Pawsey

Installing and Maintaining Your Software

Setonix Phase 1 Release

20 June 2022. Version 1.02

Australian Government

NCRIS National Research Infrastructure for Australia

CSIRO

Curtin University

Murdoch University

Government of Western Australia

ECU

The University of Western Australia

Focus for this Training

Learning outcomes:

- Discuss considerations and best practices to installing and managing your software stack on Setonix
- Identify impacts and considerations for your current software stack when migrating to Setonix
- Discuss software stack development tools, profilers, debuggers

Core Migration Training Modules:

1. Getting Started with Setonix
2. Supercomputing Filesystems
3. Using Modules and Containers
4. Installing and Maintaining Your Software
5. Submitting and Monitoring Your Job
6. Using Data Throughout the Project Lifecycle

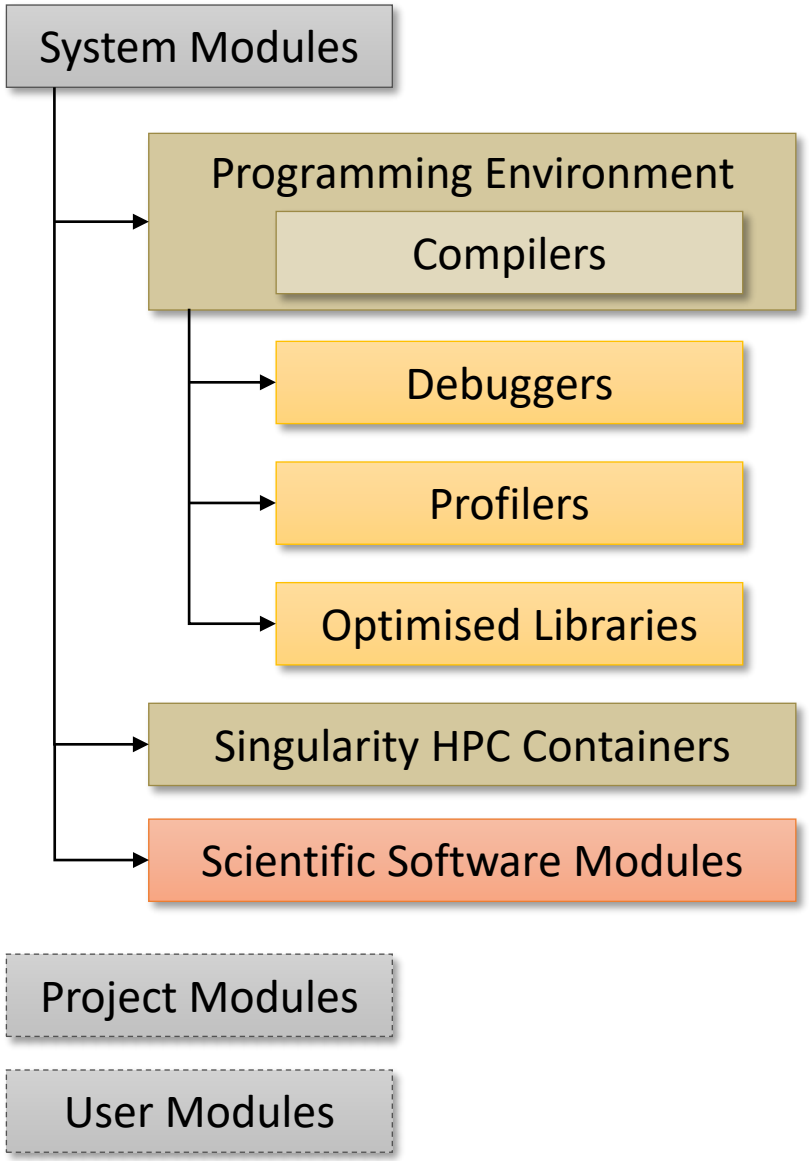
pawsey

Section 1

When and How to Install Software

Software Stack

- Software stack includes vendor-supplied operating systems, compiler suites, high-performance libraries and research-domain-specific packages, and tools such as debuggers and profilers.
- Software stack on Setonix is located on `/software` in 3 levels:
  - **System-wide**: Accessible by all users; access is by the Module system; maintained by the Pawsey staff. Use this software if possible as these are optimised for Setonix hardware. Resides under `/software/setonix`
  - **Project-wide**: Maintained by project members; accessible by all members; resides in `/software/projects/`*`project-id`*`/setonix`
  - **User**: Maintained by user (for example, to download containers). Recommended path: `/software/projects/`*`project-id`*`/`*`username`*`/setonix`
- Details available @
  - Software Stack
  - Software Stack Policies



General Best Practices for Building Your Software Stack

- Project-specific software is located on `/software` filesystem. (Previously `/group` was used for this purpose.)
- For software that we do not provide or for which a specific version/build is not provided, our recommendations are:
    - Use Spack package manager to build software and produce module files with Pawsey-supported compilers if a recipe is available (and is fit-for-purpose).
    - For software with complex build processes or that requires specific versions of libraries use Containers if one exists or one can be easily generated.
    - For Python/R software use Pip/Conda/R package manager (not Spack)
    - Otherwise, manually build software (using tools such as GNU Make, Cmake)

  More information on Spack, containers and package managers are found later in this module.

Review Setonix Modules

- Currently available modules can be searched by using the `module avail` command.
  - Module file names list versions and, when available, build options.
  - For more details @
    - Migration Training Module 3: Using Modules and Containers (Materials, Recordings)
    - Modules documentation
- Check supported software
  Popular Domain-Specific Software
- Software updates are provided via the Pawsey Technical Newsletter
- If the module is not available, refer to subsequent slides for installation

Module is present

```
$ module avail hdf5
-
/opt/cray/pe/lmod/modulefiles/mpi/crayclang/10.0/ofi/1.0/cray-
mpich/8.0 -
 cray-hdf5-parallel/1.12.0.5
- /opt/cray/pe/lmod/modulefiles/compiler/crayclang/10.0 -
 cray-hdf5/1.12.0.5
- /software/setonix/current/modules/zen3/gcc/11.2.0/libraries
-
 adios2/2.7.1-hdf5              hdf5/1.12.1-api-v110
 hdf5/1.10.7-api-v18            hdf5/1.12.1-api-v112
 hdf5/1.10.7-api-v110           hdf5/1.12.1-parallel-api-v18
 hdf5/1.10.7-parallel-api-v18   hdf5/1.12.1-parallel-api-v110
 hdf5/1.10.7-parallel-api-v110 hdf5/1.12.1-parallel-api-v112
 hdf5/1.12.1-api-v18
- /software/setonix/current/containers/modules -
 hpc-python-container/2022.03-hdf5mpi
```

Module does not exist

```
$ module avail vim
No module(s) or extension(s) found!
…

$ module spider vim
Lmod has detected the following error: Unable
to find: “vim”.
```

Use Package Managers to Install Unavailable Software

- **Spack** has an extensive list of recipes for over 5000 packages.
  - Check if the recipe exists and has the desired build using `spack list <name>` ; `spack info <name>`
  - If the fit-for-purpose version is available, build package and generate the module with Spack

- **Containers including SHPC** has over 300 containers designed for deployment on HPC systems.
  - Check if a module for an HPC container is available at https://singularityhub.github.io/singularity-hpc/.
  - If a fit-for-purpose version is available, get the container with SHPC.

- **Pip/Conda** have extensive list of Python packages.
  - Check to see if the package available @ https://pypi.org/ and install it with pip
  - Check to see if the package is available in Conda @ https://anaconda.org or with `conda search -f <name>`, and install it with Conda

- **Conda/R package manager** have extensive lists of R packages.
  - Check if the package is available on the CRAN site @ Available CRAN Packages By Name and install it with R
  - Check if the package is available in Conda and install it

pawsey

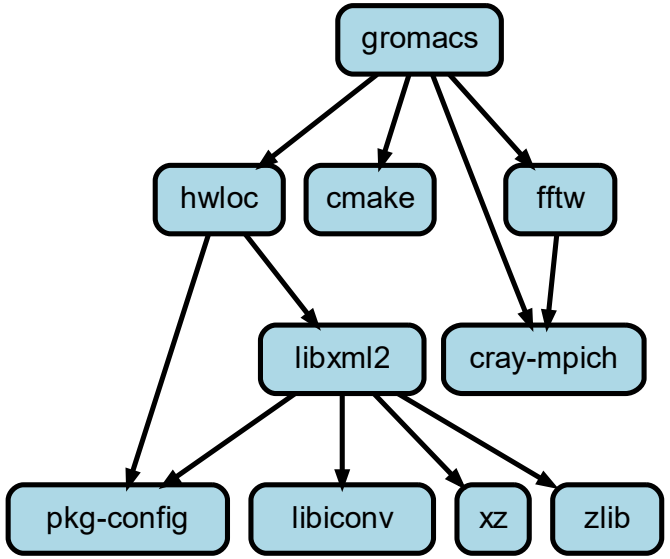
Section 2

Building with Package Managers

What is Spack?

- A package manager designed to deploy software on HPC systems.
- Spack replaces Maali, Pawsey's previous in-house package manager.
- It provides recipes for over 5000 packages.
- Spack has a user-friendly CLI (Command Line Interface).
- It automatically determines dependencies and builds them.

- More details @ https://spack.io/



Spack build of gromacs showing all the dependencies

Building and Using Software with Spack

Build step 1: Find your software in Spack

- Check package info and dependencies:

  `module load spack/0.17.0`

  `spack info` *`package-name`*

  `spack spec` *`package-name`*

Build step 2: Install with Spack

- Install into appropriate directory

  `/software/projects/`*`project-id`*`/`*`username`*`/setonix/software` using:

  `spack install` *`package-name variants`*

- Install software commonly used by the project team in

  `/software/projects/`*`project-id`*`/setonix/software` using a drop-in replacement script:

  `spack_project.sh install` *`package-name variants`*

- More details available @ Spack

```
$ spack info nano
AutotoolsPackage:   nano
Description:    Tiny little text editor
Homepage: https://www.nano-editor.org
Externally Detectable:    False
Tags:    None
Preferred version:
    4.9      …

$ spack spec nano
Input spec
--------------------------------
nano
Concretized
--------------------------------
nano@4.9%gcc@10.3.0 cflags="-O3" cppflags="-O3" fflags="-
O3 -fallow-argument-mismatch" arch=cray-sles15-zen2
    ^ncurses@6.2%gcc@10.3.0 cflags="-O3" cppflags="-O3"
fflags="-O3 -fallow-argument-mismatch" ~symlinks+termlib
abi=none arch=cray-sles15-zen2
        ^pkg-config@0.29.2%gcc@10.3.0 cflags="-O3"
cppflags="-O3" fflags="-O3 -fallow-argument-mismatch"
+internal_glib arch=cray-sles15-zen2

$ spack install nano
```

Building and Using Software with Spack (continued)

Build step 3: Generate module with Spack

- To generate modules:

  `spack module lmod refresh -y` *`package`*

Use the generated module

- Browse modules
  - `module avail`
- To use the installed software:

  `module load` *`package/version`*

- More details available @ Spack

```
$ spack install gsl@2.7
==> Installing gsl-2.7-vytnvxqt3zpzhkz6qqy6a2j36huxfh7r
==> No binary for gsl-2.7-vytnvxqt3zpzhkz6qqy6a2j36huxfh7r
found: installing from source
…
gsl: Successfully installed gsl-2.7-
vytnvxqt3zpzhkz6qqy6a2j36huxfh7r
  Fetch: 4.00s.  Build: 1m 1.18s.  Total: 1m 5.18s.

$ spack module lmod refresh -y gsl
==> Regenerating lmod module files

$ module avail gsl
 -
/software/projects/<projectid>/<user>/setonix/modules/zen3/g
cc/11.2.0 -
  gsl/2.7-vytnvxq/module

$ module load gsl/2.7-vytnvxq
```

How to Pull and Use Containers the Traditional Way

- Singularity is the container runtime available.
- The command line interface is unchanged from previous Pawsey supercomputers, and is available to:
  - Pull (download) containers
  - Execute commands in containers
- More details available @ Containers

```
$ module load singularity/3.8.6
$ singularity pull
  docker://quay.io/biocontainers/blast:2.9.0--
  pl526h3066fca_4
INFO:    Converting OCI blobs to SIF format
INFO:    Starting build...
..
INFO:    Creating SIF file...
INFO:    Build complete: blast_2.9.0--
  pl526h3066fca_4.sif
$ singularity exec blast_2.9.0--pl526h3066fca_4.sif
  blastp -help
```

What are SHPC and container modules?

- A container module allows a containerised application to be run using the same commands as its non-containerised counterpart (through the use of command wrappers — shell functions or scripts).
- Singularity Registry HPC (SHPC) automates the installation of containers as container modules.
- SHPC library provides a growing set of recipes (currently 300+):
  - Supports Singularity, Docker and Podman
  - Supports Lmod and Environment Modules
  - If needed, writing new recipes is easy
  - Library of containers available @ https://singularityhub.github.io/singularity-hpc/

- More details @ https://singularity-hpc.readthedocs.io/en/latest/

CONTAINER	DESCRIPTION
quay.io/biocontainers/bedtools	Bedtools is the swiss army knife for genome arithmetic.
quay.io/biocontainers/blast	BLAST finds regions of similarity between biological sequences.
quay.io/biocontainers/bowtie2	Bowtie 2 is an ultrafast and memory-efficient tool for aligning sequencing reads to long reference sequences.

How to Install and Use Container Modules with SHPC

Step 1: Find your software in SHPC.

```
$ module load shpc/0.0.53
$ shpc show -f package-name
```

Step 2: Install your software with SHPC.

will use appropriate directory /software/projects/*project-id*/*username*/setonix/containers

```
$ shpc install container-name:version
```

Step 3: Use your software.

```
$ module load container-name/version
$ package-command
```

- More details available @
  - Containers
  - SHPC (Singularity Registry HPC)

How to Install Python Packages with Pip and Conda

Step 1: Find your Python software.

- Check to see if the package is available.
  - For Python, use pipy or anaconda websites.

Step 2: Install Python software.

- Install into appropriate directory
  `/software/projects/project-id/username/setonix/python`
  - For Pip:

    `pip install --user package-name`
  - For Conda:

    `conda install package-name`
  - ⚠ Do not use Spack to install Python packages

More details @

- Installing Python Packages
- Conda and Reproducible Installations

Option 1: Installation with Pip

```
$ pip install --user seaborn
$ pip show seaborn
Name: seaborn
Version: 0.11.2
Summary: seaborn: statistical data
visualization
Home-page: https://seaborn.pydata.org
Author: Michael Waskom
Author-email: mwaskom@gmail.com
License: BSD (3-clause)
Location: /software/projects/project-id/user-
name/py-env/lib/python3.8/site-packages
Requires: matplotlib, numpy, pandas, scipy
Required-by:
```

Option 2: Installation with Conda

```
$ conda install seaborn
```

How to Install R Packages with R Package Manager and Conda

Step 1: Find your R software.

- Check to see if the package is available.
  - Use CRAN or anaconda websites

Step 2: Install R software.

- Install into appropriate directory
  `/software/projects/project-id/username/setonix/r/%v`
- For the R internal package manager:

```
> install.packages('package-name')
```

- For Conda:

```
conda install package-name
```

- More details available @
  - Installing R Packages
  - Conda and Reproducible Installations

Option 1: Installation with R

```
$ R #load R interpreter
> install.packages('bpa') # install basic
pattern analysis package
```

Option 2: Installation with Conda

```
$ conda install r-bpa
```

⚠ **Note:** Do not use Spack to install R packages.

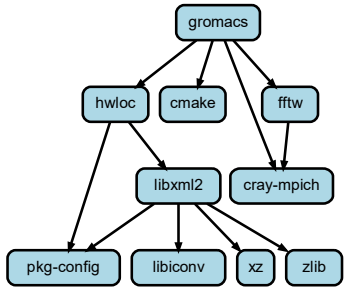
pawsey

Section 3

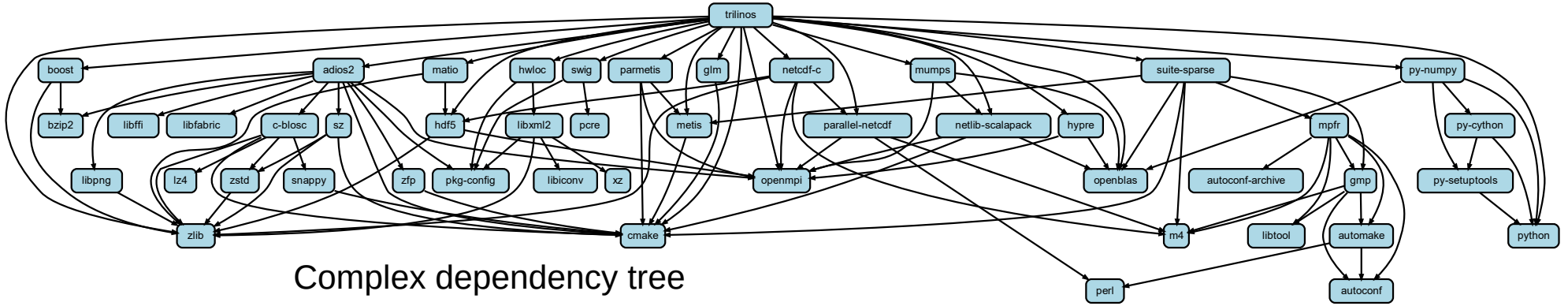
Manual Installation of Software

Decide How to Build Your Software's Dependencies

1. Determine the dependencies of your software and see if they are already available on Setonix.
2. If dependencies are not available on Setonix, check their complexity and what versions are acceptable.
   - For runtime applications with **complex** dependencies or specific versions, consider using **containers**.
     ⚠ This can restrict optimisation but otherwise is the ideal way to handle complex dependencies.
   - If the dependencies are available with **Spack**, **Pip** or **Conda**, use the appropriate package manager.
   - Otherwise, build the dependencies manually.



Simple dependency tree



Complex dependency tree

Compiling in HPE/Cray EX Programming Environment

- C, C++ and Fortran languages are supported.
- Three main programming environments are provided by the module system:
  - PrgEnv-gnu: GNU C, C++ and Fortran compilers (available by default)
  - PrgEnv-cray: Cray Clang C and C++ compilers and Cray Fortran compiler
  - PrgEnv-aocc: AMD optimised Clang C and C++ compilers
- All accessed through the Cray compiler wrappers (similar to Magnus):
  - cc provides C compiler
  - CC provides C++ compiler
  - ftn provides Fortran compiler

> ⚠ **IMPORTANT:** Always use the above wrappers to ensure correct compilation on Setonix.

- The PrgEnv-intel programming environment providing the Intel compilers is no longer available.
- Compilation instructions and optimisation flags can be found @ Compiler Optimisation Levels.

Compilers: Key Changes

Existing compiler	New compiler	Benefits OR Impacts OR Considerations….
PrgEnv-gnu (C/C++, Fortran)	PrgEnv-gnu (C/C++, Fortran)	• Setonix supports the newer GNU compilers with optimisations for AMD Milan architecture. The PrgEnv-gnu environment is available by default. • ⚠ The OpenMP flag is not activated by default. • ⚠ Optimisations are not enabled by default on Setonix (equivalent to -O0). • Pre-Setonix systems had optimisations enabled by default with OpenMP dependent on the compiler version (equivalent to compiling with –O2 –fopenmp).
PrgEnv-cray (C/C++)	PrgEnv-cray (C/C++)	• Cray C/C++ is now built on LLVM compilers. • Compiler flags are completely different (now follow clang/clang++ conventions). • ⚠ The OpenMP flag is not activated by default. • ⚠ Optimisations are not enabled by default on Setonix (i.e., -O0). • Pre-Setonix systems had OpenMP and optimisations enabled by default (equivalent to compiling with –O3 –homp).
PrgEnv-cray (Fortran)	PrgEnv-cray (Fortran)	• Cray Fortran is updated, but is not based on LLVM flang compiler. • Compiler flags remain the same. • Optimisations enabled and OpenMP flag activated by default. • **Recommendation:** Use explicit flags and do not rely on default settings.

Compilers: Key Changes (continued)

Existing compiler	New compiler	Benefits OR Impacts OR Considerations….
PrgEnv-intel (C/C++)	⚠ PrgEnv-aocc (C/C++)	• **Setonix does not support Intel.** Setonix supports AMD-optimised compilers with associated libraries, which are based on LLVM clang. • Full Fortran is not currently provided. Future releases will add AMD-optimised Fortran through flang. • Compiler flags are the same as standard LLVM Clang compiler flags. • The default optimisation is –O0 and the OpenMP flag is not activated.
PrgEnv-intel (Fortran)	⚠ (none)	• We recommend migrating to GNU Fortran.

Using Other Programming Environments

Python & R

- Use Pawsey-provided python and R modules at this time.
  - These modules provide core functionality.
  - Use a package manager to install additional packages.

⚠ **Recommendation:** Don't use Cray-provided cray-python and cray-r modules.

Other Languages

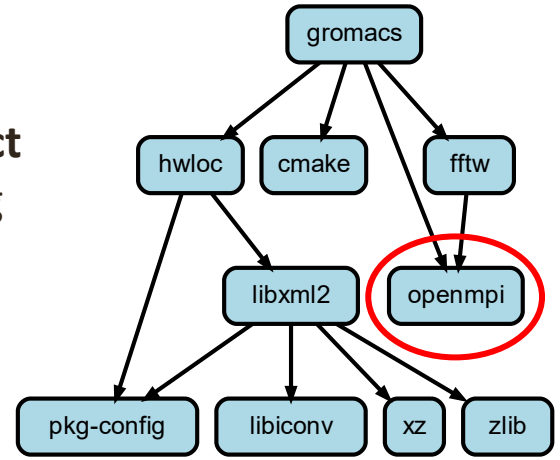
- The compiler or interpreter is provided via modules for:
  - Rust, Golang, Java, Ruby, Perl
- Any other languages will require bootstrap builds from source.

Building Parallel Codes

Building MPI-enabled codes

- HPE/Cray provides an optimised MPI library through the cray-mpich module, which is loaded by default.
- The cray-mpich module must be used for performance and multi-node communication.

Example of an **incorrect** build of Gromacs using OpenMPI



⚠ **Note:** User-built MPI, unless precisely configured, will not provide multi-node capability or performance. It is not recommended.

Building OpenMP-enabled codes

- Pre-Setonix, Cray compiler wrappers enabled OpenMP by default.
- On Setonix, Cray Clang-based C/C++ compilers do NOT enable OpenMP by default.
- Compiler flags enabling OpenMP have also changed on Setonix:
  - PrgEnv-gnu (C/C++/Fortran): -fopenmp
  - PrgEnv-cray (C/C++): -fopenmp
  - PrgEnv-cray (Fortran): -qopenmp
  - PrgEnv-aocc (C/C++): -fopenmp

Building with Optimised API Libraries on Setonix

Setonix has several optimised math libraries. We recommend using these when building software.

Modules	Provided by	Description	Key Change
FFTW3	Pawsey	`fftw/3.3.9`: optimised	(minor)
FFTW3	AMD	`amdfftw/3.0`: amd optimized	New Milan optimized library
FFTW2	Pawsey	`fftw/2.1.5`: Older FFTW API	(minor)
BLAS LAPACK SCALAPACK	Pawsey	`openblas/0.3.15`, `netlib-lapack/3.9.1`, `netlib-scalapack/2.1.0`	New optimized library
BLAS LAPACK SCALAPACK SPARSE	AMD	`amdblis/3.0`, `amdlibflame/3.0`, `amdscalapack/3.0`, `aocl-sparse/3.0`: amd optimised	New Milan optimized library
LIBM	AMD	`amdlibm/3.0`: amd optimised	New Milan optimized library

Compiling Source with Configuration Systems

1. Determine if code uses Autoconf, CMake or another configuration system.
2. Check compilation options and ensure cray compiler wrappers are used.
3. Also ensure installation paths are appropriate using MYSOFTWARE environment variable (MYSOFTWARE=/software/projects/*project-id*/*username*)

Example using Configuration Systems:

- CMake:

```
cmake -DCMAKE_INSTALL_PREFIX=${MYSOFTWARE}/some-desired-path
-DCMAKE_C_COMPILER=cc -DCMAKE_CXX_COMPILER=CC -DCMAKE_Fortran_COMPILER=ftn
other-options dir-to-CmakeLists.txt
```

- Autoconf

```
./configure CC=cc CXX=CC F77=ftn F90=ftn --prefix=${MYSOFTWARE}/some-
desired-path other-options
```

Compiling Source Without Configuration Systems

1. Determine compilation flags required and dependencies.
2. Copy source to the appropriate directory.

Example:

```
cd ${MYSOFTWARE}/some-desired-path
```

- Makefile:

```
make CC=cc CXX=CC F90=ftn FC=ftn CFLAGS=<optimisation-flags> CXXFLAGS=<…>
F90FLAGS=<…> FCFLAGS=<…>
```

- Explicit compilation:

```
cc optimisation-flags source.c –c software-name extra-library-flags
CC optimisation-flags source.cpp –c software-name extra-library-flags
ftn optimisation-flags source.f –c software-name extra-library-flags
```

Using rpaths for Libraries and Dependencies

- Libraries are loaded at runtime, and rpaths indicate the paths in which to search for runtime libraries.
- Rpaths ensure reproducible behaviour when loading libraries.

Common Practice Pre-Setonix:

```
cc optimisation-flags source.c –c software-name -Lpath-to-libs -llibraries
```

Best Practice on Setonix:

```
cc optimisation-flags source.c –c software-name -Lpath-to-libs -llibraries
-Wl,rpath=path-to-libs
```

- For more details on dynamic loading and rpaths see:
  - Compiling
  - Spack

Best Practices for Manually Building Software

- Use the compiler wrappers (`cc`, `CC`, `ftn`).
- Explicitly specify compiler optimisation flags.
- Ensure the `cray-x86-milan` architecture module is loaded.
- Compile on the same type of nodes where the software will run.
- To keep your software organised, keep the manual builds under
  - Software: `/software/projects/`*`project-id`*`/`*`username`*`/manual/software`
  - Modulefiles: `/software/projects/`*`project-id`*`/`*`username`*`/manual/modules`

- More details @
  - Compiling
  - How to manually build software

pawsey

Section 4

Software Development Tools

Developing HPC Codes: Debugging and Profiling

To assist development of HPC codes, Pawsey provides a variety of tools for debugging and profiling.
More details @ Developing Codes.

Debuggers

- Control the execution of a code to identify the location and cause of a bug in the code.
- Pawsey provides debuggers as modules:
  - gdb4hpc (`gdb4hpc`)
  - ARM DDT (`arm-forge`)
- More details @ Debugging.

Profilers

- Analyse code while it is running to provide insight and understanding of its performance, in order to optimise it.
- Pawsey provides several profilers as modules:
  - Cray Tools (`PrgEnv-cray`, `perftools-base`, `cray-stat`)
  - ARM Map (`arm-forge`)
  - ARM Performance Reports (`arm-forge`)
- More details @ Profiling.

How do I get help?

Migration Documentation & Migration Guides

- Setonix Migration Guide
- Setonix User Guide
- Supercomputing Documentation

Migration Training Materials & Video Recordings

- Upcoming Migration Training
- Recordings: Pawsey YouTube Setonix Migration Phase 1 Playlist
- Materials: Setonix Migration Training Materials (PDFs)

Help Desk

- Help Desk
- Email: help@pawsey.org.au

pawsey

Thank you for attending!

Please complete this short survey:

https://www.surveymonkey.com/r/Y3YFYHQ