Table of Contents

Module 1
Adding user module files 2

Module

Modules documentation

> `module` is a tool which helps in loading and unloading pre-configurated settings for the shell

Some programs need certain environment variables and libraries to run correctly. For example, the PATH-variable has to be extended to allow command lookup in additional directories. For this purpose, there are so-called modules which configure your shell with the necessary settings. Such modules can be load dynamically and unloaded its settings if unnecessary.

To get an idea which modules are available, run '`module avail`' in the shell.

```
$ module avail

------------------- /usr/share/Modules/modulefiles -------------------
dot         module-info null
module-git  modules     use.own

------------------------ /cluster/modulefiles ------------------------
afni-toolbox           gnuplot/4.6.7          openfoam/3.0.1b
ansys/17.0/fluent      joe/3.7                openmpi/gcc/64/1.10.1
cmake/2.8.12.2         liggghts/3.3.1         openmpi/gcc/64/1.8.4
...
```

From the available modules you can add it to your shell by using the `load` option.

```
$ module load openmpi/gcc/64/1.10.1
```

To show which modules currently are loaded in the active shell, use the `list` option.

```
[user@login ~]$ module list
Currently Loaded Modulefiles:
  1) openmpi/gcc/64/1.10.1   3) starCCM/10.06
  2) python/2.7.12
```

To unload a module, the consequent switch `unload` can be used:

```
$ module unload python/2.7.12
$ module list
Currently Loaded Modulefiles:
  1) openmpi/gcc/64/1.10.1   2) starCCM/10.06
```

`module` command can also be added to job scripts, as you may have seen in some templates already.

In some special cases, modules have to be loaded in a certain order. Moreover, avoid loading multiple modules from the same family. This could corrupt the settings for the loaded modules. Then you would need to start over with a new terminal.

If you want to know what environment changes are done by a specific module, you can use the `show` option in the following way:

```
$ module show starCCM/10.06
```

For more information, read the manual here, or by running:

```
$ man module
```

And check the module which is used to create a module:

```
$ module load modules
```

Adding user module files

You can write your own modules as well. Sometimes a new version of software is installed, but no new modulefile has been created yet. In this case, copy a previous version, and modify it to the new version.

In order to make `module` aware of your own module files, add these lines with the respective path to your `.bashrc` script.

```
## Make custom modules in <path> available
module use --append <path>
```

Replace `<path>` with your files' location.

Now, when you call `module avail`, you should see your own modules as well.

Guide, Neumann, Module