

# Lmod tools to check module syntax

Robert McLay

February 14, 2023

# Outline



- ▶ How to check module syntax when building an RPM or other software packages
- ▶ How to check your sites' entire module tree.

# Checking Modulefile Syntax

- ▶ We want to check the syntax
- ▶ Not the action!
- ▶ We ignore the actions of `loads()`, `depends_on()` ...
- ▶ How to evaluate all the syntax but not the action?

# Quick Review: How Lmod evaluates modulefile functions

- ▶ Remember that commands like `setenv("ABC", "3")` do different things
- ▶ On module load it sets ABC to 3
- ▶ On module unload it unsets ABC
- ▶ On module show it prints `setenv("ABC", "3")`

# src/modfuncs.lua: setenv()

- ▶ Almost all modulefile function work like setenv()
- ▶ They check the syntax
- ▶ Then they call a member function in the MainControl Class
- ▶ The mcp variable is global MainControl object
- ▶ mcp controls what mode Lmod is in.

```
function setenv(...)
  if (not l_validateArgsWithValue("setenv",...)) then
    return
  end
  mcp:setenv(...)
  return
end
```

# Controlling Lmod Evaluation Mode

- ▶ When loading: mcp constructs derived class MC\_Load.
- ▶ When unloading: mcp constructs derived class MC\_Unload.
- ▶ When checking syntax: mcp constructs derived class MC\_CheckSyntax

# src/MC\_Load.lua

```
M.load                = MainControl.load
M.myModuleName        = MainControl.myModuleName
M.prepend_path        = MainControl.prepend_path
M.prereq              = MainControl.prereq
M.setenv              = MainControl.setenv
M.set_alias           = MainControl.set_alias
M.unload              = MainControl.unload
M.unsetenv            = MainControl.unsetenv
```

- These are the normal mapping when loading

# src/MC\_Unload.lua

```
M.load                = MainControl.unload
M.myModuleName        = MainControl.myModuleName
M.prepend_path        = MainControl.remove_path_first
M.prereq              = MainControl.quiet
M.setenv              = MainControl.unsetenv
M.set_alias           = MainControl.unset_alias
M.unload              = MainControl.unload
M.unsetenv            = MainControl.quiet
```

- These are the normal mapping when unloading



# src/MC\_CheckSyntax.Lua

M.load	= MainControl.load
M.myModuleName	= MainControl.myModuleName
M.prepend_path	= MainControl.prepend_path
M.prereq	= MainControl.quiet
M.setenv	= MainControl.setenv
M.set_alias	= MainControl.set_alias
M.unload	= MainControl.quiet
M.unsetenv	= MainControl.unsetenv

- ▶ See that the actions of `prereq()` are ignored
- ▶ This way actions outside a module are ignored
- ▶ The syntax of all commands are checked.
- ▶ `MainControl.load` is a special case

# MainControl. load is a special case

- ▶ Loading the module under test must use it
- ▶ So load\_usr() function has this test
- ▶ This prevent module loading other modules

```
function M.load_usr(self, mA)
    local frameStk = FrameStk:singleton()
    if (checkSyntaxMode() and frameStk:count() > 1) then
        return
    end

    l_registerUserLoads(mA)
    local a = self:load(mA)
    return a
end
```

# module --checkSyntax load bad/1.0

```
$ cat bad/1.0.lua  
setenv("ONE")
```

```
$ module --checkSyntax load bad
```

```
Lmod has detected the following error:  setenv("ONE") is not valid;  
a value is required.
```

```
While processing the following module(s):
```

Module fullname	Module Filename
-----	-----
bad/1.0	/home/user/myModules/bad/1.0.lua

- ▶ We here at TACC use this when building module files in an RPM \*.spec file
- ▶ Other build tools might find this useful

# New command `check_module_tree_syntax`

- ▶ I wanted a way to check every modulefile in a site's tree.
- ▶ The command module `spider` used to do that.
- ▶ But sending site error to user is not a good idea.

# New command `check_module_tree_syntax` (II)

- ▶ Site wanted to know when module directory had multiple files marking a default
- ▶ There are upto 4 ways to do this in a modulefile directory
  1. a default symlink
  2. a `.modulerc.lua` file
  3. a `.modulerc` file
  4. a `.version` file
- ▶ The priority is in this order.

# History

- ▶ The site wanted to report to the user when this happened.
- ▶ I try hard not to report site errors to users.
- ▶ After all what are they suppose to do with this info?
- ▶ So I modified `$LMOD_DIR/spider` to walk the module tree.
- ▶ To make `$LMOD_DIR/check_module_tree_syntax`
- ▶ This reports both syntax error/warning and duplicate marked default files in a modulefile directory.
- ▶ This command uses the `MC_CheckSyntax` mode to eval each modulefiles.

# Changes to spider cache format

- ▶ Added defaultA to remember all marked defaults in directory
- ▶ This array is process to use the highest priority marked default file.
- ▶ The old format just kept the highest priority marked default file in defaultT.

# Example results

The following directories have more than one marked default file:

-----  
/Users/mclay/w/lmod/main/rt/ck\_mtree\_syntax/mf/A

The following modulefile(s) have syntax errors:

-----  
ModuleName: A/1.0, Fn: /mf/A/1.0.lua

Error: [string "setenv("MY\_VERSION", myModuleVersion())..."]:2: unexpected symbol near

ModuleName: A/2.0, Fn: /mf/A/2.0

Error: /mf/A/2.0: (A/2.0): invalid command name "nonExistantCmd"

ModuleName: hashrf/6.0.1, Fn: /mf/hashrf/6.0.1.lua

Error: [string "local name = "hashrf"..."]:16: attempt to call a nil value (global 'WTF')

ModuleName: papi/4.4.0, Fn: /mf/papi/4.4.0.lua

Error: command: help, one or more arguments are not strings.





# Conclusions

- ▶ Two ways to check for modulefile syntax errors.
- ▶ One for building modulefiles
- ▶ Another for checking site module tree.

# Future Topics

- ▶ Unknown at the moment.
- ▶ Next Meeting will be March 7th at 9:30 Central (15:30 UTC)