pawsey

Supercomputing User Training

Module 5: Using Software Modules

Pawsey Training Series

Supercomputing User Training

1. Supercomputing Introduction
2. Logging In
3. Filesystems Overview
4. Moving Data In and Out
5. Using Software Modules
6. Using Software Containers
7. Accounting Model Overview
8. Job Scheduling Overview
9. Running Jobs
10. Testing Job Runs
11. Managing Project Data

Outcomes for this Module

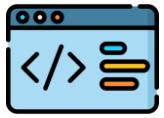
- Describe the main software categories in the Pawsey software stack
- Describe what are a module system and a module and why they are useful
- Use the `module` command to interact with software modules
- List the main `module` subcommands

✓ Prerequisite knowledge:

- ✓ **Bash shell basics**
- ✓ **User Training 02: Logging In**

Watch for These Signs!

Definition of new concepts



Hands-on coding (demo)

Best practices

Exercises and solutions

Warnings (bad practices)

Links to user documentation

Icons made by Freepik from www.flaticon.com

Pawsey

Australian Government

NCRIS National Research Infrastructure for Australia

CSIRO

Curtin University

Murdoch University

Government of Western Australia

ECU Edith Cowan University

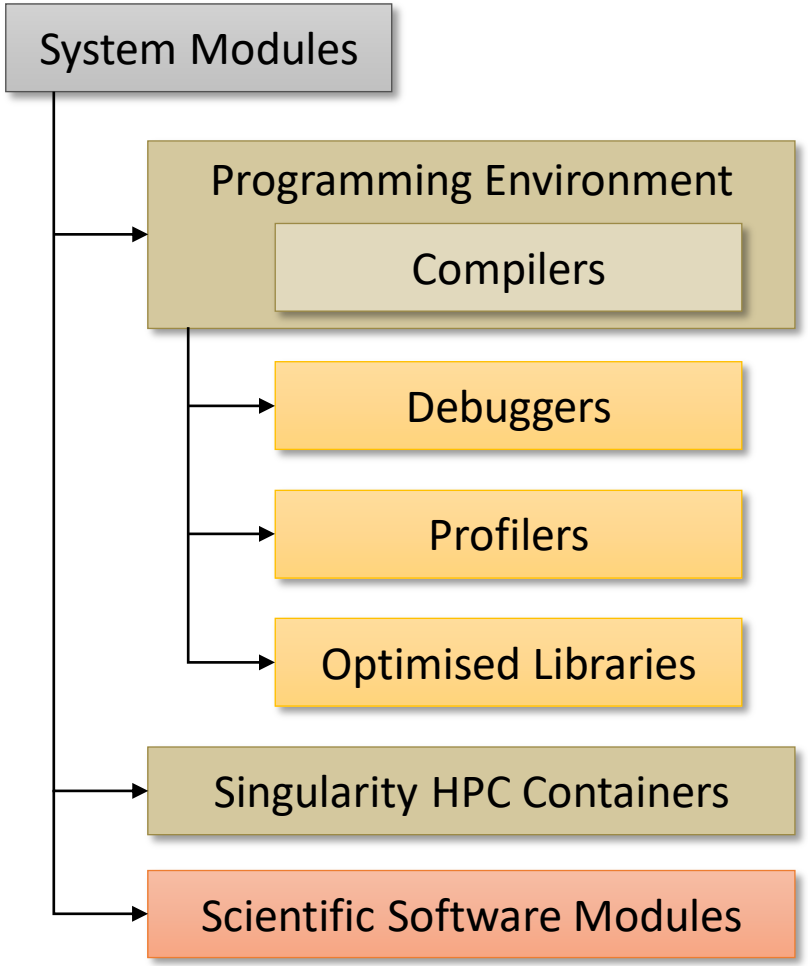
The University of Western Australia

Available Software

Software Stack Overview

Software Stack
A collection of software packages with mutual dependencies. For instance, an end-user application depends on the compiler for building, and may depend on external libraries for specific functionalities.

- HPE/Cray provides system-specific compilers, libraries and tools
- Pawsey supports a selection of popular optimised scientific applications and libraries
  - Located under: `/software/setonix`
- Research teams are responsible for installing and maintaining additional software required for their computational workflows
  - Project directory: `/software/projects/project-id`
  - User directory: `/software/projects/project-id/username`



More details @
- Software Stack

What if You Need to Install Your Own Software?

Recommended options:

- Use the Spack package manager to build software and produce module files with Pawsey-supported compilers if a recipe is available

- Use containers for software with complex build processes or that requires specific versions of libraries

- For Python/R software, use Pip/R package manager/Conda
- Otherwise, manually compile the software

NOTE: in this module we are not discussing the specifics of installing software

More details @

- How to Install Software

Pawsey

Australian Government

NCRIS National Research Infrastructure for Australia

CSIRO

Curtin University

Murdoch University

Government of Western Australia

ECU Edith Cowan University

The University of Western Australia

Software Modules

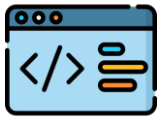
Why Do We Need Modules?

Module System
A software utility to organise and manage multiple software packages (and versions) at once.

Software Module
A specific item within a module system, that modifies the shell environment to make the corresponding software package available.

- All Pawsey supercomputers use the “Lmod” module system
- The module system allows
  - To keep the software stack tidy and organised
  - To activate only those specific software packages (and versions) that are needed by the workflow
- The shell command to interact with software modules is `module`

Browsing Loaded and Available Modules



DEMO on Setonix: `module list` and `module avail`

```
$ module list

Currently Loaded Modules:
  1) craype-x86-milan                          6) pawsey_prgenv      11) cray-libsci/21.08.1.2
  2) libfabric/1.11.0.4.75                     7) gcc/11.2.0         12) PrgEnv-gnu/8.3.2
  3) craype-network-ofi                        8) craype/2.7.14      13) pawsey
  4) perftools-base/21.12.0                    9) cray-dsmml/0.2.2   14) pawseytools
  5) xpmem/2.2.40-2.1_2.56__g3cf3325.shasta  10) cray-mpich/8.1.14
```

Upon login, some modules are already loaded

- Some are provided by HPE/Cray
- Some are provided by Pawsey

Browsing Loaded Modules

Some notable pre-loaded modules

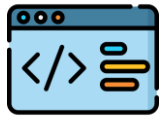
- Programming Environment and associated Compiler, one of these three:
  - GNU: `PrgEnv-gnu` , `gcc` (loaded by default at login)
  - Cray: `PrgEnv-cray` , `cce`
  - AMD: `PrgEnv-aocc` , `aocc`
  - NOTE: each Programming Environment module loads the corresponding Compiler module plus others
- MPI library (for distributed memory parallelism)
  - `cray-mpich`
- CPU-architecture specific optimisations (useful at compile time)
  - `craype-x86-milan`

Browsing Available Modules

- `module avail` without arguments
  - provides all available modules
  - output can be large and difficult to navigate
- `module avail` with sub-string argument
  - searches only for modules whose name contains the sub-string
  - e.g. searching `python` will also show `py-ipython`

```
$ module avail nwchem

--------------- /software/setonix/current/modules/zen3/gcc/11.2.0/applications ---------------
   nwchem/7.0.2
```

Loading and Unloading Modules



DEMO on Setonix: `module load` and `module unload`

- Use `module load` *`name/version`* to make the software available
- On an interactive session, you may need to disable the software when finished
  - Disable the module with `module unload` *`name/version`*
  - This avoids conflicts with other software/versions you need to use
- NOTE: software provided by Pawsey has their software dependencies resolved at build time
  - No spurious effects at runtime from shell environment
  - Improved workflow reproducibility

Always specify both name and version when using module commands.
Providing both module name and version is a good practice for workflow reproducibility. Modules provided by Pawsey forcibly require the version for loading; the load command will fail if the version is skipped.

OUTPUTS: Loading Modules

```
$ which nwchem
which: no nwchem in (/software/projects .. )

$ module load nwchem/7.0.2
$ module list
Currently Loaded Modules:
  1) craype-x86-milan                          8) craype/2.7.14          15) .cray-mpich/8.1.14-m4xyo3l (H)
  2) libfabric/1.11.0.4.75                     9) cray-dsmml/0.2.2       16) netlib-scalapack/2.1.0
  3) craype-network-ofi                       10) cray-mpich/8.1.14      17) openblas/0.3.15
  4) perftools-base/21.12.0                   11) cray-libsci/21.08.1.2  18) python/3.9.7
  5) xpmem/2.2.40-2.1_2.56__g3cf3325.shasta   12) PrgEnv-gnu/8.3.2       19) nwchem/7.0.2
  6) pawsey_prgenv                            13) pawsey
  7) gcc/11.2.0                               14) pawseytools

$ which nwchem
/software/setonix/2022.05/software/cray-sles15-zen3/gcc-11.2.0/nwchem-7.0.2-
dj3hg4oa3ihstagx7thvxlawazjowv7x/bin/nwchem
```

- The nwchem also loads dependency modules (some modules do it, useful for developer purposes, and/or when Python is a dependency)
- Unloading the module (not shown) reverts the environment to the original state

EXERCISE: Browsing, Loading and Unloading Modules

1. How many versions of `gromacs` are available?
2. Search a package string of your choice
3. Which `cmake` version is available at login on Setonix? (tip: use `cmake --version`)
4. Now search for a `cmake` module
5. Try and load it without version. Which error do you get?
6. Now load the module as *name*/*version*
7. What output is displayed when querying the version?
8. Unload the module when done

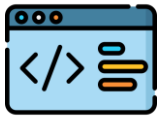
OUTPUTS: Browsing, Loading and Unloading Modules

```
$ module avail gromacs
--------------- /software/setonix/current/modules/zen3/gcc/11.2.0/applications ---------------
   gromacs/2020.4    gromacs/2021.4 (D)

$ cmake --version
cmake version 3.17.0

$ module avail cmake
---------------- /software/setonix/current/modules/zen3/gcc/11.2.0/utilities -----------------
   cmake/3.21.4

$ module load cmake
Lmod has detected the following error:  Default module versions are disabled by your systems
administrator.

         Please load this module as <name>/<version>.

While processing the following module(s):
    Module fullname  Module Filename
    ---------------  ---------------
    cmake/3.21.4     /software/setonix/current/modules/zen3/gcc/11.2.0/utilities/cmake/3.21.4.lua

$ module load cmake/3.21.4
$ cmake --version
cmake version 3.21.4
```

Handling a complex software stack: compiler dependency

Module Hierarchy
Within the Lmod module system, a mechanism to filter available modules based on a small set of “key” modules (e.g. compilers).

- Organising a module stack in hierarchies makes it easier to browse and use
- Pawsey provided modules use a Compiler-based hierarchy
  - Our focus in this session
- HPE/Cray provided modules have a more complex hierarchy, based on Compilers, MPI, Network, HDF5, ..
- The command `module swap` allows to move across branches of the hierarchy
  - In this case, across different compilers (via programming environments)



DEMO on Setonix: `module swap`

OUTPUTS: Compiler hierarchy

```
$ module swap PrgEnv-gnu PrgEnv-cray
Lmod is automatically replacing "gcc/11.2.0" with "cce/13.0.2".
Due to MODULEPATH changes, the following have been reloaded:
  1) cray-mpich/8.1.14

$ module load python/3.9.7
$ which python
/software/setonix/2022.05/software/cray-sles15-zen3/cce-13.0.2/python-3.9.7-
iih546yrigscjz4ph53haawivl62xk6t/bin/python

$ module swap PrgEnv-cray PrgEnv-gnu
Lmod is automatically replacing "cce/13.0.2" with "gcc/11.2.0".
Due to MODULEPATH changes, the following have been reloaded:
  1) cray-mpich/8.1.14     2) python/3.9.7

$ which python
/software/setonix/2022.05/software/cray-sles15-zen3/gcc-11.2.0/python-3.9.7-
otvvw3o7jd2rl5mk5dx6tiiskwhkmklo/bin/python

$ module unload python/3.9.7
```

NOTE: python is an example of module available in both hierarchy branches

OUTPUTS: Compiler hierarchy

NOTE: nwchem is an example of module available in only one hierarchy branch

```
$ module load nwchem/7.0.2
$ which nwchem
/software/setonix/2022.05/software/cray-sles15-zen3/gcc-11.2.0/nwchem-7.0.2-
dj3hg4oa3ihstagx7thvxlawazjowv7x/bin/nwchem

$ module swap PrgEnv-gnu PrgEnv-cray
Lmod is automatically replacing "gcc/11.2.0" with "cce/13.0.2".
Inactive Modules:
  1) .cray-mpich/8.1.14-m4xyo3l     2) netlib-scalapack/2.1.0     3) nwchem/7.0.2     4) python/3.9.7
Due to MODULEPATH changes, the following have been reloaded:
  1) cray-mpich/8.1.14

$ which nwchem
which: no nwchem in (/software/projects .. )

$ module swap PrgEnv-cray PrgEnv-gnu
Lmod is automatically replacing "cce/13.0.2" with "gcc/11.2.0".
Activating Modules:
  1) .cray-mpich/8.1.14-m4xyo3l     2) netlib-scalapack/2.1.0     3) nwchem/7.0.2     4) python/3.9.7
Due to MODULEPATH changes, the following have been reloaded:
  1) cray-mpich/8.1.14

$ module unload nwchem/7.0.2
```

Summary: main `module` subcommands

Command	Description
`module list`	List loaded modules
`module avail`	Show all available modules
`module avail string`	Show available modules with string in name
`module load name/version`	Activate module in shell environment
`module unload name/version`	Deactivate module in shell environment
`module swap module1 module2`	Switch between modules (useful for hierarchies, e.g. Compiler modules)
`module whatis module`	Show module specific details
`module help module`	Show module specific information

Info commands
(content interchangeable, depending on module)

Help on module command itself: `module help`

More details @
- Modules

Modules on Topaz and Garrawarla

- List of pre-loaded modules smaller than Setonix

```
$ module list

Currently Loaded Modules:
  1) cascadelake/1.0   2) pawseytools/1.28   3) slurm/20.11.9   4) gcc/8.3.0
```

- Dependency software is not determined at build time, can change at runtime
  - Modules load default dependency modules
  - Be careful of which other modules you load in addition to the ones you need to use

- Modules hierarchies are not implemented
  - If you have to swap compiler module
    - do it as the first module action, or
    - if you have loaded any other module, unload it first, or
    - start a fresh shell session

Considerations and Policies

- Pawsey provides and supports popular software modules from multiple scientific domains
- Use these modules, if possible, as they are optimised for Setonix hardware

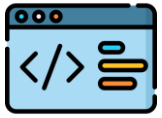
- Pawsey does not pay for licensed domain software, which is left to users
- Bring Your Own License: Pawsey can provide access to a number of HPC applications with restricted license (users have to provide proof of compliance with the license terms)

- Pawsey provides documentation for users to build and install software that is not supported by staff
- Additional assistance in installing unsupported software is discretionary and evaluated case by case, with priority given to PaCER, strategic and uptake projects

More details @
- Software Stack Policies

Summary

- Terms we learnt
  - Software Stack
  - Module System
  - Module
  - Module Hierarchy



- Tasks we learnt
  - Display available and loaded modules: `module avail` and `module list`
  - Load and unload modules: `module load` and `module unload`
  - Swap across compiler hierarchies: `module swap`

- Always specify both name and version when using module commands

Pawsey

Australian Government

NCRIS National Research Infrastructure for Australia

CSIRO

Curtin University

Murdoch University

Government of Western Australia

ECU Edith Cowan University

The University of Western Australia

Getting Help

Getting Help

https://support.pawsey.org.au

Pawsey has extensive User Support Documentation.

Areas covered include:

- System user guides
- Knowledge Base
- Pawsey-supported software list
- Maintenance logs
- Policies and terms of use

For further assistance, contact the help desk, via User Support Portal.

Help us to help you by providing details, such as:

- Which resource
- Error messages
- Location of files
- SLURM job id
- Your username if having login issues
- Never tell us (or anyone) your password!

Become a Pawsey Friend and receive our Newsletter:
https://pawsey.org.au/pawsey-friends/

Q & A Session