Environment modules

Available modules

To get a list of available modules, use `module avail` or add a string to match to shorten the list (eg: `module list FFTW`.

Tip: since modules installed through EasyBuild have the name of the build toolchain appended, looking for a compiler version or toolchain will result in many hits. Appending a / to the name will limit the results to the compiler or toolchain itself (since the compiler itself will be called e.g. `GCC/11.3.0` but a package built with it will be called something like `FFTW/3.3.10-GCC-11.3.0`)

Loading and unloading a module

To load a module, use something like: `module load GCC/11.3.0`. Usually there is a default module for a specific package, marked with `(D)` in the list of available modules. Typing just `module load` GCC will load the GCC module marked as default. Version numbers and toolchains (everything after the / in the module name) can usually be abbreviated, eg `module load GCC/11` would work as well.

To unload a module, use `module unload`. There is also a `module switch` command to swith between two modules that provide the same basic functionality, but in most cases, loading one version of something (eg GCC) will unload all earlier loaded versions, so an explicit switch command is not necessary any more.

Command `ml` is a useful frontend to the most used module commands (see `ml –help` for details; also see `module –help` for details about module commands and options).

Warnings

Be careful before adding a `module load` command to your startup files (`.bashrc` or `.tcshrc`); if a program needed for logging in (or for your graphical desktop) is incompatible with something in the loaded module, logging in may fail. This is especially true for loading any non-default python environment, which will not have the python modules needed to run several system utilities. And part of the modern desktop environments are written in python.\n So, it's probably best to load an environment only when it is needed. If you need the same set of modules often, consider making a script or shell alias for loading them easily.

Special case: MPI

MPI is available in various versions, but none is loaded by default (because doing so would make it more difficult for users to choose which version they need, and some software requires another version than other). Run `module avail mpi/` to see a list of available versions.

It might be that the software you install mentions which MPI implementation (OpenMPI or MPICH probably) and perhaps also which version is needed (eg not everything built for OpenMPI 3 will also work with version 4.*) If no special requirements are listed, the implementations that are installed with the operating system are probably the safest choice (since they will also use the system default compilers and libraries): mpi/mpich-x86_64 or mpi/openmpi-x86_64 On the other hand, if install instructions are already telling you to use some specific version of a compiler or required libraries, you may have to look a bit further to find a matching set. E.g, if a newer compiler is required, you will see for the module avail command that there is a OpenMPI/4.1.4-GCC-11.3.0 which is OpenMPI 4.1.4 installed with GCC 11.3.0. And if your project then needs more libraries, you may have to load some additional modules (eg if you need FFTW, 'module avail FFTW' will show that there is a module FFTW/3.3.10-GCC-11.3.0 to use).

See also:

- Compilers
- Sfinx modules: some special cases (for astronomical software)
- EasyBuild environment: building your own modules