深圳四博智联科技有限公司

Lua 入门教程

二〇一五年五月

目 录

一 Lua 基础知识

1.1 什么是 Lua

Lua 是一个小巧的脚本语言，是巴西里约热内卢天主教大学（Pontifical Catholic University of Rio de Janeiro）里的一个研究小组，由 Roberto Ierusalimschy、Waldemar Celes 和 Luiz Henrique de Figueiredo 所组成并于 1993 年开发。其设计目的是为了嵌入应用程序中，从而为应用程序提供灵活的扩展和定制功能。Lua 由标准 C 编写而成，几乎在所有操作系统和平台上都可以编译并运行。Lua 并没有提供强大的库，这是由它的定位决定的。所以 Lua 不适合作为开发独立应用程序的语言。Lua 有一个同时进行的 JIT 项目，提供在特定平台上的即时编译功能。

Lua 脚本可以很容易的被 C/C++代码调用，也可以反过来调用 C/C++的函数，这使得 Lua 在应用程序中可以被广泛应用。不仅仅作为扩展脚本，也可以作为普通的配置文件，代替 XML,ini 等文件格式，并且更容易理解和维护。Lua 由标准 C 编写而成，代码简洁优美，几乎在所有操作系统和平台上都可以编译，运行。一个完整的 Lua 解释器不过 200k，在目前所有脚本引擎中，Lua 的速度是最快的。这一切都决定了 Lua 是作为嵌入式脚本的最佳选择。

Lua 应用：

- Adobe Photoshop Light room uses Lua for its userinterface
- Cisco uses Lua to implement Dynamic Access Policies within the Adaptive Security Appliance
- Codea 为 ios 平台下的游戏开发程序
- 作为游戏的脚本语言
  暴雪-魔兽世界
  UBI-孤岛惊魂
  网易-大话系列
  完美-神鬼传奇
  金山-剑网 3
  巨人-征途
  Minecraft-中的 mod
  愤怒的小鸟
- web 应用：LUCI&openwrt。OpenWrt 是适合于嵌入式设备的一个 Linux 发行版，现在大多数智能路由器内运行的是 openwrt 操作系统。LuCI 是 OpenWrt 上的 Web 管理界面，LuCI 采用了 MVC 三层架构，同时其使用 Lua 脚本开发
- 在嵌入式领域：eLua

1.2 关于 eLua

eLua 就是嵌入式 Lua，在嵌入式环境下提供了 Lua 语言的全部实现，同时扩展了一些特征以便于实现高效和可移植性的嵌入式软件开发。

eLua 是一个开源项目，其项目在：http://www.eluaproject.net/.

关于 eLua 的介绍在：http://www.eluaproject.net/overview。

eLua 提供了 Lua 桌面版的全部特征，利用 Lua 的原生机制来优化嵌入式开发。

eLua 的部分特性如下：

- 可控制整个平台：不依赖 OS，可以使用 Lua 语言实现整个平台的控制
- 源码可移植：可轻松将你的代码移植到其他架构和平台上
- 开发时 PC 端不需要安装额外的开发环境，只需要通过终端或者串口将开发主机 PC 和目标板连接起来即可。
- 可实现高度灵活性的产品：可以利用现代的脚本语言，实现产品的高适应性、可编程和重新配置。

关于 eLua 还需要明确：

- eLua 不是 OS
- eLua 不是去除 Lua 的一些东西以适应嵌入式平台，eLua 拥有 Lua 桌面版的全部功能和特性
- eLua 不是针对特定平台开发的
- eLua 不是 OS 或者 RTOS 或者特定处理器的一个应用程序，而是自包含的，支持一系列处理器架构
- eLua 是开源软件，基于 MITLiscence，可以在商业产品上使用 eLua。
- eLua 支持的硬件平台，具体可参见：http://wiki.eluaproject.net/Boards

（引自：http://blog.csdn.net/tcpipstack/article/details/8259179）

二 Lua 基本语法

Lua 的语法比较简单，对于新手，也很容易理解，对于具有一定编程经验的人非常容易上手，但是其可实现的功能却非常强大。

（本节内容参考自：http://www.cnblogs.com/thinksasa/p/4039640.html）

2.1 注释

写一个程序，总是少不了注释的。在 Lua 中，可以使用单行注释和多行注释。

单行注释中，连续两个减号"--"表示注释的开始，一直延续到行末为止。相当于 C++语言中的"//"。

多行注释中，由"--[["表示注释开始，并且一直延续到"]]"为止。这种注释相当于 C 语言中的"/*...*/"。在注释当中，"[["和"]]"是可以嵌套的（在 lua5.1 中，中括号中间是可以加若干个"="号的，如[==[...]==]），见下面的字符串表示说明。

2.2 关键字

Lua 关键字很少，不能作为变量使用。

and	break	do	else	elseif	
end	false	for	function	if	

in	local	nil	not	or	
repeat	return	then	true	until	while

2.3 变量

在 Lua 中，除了保留的少数关键字，一切都是变量。Lua 支持的数据类型有以下几种：

nil	空值，所有没有使用过的变量，都是 nil。nil 既是值，又是类型
boolean	布尔值，只有两个有效值：true 和 false
number	数值，在 Lua 里，数值相当于 C 语言的 double
string	字符串，与 C 语言中 string 类型以“\0”结尾不同，string 是可以包含"\0"字符的
table	关系表类型，这个类型功能比较强大，请参考后面的内容
function	函数类型，在 Lua 中所有的函数本身就是一个变量
userdata	该类型专门用于与 Lua 的宿主打交道。宿主通常是用 C 和 C++来编写的，在这种情况下，userdata 可以是宿主的任意数据类型，常用的有 struct 和指针
thread	线程类型，在 Lua 中没有真正的线程，Lua 中可以将一个函数分成几部份运行

在 Lua 中，不管在什么地方使用变量，都不需要声明，并且所有的这些变量总是全局变量，除非我们在前面加上“local”。这一点要特别注意，因为我们可能想在函数里使用局部变量，却忘了用 local 来说明。至于变量名字，它是大小写相关的。也就是说，A 和 a 是两个不同的变量。定义一个变量的方法就是赋值。“＝”操作就是用来赋值。以下是常用类型变量的定义方式。

（1）nil 变量

Lua 中没有使用过的变量的值，都是 nil。如果需要将一个变量清除，可以直接给变量赋以 nil 值。如：

```
var1=nil
```

Tips：使用 nil 清除变量后，可以使用“collectgarbage()”显示的回收内存。

（2）boolean 变量

布尔值通常是用在进行条件判断的时候。布尔值有两种：true 和 false。在 Lua 中，只有 false 和 nil 才被计算为 false，而所有任何其它类型的值，都是 true。比如 0，空串等，都是 true。可以直接给一个变量赋以 Boolean 类型的值，如：

```
theBoolean = true
```

（3）number 变量

在 Lua 中，没有整数类型。一般情况下，只要数值不是很大是不会产生舍入误差的（比如不超过 100,000,000,000,000）。实数的表示方法，同 C 语言类似，如：

```
40.44.57e-30.3e125e+20
```

（4）string

字符串是常见的高级变量类型。在 Lua 中，可以非常方便的定义很长的字符串。字符串在 Lua 中有几种方法来表示，最通用的方法，是用双引号或单引号来括起一个字符串的，如：

```
"That's go!"或'Hello world!'
```

和 C 语言相同的，Lua 支持一些转义字符，列表如下：

\a	响铃	Bell
\b	退格	Back space
\f	换页	Form feed
\n	换行	New line
\r	回车	Carriage return
\t	水平制表	Horizontal tab
\v	垂直制表	Vertical tab
\\	一个反斜杠字符	Back slash
\”	一个单引号字符	Double quote
\’	空字符	Single quote
\[	左中括弧	Left square bracket
\]	右中括弧	Right square bracket

通常字符串需要写在一行中，不可避免的要用到转义字符。加入了转义字符的串可阅读性较差，比如：

```
"one line\nnext line\n\"in quotes\", "in quotes""
```

一大堆的“\”符号让人看起来很倒胃口。在 Lua 中，可以用另一种表示方法：用“[[”和“]]”将多行的字符串括起来。在 lua5.1 中，在中括号中间可以加入若干个"="号，如 [==[...]==]，详见下面示例），例如下面的语句所表示的是完全相同的字符串：

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

值得注意的是，在这种字符串中，如果含有单独使用的"[["或"]]"就仍然得用"\["或"\]"来避免歧义。当然，这种情况是极少会发生的。

（5）Table 类型

Table 为关系表类型，可以把这个类型看作是一个数组。与 C 语言中数组只能用正整数来作索引不同，在 Lua 中，用户可以用任意类型作数组的索引，除了 nil。同样，在 C 语言中，数组的内容只允许一种类型；在 Lua 中，用户可以用任意类型的值来作数组的内容，除了 nil。

Table 的定义很简单，它的主要特征是用"{"和"}"来括起一系列数据元素的。比如：

```
T1 = {} --定义一个空表
T1[1]=10 --可以象 C 语言一样来使用
T1["John"]={Age=27, Gender="Male"}
```

这一句相当于：

```
T1["John"]={} --必须先定义成一个表，未定义的变量是 nil 类型
```

```
T1["John"]["Age"]=27
T1["John"]["Gender"]="Male"
```

当表的索引是字符串的时候，可以简写成：

```
T1.John={}
T1.John.Age=27
T1.John.Gender="Male"
```

或

```
T1.John{Age=27, Gender="Male"}
```

在定义表的时候，可以把所有的数据内容一起写在“{”和“}”之间，这样子是非常方便，而且很好看。比如，前面的 T1 的定义，可以这么写：

```
T1=
{
10,  -- 相当于 [1] = 10
[100] = 40,
John=  -- 如果原意还可以写成：["John"] =
{
Age=27,   -- 如果你原意，你还可以写成：["Age"] =27
Gender=Male   -- 如果你原意，你还可以写成：["Gender"] =Male
},
20  -- 相当于 [2] = 20
 }
```

在写的时候，需要注意三点：

a）所有元素之间，总是用逗号“，”隔开；

b）所有索引值都需要用“[”和“]”括起来；如果是字符串，还可以去掉引号和中括号；

c）如果不写索引，则索引就会被认为是数字，并按顺序自动从 1 往后编；

表类型的构造是如此的方便，以致于常常被人用来代替配置文件。这种方法比 ini 文件要漂亮，并且强大的多。

（6）function 函数

在 Lua 中，函数的定义很简单。

典型的定义如下：

```
function add(a,b)  -- add 是函数名字，a 和 b 是参数名字
return a+b  -- return 用来返回函数的运行结果
end
```

请注意，return 语言一定要写在 end 之前。假如我们非要在中间放上一句 return，那么就应该要写成：do return end。

前面说过，函数也是变量类型。上面的函数定义，其实相当于：

```
add = function (a,b) return a+b end
```

当重新给 add 赋值时，它就不再表示这个函数了。我们甚至可以赋给 add 任意数据，包括 nil（这样赋值为 nil，将会把该变量清除）。

和 C 语言一样，Lua 的函数可以接受可变参数个数，它同样是用“...”来定义的，比如：

```
function sum (a,b,…)
```

如果想取得“...”所代表的参数，可以在函数中访问 arg 局部变量（表类型）得（lua5.1：取

消 arg，并直接用"..."来代表可变参数了，本质还是 arg）。例如：

```
sum(1,2,3,4)
```

则在函数中，

```
a = 1, b = 2, arg = {3, 4}   (lua5.1:   a = 1, b = 2, ... = {3, 4})
```

更可贵的是，它可以同时返回多个结果，比如：

```
function s( )
return 1,2,3,4
end
a,b,c,d = s()   -- 此时，a = 1, b = 2, c = 3, d = 4
```

前面说过，表类型可以拥有任意类型的值，包括函数。因此，有一个很强大的特性是，拥有函数的表，即面向对象编程的思路。Lua 可以使用面向对象编程。举例如下：

```
t =
{
  Age = 27
  add = function(self, n) self.Age = self.Age+n end
}
print(t.Age)   -- 27
t.add(t, 10)
print(t.Age)   -- 37
```

不过，t.add(t,10) 这一句实在是有点土对吧？没关系，在 Lua 中，可以简写成：

```
t:add(10)      -- 相当于 t.add(t,10)
```

（7）userdata 和 thread

这两个类型超出了本教程的内容，详细请查阅 Lua 参考手册。

2.4 控制语句

经典的“Hello world”的程序总是被用来开始介绍一种语言。在 Lua 中，写一个这样的程序很简单：

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

在 Lua 中，语句之间可以用分号“;”隔开，也可以用空白隔开。一般来说，如果多个语句写在同一行的话，建议总是用分号隔开。

Lua 常见的几种程序控制语句如下表所示。

控制语句	格式	示例
if	if 条件 then ... elseif 条件 then ... else ... end	if 1+1=2 then print("true") elseif 1+2~=3 then print("true") else print("false") end
while	while 条件 do ... end	while 1+1~=2 do print("true") end
repeat	repeat ... until 条件	repeat print("Hello") until 1+1~=2

for	for 变量=初值, 终点值, 步进 do... end	for i = 1, 10, 2 do print(i) end
for	for 变量 1, 变量 2, ... 变量 n in 表或枚举函数 do ... end	for a,b in mylist do print(a, b) end

需要注意的是，for 的循环变量总是只作用于 for 的局部变量；当省略步进值时，for 循环会使用 1 作为步进值。

使用 break 可以用来中止一个循环。

相对 C 语言来说，Lua 有几个地方是明显不同的，所以面要特别注意一下：

（1）语句块

语句块在 C 中是用“{”和“}”括起来的，在 Lua 中，它是用 do 和 end 括起来的。例如：

```
do print("Hello") end
```

可以在 **函数** 中和 **语句块** 中定局部变量。

（2）赋值语句

赋值语句在 Lua 被强化了。它可以同时给多个变量赋值。例如：

```
a,b,c,d=1,2,3,4
```

甚至是：

```
a,b=b,a -- 多么方便的交换变量功能啊。
```

在默认情况下，变量总是认为是全局的。假如需要定义局部变量，则在第一次赋值的时候，需要用 local 说明。比如：

```
local a,b,c = 1,2,3 -- a,b,c 都是局部变量
```

（3）数值运算

和 C 语言一样，支持+, -, *, /。但 Lua 还多了一个“^”表示指数乘方运算。比如 2^3 结果为 8, 2^4 结果为 16。

连接两个字符串，可以用“..”运算符。如：

```
"This a " .. "string." -- 等于 "this a string"
```

（4）比较运算

比较符号	<	>	<=	>=	==	~=
含义	小于	大于	小于或等于	大于或等于	相等	不相等

所有这些操作符总是返回 true 或 false。对于 table，function 和 userdata 类型的数据，只有“==”和“~=”可以用。相等表示两个变量引用的是同一个数据。比如：

```
a={1,2}
b=a
print(a==b, a~=b) --输出 true,false
a={1,2}
```

```
b={1,2}
print(a==b, a~=b) --输出  false, true
```

（5）逻辑运算

```
and, or, not
```

其中，and 和 or 与 C 语言区别特别大。在这里，请先记住，在 Lua 中，只有 false 和 nil 才计算为 false，其它任何数据都计算为 true，0 也是 true！and 和 or 的运算结果不是 true 和 false，而是和它的两个操作数相关。a and b：如果 a 为 false，则返回 a；否则返回 b

a or b：如果 a 为 true，则返回 a；否则返回 b。举几个例子：

```
print(4 and 5) --输出 5
print(nil and 13) --输出 nil
print(false and 13) --输出 false
print(4 or 5) --输出 4
print(false or 5) --输出 5
```

在 Lua 中这是很有用的特性，也是比较令人混淆的特性。我们可以模拟 C 语言中的语句：

```
x = a? b : c
```

在 Lua 中，可以写成：

```
x = a and b or c。
```

最有用的语句是：

```
x = x or v
```

它相当于：

```
if not x then x = v end。
```

运算符优先级，从低到高顺序如下：

```
or
and
< > <= >= ~= ==
.. (字符串连接)
+ -
* / %（取余运算）
not #(lua5.1 取长度运算) - (一元运算)
^
```

和 C 语言一样，括号可以改变优先级。

更多关于 Lua 程序设计请参阅：

lua 程序设计：http://book.luaer.cn/http://book.luaer.cn/。

Lua5.1 参考手册：http://www.codingnow.com/2000/download/lua_manual.html。

LuaTutorial：http://lua-users.org/wiki/LuaTutorial。

三 资料汇总

1） 乐鑫 ESP8266 官方网站
http://espressif.com/zh-hans/
芯片手册地址：
2） Nodemcu：
https://github.com/nodemcu
3） NodeMCU 的 API：
https://github.com/nodemcu/nodemcu-firmware
4） Lua：
http://www.lua.org/
http://baike.baidu.com/view/416116.htm
http://zh.wikipedia.org/wiki/Lua
5） eLua:
http://www.eluaproject.net/
6） LuaLoader
http://benlo.com/esp8266/index.html#LuaLoader
https://github.com/GeoNomad/LuaLoader

更多开发教程，请阅读：基于 ESP12E Dev Kit 的开发教程.pdf