VIENNA SCIENTIFIC CLUSTER

Software management with Modules

Jan Zabloudil, Moritz Siegel

Environment Modules

http://modules.sourceforge.net/

Makes **different versions** of one piece of software available.

Can **Define** or **change** environment variables required for using a piece of software:

```
PATH
CPATH
LIBRARY_PATH
LD_LIBRARY_PATH
...
```

Find Modules

Use `module avail` to find installations of a package:

```
skylake trainee00@l44:~$ module avail intel-oneapi-mpi
zen trainee00@l55:~$ module avail amdblis
```

This is similar to `spack find`, and mostly shows the **same** packages.

Some software is not installed with spack so you can **only** load it with modules!

Package names are **case sensitive**, like **M**atlab, **M**athematica, ...

Load Your Module

Use **`module load`** to load your package:

```
skylake trainee00@l44:~$ module load openmpi/4.1.4-gcc-12.2.0-xt53foa
```

Always state the whole line:

```
skylake trainee00@l44:~$ module load openmpi/4.1.4
ERROR: Unable to locate a modulefile for 'openmpi/4.1.4'
```

Module names differ depending on cluster (CPU architecture):

```
skylake trainee00@l44:~$ module load python/3.11.3-gcc-12.2.0-rtzvjko
zen trainee00@l55:~$ module load python/3.11.3-gcc-12.2.0-hn7p65z
cuda-zen trainee00@l55:~$ module load python/3.9.15-gcc-12.2.0-dnctq7y
```

List Loaded Modules

Type **`module list`** to show loaded modules:

```
zen trainee00@l55:~$ module list
zen trainee00@l55:~$ module list -t
Currently Loaded Modulefiles:
netlib-lapack/3.10.1-gcc-12.2.0-4qrxbdw
python/3.9.15-gcc-12.2.0-my6jxu2
py-setuptools/63.0.0-gcc-12.2.0-jru4czh
py-numpy/1.24.3-gcc-12.2.0-muackhh
```

Unload & Remove Modules

Type **module purge** to remove all loaded modules:

```
1 module purge
```

Recommended in job scripts before loading any modules (reproducibility).

Use **module rm** or **module unload** to remove/unload a list of modules:

```
2 zen traineeOO@l55:~$ module rm <modulename> <modulename> ...
3 zen traineeOO@l55:~$ module unload <modulename> <modulename> ...
```

Inspect Modules

Use **`module show`** to show the variables added to your environment by a module:

```
zen trainee00@l55:~$ module show python/3.11.3-gcc-12.2.0-hn7p65z
-------------------------------------------------------------------
/opt/sw/zen/spack-0.19.0/share/spack/modules/linux-almalinux8-zen3...

module-whatis   {The Python programming language.}
prepend-path    PATH /gpfs/opt/sw/zen/spack-0.19.0/opt/spack/linux...
prepend-path    LIBRARY_PATH /gpfs/opt/sw/zen/spack-0.19.0/opt/spa...
prepend-path    CPATH /gpfs/opt/sw/zen/spack-0.19.0/opt/spack/linu...
prepend-path    MANPATH /gpfs/opt/sw/zen/spack-0.19.0/opt/spack/li...
...
```

No need to load a module before inspecting it.

Dependencies

Sometimes a module also **needs** dependencies:

```
zen trainee00@l56:~$ module load py-numpy/1.24.3-gcc-12.2.0-muackhh
Loading py-numpy/1.24.3-gcc-12.2.0-muackhh
  Loading requirement: netlib-lapack/3.10.1-gcc-12.2.0-4qrxbdw
      python/3.9.15-gcc-12.2.0-my6jxu2
      py-setuptools/63.0.0-gcc-12.2.0-jru4czh
```

Add `--auto` to automatically load all the dependencies:

```
zen trainee00@l56:~$ module load --auto py-numpy/1.24.3-gcc-12.2.0-muackhh
```

Write the line `export MODULES_AUTO_HANDLING=1` to your `~/.bashrc` to **always** load all dependencies.

Exercises

- ☐ login to VSC-4/5
- ☐ find a recent **python** module.
- ☐ load a recent **mathematica** module.
- ☐ load a recent **Matlab** module.
- ☐ list all **loaded** modules.
- ☐ unload **all** modules.
- ☐ find a **matplotlib** *python* module.
- ☐ find an **openfoam** module compiled with **gcc**.
- ☐ find an **openfoam** module compiled with **intel**.
- ☐ what version of **numpy** does the module `py-matplotlib/3.6.2-gcc-12.2.0-wbe7m7i` need.