Environment Modules

TeideHPC is a cluster 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 on your own server. .

Environment Modules is a software management system that allows users of Unix-like systems, such as Linux and macOS, and particularly in high-performance computing (HPC) environments, **dynamically modify their environment, access different compilers, libraries and software that may have multiple versions and dependencies.**

There are different implementations of Environment Modules but the one currently used in TeideHPC is ***Lmod***.

Lmod

Lmod is an acronym for **Lua Module System** is a specific implementation of the Environment Modules concept for managing software environments in high-performance systems (HPC).

Lmod allows users to:

- load and download softwaremodules dynamically.
- Change environment variables like PATH and LD_LIBRARY_PATH.
- facilitate the use of different versions of applications and libraries without interference between them.

You can learn more about Lmod at https://lmod.readthedocs.io/en/latest/.

How is the software organized at TeideHPC?. Flat nomenclature.

In TeideHPC there are 2 clusters, TeideHPC and AnagaGPU

For this reason, the set of modules available **depends on the Cluster you are going to use.**

Each cluster has its own login nodes and **they do not have the same software.**

The module schema uses specific nomenclature for its modules, which typically follows the format:

```
<software_name>/<version>-<toolchain>[-<variations>]
```

- **Software_name**: This is the name of the software or library that is being installed.
- **Version**: It is the specific version of the software.
- **Toolchain**: The compilation toolchain that includes the compiler, the MPI library (if applicable), and other performance optimization libraries used to compile the software. For example, foss is a common toolchain that includes GCC, OpenMPI, and others.
- **Variants**: These are additional modifiers that indicate specific variants of the installation, such as specific hardware optimizations or the inclusion of certain features.

For example, a module for Unzip version 6.0 compiled with the GCCcore 12.2.0 toolchain might have module nomenclature like

`UnZip/6.0-GCCcore-12.2.0`

An example of a module with different variants and toolchain is the following:

```
module spider 'Python'

----------------------------------------------------------------------------------------------------------------
  Python:
----------------------------------------------------------------------------------------------------------------
    Description:
      Python is a programming language that lets you work more quickly and integrate your systems more
      effectively.

     Versions:
        Python/2.7.18-GCCcore-11.2.0-bare
        Python/2.7.18-GCCcore-11.2.0
        Python/2.7.18-GCCcore-12.2.0-bare
        Python/3.8.6-GCCcore-11.2.0
        Python/3.8.6-intel-2021b
        Python/3.9.6-GCCcore-11.2.0-bare
        Python/3.9.6-GCCcore-11.2.0
        Python/3.9.6-intel-2021b
        Python/3.10.8-GCCcore-12.2.0-bare
        Python/3.10.8-GCCcore-12.2.0
```

What is a *Toolchain*?

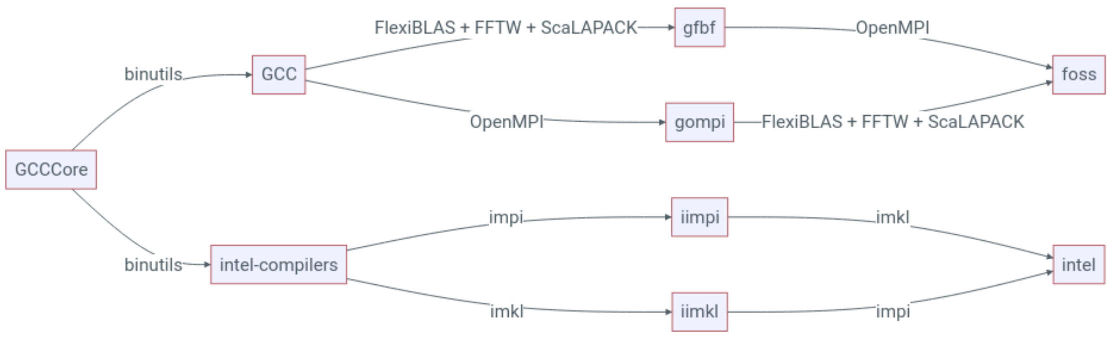
The term "toolchain" refers to the set of software tools used to compile and link programs.

A toolchain typically includes compilers for different programming languages, mathematical software libraries, and linking and debugging tools.

Toolchains are defined by a set of components that include:

- **Compilers**: GCC (GNU Compiler Collection), Intel Compiler, etc.
- **MPI communications libraries**: For example, OpenMPI, MPICH, Intel MPI, etc.
- **Mathematics libraries**: BLAS (Basic Linear Algebra Subprograms) and LAPACK (Linear Algebra Package).
- **Other libraries and tools**: Such as GMP (GNU Multiple Precision Arithmetic Library), FFTW (Fastest Fourier Transform in the West), etc.

The following diagram can see how the software is divided depending on the chosen Toolchain. **Basically 2 branches are identified depending on the main compiler: GNU GCC or Intel GCC**



Likewise, **there are different generations of these toolchains that are incompatible with each other.**

foss toolchain

The *foss* toolchain consists entirely of open source software (hence the name, derived from the common term 'FOSS', which is short for "Free and Open Source Software").

At TeideHPC we can identify the following generations for *foss*

foss	**binutils**	**GCC**	**Open MPI**	**FlexiBLAS**	**OpenBLAS**	**LAPACK**	**ScaLA**
2021b	2.37	11.2.0	4.1.1	3.0.4	0.3.18	(incl. with OpenBLAS)	2.1.0
2022a	2.38	11.3.0	4.1.4	3.2.0	0.3.20	(incl. with OpenBLAS)	2.2.0
2022b	2.39	12.2.0	4.1.4	3.2.1	0.3.21	(incl. with OpenBLAS)	2.2.0

intel toolchain

The *Intel* Toolchain consists of Intel compilers and libraries.

At TeideHPC we can identify the next generations for *intel*

En TeideHPC podemos identificar las siguientes generaciones para *intel*

intel	**binutils**	**GCC**	**Intel compilers**	**Intel MPI**	**Intel MKL**
2021b	2.37	11.2.0	2021.4.0	2021.4.0	2021.4.0
2022a	2.38	11.3.0	2022.1.0	2021.6.0	2022.1.0
2022b	2.39	12.2.0	2022.2.1	2021.7.1	2022.2.1

In the next section you can see how to use modules to view and load the software available in TeideHPC.

`module spider foss`

```
----------------------------------------------------
  foss:
----------------------------------------------------
    Description:
      GNU Compiler Collection (GCC) based compiler toolchain, including OpenMPI for MPI
support, OpenBLAS (BLAS
      and LAPACK support), FFTW and ScaLAPACK.

     Versions:
        foss/2021b
        foss/2022b
----------------------------------------------------
  To find other possible module matches execute:

      $ module -r spider '.*GCC.*'
```

```
module spider intel
```

```
-----------------------------------------------------
  intel:
-----------------------------------------------------
    Description:
      Compiler toolchain including Intel compilers, Intel MPI and Intel Math Kernel Library
(MKL).

     Versions:
        intel/2021b
        intel/2022b
     Other possible modules matches:
        intel-compilers

-----------------------------------------------------
  To find other possible module matches execute:

      $ module -r spider '.*intel.*'
```