Software

The cluster runs the Scientific Linux 7 (SL7) distribution, which is based on the commercial Red Hat Enterprise Linux 7 (RHEL7) product. This means that all of the software provided with RHEL7 is available to the cluster as well as any commercial software that supports RHEL7. In addition to this, any free, libre or open source software that targets UNIX environments should also run on the cluster.

The software on the cluster falls into one of two classes:

1. Provided with the SL7 distribution
2. Not provided by the distribution

Examples of software in the first class are editors (vim, emacs, nano), compilers (gcc, gfort, g++) and scripting language interpreters (perl, python, ruby). These pieces of software are available by just typing the relevant command. The second class of software is provided by the GRID team, and require some additional work to use. This additional work is required so that we can provide multiple versions of the same software without requiring you to carry out a lot of tedious and error prone configuration. You might need multiple versions of the same software because it might need to be compiled with different compilers or different compiler options, or it may be that an older release is better suited to solving a particular class of problem. Selecting which version to use is managed by the Environment Modules tool.

The installation and support of personal applications, whether or not existing in the official distribution repository, is the responsibility of the user. The scope of the Grid team support is limited to infrastructure and base software with official packages for the distribution of the Operating System.

Environment Modules

The Environment Modules tool is an open source project for managing environments. It works through configuration files called module files which describe how to add or remove environment variables that are required for each program. The module command then uses these files to apply the changes. To see all of the software available, you can use the avail argument to the module command:

```
$ module avail

------------------------------------ /usr/share/Modules/modulefiles --------
----------------------------
dot         module-git  module-info modules     null        use.own

---------------------------------------------- /etc/modulefiles ---------------
----------------------------
abaqus/2017           hdf/5.1.10.1          mpi/mpich-x86_64      python/2.7
ansys/18.2            intel/2017.2.174      mpi/openmpi-x86_64    python/3.4
dl_poly/4.03          java/1.6              nomad/3.8.1
salome/8.3.0
dl_poly/4.03-mpi      java/1.8              octave/octave-4.2.1
simgrid/3.18
gcc/7.2.1             maple/13              octave/octave-mpich
wien/wien2k
gmsh/3.0.6            matlab/2017b          octave/octave-openmpi
```

```
gromacs/2018          meep/1.4.3            openfoam/2.3.0
hadoop/1.1.2          mpfr/mpfr-4.0.0       openfoam/v1712
hadoop/1.2.1          mpi/mpich-3.0-x86_64  opensees/20130328
```

You can then load a module with the load command:

```
$ maple
-bash: maple: command not found
$ module load maple/13
$ maple
    |\^/|     Maple 13 (X86 64 LINUX)
._|\|   |/|_. Copyright (c) Maplesoft, a division of Waterloo Maple Inc.
2009
 \  MAPLE  /  All rights reserved. Maple is a trademark of
 <____ ____>  Waterloo Maple Inc.
      |       Type ? for help.
>
```

If you don't specify the version, then the module system will normally load the latest version, so *module load maple* is the same as *module load maple/13*. You can check which modules are currently loaded with the *list* command:

```
$ module list
Currently Loaded Modulefiles:
  1) maple/13
```

Normally, you can't load different versions of the same package as this doesn't make sense. If you try to do this it will generate a warning:

```
$ module load java/1.6
$ module load java/1.8
java/1.8(5):ERROR:150: Module 'java/1.8' conflicts with the currently loaded
module(s) 'java/1.6'
java/1.8(5):ERROR:102: Tcl command execution failed: conflict        java
```

If you want to stop using a module, you can use the remove command:

```
$ module list
Currently Loaded Modulefiles:
  1) maple/13   2) java/1.6
$ module remove java
$ module list
Currently Loaded Modulefiles:
  1) maple/13
```

From:
https://grid.fe.up.pt/dokuwiki/ - **GRID FEUP**

Permanent link:
https://grid.fe.up.pt/dokuwiki/doku.php?id=documentation:software

Last update: **2021/03/03 17:24**