Lua

Vuonna 1993 Brasiliassa kehitetty skriptauskieli, jota pääasiallisesti käytetään muilla kieleillä kirjoitettujen ohjelmien laajentamiseen. Kielen mukana tulee vain hyvin kevyet peruskirjastot, eikä se sisällä valmiita työkaluja mm. olio-ohjelmoinnin toteuttamiseen, mutta se on hyvin helposti laajennettavissa lähes mitä tahansa käyttötarkoitusta varten.

* Erittäin kevyt

C-kielellä toteutettu tulkki on kokonaisuudessaan vain 247 kilotavua. Lua ei sisällä samanlaisia kattavia kirjastoja kuin esimerkiksi Python, ja käyttäjän odotetaan itse muokkaavan ja laajentavan kieltä omaa käyttöä varten.

* Dynaamisesti tyypitetty skriptauskieli

Lua on tarkoitettu nopeaan skriptaukseen ja muiden ohjelmien laajennukseen. Se sisältää monia käyttäjää helpottavia ominaisuuksia, kuten dynaamisen tyypityksen ja vahvat implisiittiset tyyppimuunnokset.

* Tulkattava

Luaa ei tarvitse manuaalisesti kääntää; se tulkataan ajon aikana tavukoodiksi jota tulkki ajaa. Vaikka se on alunperin toteutettu C:llä, tulkkeja löytyy myös lukuisille muille kielille.

* Assosiatiiviset taulukot

Kielen ainoana varsinaisena tietorakenteena on “table”, joka vastaa mm. Pythonin Dictionaryä ja Javan Map-luokkaa. Lähes kaikissa muissa kielissä olevat perustaulukot (array) ovat Luassa käytännössä tableja joiden avaimena on kokonaisluku 0 - n.

* Metataulukot

Yksi kielen kiinnostavimmista ominaisuuksista on ns. Metataulukot: assosiatiiviset taulukot joiden käyttäytymistä mm. arvoja sijoittaessa tai etsittäessä voidaan vapaasti muokata. Metataulukot sallivat kielen laajentamisen hyvinkin erilaisiin suuntiin, ja esimerkiksi monet olio-ohjelmoinnin perusteet kuten luokkien periminen ovat toteutettavissa metataulujen kautta.

* Oletuksena globaalit muuttujat

Ellei Luan muuttujia eksplisiittisesti määritä paikallisiksi, ne ovat aina näkyvyydeltään globaaleja.

Lua on suosittu skriptauskieli korkean tason toiminnallisuudelle monia eri käyttökohteita varten. Esimerkiksi Internet of Things-maailmassa monet WiFi-ohjaimet ovat toteutettu Lualla, ja erityisesti videopelimaailmassa Lua on erittäin suosittu mm. modauksen toteuttamiseen, tai itse pelin sisällön implementoimiseen. Esimerkiksi Angry Birds -mobiilipelin eri tasot ovat Lualla skriptattuja, kuten on mm. World of Warcraft -roolipelin käyttöliittymä.

Lua on yksinkertainen ja sillä on erittäin helppo tehdä nopeita muutoksia ja prototyyppejä, mutta yksinkertaisuudella on hintansa. Implisiittiset tyyppimuunnokset saattavat aiheuttaa outoja, vaikeasti löydettäviä virheitä; Luan numerojärjestelmä pohjautuu puhtaasti liukulukuihin, jotka saattavat aiheuttaa varomattomasti käytettynä pyöristysvirheitä; oletuksena globaalit muuttujat tekevät kielestä virheherkän laajemmissa projekteissa.

Verrattuna esimerkiksi Pythoniin, jota usein samaan tapaan käytetään muiden ohjelmien laajentamiseen, Lua on huomattavasti kevyempi ja vieläkin yksinkertaisempi kirjoittaa – mutta se on herkempi virheille ja siitä puuttuu monia perusominaisuuksia mitä sen monissa kilpailijoissa on. Silti, Lua on äärimmäisessä kepeydessään ja laajennettavuudessaan varsin uniikki, ja se on pysynyt suosiossa vuosi vuoden jälkeen, loistaen omassa lokerossaan.

12.12.2019 Riku Alho, Topi Törnroos, Simo Auvinen

Esimerkki:

```
load("config.lua")
load("boot.lua")

local buffer = ""
math.randomseed(os.time())

socket = connect_socket(read_hostname(HOST), PORT)
nick(NICK)
socket_send(socket, "USER " .. IDENT .. " " .. HOST .. " bla :" .. REALNAME .. "\r\n")
if NICKSERV_PASSWORD then
  send_privmsg("nickserv", "identify " .. NICKSERV_PASSWORD, true)
end
join(CHANNEL)

::loop::
  local recv, len = socket_receive(socket)
  buffer = buffer .. recv
  local _, cutoff
  repeat
      _, cutoff = string.find(buffer, "\r\n")
      if cutoff then
          local message = string.sub(buffer, 1, cutoff)
          buffer = string.sub(buffer, cutoff + 1)
          local msg_nick, msg_user, msg_server, msg_command, msg_suffix = parsemsg(message)

          if msg_command == "PING" then -- Responds to server PINGs
              print("PING? PONG!")
              print("PONG " .. string.sub(msg_suffix, 0, -1) .. "\r\n")
              socket_send(socket, "PONG " .. string.sub(msg_suffix, 0, -1) .. "\r\n")

          else
              print (string.sub(message, 1, -3))
          end
      end
  until not cutoff
goto loop
```