TEXAS ADVANCED COMPUTING CENTER

WWW.TACC.UTEXAS.EDU

TEXAS
The University of Texas at Austin

Using Personal Modules and Inherit() w/ the Software Hierarchy

Robert McLay

March 7, 2023

Outline

- How to correctly use personal modules so that Lmod will find and use them.
- How to setup a personal library/application in the software hierarchy
- Why this is a PITA!

Creating Personal modules

- What is the big deal?
- Easy: Create a directory (say $HOME/my_modules)
- Create a directory ($HOME/my_modules/acme)
- Create a create modulefile: $HOME/my_modules/acme/3.2.lua)
- `$ module use $HOME/my_modules`
- `$ module load acme/3.2`
- Easy right!?

Testing a personal copy of a system module

- Suppose that acme/3.2 is already on your system
- And acme/3.2 is a marked default
- The command `module load acme/3.2`
- Will load the system one and not yours
- Even though $HOME/my_modules is listed first in $MODULEPATH
- Why?

Why?

- While Lmod does look in $MODULEPATH order
- So the first module found is usually picked.
- *However*, marked defaults ALWAYS win in N/V module layouts (Best found)
- Note not in N/V/V layouts. (First found)
- A marked default is where there is either:
  1. A default symlink
  2. .modulerc.lua
  3. .modulerc
  4. .version
- I recently updated the documentation https://lmod.readthedocs.io/en/latest/060_locating.html to explain this

Getting Around a System Marked Defaults

- Make your own marked default.
- Easiest way is to make a default symlink

```
$ cd $HOME/my_modules/acme
$ ln -s 3.2.lua default
```

Checking with module avail

```
---------- /home/user/my_modules -----------
acme/3.2 (D)

---------- /opt/apps/modulefiles ------------
StdEnv    acme/3.2
```

- Make sure that the (D) is next to your acme module
- And not the system one.

Bigger issue: Testing a compiler dependent `boost/1.85.0`

- And you want it part of the software hierarchy
- How can you do this `without` modifying the system modulefiles?
- In particular you only want the correct version of boost available when you load the correct compiler.

The short answer: inherit()

- You can use the inherit() function to simplify this a little
- This is discussed in detail in https://lmod.readthedocs.io/en/latest/340_inherit.html

Overview

- We want to test/use boost install from our own account.
- And have it load when the “right” compiler is loaded
- This assumes that your site is using the software hierarchy
- How can we get the system compiler to load our directory into $MODULEPATH?
- Suppose we want to test a boost version with the intel 19.1 and gcc 12.2 compilers

Building and Installing boost in your account

- Build boost 1.85.0 with gcc 12.2 $\Rightarrow$ ~/pkg/gcc-12/boost/1.85.0
- Build boost 1.85.0 with intel 19.1 $\Rightarrow$ ~/pkg/intel-19/boost/1.85.0

Choice 1: Copy Each compiler modulefiles into your account

- Easy to do
- Add your hierarchical directory into $MODULEPATH
- Problem: you are now responsible to keep your copy up-to-date
- As the sys-admin might change them w/o you knowning

Choice 2: Use inherit()

- Create your own compiler module and inherit from the system one.
- The inherit() function take NO arguments
- Lmod looks for the exact same name in $MODULEPATH
- This way it includes the system one with your changes.

Inherit() part 2

- Create gcc/12.2 and intel/19.1 in your own directory structure
- Then create two boost modulefiles for each compiler

Steps for gcc/12.2

```
   $ mkdir -p ~/my_modules/{Core,Compiler,MPI}
```

You also set the following environment variable:

```
   $ export MY_MODULEPATH_ROOT=$HOME/my_modules
```

When this is set up you will do:

```
   $ module use ~/my_modules/Core
```

gcc/12.2.lua

Then in the file ~/my_modules/Core/gcc/12.2.lua you have::

```
inherit()
local compiler = "gcc"
local MP_ROOT  = os.getenv("MY_MODULEPATH_ROOT")
local version  = "12"

prepend_path("MODULEPATH", pathJoin(MP_ROOT, "Compiler",compiler,version))
```

- Note that I'm assuming that I'll have the same libraries
- for all versions of gcc 12.*

intel/19.1.lua

```
Suppose you also have the system intel/19.1  module.  Then you would
need at ~/my_modules/Core/intel/19.1.lua you have::

   inherit()
   local compiler = "intel"
   local MP_ROOT  = os.getenv("MY_MODULEPATH_ROOT")
   local version  = "19"

   prepend_path("MODULEPATH", pathJoin(MP_ROOT, "Compiler",compiler,version))
```

- Note that I'm assuming that I'll have the same libraries
- for all versions of intel 19.*

Protect against system marked defaults

```
$ cd ~/my_modules/Core/intel; ln -s 19.1.lua default
$ cd ~/my_modules/Core/gcc;   ln -s 12.2.lua default
```

Now support two versions of boost

- gcc boost:
- ~/my_modules/Compiler/gcc/12/boost/1.85.0.lua
- intel boost
- ~/my_modules/Compiler/intel/19/boost/1.85.0.lua

What about mpi libraries and applications

- Similar to what was shown above
- See https://lmod.readthedocs.io/en/latest/340_inherit.html for more details

Conclusions

- It is some work but it is possible to include personal libs/apps
- See the documentation
- Or see rt/user_inherit for a full example

Future Topics

- Matthew will talk about how benchpro interacts with Lmod
- Next Meeting will be April 4th at 9:30 Central (14:30 UTC) (back to Summer time!)