**lua5.2 改动**

1. 第三方cmodule，如使用luaL\_register需要改为 luaL\_newlib。如lfs库luaL\_register (L, "lfs", fslib) 改为luaL\_newlib(L,fslib);。  
   这里本来第二个参数是表明，非nil是把所有接口放到一个全局变量table中，nil就是所有接口都是全局函数。现在是强制取消全局接口了。
2. 所有第三方库没有gloable的函数了，所以使用时都要赋给一个表。如lfs， local lfs = require"lfs"。
3. 以前thread， function， userdata可以有env。现在没有了。
4. getfenv、setfenv没了，只能使用\_ENV，不能完全替代。比如给不同func加env不行了，以前可以传入不同函数，用setfenv给他们加上相同env的

setfenv = setfenv or function(f, t)  
     f = (type(f) == 'function' and f or debug.getinfo(f + 1, 'f').func)  
     local name  
     local up = 0  
     repeat  
         up = up + 1  
         name = debug.getupvalue(f, up)  
     until name == '\_ENV' or name == nil  
     if name then  
         debug.upvaluejoin(f, up, function() return name end, 1)  -- use unique upvalue  
         debug.setupvalue(f, up, t)  
     end  
end

1. userdata用lua\_getuservalue代替lua\_setfenv。
2. local ss = "aa/bb/cc" ss:gsub('/', '%.')  5.1能运行，5.2必须把%去掉。
3. table.maxn下个版本要去掉了
4. lua\_objlen ->lua\_rawlen
5. [module (name [, ···])](http://www.lua.org/manual/5.1/manual.html#pdf-module) deprecated

[**坏处**](http://lua-users.org/wiki/LuaModuleFunctionCritiqued)**：**

1.使用package.seeall会破坏模块内聚性，有可能随意访问或改变全局变量。

2.直接把包以指定名字（可以AA.BB.CC的名字）加到了全局表。  
原来的module("mymodule")等同于：  
local modname = “mymodule”     – 定义模块名   
local M = {}                               -- 定义用于返回的模块表    
\_G[modname] = M                      -- 将模块表加入到全局变量中    
package.loaded[modname] = M    -- 将模块表加入到package.loaded中，防止多次加载    
setfenv(1,M)                               -- 将模块表设置为函数的环境表，这使得模块中的所有操作是以在模块表中的，这样定义函数就直接定义在模块表中   
新的方式：  
local base = \_ENV  
local modname = {}  
local \_ENV = modname  
return modname

1. local socket = require("socket.core")  
   module("socket") 如luasocket这样的定义在5.1中，socket是 socket.core返回的table加上本module内定义的接口。5.2中设置了兼容，能用module，但是socket.core中的table直接被空table覆盖，不会暴露。
2. #define LUA\_GLOBALSINDEX    LUA\_RIDX\_GLOBALS  
   #define luaL\_reg         luaL\_Reg  
   #define luaL\_putchar(B,c)   luaL\_addchar(B,c)  
   #define lua\_open         luaL\_newstate
3. luaL\_typerror没了，改为

LUALIB\_API int luaL\_typerror (lua\_State \*L, int narg, const char \*tname) {  
  const char \*msg = lua\_pushfstring(L, "%s expected, got %s",tname, luaL\_typename(L, narg));  
  return luaL\_argerror(L, narg, msg);  
}

1. luaL\_openlib(L, NULL, func, 0);   => luaL\_setfuncs(L, func, 0);