CFM : A Console File Manager for POSIX

Joseph Manning

Department of Computer Science
University College Cork
Ireland

Lua Workshop 2013

CFM : The Context

console : text-based, runs in terminal

Why a console file manager? (it's soooo 1980's!)

- keyboard is faster and more ergonomic than mouse
- fits more onto the screen, no space taken by icons
- more light-weight, can help on small or slow computers
- easier and faster over a remote connection
- useful on servers which lack GUI software

Why yet another console file manager?

- already: Midnight Commander, FDclone, vifm, ytree, . . .
- but none of these really suited me (!)
- so let's look at the goals for CFM . . .

CFM : Goals

Design Goals

- simple ergonomic interaction
- consistency of commands
- instant seamless response
- clean minimalist appearance
- silent handling of harmless errors
- modest set of commonly-used features

Personal Goals

- create a file manager to match my wishes
- explore ‘curses’ programming
- gain more experience with Lua

CFM : Demonstration

CFM : The Program

- written entirely in Lua
- runs under Lua 5.1 / Lua 5.2 / LuaJIT
- uses the ‘curses’ and ‘posix’ libraries
- comparison with other console file managers:

Name	Language	Files	Lines
FDclone	C	102	94 586
Midnight Commander	C	325	92 228
vfu	C	56	14 948
ytree	C	58	13 970
vifm	C	40	9 010
CFM	Lua	1	718

CFM : Binding Keys to Actions

```
KeyActions = {
               a = function( )
                      ToggleActive( "access" )
                      end,
               q = function( )
                      running = false
                      end,
               z = function( )
                      if #Items > 0 then
                         local item = Items[ focuspos ]
                         item.marked = not item.marked
                         end
                      end,
             }

setmetatable( KeyActions,
              { __index = function( )
                             return function( ) end
                             end } )
```

CFM : Main Program

```
Setup( )
while running do
    UpdateDisplay( )
    KeyActions[ ReadKey( ) ]( )
    end
CloseDown( )
```

Speed of Computing `#t`

- `#t` computed repeatedly, but with no change in `t`
- optimisation! `local len_t = #t`
- code a bit messier . . . but maybe worth it for the speed?
- no, not at all!
- can compute `#t`
  where `t` has length 10,000
  a total of 1,000,000 times
  in just 0.1 seconds
- `+1` : binary search
- `-1` : tables with holes

CFM : The Need for Speed

- crucial to achieve instant seamless response
- yet need only operate on a *human* time-scale
- less than 1/30th second $\equiv$ instantaneous
- “fast enough is fast enough”
- even on a little 5-year-old €195 netbook . . .
- . . . which is also running 2 infinite loops
- raw speed of Lua allows clean coding of CFM

CFM : User Configuration

- a single configuration file, processed by **`dofile`**
- defines a string **`terminal`** and a table **`OpenProg`**
- example ('**#**' = placeholder for name of file being opened):

```
terminal = "urxvtc"

OpenProg = {
             dvi    = "xdvi #",
             html   = "iceweasel #",
             jpg    = "display #",
             odt    = "libreoffice #",
             pdf    = "zathura # 2>/dev/null",
             wmv    = "mplayer #",
             ["*"]  = "elvis # 2>/dev/null"
           }
```

CFM : Reflections on Use of Lua

- small language, clear orthogonal features, easy to grasp
- f-a-s-t ! ... although “fast enough is fast enough”
- the table data structure (combines ‘arrays’ and ‘records’)
- default counting-from-one
- use of dispatch table of anonymous functions
- use of `__index` metamethod for missing table keys
- simplicity of writing and processing configuration file

CFM : Current Status

- still a work-in-progress, although quite useable already
- portability barely tested
- error-checking incomplete
- documentation incomplete
- but if you *still* want a copy, then e-mail me at

  `manning@cs.ucc.ie`

  or see me today with a USB stick

Thanks for Listening !