Lua

A lightweight,
portable, dynamic
scripting language

Scott Vokes - BarCamp GR, Aug. 21 2009

"I like Lua for the same reason I like WikiWiki:
it's clean, simple, and light. The authors had a
pure vision and *left out* the right things."

- Ashley Fryer

Real world usage

- Adobe Lightroom
- World of Warcraft (and several other games)
- nmap, monotone, tokyocabinet, awesome, ion, ...

Features

- Designed primarily for embedding:
  A 200k dynamic library (800k source)
- Trivial to integrate with existing C projects
- Incremental garbage collector
- Portable to any platform w/ ANSI C compiler
- Excellent for rapid prototyping
- “The Good Parts”

Serious computer science juju

- First-class functions
- Tail-call optimization
- Lexical scope, with proper closures
- One-pass byte-compiler
- Register-based virtual machine
- Lightweight, assymetric co-routines
- Dynamic linking
- Scheme-like semantics

History

- “SOL” – Simple Object Language, c. 1993
- PUC-Rio and Tecgraf
- Petrobras
- Major influences:
  SNOBOL, Icon, AWK, Bibtex, Scheme

Example code

```
-- Flatten the whitespace in a line.
function flatten_whitespace(line)
   line = string.gsub(line, "[ \t\n]+", " ")
   if string.sub(line, 1, 1) == " " then
      return string.sub(line, 2)
   else
      return line
   end
end
```

Core data types

- Number
  - Doubles by default (precise to 2^53, then floating)
  - Not boxed in a pointer, numeric ops are fast
- String
  - Array of raw bytes w/ a given length
  - Interned (comparison by pointer)
  - Can be UTF-8, raw JPEG data with \0s, etc.

Core data types, cnt'd.

- Boolean
  - True, false
- Nil
- Function
- Tables
  - Assoc. array, like Python's "dictionary"
- Thread (co-routine)
- Userdata
  - Opaque reference to external C pointer
  - Full userdata have metatables, garbage-collection

Metatables

- Hooks to define behavior for tables and udata
- __index: Called when a slot doesn't exist
  - Good for e.g. inheritance, proxying, error logging
- __newindex: Called for creating a slot
  - Read-only tables, proxy to key-value database, etc.
- __call: Using a table/udata as a function
  - "functables"

Environment control

- You can sandbox a function (e.g. compiling and running end-users' code) with setfenv and a custom environment

- Environments *are just tables*, so you can use metatables...

Data serialization

JSON:

```
{
      "firstName": "John",
      "lastName": "Smith",
      "address": {
          "streetAddress": "21 2nd Street",
          "city": "New York",
          "state": "NY",
          "postalCode": "10021"
      },
      "phoneNumbers": [
          { "type": "home", "number": "212 555-1234" },
          { "type": "fax", "number": "646 555-4567" }
      ]
 }
```

Data serialization in Lua

Same, in Lua:

```
{
    firstname="John",
    lastname="Smith",
    address = { streetAddress="21 2nd Street",
          city="New York",
          state="NY",
               postalCode=10021
             },
    phoneNumbers = {
  { type="home", number="212 555-1234" },
        { type="fax", number="646 555-4567" },
    }
}
```

Configuration file format

```
device="ral0"
sudo=true

Network { key="home", nwid="bluebuddha", wpa="secret-o-rama" }
Network { key="founders", nwid="Founders", wep="…" }
Network { key="sparrows", nwid="The Sparrows" }
Network { key="schulersa", nwid="schuler-alpine" }
Network { key="schulersd", nwid="Schuler Downtown" }
Network { key="work", nwid="configuraguestnetwork", wpa="…" }
Network { key="itsagrind", nwid="It's A Grind" }
Network { key="madcap", nwid="secure", wpa="peoplebeforeprofit" }
Network { key="ferris", nwid="riverfront" }
```

Some downsides

- The usual “dynamic language” tradeoffs
- The module system came late, may be too simple (no linker, just a global table of pkgs)
- Tables are indexed from 1
- Vars are global unless “local” is used
- Any missing args are nil, so APIs are too fluid
- Since you can “fork your own Lua”, there aren’t universal code conventions

Going deeper

- The C API
- The debug API
- LPEG
- LuaJIT

Questions?

For further information:

lua.org

lua-users.org