

Software Modules

Peter Weisbrod, SCC, KIT



Software (=Environment) modules

By default manual setup of \$PATH, \$LD_LIBRARY_PATH ... for compilers, libraries and software packages etc.

→ Getting complicated if multiple versions of same software installed

Solution:

- dynamic modification of the session environment by
→ instruction sets stored in *modulefiles*

HowTo?

- *load* and *unload* instruction sets (= modulefiles)
- How to use modulefiles in general?

```
$ module help
```

- More information:
 - about the tool in use: Lmod → <https://lmod.readthedocs.io/en/latest/>

modulefiles: available / search

■ Display all available modulefiles

```
$ module avail = $ ml av
```

```
----- /opt/bwhpc/common/modulefiles/Core -----  
bio/freesurfer/6.0.0  
bio/fsl/6.0.4  
bio/nest/2.18.0 (T)  
bio/nest/2.20.1 (T,D) (D)  
cae/ansys/2022R1_no_license  
cae/ansys/2022R2_no_license  
cae/cgns/3.4.1-intel-19.1  
cae/cgns/4.1.2-gnu-8.3 (D)  
cae/openfoam/v2006-impi  
cae/openfoam/v2006  
cae/openfoam/v2012  
cae/openfoam/v2106-impi  
cae/openfoam/v2112  
cae/openfoam/v2206 (D)  
cae/openfoam/4.1-extend  
devel/python/3.8.6_gnu_10.2  
devel/python/3.8.6_intel_19.1  
devel/python/3.10.0_gnu_11.1  
devel/python/3.10.0_intel_19.1  
devel/reports/20.0  
devel/scorep/7.1-gnu-11.2-openmpi-4.1  
devel/scorep/7.1-intel-2021.4.0-impi-2021.4.0  
devel/scorep/7.1-intel-2021.4.0-openmpi-4.1  
devel/scorep/7.1-llvm-12.0-openmpi-4.1 (D)  
devel/swig/4.0.2  
devel/tbb/2021.4.0  
devel/valgrind/3.19.0  
devel/vampir/9.10  
devel/vampir/10.1 (D)  
devel/vasm/1.3
```

■ Search: Display all available „compiler“ modulefiles

```
$ module avail compiler
```

```
----- /opt/bwhpc/common/modulefiles/Core -----  
compiler/clang/9.0  
compiler/gnu/8.3.1  
compiler/gnu/9.3  
compiler/gnu/10.2 (D)  
compiler/gnu/12.1  
compiler/intel/19.0  
compiler/intel/19.1  
compiler/intel/2021.4.0_llvm  
compiler/llvm/12.0 (D)  
compiler/llvm/13.0  
compiler/llvm/14.0  
compiler/llvm/15.0
```

modulefiles: spider / search (1)

- Display all **possible** modulefiles

```
$ module spider
```

The following is a list of the modules and extensions currently available:

```
bio/freesurfer: bio/freesurfer/6.0.0

bio/fsl: bio/fsl/6.0.4

bio/nest: bio/nest/2.18.0, bio/nest/2.20.1
  NEST is a command line tool for simulating neural networks

cae/abaqus: cae/abaqus/2020

cae/ansys: cae/ansys/2020R2, cae/ansys/2021R2

cae/cgns: cae/cgns/3.4.1-intel-19.1, cae/cgns/4.1.2-gnu-8.3

cae/comsol: cae/comsol/5.6

cae/cst: cae/cst/2020

cae/lsdyna: cae/lsdyna/9.3.1, cae/lsdyna/12.0.0

cae/openfoam: cae/openfoam/v2006-impi, cae/openfoam/v2006, cae/openfoam/v2012, cae/openfoam/v2106-impi, ...
```

- Search: Display all **possible** „gnu compiler“ modulefiles

```
$ module spider compiler/gnu
```

```
compiler/gnu:
```

```
Versions:
```

```
compiler/gnu/8.3.1
compiler/gnu/9.3
compiler/gnu/10.2
compiler/gnu/10.3
compiler/gnu/11.1
compiler/gnu/11.2
```

modulefiles: spider / search (2)

- Display all **possible variants** of a modulefiles

```
$ module spider mpi/openmpi/4.0
```

```
-----  
mpi/openmpi: mpi/openmpi/4.0  
-----
```

You will need to load all module(s)
on any one of the lines below before
the "mpi/openmpi/4.0" module is available
to load.

```
compiler/clang/9.0  
compiler/gnu/10.2  
compiler/gnu/8.3.1  
compiler/gnu/9.3  
compiler/intel/19.0  
compiler/intel/19.1  
compiler/llvm/10.0  
compiler/pgi/2020
```

modulefiles: help / whatis

- Show help of modulefiles, e.g. `$ module help chem/turbomole`

```
--Module Specific Help for "chem/turbomole/7.6.1" -----  
-----  
| Loading Parallel version          |  
-----  
* Code_words are: SMP (shared memory parallel) and  
                  MPI (message passing interface)  
* To load for e.g. SMP, execute:  
  export PARA_ARCH=SMP  
  module load chem/turbomole/7.6.1  
...  
...  
-----  
| Support                          |  
-----  
...
```

Version fallback is the defined default (here 7.6.1)

- Show short info modulefile

```
$ module whatis chem/turbomole
```

```
chem/turbomole/7.6.1 : Quantum chemistry package Turbomole version 7.6.1
```

modulefiles: show

- Show all instructions of modulefile

```
$ module show compiler/gnu/11
```

```
-----  
/opt/bwhpc/common/modulefiles/Core/compiler/gnu/11.lua:  
-----  
...  
setenv("CC", "/opt/gcc/11/bin/gcc")  
setenv("CFLAGS", "-O2 -march=native")  
setenv("OMP_PROC_BIND", "true")  
...  
prepend_path("PATH", "/opt/gcc/11/bin")  
prepend_path("LD_LIBRARY_PATH", "/opt/gcc/11/lib64")  
...  
whatis("Sets up GCC C/C++/Fortran compiler version 11 in your environment...  
help([[The GNU Compiler Collection includes front ends for C, C++,  
Objective-C, Fortran, Java, Ada, and Go, as well as libraries for these  
Languages (libstdc++, libgcj,...). GCC was originally written as the  
compiler for the GNU operating system. The GNU system was developed  
to be 100% free software, free in the sense that it respects the  
user's freedom.  
  
In case of problems, please contact: Hartmut Häfner <hartmut.haefner@kit.edu>  
SCC support end: As soon as GNU compiler version 13 is available!  
]])
```

Setting environment
variables

Modifying environment
variables

Content of printout
functions

module show does NOT load the modulefile

modulefiles: show

■ Show all instructions of modulefile

```
$ module show compiler/gnu/11.2
```

```
-----  
/opt/bwhpc/common/modulefiles/Core/compiler/gnu/11.2.lua:  
-----  
setenv("GNU_VERSION","11.2.0")  
setenv("GNU_HOME","/opt/bwhpc/common/compiler/gnu/11.2.0")  
setenv("GNU_BIN_DIR","/opt/bwhpc/common/compiler/gnu/11.2.0/bin")  
...  
prepend_path("PATH","/opt/bwhpc/common/compiler/gnu/11.2.0/bin")  
prepend_path("LD_LIBRARY_PATH","/opt/bwhpc/common/compiler/gnu/11.2.0/lib64")  
...  
conflict("compiler/intel")  
conflict("compiler/pgi")  
whatis("GNU compiler suite version 11.2.0 (gcc, g++, gfortran,...  
help([[This module provides the GNU compiler collection version 11.2.0  
via commands gcc, g++, gfortran and gccgo. The GNU compiler has been build ...  
...  
cpp      - GNU pre processor  
gcc      - GNU C compiler  
g++     - GNU C++ compiler  
gfortran - GNU Fortran compiler (Fortran 95/2003/2008 ...  
...  
In case of problems, submit a trouble ticket at  
'https://bw-support.scc.kit.edu'.  
  
The full version is: compiler/gnu/11.2.0  
]])
```

Setting environment
variables

Modifying environment
variables

Conflict setup

module show does NOT load the modulefile

modulefiles: categories & dependencies

- Module names already implicate dependencies:

→ **Category/softwarename/version_attributes-dependencies**

e.g. **numlib/petsc/3.13.4-gnu-10.2-openmpi-4.0**

→ PETSc package version 3.13.4, compiled with GNU 10.2 and OpenMPI 4.0

- Categories:

compiler/	for compiler, e.g. intel, gnu, pgi, open64
devel/	for debugger, e.g. ddt, and development tools, e.g. cmake, itrac
mpi/	for MPI libraries, e.g. impi, openmpi, mvapich(2)
numlib/	for numerical libraries, e.g. Intel MKL, ACML, nag, gsl, fftw
lib/	for other libraries, e.g. netcdf, global array
bio/	for biology software, e.g. bowtie, abyss, mrbayes
cae/	for CAE software, e.g. ansys, abaqus, fluent
chem/	for chemistry software, e.g. gromacs, dacapo, turbomole
math/	for mathematics software, e.g. matlab, R
phys/	for physics software, e.g. geant4
vis/	for visualisation software, e.g. vmd, tigervnc

Exercise 1

- 1. Find all modulefiles that start with „mpi“

Exercise 1 - Solution

- 1. Find all modulefiles that start with „mpi“

```
$ module -t -r spider '^mpi'
```

```
mpi/impi/2019  
mpi/impi/2020  
mpi/impi/2021.4.0  
mpi/openmpi/default  
mpi/openmpi/4.0  
mpi/openmpi/4.1
```

modulefiles: load / list

- Modulefiles are sorted in categories, software name and versions:

```
$ module load <category>/<software_name>/<version>
```

- Load a **default** software:

```
$ module load <category>/<software_name>
```

- e.g. Intel compiler

```
$ module load compiler/intel mpi/impi
```

→ loads currently Intel compiler suite 19.1

→ loads currently Intel-MPI 2020 for Intel compiler suite 19.1

- Display all loaded modules

```
$ module list = $ ml
```

Currently Loaded Modules:

1) compiler/intel/2021.4.0 2) mpi/impi/2021.4.0



modulefiles: load dependencies /conflicts (1)

■ Dependencies

- e.g.: some applications require particular compiler libraries

```
$ module load numlib/gsl/2.6-intel-19.1  
$ module list
```

Currently Loaded Modules:

1) compiler/intel/19.1 2) numlib/gsl/2.6-intel-19.1

Autoloaded Intel-Suite 19.1

■ Conflicts:

- a) load different software version in the same session, e.g. Intel:

```
$ module load compiler/intel/19.0  
$ module load compiler/intel/2021.4.0
```

The following have been reloaded with a version change:
1) compiler/intel/19.0 => compiler/intel/2021.4.0

- b) load module with dependencies on other modules

```
$ module load compiler/intel/2021.4.0  
$ module load numlib/gsl/2.6-intel-19.1
```

Requires Intel-Suite 19.1

The following have been reloaded with a version change:
1) compiler/intel/2021.4.0 => compiler/intel/19.1

Exercise 2

- 1. Load latest OpenMPI with default INTEL compiler (Hint: Option ,-d‘ to show only default version)

Exercise 2 - Solutions

1. Load latest OpenMPI with default INTEL compiler

```
$ module -d avail compiler/intel  
compiler/intel/2021.4.0  
  
$ module load compiler/intel/2021.4.0  
  
$ module -r spider 'mpi/openmpi.*'  
→ mpi/openmpi/4.1  
  
$ module load mpi/openmpi/4.1
```

Pitfall: Loading openmpi before compiler

```
$ module load mpi/openmpi/4.1
```

Lmod has detected the following error:

These module(s) or extension(s) exist but cannot be loaded as requested:

"mpi/openmpi/4.1"

Try: "module spider mpi/openmpi/4.1" to see how to load the module(s).

modulefiles: unload/swap/purge

- To remove module *foo*:

```
$ module unload foo
```

→ be aware that you might create **inconsistencies**

```
$ module load numlib/gsl/2.6-intel-19.1  
$ module unload compiler/intel/19.1
```

```
-----  
The following dependent module(s) are not currently loaded:  
compiler/intel/19.1 (required by: numlib/gsl/2.6-intel-19.1)  
-----
```

- Swap = remove + load

e.g.:

```
$ module swap compiler/gnu compiler/intel
```

Removes default GNU
version and loads
default INTEL version

- To remove **ALL** modules at once:

```
$ module purge
```

```
$ module list  
No modules loaded
```


Exercise 3 - Solution

- 1. Determine system gcc version (without modulefile system)

```
$ module purge
$ module list
No modules loaded
$ gcc --version
gcc (GCC) 8.4.1 20200928 (Red Hat 8.4.1-1)
```

- 2. Load newest gcc version (hint: newest → highest version number of compiler/gnu)

```
$ module spider compiler/gnu
Versions:
  compiler/gnu/8.3.1
  compiler/gnu/9.3
  ...
  compiler/gnu/11.2
  compiler/gnu/12.1
$ module load compiler/gnu/12.1
$ gcc --version
gcc (GCC) 12.1.0
```

Private modulefiles

- Each user can create own modulefiles:

e.g. modulefiles that adds path of own programs, `$HOME/special`, to `$PATH`

→ content of this modulefile „*mybin.lua*“

```
-- Own Lua modulefile to prepend $HOME/special to $PATH
--
prepend_path("PATH", os.getenv("HOME") .. "/special")
```

→ place „*mybin.lua*“ under `$HOME/privatemodules`

→ to make all own modules visible to “module avail” command, enter:

```
$ module use $HOME/privatemodules
```

→ note: own module have higher priority than systems ones

```
$ module avail
```

```
--- /home/kit/scc/ab1234/privatemodules ---
    mybin
-----
```

- Remove own modules:

```
$ module unuse $HOME/privatemodules
```