Using Environment Modules

This page describes high level usage of SLAC's implementation of environment modules.

environment modules are a way of dynamically loading arbitary programs into your unix shell environment. examples include the ability to switch between different versions of software or entire software installs.

You can use the MODULEPATH environment to control where environment modules should fine 'modulefiles' - text files that define what and how to load certain programs.

```
export MODULEPATH=/usr/share/Modules/modulefiles:/etc/modulefiles:/opt/modulefiles:/afs/slac/package/singularity
/modulefiles
```

In the above example, we pick up these modulefiles from the two default paths under /usr and /etc, and then we have SLAC's standard modules (base libraries etc) followed by some containerised applications under /afs/slac/package/singularity/modulefiles.

You can view all available modulefiles by running:

```
$ module avail
------------------------------------------------------- /usr/share/Modules/modulefiles
------------------------------------------------------
dot         module-git  module-info modules     null        use.own
------------------------------------------------- /afs/slac/package/singularity/modulefiles
-------------------------------------------------
amira/6.7.0             eman2/20190418           motioncor2/1.2.1         rclone/1.44              rosetta
/2018.48
chimera/1.13.1          eman2/20190603           motioncor2/1.2.2         relion/2.1               rosetta/3.10
ctffind/4.1.10          emClarity/1.0.0          motioncor2/1.2.3-intpix  relion/3.0               scipion/1.
2.1
ctffind/4.1.13          git/2.13.0               phenix/1.14-3260         relion/3.0.2             slac-ml
/20181002.0
eman2/20181015          icon-gpu/1.2.9           protomo/2.4.2            relion/3.0.4             slac-ml-
devel/20181002.0
eman2/20190320          imod/4.9.10              pymol/2.1.1              relion/3.0_beta-20181121 xds/20190315
eman2/20190324          imod/4.9.11              pymol/2.2                resmap/1.95
------------------------------------------------------------- /opt/modulefiles
-------------------------------------------------------------
boost/1.69.0-openmpi-3.1.2-gcc-4.8.5        gcc/6.3.1                                openmpi/3.1.3-gcc-
7.3.1
boost/1.69.0-openmpi-3.1.2-gcc-7.3.1        gcc/7.3.1(default)                       parallel-hdf5/1.10.4-
openmpi-3.1.2-gcc-4.8.5
cuda/10.0(default)                          intel/2019.4.227(default)                parallel-hdf5/1.10.4-
openmpi-3.1.2-gcc-7.3.1
cuda/9.0                                    intelmpi/2019.4.227(default)             parallel-hdf5/1.10.4-
openmpi-3.1.3-gcc-4.8.5
cuda/9.2                                    lsf                                      parallel-hdf5/1.10.4-
openmpi-3.1.3-gcc-7.3.1
fftw3/3.3.8-openmpi-3.1.2-gcc-4.8.5         openmpi/3.1.2-gcc-4.8.5                  PrgEnv-gcc/4.8.5
(default)
fftw3/3.3.8-openmpi-3.1.2-gcc-7.3.1         openmpi/3.1.2-gcc-7.3.1                  PrgEnv-gcc/7.3.1
gcc/4.8.5                                   openmpi/3.1.3-gcc-4.8.5                  python/anaconda3
(default)
```

Here, you will see the modulesfiles orgranised by their top level directory. the typical format is the name of the application, slash, version of application.

you can then load the module with

```
module load slac-ml/20181002.0
```

at which point, the various binaries associated with that application should now be available in your path.