luawrap

Self-executing lua scripts

David Whale

Londonlua 1st Nov 2012

Overview/Requirements

- Scripting Language
  - A single language
  - Powerful enough for file, sockets, apps
- Distribution
  - To customers
  - No-install
  - No-dependencies
  - Hiding clever stuff
- Sharing
  - Others can do the same
  - … without special tools

Hello.exe (DEMO)

- Hello world (hello.lua->hello.exe)
- Hello with args (args.lua->args.exe)

Why Lua?

- Small
  - 200kB on Win32 (inc libs, socket, compiler, interpreter)
- Simple
  - Few language constructs to learn
  - Well maintained by core community
- Flexible
  - Good C interface (libs)
  - Tables and metatables
  - Object orientation as design pattern
  - Very extensible
- Open source
  - Standing on the shoulders of giants
  - Source is well written
  - Around for 20 or so years

Prior Work

- srlua (2004)
  - (http://www.tecgraf.puc-rio.br/~lhf/ftp/lua/#srlua)
  - Separate "glue" and "interpreter" exe's
  - Doesn't have the compiler built in.
  - Requires you to build it - some prebuilt exe's around
  - Only supports a single resource
  - Very simple
  - Appends resources to end of file

Prior Work

- L-Bia (2010) (http://l-bia.sourceforge.net/)
  - Strips whitespace and comments from scripts and compresses them
  - Attempts to find and embed all libraries (so you have to have them already)
  - Makes a very complex attempt to automate everything into an overlay
  - No source of latest version (only of archives)
  - *“WARNING: This version doesn't do that, it only run a lua script with the same name as the executable. I'm rewriting it due to many bugs in previous versions.“*

Prior Work

- squish (2010)
  - (http://matthewwild.co.uk/projects/squish/home)
  - nice way to pack everything into a single lua script
  - still need separate runtime
- MurgaLua (2008)
  - (http://www.murga-projects.com/murgaLua/murgaLua.html)
  - single prebuilt exe with lots of embedded libs (GUI etc)
  - but still need separate runtime and scripts.
- LuaJit
  - might be a way to append script to end and bootstrap with command line script, but not double-click to run.

Prior Work

- Lots of chatter on forums
  - references to above all talking about "append script to end and hack the interpreter"
- Modify lua interpreter to include scripts
  - bin2c used for this
  - my first solution did this (“luaembed”)
  - but you need compilers and source installed

First Hack (DEMO)

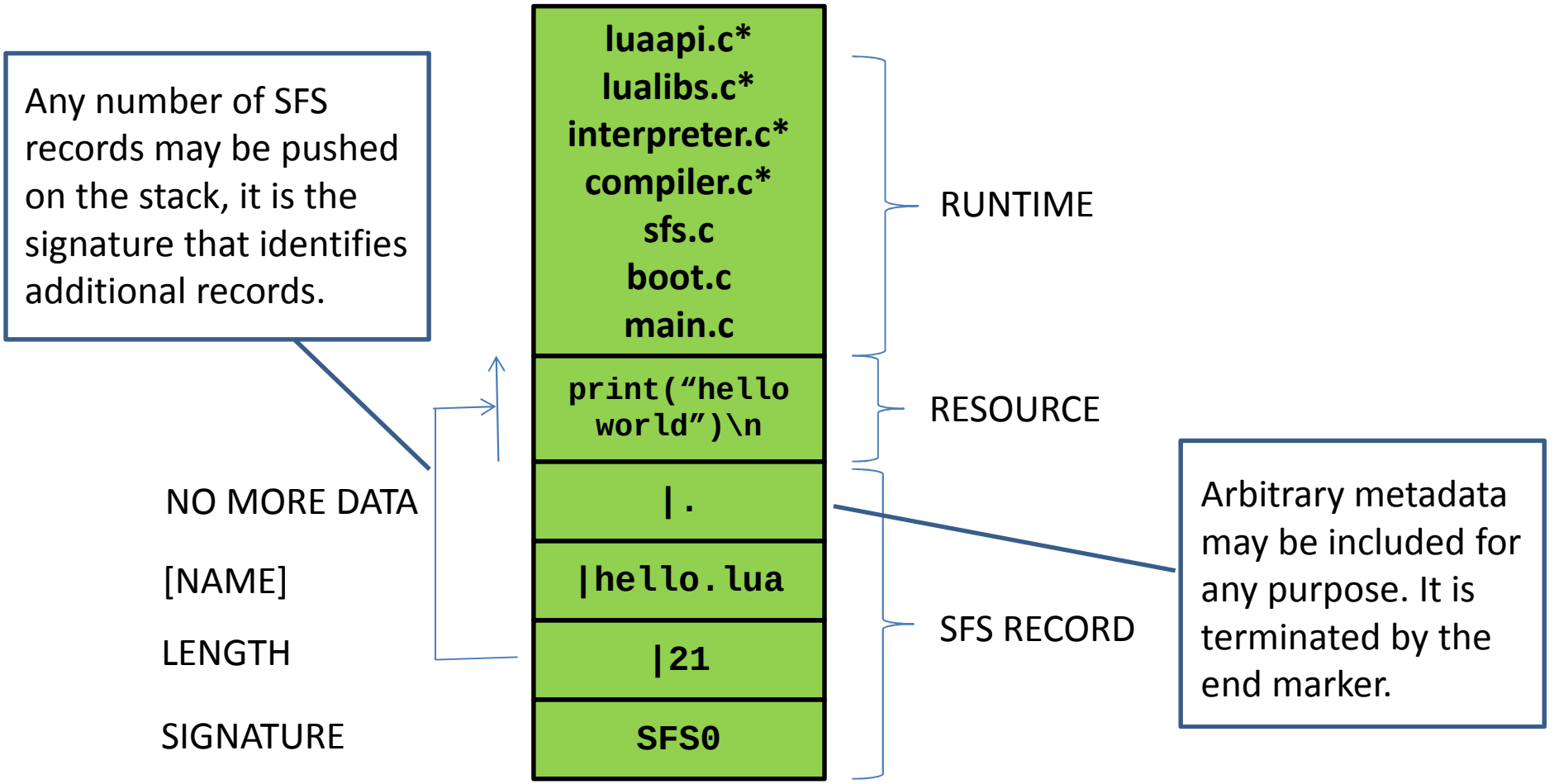
- "luaembed" - bin2c in a makefile, compile along with a tiny main()
- makefiles to knit it all together
- gets tedious writing a makefile for each new script
- but packaged all the libs & libconf as static link
- rewrote "bin2c" in lua and created bin2c.exe with bin2c.lua inside

No-Compiler Requirement

- want to enable others to do it
- those ‘others’ are not likely to want to install and use compiler tools
- looked at parsing exe headers to embed resource inside
- best option seems to be to tack it all on the end and work backwards
- wrote a SFS (stack filing system)
- expandable to multiple resources in future

Stack Filing System

- Records are pushed on to the end of the file



Manually Adding Resources (DEMO)

- print("hello world")|.|hello.lua|lenSFS0
- `copy /b luawrap.exe + hello.lua + hello.rec hello.exe`
- What does this look like for real?

Eat Your Own Dog Food (DEMO)

- rewrite manual wrapping process in lua
- append luawrap.lua to lua.exe
- call it luawrap.exe
- Single executable luawrap.exe
  - Hmm, could just call it lua.exe?

Case Study – buildnum (DEMO)

- buildnum
  - Modify a buildnum.txt on each run
  - Generate buildnum.h to include in C files
  - Originally written in C
  - Rewritten in lua and wrapped in about 10 mins

```
- #ifndef _BUILDNUM_H
- #define _BUILDNUM_H
- #define BUILDNUM_NUMBER 1
- #define BUILDNUM_STR “1”
- #endif
```

Some Fun (DEMO)

- Re-wrapping scripts
  - Because it's a stack, you can push other things on
- But how do you get to earlier scripts?

Boot Monitor (DEMO)

- Boot.c introduces:

```
luawrap.exe --lua list
luawrap.exe --lua boot <name> <args>
luawrap.exe --lua interpret <name> <args>
luawrap.exe --lua compile <name> <args>
```

- Or even

```
hello.exe –lua list
```

- Embedding compiled lua:

```
luawrap –lua compile –o hello.lbin hello.lua
luawrap luawrap.exe hello.lbin hello.exe
hello.exe
hello.exe –lua list
```

Comments on Lua Sources

- libconf process had to be tidied up to simplify library set selection
- Lua standalone and Lua compiler extracted into mini packages/interfaces
- minimal whitespace or comments in the source code(!)
- doxygen helps to find things
- naming convention bit weird (single letter lua package prefixes)
- ‘make’ process was changed to build static libraries for useful chunks

Conclusions

- I Had some specific requirements
- Met those with luawrap
- Other solutions do work, but mine seems simpler and more flexible
- Single executable does everything you need

- More to do to enable multi-file lua programs
  - (could use squish, but see future work…)

Future Work

- file access
  - internal/external/pictures/sounds sfs.c sfs.lua
- library load
  - internal/external
- bootmonitor
  - rewrite in lua, special status for first pushed script "boot0"
- L-Bia can link .dll and .so?
- Packaging and library loading
  - socket/core.dll vs socket.lua
- Protection/sandbox
  - Encryption/decryption/compression, limit access to int/ext resources
- linux version – just a makefile?
- virus checker implications? None found yet
- Publication (source/user guide)

Questions

- http://www.thinkingbinaries.com/luawrap