HPC Software – Modules, Libraries & Software

Introduction to supercomputing at JSC

November 12, 2024 | R. Partzsch | JSC

Member of the Helmholtz Association

JÜLICH Forschungszentrum

Outline

Modular Programming

Software Implementation & Libraries

A **library** is a collection of resources.
In computer science: configuration data, documentation, help data, message templates, pre-written code and subroutines, classes, values or type specifications

- **Don't reinvent the wheel**: Recoding is time-consuming and error-prone
- **Best practice for own code**: Design interfaces and do different implementation separately (ideally separate files)

Modular Programming

Compiling & Linking

Example:

- Code decomposition into an executable and an outsourced subprogram *(C++: Link object files for different modules)*

```
g++ -g -c -o main.o main.cpp
g++ -g -c -o alibrary.o alibrary.cpp
g++ -g main.o alibrary.o -o main
```

- Copy *.o* and *.h* to separate directories (e.g. */base/lib/alibrary.o*) and add
  - The **-L** option for the path to the library's object, the object code using **-l**

```
g++ -g -L/base/lib -o main main.o -lalibrary
```

  - The **-I** flag for include directories

Modular Programming

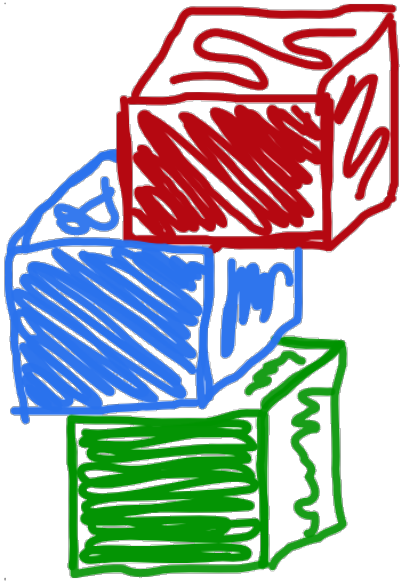
Compiling & Linking

- For libraries that are not in standard directories, you need to set **-I**/base/include and **-L**/base/lib for compiling/linking
- Or define linux **environment variables**:
  - You either enter the *export*-commands on the linux prompt before compiling, or, copy them into to the .bashrc file in your home folder
- Usually you do not need **-I** or **-L** for libraries accessed using the *module load* command on our supercomputers

Modular Programming

Installing Libraries from Source

- Library is not in the software module stack
- Common installation procedure:
  - make
    ./configure –prefix=base
    make; make install
  - cmake
    cmake -DCMAKE_INSTALL_PREFIX=base
    make; make install
  - Choose *base* as a directory you have write permissions



easybuild

Preinstalled HPC Software Packages

EasyBuild

Preinstalled HPC Software Packages

Navigating Modules

- Tools are available through **"modules"**
  - Allows to easily manage different versions of programs
  - Works by dynamic modification of a user's environment
- Module setup based on **EasyBuild** and **lmod**
  - **"Staged"**, hierarchical setup
  - Automatically manages dependencies via **"toolchains"**
- Consistent setup on JURECA, JUSUF and JUWELS (Cluster & Booster)

Preinstalled HPC Software Packages

Toolchains: Underlying Compiler and MPI Libraries

- **Base:** GCCcore
- **Compiler:**
  - Intel compiler
  - GNU compiler
  - NVIDIA (CUDA) compiler
- **MPI libraries:**
  - ParastationMPI
  - Intel MPI
  - OpenMPI
- **Math libraries:** e.g. FlexiBLAS

Preinstalled HPC Software Packages

Toolchains: Underlying Compiler and MPI Libraries

Current software stage is 2024, stage 2025 is under construction and can already be used

- **Base:** GCCcore *(12.3.0)*
- **Compiler:**
  - Intel compiler *(Intel 2023.2.1)*
  - GNU compiler *(GCC 12.3.0)*
  - NVIDIA (CUDA) compiler *(NVHPC 23.7)*
- **MPI libraries:**
  - ParastationMPI *(ParaStationMPI 5.9.2-1)*
  - Intel MPI *(Intel MPI 2021.10.0)*
  - OpenMPI *(OpenMPI 4.1.5)*
- **Math libraries:** e.g. FlexiBLAS *(MKL 2023.2.0, BLIS 0.9.0, OpenBLAS 0.3.23)*

Preinstalled HPC Software Packages

Modules Environment

Available Compiler/MPI Combinations in Stages/2024

Compiler	MPI	Cuda available
GCC	ParaStationMPI	yes
GCC	OpenMPI	yes
NVHPC	ParaStationMPI	yes
NVHPC	OpenMPI	yes
Intel	ParaStationMPI	yes
Intel	IntelMPI	no
Intel	OpenMPI	yes

Preinstalled HPC Software Packages

Toolchains: Underlying Compiler and MPI Libraries

The most important module command

- module load <*name*> or ml <*name*>

- **GCCcore** is preloaded, which enables a lot of base software
- For HPC software you have to load a **compiler**, to expand the module tree
  e.g. `ml GCC` *(Default is 12.3.0)*
- Then you load an **MPI** version
  e.g. `ml ParaStationMPI` *(Default is 5.9.2-1)*
- Then you can load any other **math or application package**
  e.g. `ml PETSc/3.20.0`

Preinstalled HPC Software Packages

Toolchains: Underlying Compiler and MPI Libraries

The most important module command

- module load <*name*> or ml <*name*>

Hierarchical modules

- **GCCcore** is preloaded, which enables a lot of base software
- For HPC software you have to load a **compiler**, to expand the module tree
  e.g. `ml GCC` *(Default is 12.3.0)*
- Then you load an **MPI** version
  e.g. `ml ParaStationMPI` *(Default is 5.9.2-1)*
- Then you can load any other **math or application package**
  e.g. `ml PETSc/3.20.0`

Preinstalled HPC Software Packages

Toolchains: Underlying Compiler and MPI Libraries

The most important module command

- module load <*name*> or ml <*name*>

Hierarchical modules

- **GCCcore** is preloaded, which enables a lot of base software
- For HPC software you have to load a **compiler**, to expand the module tree
  e.g. `ml GCC` *(Default is 12.3.0)*
- Then you load an **MPI** version
  e.g. `ml ParaStationMPI` *(Default is 5.9.2-1)*
- Then you can load any other **math or application package**
  e.g. `ml PETSc/3.20.0`

Preinstalled HPC Software Packages

Toolchains: Underlying Compiler and MPI Libraries

The most important module command

- module load <*name*> or ml <*name*>

Hierarchical modules

- **GCCcore** is preloaded, which enables a lot of base software
- For HPC software you have to load a **compiler**, to expand the module tree
  e.g. `ml GCC` *(Default is 12.3.0)*
- Then you load an **MPI** version
  e.g. `ml ParaStationMPI` *(Default is 5.9.2-1)*
- Then you can load any other **math or application package**
  e.g. ml PETSc/3.20.0

Preinstalled HPC Software Packages

Toolchains: Underlying Compiler and MPI Libraries

The most important module command

- module load <*name*> or ml <*name*>

Hierarchical modules

- **GCCcore** is preloaded, which enables a lot of base software
- For HPC software you have to load a **compiler**, to expand the module tree
  e.g. `ml GCC` *(Default is 12.3.0)*
- Then you load an **MPI** version
  e.g. `ml ParaStationMPI` *(Default is 5.9.2-1)*
- Then you can load any other **math or application package**
  e.g. `ml PETSc/3.20.0`

Preinstalled HPC Software Packages

Modules Environment

- `ml spider` *name* shows whether a library is available in the current stage and in which versions

  e.g. module spider petsc

  ⇒ PETSc/3.18.5,

  PETSc/3.20.0, ...

- `ml spider` *name/version* shows which environment you have to load before you can load that version

  e.g. module spider PETSc/3.20.0

  ⇒ Stages/2024 + GCC/12.3.0 + ParaStationMPI/5.9.2-1, ...

- Some packages are hidden. To see them use

  `ml -show-hidden spider` *name*

Preinstalled HPC Software Packages

Modules Environment

- `ml spider` *name* shows whether a library is available in the current stage and in which versions
  e.g. module spider petsc
  ⇒ *PETSc/3.18.5, PETSc/3.20.0, ...*
- `ml spider` *name*/*version* shows which environment you have to load before you can load that version
  e.g. module spider PETSc/3.20.0
  ⇒ *Stages/2024 + GCC/12.3.0 + ParaStationMPI/5.9.2-1, ...*
- Some packages are hidden. To see them use
  `ml -show-hidden spider` *name*

Preinstalled HPC Software Packages

Modules Environment

- `ml spider` *name* shows whether a library is available in the current stage and in which versions
  e.g. module spider petsc
  ⇒ *PETSc/3.18.5, PETSc/3.20.0, ...*
- `ml spider` *name*/*version* shows which environment you have to load before you can load that version
  e.g. module spider PETSc/3.20.0
  ⇒ *Stages/2024 + GCC/12.3.0 + ParaStationMPI/5.9.2-1, ...*
- Some packages are hidden. To see them use
  `ml -show-hidden spider` *name*

Preinstalled HPC Software Packages

Modules Environment

Stages

- The whole software stack of JURECA, JUSUF, JUWELS Cluster and Booster will be updated regularly
- Current stage is 2024 (*2025 is default soon*)
- To check availability in other stages first type

```
ml use $OTHERSTAGES
```

Preinstalled HPC Software Packages

Mathematical Libraries: FlexiBlas

FlexiBlas: wrapper library, includes MKL, BLIS, OpenBLAS

- Linear Algebra Packages (LAPACK, ScaLAPACK, ...)
- Iterative Sparse Solvers, Trust Region Solver
- Vector Math Library
- Vector Statistical Library
- Fourier Transform Functions
- Trigonometric Transform Functions
- GMP routines, Poisson Library, ...

Preinstalled HPC software packages

Mathematical Libraries: Sequential Packages

Public domain Libraries

- LAPACK (Linear Algebra PACKage)
- ARPACK (ARnoldi PACKage)
- GSL (Gnu Scientific Library)
- GMP (Gnu Multiple Precision Arithmetic Library)
- METIS (Serial Graph Partitioning and Fill-reducing Matrix Ordering)

Preinstalled HPC Software Packages

Mathematical Libraries: Parallel Packages

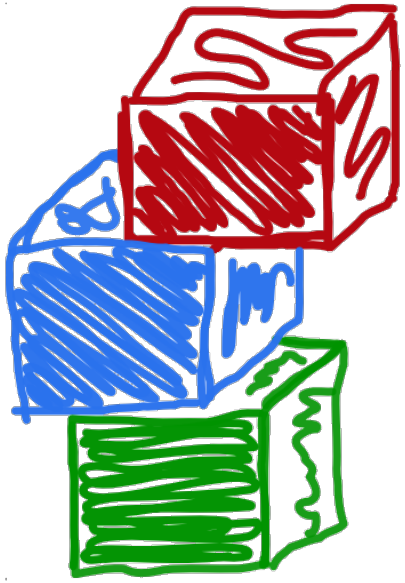
- ScaLAPACK (Scalable Linear Algebra PACKage)
- ELPA (Eigenvalue SoLvers for Petaflop-Applications)
- FFTW (Fastest Fourier Transform of the West)
- MUMPS (MUltifrontal Massively Parallel sparse direct Solver)
- ParMETIS (Parallel Graph Partitioning)
- SCOTCH (Parallel Graph Partitioning)
- Hypre (high performance preconditioners)
- ARPACK (Parallel ARPACK)
- SPRNG (Scalable Parallel Random Number Generator)
- SUNDIALS (SUite of Nonlinear and DIfferential/ALgebraic equation Solvers)

GPU Library

- MAGMA, Matrix Algebra on GPU and Multicore Architectures

Parallel Systems

- PETSc, toolkit for partial differential equations
  - PETSc for Python (petsc4py)
- SLEPc Scalable Library for Eigenvalue Problem Computations
  Extension to PETSc for the computation of eigenvalues and eigenvectors
- MATLAB for High-Performance Computing
  - `https://www.fz-juelich.de/en/ias/jsc/services/user-support/software-tools/matlab`



easybuild

User Installations at JSC

EasyBuild

EasyBuild

How it is used at JSC

- Used by the software team to create software stacks since 2014
  - **Install in Production stage:**
    ml Stages/2024
    ml Developers
    eb packages-1.2.3.eb
- Allow users to install software on-top of the available modules
  - **Install in User space:**
    ml Stages/2024
    export USERINSTALLATIONS=/p/project/yourproject/user
    ml UserInstallations
    eb packages-1.2.3.eb

Excerpt from PETSc easyconfig

https://github.com/easybuilders/JSC/blob/2024/Golden_Repo/p/PETSc/PETSc-3.20.0-gpsfbf-2023a.eb

```
04 name = "PETSc"
05 version = "3.20.0" # PETSc/3.20.0

#see Golden_Repo: g=GCC, ps=ParaStationMPI, fbf=FlexiBlas
18 toolchain = 'name': 'gpsfbf', 'version': '2023a'

# where the installation files are downloaded
23 source_urls = ['https://web.cels.anl.gov/...
24 sources = ['petsc-%s.tar.gz' % version]

#deps that are already present at stage 2023a
35 dependencies = [('METIS', '5.1.0'), ...

53 configopts = '–with-large-file-io'
```

Create new PETSc installation

PETSc-3.20.0-gpsfbf-2023a-myVersion.eb

```
04 name = "PETSc"
05 version = "3.20.0"
06 versionsuffix = '-myVersion' # PETSc/3.20.0-myVersion
18 toolchain = 'name': 'gpsfbf', 'version': '2023a'

# where the installation files are downloaded
23 source_urls = ['https://web.cels.anl.gov/...
24 sources = ['petsc-%s.tar.gz' % version]

#deps that are already present at stage 2023a
35 dependencies = [('METIS', '5.1.0'), ...

53 configopts = '–with-large-file-io'
54 configopts += '–something_fancy '
```

Create and run new PETSc installation

PETSc-3.20.0-gpsfbf-2023a-myVersion.eb

- **Install in User space:**
  ml Stages/2024
  export USERINSTALLATIONS=/p/project/yourproject/user
  ml UserInstallations
  eb PETSc-3.20.0-gpsfbf-2023a-myVersion.eb
- **Load and Use of new Software:**
  export USERINSTALLATIONS=/p/project/yourproject/user
  ml Stages/2024 && ml GCC && ml ParaStationMPI
  ml PETSc/3.20.0-myVersion

EasyBuild: Further Links

Documentation:

- EasyBuild Documentation
  `https://docs.easybuild.io`
- EasyBuild Tutorial
  `https://easybuild.io/tutorial`

Where do I find easyconfigs?

- JSC repository
  `https://gitlab.jsc.fz-juelich.de/software-team/easybuild`
- JSC public mirror
  `https://github.com/easybuilders/JSC`
- Upstream
  `https://github.com/easybuilders/easybuild-easyconfigs`

Containers

Apptainer

Containers

What they provide

- Containers package up pieces of software in a way that is **portable and reproducible**, they ...
  - manage different versions of programs
  - are more lightweight than virtual machines
  - provide the ability to build, ship, and run applications
- Some examples are Docker, Shifter, and **Apptainer/Singularity**
- They typically use so-called **"images"**
  - contain a file system including a minimal operating-system, the application, and some metadata

Apptainer Containers

First steps

- We provide an up-to-date version of **Apptainer**
- To be granted access to the container runtime, you have to go to our user portal **JuDoor**
- JSC provides a build system that can build images on behalf of the user, based on a Docker- or Singularity-file
- For further information see `https://apps.fz-juelich.de/jsc/hps/jureca/container-runtime.html`

Apptainer Containers

First steps

Download an image:

- Use the pull command to download pre-built images from an external resource like Docker Hub

```
apptainer pull centos.sif docker://centos:7
```

Call an executable:

- The shell command allows you to spawn a new shell within your container and interact with it

```
srun -N1 -p <part> –gres gpu:1 –pty apptainer shell –nv centos.sif
```

- To Slurm, Singularity is just another executable and can be called as such

Further information and JSC-people

http://www.fz-juelich.de/ias/jsc/jureca
http://www.fz-juelich.de/ias/jsc/juwels
http://www.fz-juelich.de/ias/jsc/jusuf

r.partzsch@fz-juelich.de

Supercomputer support:
sc@fz-juelich.de

HPC Software – Modules, Libraries & Software

Introduction to supercomputing at JSC

November 12, 2024 | R. Partzsch | JSC

Member of the Helmholtz Association

JÜLICH Forschungszentrum