TEXAS ADVANCED COMPUTING CENTER

WWW.TACC.UTEXAS.EDU

TEXAS
The University of Texas at Austin

The complicated story about TCL break

Robert McLay

May. 3, 2022

Outline

- Let's talk about TCL break (and LmodBreak)
- Lmod didn't really support TCL break at all until Lmod 8.6 (really Lmod 8.7)
- Years ago mailing list question: support for break
- Lmod 6 and below could not support break
- Why?

Reminder: How Lmod works

- In order to have a command effect the current shell
- A simple module command for bash is given below
- The $LMOD_CMD command generate shell commands as text
- The eval "..." evaluate the text to change the current shell
- For the rest of this talk: focus on what $LMOD_CMD produces

```
module () { eval "$($LMOD_CMD bash "$@")"; }
```

Reminder: How Lmod TCL processing works

- Internally Lmod knows when a file is a TCL modulefile
- No *.lua extension ⇒ TCL modulefile
- The program tcl2lua.tcl is called to process the tcl
- It converts TCL module command into Lua with Lmod module commands

```
setenv FOO bar ⇒ setenv("FOO","bar")
prepend-path PATH /prgm/bin ⇒ prepend_path("PATH","/prgm/bin")
break ⇒ LmodBreak() -- Only for bare breaks
```

TCL Break

```
for {set i 0} {$i $<$ 5} {incr i} {
  puts stderr "$i"
  if { $i == 3 } {
      break  # This breaks out of the loop
  }
}
break # This causes the modulefile
      # to stop being processed.
```

Why was TCL break such a problem for Lmod?

- TCL break stops processing the current module
- It ignores any changes in a module that has a break
- But it keeps all other modules loaded.
- `module load A B C D`
- Where C has a break
- Then A B are loaded but C and D are not.

LmodError is different

- `module load A B C D`
- Where C has an LmodError()
- No modules are loaded.

Lmod waits to produce output

- When loading several modules, Lmod waits
- All module actions are completed internally
- Then Lmod generates shell command output.
- Lmod 6 and earlier wouldn't know what changes to ignore when processing a break.
- All Lmod's produce either an error or environment changes not both.

Lmod 7+ was a complete re-write of Lmod

- It was needed to support Name/Version/Version (N/V/V) modulefiles
- Before Lmod only supported N/V or C/N/V
- Lmod 7+ now has a frameStk (AKA the stack-frame)
- The frameStk contains a stack of the environment var table (varT) and the module table (mt)

FrameStk: varT and mt

- The table varT contains key-value pairs that represent the new env. var values
- The table mt is the module table containing the currently loaded modules among other things
- The Module Table is stored in the environment via $_ModuleTable001_ etc.

FrameStk

- Before each module: Deep Copy previous varT and mt to top of FrameStk.
- Normal assignment copy references.
- A deepcopy() is required for a new array.
- Each evaluation of modulefile is updated on the top of the FrameStk
- When the current modulefile evaluation is completed
- The FrameStk is pop'ed
- The previous stack values are replaced with current

LmodBreak or TCL break

- If LmodBreak() is called, the current module changes are ignored
- LmodBreak() causes the previous values to be current
- FrameStk:pop() pops the stack.
- The code is shown below:

```
function M.LmodBreak(self)
   local stack           = self.__stack
   local count           = self.__count
   stack[count].mt       = deepcopy(stack[count-1].mt)
   stack[count].varT     = deepcopy(stack[count-1].varT)
end

function M.pop(self)
   local stack           = self.__stack
   local count           = self.__count
   stack[count-1].mt     = stack[count].mt
   stack[count-1].varT   = stack[count].varT
   stack[count]          = nil
   self.__count          = count - 1
end
```

Support for TCL break

- Lmod 8.6+ added support LmodBreak()
- Lmod 8.6+ added support a bare TCL break
- Lmod 8.7+ added support for regular break and bare break

TCL Break strangeness

```
for {set i 0} {$i $<$ 5} {incr i} {
  puts stderr "$i"
  if { $i == 3 } {
      break  # This breaks out of the loop
  }
}
break # This causes the modulefile
      # to stop being processed.
```

- TCL treats a bare break as an error
- Tmod 3, 4 and 5 catch the error

To support regular and bare break in TCL in tcl2lua.tcl

```
set sourceFailed [catch {source $ModulesCurrentModulefile } errorMsg] # (1)
set returnval 0
if { $g_help && [info procs "ModulesHelp"] == "ModulesHelp" } {
   # handle module help
   ...
}
if {$sourceFailed} {
    if { $sourceFailed == 3 || $errorMsg == {invoked "break" outside of a loop}} {
        set returnVal 1
        myBreak
        showResults
        return $returnVal
    }
    reportError $errorMsg
    set returnVal 1
}
showResults
return $returnVal
```

- line 1 evaluate the TCL modulefile
- $sourceFailed will be non-zero for TCL errors
- $sourceFailed == 3 means a bare break has been found.