1 建立动态库，与 lua 无关，库中的函数和全局变量供 luajit 调用

mylib.h

[html] view plain copy

```
#ifdef __cplusplus
extern "C" {
#endif
  typedef struct{
   int len;
   char value[128];
  }MY_STR;

  typedef int (* FPROC)(MY_STR[32]);
  void printmy();
  void testAdd();
  void testNoHFile();/*如果没有放在 extern 中，则无法在 lua 中使用函数。*/

  int g_directOpr=0;
#ifdef __cplusplus
}
#endif
```

mylib.cpp

[html] view plain copy

```
extern "C"
{
 #include <stdlib.h>
 #include <stdio.h>
 #include <string.h>
 #include <errno.h>
 #include "mylib.h"
}

/*全局变量 即使不在头文件声明，也可以直接在 lua 中直接使用*/
int g_Num=100;
void printmy(){
     printf("lib print [%d]\n",g_Num);
}

void testAdd(){
   g_Num++;
}

void testNoHFile(){
    printf("test no head file \n");
}
/*直接在函数前使用 extern C,则可以直接在 lua 中访问了。不必在头文件中。*/
```

```
extern "C"  void testNoHFile2(){
    printf("test no head file 2 \n");
}
```

2 建立一个 bin，与 lua 有关
[html] view plain copy

```
extern "C"
{
 #include <stdlib.h>
 #include <string.h>
 #include <errno.h>
 #include "mylib.h"

#include "/data/project/LuaJIT-2.1.0-beta2/src/lua.h"
#include "/data/project/LuaJIT-2.1.0-beta2/src/lualib.h"
#include "/data/project/LuaJIT-2.1.0-beta2/src/lauxlib.h"
}

/*int g_Num=100;定义全局变量与 mylib.so 中重复，则 lua 中直接无法操作 g_Num*/

int g_clientBinNum=1000;

int main()
{
    lua_State *L=luaL_newstate(); /*创建一个解释器句柄*/
    luaL_openlibs(L);            /*打开所有的 Lua 库*/

     luaL_loadfile(L,"script.lua"); /*调入 Lua 脚本文件*/

     lua_pcall(L,0,0,0); /*执行 Lua 脚本*/
     lua_close(L);       /*关闭句柄*/
     return 0;
}
```

3 建立 lua 脚本
[html] view plain copy

```
print("case2 lua 访问 C 中的函数和全局变量")
--三种访问库的方式都可以
--local mylib = ffi.load("/data/mytest/src/test/libmylib.so");
--local mylib = ffi.load("./libmylib.so");
local mylib = ffi.load("libmylib.so");
print("test so global variable-g_Num direct");
print(mylib.g_Num);
mylib.g_Num=mylib.g_Num+1;
print(mylib.g_Num);

print("test so global variable-g_directOpr direct");
print(mylib.g_directOpr);
mylib.g_directOpr=mylib.g_directOpr+1;
```

```
print(mylib.g_directOpr);
mylib.g_directOpr=mylib.g_directOpr+10;
print(mylib.g_directOpr);


print("test so global variable- g_Num test by func");
mylib.printmy();
mylib.testAdd();
mylib.printmy();
mylib.testAdd();
mylib.printmy();

print("test bin global variable-int g_clientBinNum");
print(g_clientBinNum);--显示 nil

mylib.testNoHFile();
mylib.testNoHFile2();
```

1 luajit 直接支持 C，所以如果使用 C++的.cpp，则需要在函数前使用 extern C

2 ffi.load 中可以直接配置路径，具体见 api 说明

http://luajit.org/ext_ffi_api.html
3 luajit 使用动态库中的变量和函数。但没有找到直接使用 bin 程序中程序和变量的简单方法。