Loading Environments for Using Codes

Description of Environment Modules

Environment Modules provide a convenient way to dynamically change the users' environment through modulefiles. This includes easily adding or removing directories to the PATH environment variable.

A modulefile contains the necessary information to allow a user to run a particular application or provide access to a particular library. All of this can be done dynamically without logging out and back in. Modulefiles for applications modify the user's path to make access easy. Modulefiles for Library packages provide environment variables that specify where the library and header files can be found.

Packages can be loaded and unloaded cleanly through the module system. It is also very easy to switch between different versions of a package or remove it.

Default Settings

For your convenience, the cluster is set up with a default environment in place. The default environment setup loads environment modules for:

- autotools
- prun
- gnu7
- openmpi3

Which Modules to Load for Codes

On the main cluster page is a listing of codes currently installed on the cluster. For many of the codes, the last column shows the environment module commands which must be run to use that code. Some codes will require more than one environment module. And occasionally you will need to run an additional non-module command as shown in the last column of the codes listing.

When to Load Modules for Codes

If you will be running an interactive batch job, load any modules on the head node, nanolab, prior to running the *srun* command. Your loaded environment modules will stay with you on the compute nodes.

Commands for Environment Modules

List currently loaded modules

```
$ module list
```

List modules available to load

```
$ module avail
```

Load a module

```
$ module load <package1> <package2> ...
```

Unload a module

```
$ module unload <package1> <package2> ...
```

Swap loaded modules

```
$ module swap <swapping_out_package> <swapping_in_package>
```

Display module help information

```
$ module help <packageName>
```

Search for a module where the name contains a string

```
$ module avail <string_to_search_in_name>
```

Display module whatis description

```
$ module whatis <packagename>
```

Module keyword search of help message and whatis description

```
$ module keyword <word1> <word2> ...
```