Environmental Modules

Intro

One problem with UNIX is that it requires (or allows) command execution to be tweaked using environmental variables. For example, the PATH variable is consulted when you run a command like ls. Since this is not a full path name, the kernel will consult the path names listed in PATH until it finds one that contains a command called ls which it then executes. The man command expects that the MANPATH variable will contain a list of path names that will contain man pages that it can display. In some cases, these are variables you have just learn to set yourself. In most cases, setting PATH, MANPATH, and other variables can be a tiresome task for users. It can also be a burden when multiple versions of software exists in which case you could continually need to be updating your environment. For this reason, CCBB provides the module command which lets you examine software that we have available, query it, and dynamically prepare your environment to use or not use it as needed.

Usage

To prepare to use a piece of software you first run the module command with the load argument and the name of a module that you want to load. For example, if you wanted to run java, you would run

```bash
module load java
```

which loads the default version of java. You can also load a specific version with

```bash
module load java/1.6.0_22
```

which loads the 1.6.0 update 22 version of Sun’s JDK. To see all of the java installations you can run

```bash
module avail java
```

and you can run

```bash
module avail
```

to see all of the modules we have provided. If you do not see the software you need, please let us know so that we can confirm whether it is installed, and if not whether we can install it. Notice that we explicitly make one version of a module a default which is what you get if you don’t provide a version number (by default the module command would just sort the names, so a default would be found anyways). The particular choice of a default is subject to change.

Once a module is loaded, then you can just type the short name of the command. This can seen by loading one of the above java modules, and then running

```bash
java -version
```

If you no longer need to use a given software product, it is usually best to unload it. This prevents problems with environmental variables being set to values to cause one program to work, and another to fail. The command to unload is
module unload java

If multiple configurations are provided you can swap between them with

module swap java/1.6.0_22

Finally, you can use

module apropos text

which like the apropos command searches all of the modules to find one that might be applicable. You can also use

module help java/1.6.0_22

to read the help file for a software product. This should then tell what the module file provides, as well as any other usage issues that you might need to know. It should also have links to the home page for the product which you can then refer to for further help.

**Modules and SGE**

The module command can also be used in job scripts on the cluster. To do so, you must first determine what modules you need. This can be done by experimenting on the commandline, and determine what modules you have to load to support all of the products used in your job script. By running

```
module list
```

you will see what modules you have loaded. Then in the script place the lines

```
. /etc/profile
module load foo
module load bar
```

before you start executing commands.