



蛭子

alacner@gmail.com

Alacner.com



怎么读?
/'luɑ/ (噜啊)



1993 年在巴西里约热内卢天主教大学(Pontifical Catholic University of Rio de Janeiro in Brazil)诞生了一门编程语言，发明者是该校的三位研究人员。

Roberto Ierusalimschy



Waldemar Celes



Luiz Henrique de Figueiredo

History

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



A lightweight, portable, dynamic scripting language

简洁、轻量、可扩展的脚本语言





ZEN X-Fi 2



```
print ("Hello world!")
```



```
print "Hello world!"
```



一切都是变量，除了关键字

Lua的关键字不多，就以下21个：

local function if not **nil** and true or false
then else elseif for repeat until while in do
break return end

注意：大小写敏感:And,AND就不是关键词

+ - * / % ^ # == ~= <= >= < > = () { } [] ; : ,



字符串 string

```
a = 'alo\n123'"
a = "alo\n123\''
a = '\97lo\10\04923'''
a = [[alo 123'']]
a = [==[ alo 123'']]
a = 'alo\n\
123'''
```

注意：字符串内的最后的一个反斜杠与c类似，是说明本行未结束的意思。



数字 number

3

3.0

3.1416

314.16e-2

0.31416E1

0xff

0x56



数组 table

1. {value1, value2,value3,...}
2. {[‘key1’]=value1, [‘key2’]=value2, [‘key3’]=value3,...}
3. {key1 = value1, key2 = value3,...}
4. {0=value1, 1=value2, 2=value3,}



赋值 Assignment

Lua 允许多重赋值。因此，赋值的语法定义是等号左边放一系列变量，而等号右边放一系列的表达式。两边的元素都用逗号间开。

原则：先运算再赋值

例子：

`i = 3`

`i, a[i] = i+1, 20`

`x, y = y, x` --交换 `x` 和 `y` 中的值



逻辑操作符 Logical Operators

10 or 20 --> 10

10 or error() --> 10

nil or "a" --> "a"

nil and 10 --> nil

false and error() --> false

false and nil --> false

false or nil --> nil

10 and 20 --> 20

总结：逻辑操作符把 **false** 和 **nil** 都作为假，而其它的一切都当作真



函数 *function*

函数定义：是一个可执行的表达式，执行结果是一个类型为 *function* 的值。

```
function f(a, b) end
```

```
function r(...) return 1,2,3 end
```

```
f = function (a, b) end
```

```
r = function (...) return 1,2,3 end
```



注释 comment

--

-- This is comment

--[[These
are
comments]]



内在函数 Lua functions

- 独立函数: assert, pairs, print, type, require, ...
- 协同函数: coroutine.*
- 调试函数: debug.*
- 文件函数: io.*, file:*
- 数学函数: math.*
- 系统函数: os.*
- 模块函数: package.*
- 字符函数: string.*
- 数组函数: table.*
- C API函数: lua_***
- 辅助函数: luaL_***



更多话题 More topics

Metatable（元表）
Coroutine（协同例程）
程序接口（C-API）

.....

```
function create_a_counter()
    local count = 0
    return function()
        count = count + 1
        return count
    end
end

local counter = create_a_counter()
print(counter()) - 1
print(counter()) -- 2
print(counter()) - 3
.....
```

```

--[[
unreserved = ALPHA / DIGIT / "-" / "." / "_" / "~"
ALPHA (%41-%5A and %61-%7A) -> [78-90 97-122]
DIGIT (%30-%39) -> [48-57]
hyphen (%2D) -> 45
period (%2E) -> 46
underscore (%5F) -> 95
tilde (%7E) -> 126
--]]
local escape_unreserved = {}
for i=78,90 do escape_unreserved[i] = true end
for i=97,122 do escape_unreserved[i] = true end
for i=48,57 do escape_unreserved[i] = true end
for i=45,46 do escape_unreserved[i] = true end
escape_unreserved[95] = true
escape_unreserved[126] = true

function escape(str) -- rfc3986
  if not str then return "" end
  --str = string.gsub (str, "\n", "\r\n")
  str = string.gsub (str, "([^\w])",
    function (c)
      local b = string.byte(c)
      if escape_unreserved[b] then return c end
      return string.format ("%%%02X", b)
    end
  )
  return str
end
end

```

```

function print_r(sth)
    if type(sth) == "table" then
        print(sth)
        return
    end

    local space, deep = string.rep(' ', 4), 0
    local function _dump(t)
        for k,v in pairs(t) do
            local key = tostring(k)

            if type(v) == "table" then
                deep = deep + 2
                print(string.format("%s[%s] => Table\n%s(",
                    string.rep(space, deep - 1),
                    key,
                    string.rep(space, deep)
                ))
                _dump(v)
            else
                print(string.format("%s[%s] => %s",
                    string.rep(space, deep + 1),
                    key,
                    tostring(v)
                ))
            end
        end
    end

    print(string.format("%s)", string.rep(space, deep)))
    deep = deep - 2
    _dump(sth)
    print(string.format(""))
end

```

example

```
function p()  
    print("Hello World")  
end
```

```
local co = coroutine.create(p)  
print(co)           --> thread: 003FBBF0  
print(coroutine.status(co))  --> suspended  
coroutine.resume(co)  --> Hello World  
print(coroutine.status(co))  --> dead
```

```

#include <lua.h>
#include <luauxlib.h>

/* Compatibility between Lua 5.1+ and Lua 5.0 */
#ifndef LUA_VERSION_NUM
#define LUA_VERSION_NUM 0
#endif
#if LUA_VERSION_NUM < 501
#define luaL_register(a, b, c) luaL_openlib((a), (b), (c), 0)
#endif

static int Lsleep (lua_State *L) {
    lua_Number s = luaL_checknumber(L, 1);
    sleep(s);
    lua_pushnumber(L, s);
    return 1;
}

static const luaL_reg nix[] = {
    { "sleep",    Lsleep },
    { NULL, NULL }
};

int luaopen_demo(lua_State *L) {
    luaL_register(L, "demo", demo);
    return 0;
}
~

```

调用:

```
require "demo"
demo.sleep(5)
```

example

```
cc -o demo.so -shared `pkg-config lua5.1 --libs` `pkg-config lua5.1 --cflags` demo.c
```



For further information:

lua.org

lua-users.org

<http://zh.wikipedia.org/zh-cn/Lua>