_directory
export LD_LIBRARY_PATH=/home/abc123/custom_lib_directory
```

System modules

On complex computer system, it is necessary to make available a wide choice of software packages with possible multiple versions. It can be quite difficult to set up the user environment so as to always find the required executables and libraries. (This is particularly true where different versions use the same names for files). Environment modules provide a way to selectively activate and deactivate modifications to the user environment which allow particular packages and versions to be found.

The basic command to use is module:

```
module
   (no arguments)              print usage instructions
   avail|av                    list available software modules
   add|load <modulename>       add a module to your environment
   rm|unload <modulename>      remove a module
   list|li                     list currently loaded modules
   purge                       remove all modules
   whatis                      as above with brief descriptions
```

Since the home directory is shared with the compute nodes, the same default environment is inherited when a user's job commences via the queuing system. It is good practice to explicitly set up the module state in the job submission script to remove ambiguity.

Using modules in batch files

To use the software package xxx in a job submission script it is a simple matter of specifying

```
module load xxx
```

The naming convention for modules are [*software/version-compiler*]. Below is a list of the extensions:

1. –gnu, software compiled with standard GNU compiler.
2. –intel, software compiled with Intel compiler.
3. –gcc485, software compiled with GNU v4.8.5 compiler.
4. No extension, is commercial software that wasn't compiled.

Creating your own modules

Apart from the available system environment modules, you can define your own modules. If you load the *use.own* module, as below,

```
module load use.own
```

this will change your *MODULEPATH* environment variable and add the directory *$HOME/privatemodules* to it. This is the default location of private modules. See

```
man modulefile
```

for further information on writing your own modules.