

Introducere in Lua

Bogdan Marinescu
Autor proiect eLua

BLUG·OS·CON
we're open

Introducere

- Creat in 1993 la PUC Rio (Brazilia)
 - versiunea curenta: 5.1.4
 - <http://www.lua.org>
- Open source (MIT)
- Limbaj imperativ cu tipuri dinamice
- Limbaj interpretat (compilabil la bytecode), garbage collected
- Compact (interpretor ~200k pe x86)
- Unul din cele mai rapide limbaje interpretate
 - <http://dada.perl.it/shootout/craps.html>
- Minimalist, usor de invatat
- Portabil (Windows, Unix, OSX...)

Utilizare

- Modele de utilizare:
 - standalone
 - embedded: functii Lua apelate din C
- Domeniu principal: embedded pentru scripting in jocuri
 - World of Warcraft, Far Cry, Runes of Magic ...
 - scripting pe alte aplicatii: nmap, SciTE, VLC
- Framework-uri compacte pentru Web:
 - Kepler (<http://www.keplerproject.org/>)
 - Orbit (<http://orbit.luaforge.net/>)
- Wiki: Sputnik (<http://sputnik.freewisdom.org/>)
- GUI: IUP (<http://www.tecgraf.puc-rio.br/iup/>)
- Utilizari non-desktop: LuCI (OpenWRT)

'Hello, World!' in Lua

```
$ lua  
Lua 5.1.4 Copyright (C) 1994-2008 Lua.org, PUC-Rio  
> print "Hello, World!"  
Hello, World!
```

```
$ echo 'print "Hello, World!"' > hello.lua  
$ lua hello.lua  
Hello, World!
```

Tipuri de date

- **nil**
- **boolean:** doar **false** si **nil** sunt considerate “false”
- **number**
 - un singur tip de numere, implicit 'double'
 - tipul de numere poate fi schimbat la compilare
- **string:** poate fi oricat de lungi si poate contine 0
- **function**
 - functiile sunt valori 'first-class' in Lua
- **table:** tip de date central in limbaj
 - perechi (cheie, valoare) accesate eficient print hashing
 - cheile si valorile pot fi de orice tip, mai putin 'nil'
 - folosit pentru a implementa array-uri (tabele cu chei intregi), variabile globale, modele OOP ...

Operatori

- binari : +, -, *, /, % (modulo), ^ (ridicare la putere)
- unari : -
- concatenare string-uri: ..
- relationali : ==, ~=, <, >, <=, >=
- logici: **and**, **or**, **not**
- #: obtine lungimea unui string sau al unui array
- nu contine suport pentru operatii pe biti
 - se pot adauga prin module aditionale

Structuri de control

- blocuri de cod : **do** **end**
- **for** numeric
 - **for** var = initial, final, pas **do** **end**
- **for** cu iteratori
 - **for** v1, v2, ..., vn **in** *iterator*(params) **do** **end**
 - iteratori generici (iterare peste tabele, linii din fisiere...)
 - suporta break, dar nu continue
- **while** conditie **do** **end**
- **repeat** **until** conditie
- **if** conditie **then** [**elseif** conditie **then**] **end**
- **return**

Tabele

```
-- Inicializare
t = {}
t["a"] = 10
t["b"] = 20

-- Inicializare echivalenta
t = {a = 10, b = 20}

-- Accesare elemente
print(t.a, t.b) -- sau t["a"], t["b"]

-- Inicializare array (index de start 1)
t = {1, 2, 3, 4, 5}

-- Iterare tabela
for k, v in pairs(t) do print(k, v) end

-- Variabilele globale sunt chei in tabela predefinita _G
v = 20
_G.v = 10
print(v) -- 10

-- Descriere structura de date folosind o tabela
book = { title = "Programming in Lua, Second Edition",
         author = "Roberto Ierusalimschy",
         ISBN = "8590379825" }
```

Funcții

- Funcțiile sunt valori 'first-class' in Lua
 - pot fi asignate unei variabile sau trimise ca parametrii
 - pot fi intoarse ca rezultat al unei alte functii
 - “lexical scoping” (acces la variabilele functiei de nivel superior)

```
function f(x) print(x) end
-- Echivalent
f = function(x) print(x) end

-- Functie ce returneaza o alta functie (lexical scoping)
function sum_factory(x)
  return function(n) return x + n end
end
f = sum_factory(2)
print(f(5)) -- 7
f = sum_factory(5)
print(f(5)) -- 10

-- Functie anonima
table.sort(t, function(e1, e2) return e1 < e2 end)

-- Apel functie cu parametru table (simulare parametri cu nume)
create_window{x = 0, y = 0, width = 100, height = 100, name = "MyWin"}
```

Module

- Extind functionalitatea Lua cu functii din librarii externe
 - similar conceptual cu “import” in Python
 - foloseste tabele pentru implementare
- Modulele pot fi scrise in Lua sau in C (folosind un API special)

```
require "lfs"
lfs.chdir "/usr/local/bin"
```

- LuaRocks: sistem centralizat de management pentru module
 - instalare (cu compilare locala)/dezinstalare module

```
$ sudo luarocks install bitlib
$ sudo luarocks remove bitlib
```

Module built-in

- Module incluse automat in interpretorul Lua la compilare
 - nu este nevoie de instalare separata si de “require”
- **string**: operatii cu string-uri (inclusiv expresii regulate)
- **table**: manipulare tabele
- **math**: functii matematice din libm (sin, cos, exp...)
- **io**: operatii de intrare/iesire (pentru consola si fisiere)
- **os**: servicii ale sistemului de operare (execute, exit, ...)
- **debug**: depanarea programelor scrise in Lua
- **coroutine**: corutine

Coroutine

- Mecanism de multitasking cooperativ inclus in limbaj
 - o corutina isi poate suspenda executia (yield)
 - o corutina poate restarta alta corutina (resume)
 - o corutina nu poate intrerupe alta corutina

```
co = coroutine.create(function ()
  for i=1,10 do print("co", i) coroutine.yield() end
end)
```

```
coroutine.resume(co)    -- co    1 (ruleaza pana la yield)
coroutine.resume(co)    -- co    2
.....
coroutine.resume(co)    -- co    10
coroutine.resume(co)    -- nu tipareste nimic (corutina terminata)
```

- Folosite si pentru a implementa iteratori pentru **for**

Metatabele si metametode

- **Metatabele:** schimba comportarea unei tabele
 - un set de **metametode** apelate automat in situatii predefinite

```
t = {}
print(t.v) -- nil, deoarece "v" nu este o cheie in tabela t
mt = {__index = function(table, key) return "default" end}
setmetatable(t, mt)
print(t.v) -- "default", deoarece __index se apeleaza automat
```

- Utilizari:
 - implementare modele OOP bazate pe prototip
 - tabele cu valori "on-demand"
 - tabele read-only (__newindex)
 - tabele cu valori implicite
 - "supradefinire" operatori (__add, __sub, __le, __lt)
 - tipuri noi de date (seturi, liste dublu inlantuite, ...)

Resurse Lua

- On-line
 - website : <http://www.lua.org/>
 - manual complet: <http://www.lua.org/manual/5.1/>
 - cartea "Programming in Lua": <http://www.lua.org/pil/>
 - doar prima editie este disponibila gratuit on-line
 - lista de discutii: <http://www.lua.org/luail.html>
 - wiki: <http://lua-users.org/wiki/>
 - diverse tutorial-uri
- Carti tiparite
 - "Programming in Lua, Second Edition" (8590379825)
 - "Lua Programming Gems" (8590379841)
 - "Beginning Lua Programming" (0470069171)

do return end

Intrebari?