TEXAS ADVANCED COMPUTING CENTER
WWW.TACC.UTEXAS.EDU

The University of Texas at Austin

Lmod Debugging & Module evaluation

Robert McLay

Sept. 7, 2021

Debugging Modulefiles & How are Modulefiles evaluated?

- Debugging Modulefiles Tricks
- How are Modulefile evaluated?

Debugging Modulefiles Tricks

- https://lmod.readthedocs.io/en/latest/160_debugging_modulefiles.html
- Trick 1: Printing: `$LMOD_CMD bash load` *module*
- Trick 2: Tracing: `ml -T` *module*

How many ways are Modules evaluated?

- There are 3 main ways: load, unload, show
- There are 10 total ways (src/MC_*)
- How does Lmod handle this mess?

The simple goals of an Env. Module System

- Change User's environment
- One text file that's independent of shell (bash, zsh, csh, ...)
- Instead of separate shell scripts for each shell (like intel scripts)
- Great Feature: Unloading a module restores User's env. (kinda?!)

Example Modulefile: phdf5

```
setenv("TACC_HDF5_DIR","/apps/.../phdf5/1.12.1")
setenv("TACC_HDF5_INC","/apps/.../phdf5/1.12.1/inc")
setenv("TACC_HDF5_INC","/apps/.../phdf5/1.12.1/inc")
setenv("TACC_HDF5_LIB","/apps/.../phdf5/1.12.1/lib")
prepend_path("PATH","/apps/.../phdf5/1.10.4/bin")
prepend_path("LD_LIB_PATH","/apps/.../phdf5/1.10.4/lib")
help([[Help Message for Parallel HDF5 ...]])
```

Bash: Module load phdf5

```
export TACC_HDF5_DIR=/apps/.../phdf5/1.12.1
export TACC_HDF5_INC=/apps/.../phdf5/1.12.1/inc
export TACC_HDF5_LIB=/apps/.../phdf5/1.12.1/lib
export PATH=/apps/.../phdf5/1.10.4/bin:/usr/bin:/bin
export LD_LIB_PATH=/apps/.../phdf5/1.10.4/lib:...
```

Bash: Module unload phdf5

```
unset TACC_HDF5_DIR
unset TACC_HDF5_INC
unset TACC_HDF5_LIB
export PATH=/usr/bin:/bin
export LD_LIB_PATH=...
```

Lua Object Oriented Programing

- Lua’s OOP model is OOP lite
- It is simplier than Python’s
- There is very little Magic
- It is an extension of Lua’s Hash Tables (AKA Dictionaries)
- Functions are First Class Object
- They can be assigned to variables

How does Lmod evaluate modulefile functions etc

- Note that TCL modules are converted to Lua automatically
- Each module function calls Lua functions (like setenv())
- Inside each function dynamically calls the correct operation for load, unload, etc.

Lmod finds and reads phdf5/1.12.1.lua

- loadModuleFile.lua reads modulefile into a string *whole*
- `status, msg = sandbox_run(whole)`
- Each line in sandbox is evaluated by the lua interpreter

How does Lmod handle setenv()?

- Lmod could have check the mode() in each function
- Instead Lmod builds MasterControl object (mcp) based on mode()
- There is a derived class for Load, Unload, Show etc.

How does Lmod handle setenv()?

```
--src/modfunc.lua
function setenv(...)
    -- check args
    mcp:setenv(...)
end

--src/MasterControl.lua
function M.setenv(self, name, value)
   local frameStk = FrameStk:singleton()
   local varT     = frameStk:varT()
   if (varT[name] == nil) then
      varT[name] = Var:new(name)
   end
   varT[name]:set(tostring(value))
end

function M.unsetenv(self, name, value)
   local frameStk  = FrameStk:singleton()
   local varT      = frameStk:varT()
   if (varT[name] == nil) then
      varT[name]   = Var:new(name)
   end
   varT[name]:unset()
end
```

What is mcp? How does load work?

```
--src/lmod.in.lua
   MCP = MasterControl.build("load")
   mcp = MasterControl.build("load")

--src/cmdfuncs.lua
function Load_Usr(...)
   local mcp_old = mcp
   mcp = MCP
   mcp:load_usr(...)
   mcp = mcp_old
end

--src/MC_Load.lua
...
M.setenv               = MasterControl.setenv
```

How does unload work?

```
--src/cmdfuncs.lua
function UnLoad(...)
   local mcp_old = mcp
   mcp = MasterControl.build("unload")
   MCP:unload_usr(...)
   mcp = mcp_old
end

--src/MC_Unload.lua
...
M.setenv               = MasterControl.unsetenv
```

Other internal Lmod Topics

- Rules for finding modulefiles for load?
- The MName object?
- The ModuleTable stored in the Environment
- sandbox() ?
- FrameStk?
- tcl to lua translation?
- Lmod testing system?