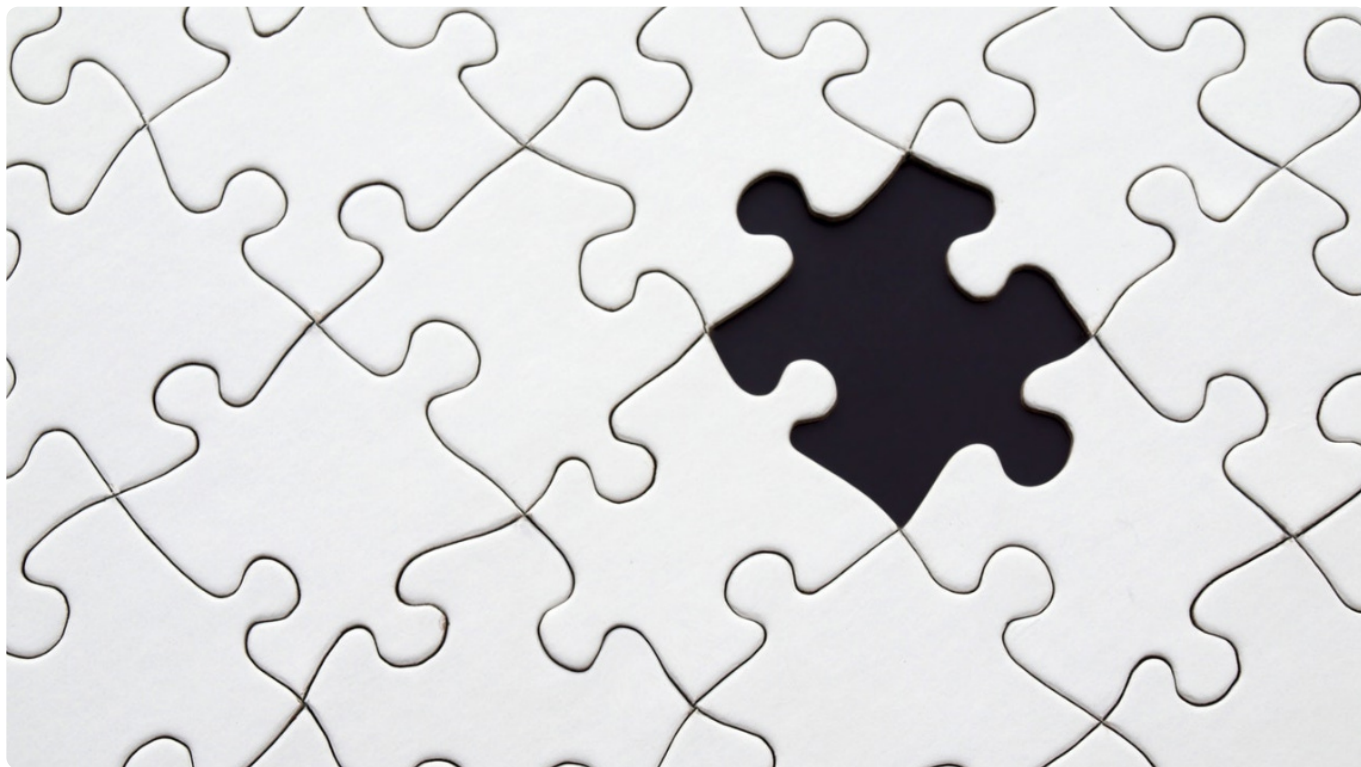


第24讲 | 如何嵌入脚本语言？

2018-07-19 蔡能

从0开始学游戏开发

[进入课程 >](#)



讲述：蔡能

时长 09:25 大小 4.32M



从 2005 年开始，逐渐流行使用 C/C++ 语言结合脚本语言（Lua、Python、Ruby 等等）编写游戏。这是因为用 C/C++ 编写游戏的传统方式，硬代码太多，而使用硬代码编写的游戏，更新难度很大，除非重新编译一次程序。

于是，就有人开始使用配置文件来做活动逻辑。比如填写好配置表、玩家等级多少、攻击力如何、等于多少的伤害力等等，一开始就将这些内容都读取进代码，在游戏中实时计算出来。

但是这种方法其实也并不方便。很久以前的游戏，由于硬件资源限制，所以一般都加载 WAV 格式。而加载 MP3 则需要机器对音乐文件进行解压缩再播放，如果机器硬件计算能力不好的话，会由于解压缩而导致整个游戏的运行效率下降。

脚本语言也是如此，如果机器硬件能力不好的话，会由于脚本语言的虚拟机要解释程序，导致游戏运行效率下降。随着电脑硬件的提升，我们在游戏中加载 MP3 音乐文件成为可能，而在游戏中加载脚本语言进行逻辑编写当然也是可以的。

《魔兽世界》就是使用 Lua 脚本语言编写的。类似《GTA》等大型游戏，都拥有一套自己的脚本语言和体系。 **使用脚本语言，是为了能够在编写硬代码的同时，也能很方便地、不需要重新编译地编写逻辑代码。** 事实上，现在很多大型游戏都使用这种方式来编写代码，甚至一些游戏引擎本身，也支持脚本语言和引擎本身所提供的语言分离编写。比如引擎用 C++ 语言编写，脚本语言用 Lua 编写。

为什么使用 Lua 脚本嵌入 C/C++ 硬代码？

今天我就来教你使用 Lua 脚本来嵌入 C/C++ 硬代码。为什么我要选择 Lua 脚本语言来编写代码呢？

因为**Lua 脚本足够轻量级，几乎没有冗余的代码。Lua 虚拟机的执行效率几乎可以媲美 C/C++ 的执行效率。**如果选择 Python、Ruby 等常用脚本语言来嵌入，并不是不行，而是要付出执行效率作为代价。因为 Python、Ruby 的执行效率远逊于 Lua。

如果没有非常多的编码经验，你可能会问，为什么 Python、Ruby 的执行效率远逊于 Lua 呢？这个问题，用一本书的篇幅恐怕才能彻底讲明白。我这里只简要说一下原因。

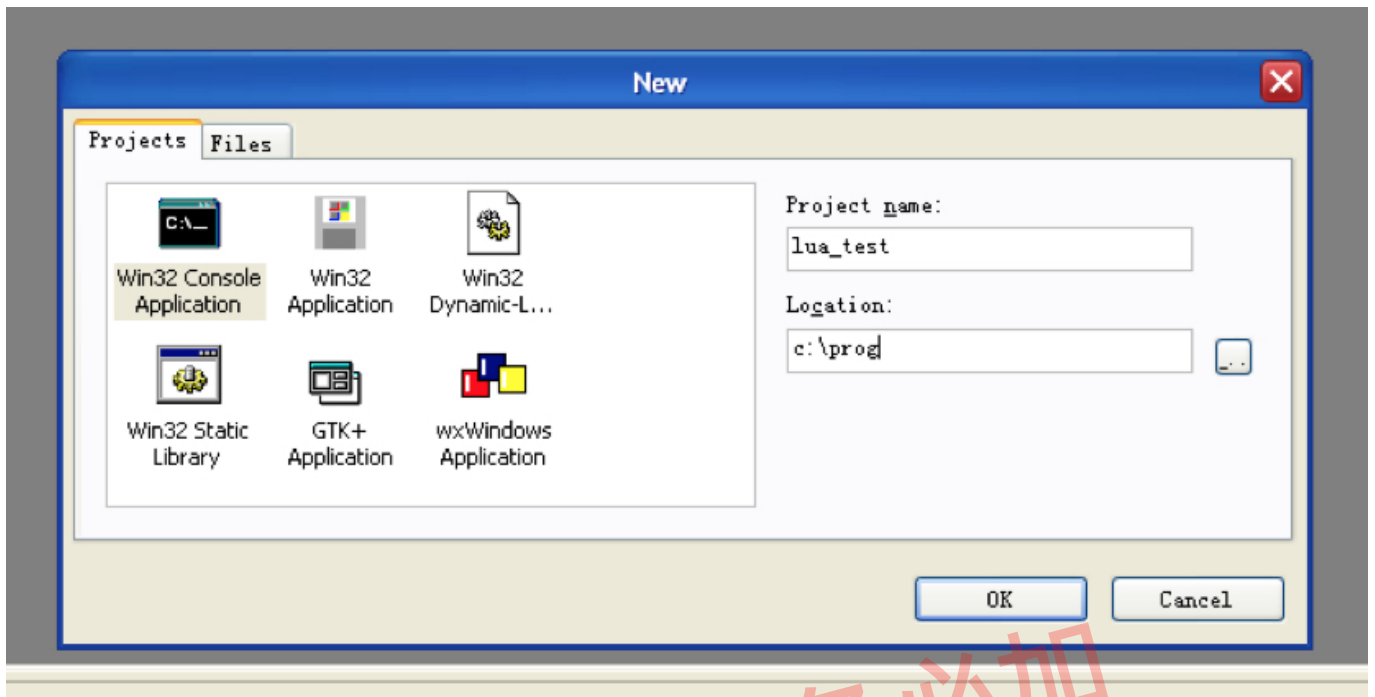
Lua 的虚拟机很简单，指令设计得也精简，Lua 本身是基于寄存器的虚拟机实现，而 Python 等其他脚本语言是基于堆栈的虚拟机，而基于寄存器的虚拟机字节码更简单、高效。因为字节码，一般会同时包含指令、操作数、操作目标等内容。

另一方面，Python、Ruby 之所以应用范围广，是因为它们拥有大量的成熟库和框架，而 Lua 只是一种很纯粹的脚本语言。因为 Lua 没有过多的第三方库，只提供最基础的 I/O 处理、数学运算处理、字符串处理等，别的与操作系统相关度密切的，例如网络、多线程、音频视频处理等等都不提供。

我在[第 6 讲](#)里，已经非常详细地讲过，如何将 Lua 脚本编译成为静态库，如果不记得的话，可以回去复习一下。编译好静态库 liblua.a 之后，我们就可以在编程中使用它了。


你也可以选择在解压缩出来的目录内，使用 make 命令来直接编译，编译会生成 Lua 虚拟机的执行文件 lua.exe、luac.exe，当然这需要一整套 MinGW 的环境支持。

开始，我们还是使用 MinGW Development Studio 来创建一个工程。由于只是示例，所以名字可以任意取。我取一个叫作 lua_test 的工程名，并且将工程设置为 Win32 Console Application。你可以看这个示例图。



建立好了工程之后，我们新建一个 test.c 文件。这个文件位于 lua 源代码路径下。我们将 liblua.a 文件也放到同一个目录下，以方便后续链接时候调用。


在包含 Lua 头文件之前，我们需要将头文件写在某一个.hpp 文件下，以便一次性包含进去，我们的代码可以这么写。

 复制代码

```
1 #ifdef __cplusplus
2 extern "C" {
3 #endif
4 #include "src/lua.h"
5 #include "src/lualib.h"
6 #include "src/lauxlib.h"
7 #ifdef __cplusplus
8 }
9 #endif
```

你可以看到，这里面包含了三个代码。这三个代码来自 src 目录下，其中最后一个 lauxlib.h 包含了大量的 C 语言形式的接口以及扩展接口。而定义 extern "C" 的意思是，使用 C 的方式进行链接，前置条件是，你的语言是 C++ 语言 (ifdef __cplusplus)。

定义好了这个 hpp 文件后，我们可以在 C 或者 C++ 语言中进行包含。

 复制代码

```
1 #include "lua.hpp"
```

你需要了解三个 Lua 语言的细节问题


写完定义之后，我们就可以开始对 Lua 进行一系列的绑定操作了。在编程之前，我先用一些你能看得懂的语言，对 Lua 语言的细节进行一些描述。有三个点，需要你着重记一下。

首先，**Lua 的下标都是以 1 为最初始的值**（当然反向可以使用 -1 为下标），而不是我们所熟悉的 0。有个传言说，是因为作者当时编写最初版本的 Lua 时，计算错误才导致的，所以就这么一直沿用下来了，这个说法虽然不可考，但也算是一种解释。

其次，在 C/C++ 内嵌 Lua 的做法中，**Lua 有两种读取脚本的方法**。

一种方式是**读取后直接运行，调用的函数是 luaL_dofile**。使用这个函数，脚本会在读取完毕后直接运行。当然如果出现错误，你也不知道错误的具体位置在哪里，调试起来不是很方便。

第二种方式是**将脚本代码压到栈顶，然后使用 pcall 操作运行脚本，这个函数叫 luaL_loadfile**。事实上第一种方式也是使用这种方式并且将 pcall 操作直接调用起来，第一种方式的代码一看你就能明了。

 复制代码

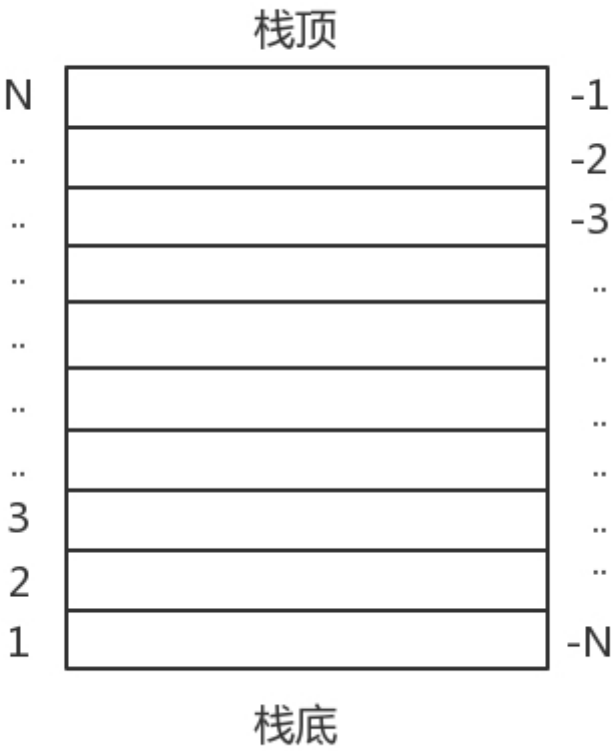
```
1 #define luaL_dofile(L, fn) \
2     (luaL_loadfile(L, fn) || lua_pcall(L, 0, LUA_MULTRET, 0))
```

这行代码在 lauxlib.h 中能找到。这段代码写得非常精妙，它的意思是，如果 loadfile 成功，那么就运行 pcall 函数，中间这个 ||（或者）已经直接判断了 loadfile 是否成功。因为 loadfile 函数操作成功就返回 0，否则就返回 1。

而在“或者”这个逻辑判断下，只要是 0，就继续往下判断；只要是 1，就直接返回条件为真。所以，在这行代码下，只要是 1，就中断 dofile 这个宏的操作；只要是 0，就进行

pcall 操作。


最后，我要说一下**Lua 的堆栈**。理解了堆栈的计数方式，就能很容易地理解我后续要讲解的代码中的计数方式。Lua 的堆栈可以从这个图里看出来，从栈底往上表示可以用 1、2、3、4、5，而从栈顶往下表示是 -1、-2、-3、-4、-5。



如何使用 Lua 以及 liblua.a 来进行与 C 语言的绑定操作？

我们现在开始使用 Lua 以及 liblua.a 来进行与 C 语言的绑定操作。


首先，我们需要包含之前我们所定义的 lua.hpp 头文件，随后我们开始在 main 入口函数处，定义一些变量。

 复制代码

```
1 #include "lua.hpp"
2 int main(int argc, char ** argv)
3 {
4     int r;
5     const char* err;
6     lua_State* ls;
7     ...
8 }
```

在这里，我们定义了三个变量，其中 `r` 是用来接收返回值的；`err` 是一个常量字符串，用来接收错误字符串并打印出来；而 `lua_State* ls` 就是 Lua 虚拟机的指针了。


我们再来看接下来的代码。

 复制代码

```
1 ls = luaL_newstate();
2 luaL_openlibs(ls);
```

在这两行代码中，首先初始化一个虚拟机（在 Lua 5.1 中，使用的函数是 `lua_open` 来新建虚拟机），并且将虚拟机地址赋值给 `ls` 指针。随后，我们拿到这个指针之后，就在之后的代码中“打开”Lua 所需要用到的各种库。我们用到 `luaL_openlibs`。我现在只是给你示范，你可以一个一个库单独打开。

我们新建了虚拟机，并且打开了 Lua 类库。我们继续看下面的代码。

 复制代码

```
1 r = luaL_loadfile(ls, argv[1]);
2 if(r)
3 {
4     err = lua_tostring(ls, -1);
5     if(err)
6         printf("err1: %s\n", err);
7     return 1;
8 }
9 r = lua_pcall(ls, 0, 0, 0);
10 if(r)
11 {
12     err = lua_tostring(ls, -1);
13     if(err)
14         printf("err2: %s\n", err);
15     return 1;
16 }
17 lua_close(ls);
```


我来具体解释一下。这段代码中，`argv[1]` 的是命令行输入的第一个内容。比如我们的程序叫 `lua_test`，那么我们在 Windows 命令行中，输入 `lua_test a.lua`，那么其中 `a.lua` 就是 `argv[1]` 这个内容。

`luaL_loadfile` 我们在前面介绍过，就是载入文件并不运行。当然在这个期间，它会检查基础的语法。如果你少一个括号或者多一个引号，就会在这个时候给你一个错误信息，这个错误信息就是利用 `r` 这个变量判断的。如果 `r` 的返回值不等于 0 的话，那就是出错了。出错的时候，Lua 会将出错信息压栈顶，而栈顶是从 -1 开始表示，所以我们要取出栈顶的错误信息 `lua_tostring(ls, -1);`，并且将它赋值给 `err`，最后由 `err` 打印出来。

认为没有错误之后，就是过了这一关。第二关我们需要使用 `lua_pcall` 函数，来调用 Lua 脚本文件，其中第一个参数是虚拟机指针，第二个参数是传递多少参数给 Lua，第三个参数是这个脚本返回多少值，第四个是错误处理函数，可以是 0，那就是无处理函数。

`pcall` 的返回值也是一样，如果不是 0 的话，就说明出错了。和之前的 `luaL_loadfile` 不同，这时候一般是运行时错误，比如运行时类型错误等等。同样的，`pcall` 也会把错误信息压到栈顶，我们直接去将栈顶的内容转成 `string` 就可以打印出来了。最后，我们将 Lua 虚拟机通过 `lua_close` 关闭。

按常理来说，我们现在可以来运行一下效果了，你可以先等等，我们先写一段错误的 Lua 代码，来看看执行起来会发生什么情况。

 复制代码

```
1 print "test running")
```

我们故意少写一个括号，然后将源代码命名为 `a.lua`，我们来运行看看。会出现一个这样的错误信息：

```
err1: a.lua:2: unexpected symbol near '>'
```

在发现语法错误后，程序就会报错，另外，如果你输入了一个根本不存在的文件，比如我们这么运行，`test_lua xxx.lua`，也会在 `loadfile` 的时候出错。

小结

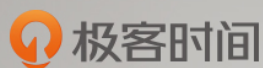
我们今天的内容就到这里。下次我会进一步把 Lua 的脚本嵌入的细节呈现在你面前。我们来总结一下今天的内容。

1. 因为 Lua 脚本足够轻量级，几乎没有冗余的代码。Lua 虚拟机的执行效率几乎可以媲美 C/C++ 的执行效率。所以我们选择使用 Lua 脚本来嵌入 C/C++ 硬代码。
2. Lua 脚本在 C/C++ 语言里面嵌入，需要先声明一个虚拟机并且赋值给指针。
3. Lua 脚本需要先 loadfile 再 pcall 调用脚本文件，loadfile 会检查最基本的脚本文件内容，比如文件是否存在，比如脚本代码是否出错，而 pcall 会在运行时出错的时候将错误压至栈顶。
4. Lua 错误会将错误压制栈顶，我们要取出来，需要使用 -1 下标取出栈顶的内容，并转成 string 打印。

给你留一个小问题吧。

如果直接使用 luaL_dofile，相对于把 loadfile 和 pcall 分开写，这样有什么优劣呢？

欢迎留言说出你的看法。我在下一节的挑战中等你！



从 0 开始学游戏开发

你的游戏开发入门第一课

蔡能 原网易游戏引擎架构师
资深游戏底层技术专家



新版升级：点击「👤 请朋友读」，10位好友免费读，邀请订阅更有**现金**奖励。

上一篇 复习课 | 带你梳理客户端开发的三个重点

下一篇 第25讲 | 热点剖析（六）：AR和人工智能在游戏领域有哪些应用？

精选留言 (4)

写留言



wusiration

2018-07-30

1

直接使用luaL_dofile无法体现出错误出现在哪，而且是加载lua文件并运行；而用luaL_loadfile和pcall则会体现出语法错误或者是逻辑错误，并将lua文件的加载和运行分开。

展开



宋桓公

2018-08-02

1

编译失败，找不到那些函数
请问这里面不用写关联.a 这个库的代码吗？

作者回复: 要写的，在IDE里面设置库文件



Geek_King@...

2018-07-21

1

想请教一下lua和c的变量怎么共享呢

展开



云学

2018-07-19

1

今天讲解的是在C中调用lua，请问可以在lua中调用c++程序吗

展开

作者回复: 你得说明白是调用执行文件？还是调用类库？

