Lmod: A New Environment Module System

As we said in previous sections, TeideHPC is an environment shared by hundreds of users, so the way to provide multiple applications in their multiple versions to a multitude of users in a distributed computing environment is not the same as on a personal computer or your own server.

The tool for managing these environments is called ***Environment Modules*** and ***Lmod*** is an implementation of Environment Modules.

Lmod is a Lua based module system that easily handles the MODULEPATH using **Hierarchical module naming scheme** and this provides a convenient way to dynamically change the users' environment through modulefiles. This includes easily adding or removing directories to the PATH environment variable.

- https://lmod.readthedocs.io/en/latest/

What is Hierarchical module naming scheme?

Flat vs hierarchical

The default module naming scheme used until now is an example of regular "flat" module naming scheme, which is characterized by:

- all module files are directly available for loading;
- each module name uniquely identifies a particular installation;

```
$ module av

------------------------------------------ /opt/envhpc/modulefiles/.rhel6 ------------------------------------------
admixtools/7.0.2/gcc           hdf5/1.8.16/icc               perl/5.34.0/gcc
admixture/1.3.0                hdf5-parallel/1.12.1/intelmpi   picard/2.25.6
angsd/0.935/gcc                hdf5-parallel/1.8.13/intelmpi   platypus/0.8.1/gcc
....
```

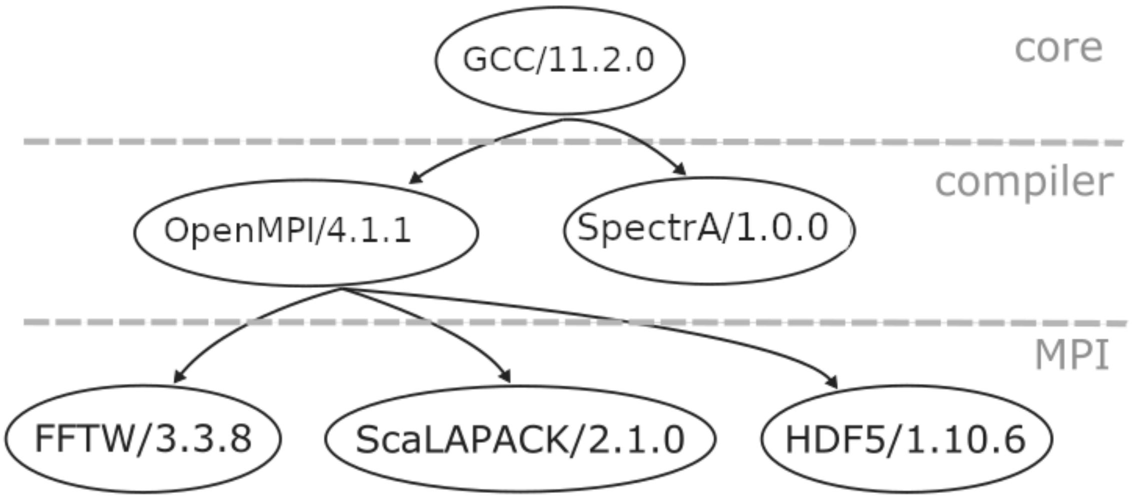
In contrast, a **Hierarchical module naming scheme consists of a hierarchy of module files**.

The typical module hierarchy has 3 levels:

- **CORE** level: where module files for software that was installed using the system;

- **COMPILER** level: where module files for software that was installed using a certein compiler-only;
- **MPI** level: which houses module files for software that was installed using (at least) a certain compiler and MPI component;

Here is a simple example of such a 3-level module hierarchy:



In this example the core level only includes a single module GCC/11.2.0, while the compiler level includes two modules: OpenMPI/4.1.1 and SpectrA/1.0.0. In the MPI level, three modules are available: one for FFTW, one for ScaLAPACK, and one for HDF5.

As you will notice, at **every level we select the module of the layer we are entering**. At core level we select our compiler. When in the compiler level we select our MPI implementation, and within the MPI level we select our software and at MPI leve we select software.

Some modules in the top level of the hierarchy act as a **"gateway"** to modules in the next level below. **To make additional modules available for loading one of these gateway modules has to be loaded.** in the case of the previous image, to load OpenMPI/4.1.1, it is necessary to preload the core level GCC/11.2.0 and in the case of wanting to use ScaLAPACK/2.1.0 it is necessary to preload GCC/11.2.0 and OpenMPI/4.1.1.

Characteristics of a module hierarchy are:

- not all module files are directly available for loading;
- some modules serve as a gateway to more modules;
- to access some software installations you will first need to load one or more "*gateway*" modules in order to use them;

- "automatic" conflict resolution by loading modules

You can probably think of other ways to organize module files in a hierarchical module tree, but here we will stick to this widely used ***CORE / COMPILER / MPI*** hierarchical standard.

How to use Lmod and understand the Hierarchical Scheme?

You will see in later sections how they are used in conjunction with job scheduler Slurm. To familiarize yourself with this tool you can log in and run the following more useful commands to understand how it works:

```
$module spider modulename          -- search a module
```

```
$module avail (o av) (modulename)   -- shows available modules
```

```
$module load modulename            -- load a specific module / application
```

```
$module unload modulename (or rm)   -- remove a loaded module
```

```
$module purge                      -- remove all loaded modules
```

```
$module list                       -- list the modules loaded by the user
```

```
$module show modulename            -- show module information
```

```
$module update modulename          -- reload module
```

```
$module (--raw) show modulename     -- show the contents of a module
```

```
$module reset                      -- go back to an initial set of modules
```

If there are many modules on a system, it can be difficult to see what modules are available to load. Lmod provides the overview command to provide a concise listing. For example

```
$module overview                   -- show an overview with all available modules
```

For more information about *lmod* this is the official documentation for this tool. https://lmod.readthedocs.io/en/latest/010_user.html

How does Hierarchical scheme work?

Search software at CORE level

At CORE level you can see software that depend on the system only such us compiler and other aplications

```
$ module av
```

```
----------------------------------- /opt/envhpc/modulefiles/.rhel7-eb/all/Core ------------------------------------
   ADMIXTURE/1.3.0                     M4/1.4.18                         gettext/0.21
   Bison/3.5.3                         M4/1.4.19                 (D)     gompi/2021b
   Bison/3.7.1                         Miniconda2/4.7.10                 iimpi/2021b
   Bison/3.7.6               (D)       Miniconda3/4.9.2                  imkl/2021.4.0
   FastQC/0.11.8-Java-1.8              OpenSSL/1.1                       intel-compilers/2021.4.0
   FastQC/0.11.9-Java-11     (D)       PLINK/1.07-x86_64                 intel/2021b
   GATK/3.8-0-Java-1.8.0_281           PLINK/1.9b_6.21-x86_64            ncurses/6.2
   GCC/10.2.0                          PLINK/2.00a2.3_x86_64     (D)     picard/2.25.5-Java-13
   GCC/11.2.0                (D)       SHAPEIT/2.r837.GLIBCv2.12         pkg-config/0.29.2
   GCCcore/10.2.0                      Trimmomatic/0.39-Java-1.8         popt/1.18
   GCCcore/11.2.0            (D)       binutils/2.35                     snpEff/4.3t-Java-1.8
   GCTA/1.94.0beta                     binutils/2.36.1                   tbl2asn/20200302-linux64
   Java/1.8.0_281            (1.8)     binutils/2.37             (D)     zlib/1.2.11
   Java/11.0.2               (11)      flex/2.6.4
   Java/13.0.2               (D:13)    foss/2021b

  Where:
   Aliases:  Aliases exist: foo/1.2.3 (1.2) means that "module load foo/1.2" will load foo/1.2.3
   D:        Default Module

If the avail list is too long consider trying:

"module --default avail" or "ml -d av" to just list the default modules.
"module overview" or "ml ov" to display the number of modules for each name.

Use "module spider" to find all possible modules and extensions.
Use "module keyword key1 key2 ..." to search for all possible modules matching any of the
"keys".
```

Search software at COMPILER level

After load a compiler, for example a any version of GCC, it is possible to see the compiled software stack for this compiler version.

```
$ module load GCCcore/11.2.0
$ module av
```

```
------------------------- /opt/envhpc/modulefiles/.rhel7-eb/all/Compiler/GCCcore/11.2.0
----------------------------
   Autoconf/2.69                    PMIx/4.1.0                        hypothesis/6.14.6
   Autoconf/2.71             (D)    PROJ/8.1.0                        intltool/0.51.0
   Automake/1.16.2                  Pango/1.48.8                      jbigkit/2.1
   Automake/1.16.4              (D)    Perl/5.32.1                        libGLU/9.0.2
   Autotools/20200321                  Perl/5.34.0                 (D)    libarchive/3.4.3
   Autotools/20210726           (D)    ProbABEL/0.5.0                     libarchive/3.5.1    (D)
   BioPerl/1.7.8-Perl-5.34.0           PyOpenGL/3.1.5                     libcerf/1.17
   BioPerl/1.7.8                (D)    Python/2.7.18-bare                 libdrm/2.4.107
   Bison/3.7.1                         Python/2.7.18                      libevent/2.1.12
   Bison/3.7.6                  (D)    Python/3.8.6                       libfabric/1.13.2
   Brotli/1.0.9                        Python/3.9.6-bare                  libffi/3.4.2
   CMake/3.18.4                        Python/3.9.6                (D)    libgd/2.3.1
...
------------------------------------ /opt/envhpc/modulefiles/.rhel7-eb/all/Core ------------------------------------
   Bison/3.5.3                         M4/1.4.19                          gompi/2021b
   Bison/3.7.1                         Miniconda2/4.7.10                  iimpi/2021b
   Bison/3.7.6                         Miniconda3/4.9.2                   imkl/2021.4.0
   FastQC/0.11.8-Java-1.8              OpenSSL/1.1                        intel-compilers/2021.4.0
   FastQC/0.11.9-Java-11      (D)      PLINK/1.07-x86_64                  intel/2021b
   GATK/3.8-0-Java-1.8.0_281           PLINK/1.9b_6.21-x86_64             ncurses/6.2
   GCC/10.2.0                          PLINK/2.00a2.3_x86_64       (D)    picard/2.25.5-Java-13
   GCC/11.2.0                 (D)      SHAPEIT/2.r837.GLIBCv2.12          pkg-config/0.29.2
   GCCcore/10.2.0                      Trimmomatic/0.39-Java-1.8          popt/1.18
   GCCcore/11.2.0             (L,D)    binutils/2.35                      snpEff/4.3t-Java-1.8
   GCTA/1.94.0beta                     binutils/2.36.1                    tbl2asn/20200302-linux64
   Java/1.8.0_281             (1.8)    binutils/2.37                      zlib/1.2.11
   Java/11.0.2                (11)     flex/2.6.4
   Java/13.0.2                (D:13)   foss/2021b
```

Search software with spider

Lmod has a very useful tool for this, the command **spider** which provides the information needed to load a module.

- Software depends on CORE / COMPILER level

```
$ module spider cmalt
```

```
  cMALT: cMALT/0.3.8-Java-1.8
----------------------------------------------------------------------------------------------------------------
    Description:
      The Genome Analysis Toolkit or GATK is a software package developed at the Broad Institute to analyse
     ....
    You will need to load all module(s) on any one of the lines below before the "cMALT/0.3.8-Java-1.8" module is available to load.

      GCCcore/11.2.0

    Help:
      Description
      ===========
      The Genome Analysis Toolkit or GATK is a software package developed at the Broad Institute
    ...
```

To load the module you will need to load all listed modules

```
$ module load GCCcore/11.2.0 cMALT/0.3.8-Java-1.8
```

- Software depends on CORE / COMPILER / MPI

```
$ module spider eigen/3.4
```

```
  Eigen: Eigen/3.4.0
------------------------------------------------------------------------------------------------------------------
    Description:
      Eigen is a C++ template library for linear algebra: matrices, vectors, numerical solvers, and related
      algorithms.


    You will need to load all module(s) on any one of the lines below before the "Eigen/3.4.0" module is available to load.

      GCC/11.2.0  OpenMPI/4.1.1
      GCCcore/11.2.0

    Help:
    ...
```

To load the module you will need to load all listed modules in order ***CORE / COMPILER / MPI / SOFTWARE***

```
$ module load GCCcore/11.2.0 GCC/11.2.0 OpenMPI/4.1.1 Eigen/3.4.0
```

- Another option is to load **gompi/2021b**.

gompi means *GNU Open MPI* and as you can see, now all software compiled under GCC and OpenMPI appears.

```
$ module load gompi/2021b
$ module av
```

```
------------------------/opt/envhpc/modulefiles/.rhel7-eb/all/MPI/GCC/11.2.0/OpenMPI/4.1.1 ------------------------
   ATLAS/0.9.9                     Infernal/1.1.4                      SciPy-bundle/2021.10
   AdmixTools/7.0.2                Kraken2/2.1.1                       XML-LibXML/2.0132-Perl-5.32.1
   Armadillo/10.5.3                LAMPLD/1.0                          XML-LibXML/2.0132-Perl-5.34.0
   Arrow/6.0.0                     MAFFT/7.487-with-extensions         arpack-ng/3.8.0
   BLAST+/2.11.0                   MitoZ/2.3-Python-3.9.6              arrow-R/6.0.0.2-R-4.1.2
   Beast/2.5.1                     MrBayes/3.2.7                       beagle-lib/3.0.2
   Biopython/1.79-Python-3.9.6     Ninja/1.10.1                        git/2.33.1              (D)
   ....
---------------------------- /opt/envhpc/modulefiles/.rhel7-eb/all/Compiler/GCC/11.2.0 -----------------------------
```

```
   BCFtools/1.10.2        GCTA/1.94.0beta             (D)    SAMtools/1.14
   BCFtools/1.12   (D)    GEOS/3.9.1                         Shapely/1.8.0
   BEDTools/2.30.0        GSL/2.7                            SpectrA/0.9.0
   BLIS/0.8.1             HTSlib/1.14                        SpectrA/1.0.0              (D)
   BWA/0.7.17             LAPACK/3.9.1                       VCFtools/0.1.16
   BamTools/2.5.2         LAPACK/3.10.1               (D)    XML-LibXML/2.0132-Perl-5.32.1
   ....
-------------------------- /opt/envhpc/modulefiles/.rhel7-eb/all/Compiler/GCCcore/11.2.0
---------------------------
   Autoconf/2.69                    PMIx/4.1.0                (L)      hypothesis/6.14.6
   Autoconf/2.71             (D)    PROJ/8.1.0                         intltool/0.51.0
   Automake/1.16.2                  Pango/1.48.8                       jbigkit/2.1
   Automake/1.16.4           (D)    Perl/5.32.1                        libGLU/9.0.2
   Autotools/20200321               Perl/5.34.0               (D)      libarchive/3.4.3
   Autotools/20210726        (D)    ProbABEL/0.5.0                     libarchive/3.5.1    (D)
   ....
------------------------------------ /opt/envhpc/modulefiles/.rhel7-eb/all/Core ------------------------------------
   ADMIXTURE/1.3.0                  M4/1.4.18                          gettext/0.21
   Bison/3.5.3                      M4/1.4.19                          gompi/2021b              (L)
   Bison/3.7.1                      Miniconda2/4.7.10                  iimpi/2021b
   Bison/3.7.6                      Miniconda3/4.9.2                   imkl/2021.4.0
```

Meaning of some important modules

- **GCC/11.2.0**:

GCC/11.2.0 defines the module that groups all the CORE of GCC.

- **GCCCore/11.2.0**:

GCCCore/11.2.0 defines the GCC CORE and all software compiled with it.

- **foss/2021b**

The foss common compiler consists entirely of common term *FOOS* which is short for *Free and Open source software*.

This module consists of :

- binutils
- the GNU Compiler Collection GCC (C), g++ (C++) and gfortran (Fortran)
- the Open MPI library
- the FlexiBLAS library, with OpenBLAS + LAPACK as backend
- the ScaLAPACK
- the FFTW library (Fourier transform (DFT) in one or more dimensions)

```
$ module load foss/2021b
$ module list
Currently Loaded Modules:
  1) GCCcore/11.2.0   5) numactl/2.0.14      9) hwloc/2.5.0      13) libfabric/1.13.2  17) 
FlexiBLAS/3.0.4
  2) zlib/1.2.11      6) XZ/5.2.5           10) OpenSSL/1.1      14) PMIx/4.1.0        18) FFTW/
3.3.10
  3) binutils/2.37    7) libxml2/2.9.10     11) libevent/2.1.12  15) OpenMPI/4.1.1     19) 
ScaLAPACK/2.1.0-fb
  4) GCC/11.2.0       8) libpciaccess/0.16  12) UCX/1.11.2       16) OpenBLAS/0.3.18   20) foss/
2021b
```

- **gompi/2021b**:

The *gompi* module preloads all software compiled with GNU GCC and OpenMPI

```
$ module load gompi/2021b
$ module list
Currently Loaded Modules:
  1) GCCcore/11.2.0   5) numactl/2.0.14      9) hwloc/2.5.0      13) libfabric/1.13.2
  2) zlib/1.2.11      6) XZ/5.2.5           10) OpenSSL/1.1      14) PMIx/4.1.0
  3) binutils/2.37    7) libxml2/2.9.10     11) libevent/2.1.12  15) OpenMPI/4.1.1
  4) GCC/11.2.0       8) libpciaccess/0.16  12) UCX/1.11.2       16) gompi/2021b
```

- **intel/2021b**:

This common module preloads Intel compilers and libraries.

- the Intel C/C++/Fortran compilers (icc, icpc and ifort)
- binutils
- GCC, which serve as a base for the Intel compilers
- Intel MPI library
- Intel Math Kernel Library (MKL) for BLAS/LAPACK/FFT functionality

```
$ module load intel/2021b
$ module list
Currently Loaded Modules:
  1) GCCcore/11.2.0   3) binutils/2.37              5) numactl/2.0.14   7) impi/2021.4.0   9) imkl-
FFTW/2021.4.0
  2) zlib/1.2.11      4) intel-compilers/2021.4.0   6) UCX/1.11.2       8) imkl/2021.4.0  10) intel/
2021b
```