Environment modules on Cheyenne

The **module** utility helps you identify software that is available on the system and then load compatible packages. It manages complex combinations of paths, variables, and dependencies so you can compile and run jobs efficiently and make the most of your allocation.

Some modules are loaded by default. To see which modules those are, run **module list** when you log in. Depending on the work you need to do, you can load additional modules or different modules, or you can create and save multiple customized environments as described below.

> Do not use personalized start files to load *environment modules*; it can cause conflicts. Instead, set up any unique environments that you need as described in our Customized environments documentation. Use that approach to create and save different environments for various aspects of your work – one for model runs and another for data analysis, for example.

Page contents

Essential module commands

Following are descriptions of commonly used module commands.

module av – Show which modules are available for use with the currently loaded compiler.

```
--------------- /glade/u/apps/ch/modulefiles/default/compilers -----
  gnu/5.4.0    gnu/6.3.0    (D)    intel/16.0.3 (L,D)
  gnu/6.2.0    intel/16.0.1

----------------- /glade/u/apps/ch/modulefiles/default/idep --------
  R/3.3.2       ncarenv/1.0 (L)    nco/4.6.2       python/2.7.13
  git/2.10.2    ncl/6.3.0          ncview/2.1.7

------------- /glade/u/apps/ch/modulefiles/default/intel/16.0.3 ----
  impi/5.1.3.210    mpt/2.15            (L)    netcdf/4.4.1
  mkl/11.3.3        ncarcompilers/0.3.5 (L)    netcdf/4.4.1.1 (L,D)

--------- /glade/u/apps/ch/modulefiles/default/mpt/2.15/intel/16.0.1
  netcdf-mpi/4.4.1.1    pnetcdf/1.8.0

 Where:
  L:  Module is loaded
  D:  Default Module
```

module help – List options and subcommands for the module utility; or specify a modulefile by name for help with an individual module.

```
module help
module help netcdf
```

module list – List the modules that are loaded.

module load – Load the default version of a software package, or load a specified version.

```
module load modulefile_name
module load modulefile_name/n.n.n
```

module purge – Unload all modules. Some users include this command in a batch script, followed by a sequence of **module load** commands to establish a customized environment for the job being submitted.

module spider – List all modules that exist on the system. This does not give you information on module dependencies or tell you which modules can be loaded without conflicts at that point. Output from this command is published on these web pages daily for easy reference when you are not logged in: Cheyenne and Casper.

module swap – Unload one module and load a different one. Example:

```
module swap netcdf pnetcdf
```

module unload – Unload the specified software package.

```
module unload modulefile_name
```

module whatis – Get a short description of a module.

Customized environments

If you have created your own environment or want to have multiple collections of modules for various tasks, you can save those customized environments for easy re-use.

To save a customized environment as your **default** environment, load the modules that you want to use, then simply run **module save** or **module s**.

```
module save
```

If you plan to set up additional custom environments for other needs, give each collection of modules a unique name.

```
module save environment_name
```

To see a list of your customized, saved environments, use **module savelist**.

```
module savelist
```

To use one of the custom environments from that list, use **module restore**, or **module r**, followed by the name.

```
module restore environment_name
```

To see which modules you've saved in a custom environment, use **module describe** as shown.

```
module describe environment_name
```

To remove a customized environment that you have saved:

- Change to your **.lmod.d** directory.
- List the files.
- Use **rm** to delete what you no longer need.

```
cd /glade/u/home/username/.lmod.d
ls -l
rm environment_name
```

To revise a customized environment:

- Restore (change to) that environment.
- Unload, load, or swap modules as needed.
- Save the environment as a default environment again with the same name.

```
module restore myenvironment
module load additional_module
module save myenvironment
```

The previously saved environment will be renamed automatically with the addition of a tilde (~). In the example just above, the previously saved environment would be renamed to **myenvironment~**.

Troubleshooting tips

Situation: You load a custom default module collection (for example, **module restore myenvironment**). You receive a warning similar to this:

```
Lmod Warning: The following modules have changed: pgi
Lmod Warning: Please re-save this collection
```

What to do: “Re-save” the customized module collection by running **module save** and using the same environment name, as follows.

```
module save myenvironment
```