

第26讲 | 脚本语言在游戏开发中有哪些应用？

2018-07-24 蔡能

从0开始学游戏开发

[进入课程 >](#)



讲述：蔡能

时长 06:31 大小 2.99M



上一次，我们谈到了如何在游戏中嵌入脚本语言，我们用的语言是 Lua。Lua 语言具有轻量级、速度快的特点，而且 API 的调用也非常方便和直观。现在，我们仍然拿 Lua 脚本，试着把它应用在我们开发的游戏。


我们使用 C 语言来对 Lua 脚本的绑定做一次深入的剖析，然后来看一下，在游戏开发中绑定了脚本语言后，脚本语言能做什么事情。

首先，我们要明白一点，事实上**任何模块都可以使用脚本语言编写**。当然在游戏开发的过程中，需要分工明确，如果不分工的话，效率可能会比较低。

在需要某些效率要求非常高的情况下，一般是用 C、C++ 或 ASM 语言，将底层模块搭建好，然后将一些逻辑部分分出来，给脚本语言处理。比如我们熟知的服务器端，可以使用

C/C++ 来编写服务器端的 IOCP 或者 epoll 处理；而接收、发送、逻辑处理等等，都可以使用绑定脚本的方式编写。

我们在编写的过程中，需要对 C/C++ 的语言和代码有个了解，我们需要先考虑这个函数。

 复制代码

```
1 int test_func(lua_State *L)
2 {
3     return 0;
4 }
```

这只是一个空的 C 函数，在这个函数里面，我们看到它的传入参数是 lua_State，接受一个指针 L。随后，这个函数返回一个 0。

lua_State 是 Lua 虚拟机的对象指针，也就是我们需要把前面 new 出来的一个虚拟机传进去，才可以保证在这个函数里面，使用的是一致的虚拟机。

这个函数的作用是，**只要注册到了 Lua 虚拟机里面，它就是 lua 的一个函数，其中在 lua 函数中，传入的参数由函数内部决定。**

比如我可以这么写：


 复制代码

```
1 int test_func(lua_State *L)
2 {
3     const char *p1 = lua_tostring(L, 1);
4     const char *p2 = lua_tostring(L, 2);
5     // .... do something
6     lua_pushstring(L, "something");
7     return 1;
8 }
```

这里面，lua_tosting 就是这个函数的传入参数，传入的是一个字符串的参数；第二个参数也是字符串参数，其中 lua_tosting 的第二个参数 1 或者 2，表明的是在 Lua 虚拟机的堆栈中从栈底到栈顶开始计数，一般先压入的参数在第一个，后压入的在第二个，以此类推。

返回 1 的意思是，这个函数会返回一个参数，这个参数就是我们前面 lua_pushstring 后压入的这个内容 something，这就是返回的参数。

那么这个函数究竟怎么注册成为 Lua 函数呢？我们来看这段代码。

 复制代码

```
1 lua_register(L, "test", &test_func);
```


lua_register 函数的功能是，注册 C 函数到 Lua 虚拟机。其中 L 是虚拟机指针。这个在前面的代码都有说到，而第二个参数 test 就是注册在 Lua 虚拟机中的函数名，所以这个函数名叫 test。第三个参数是函数指针，我们把 test_func 这个函数传入到 lua_register 函数中。这样，一个函数就注册好了。

那么，如果我们在游戏中有许多许多的函数需要注册到 Lua 中，那么这种写法是不是太慢了，有没有一种快捷的写法来支持注册等操作呢？

如果你没有 C/C++ 的语言基础，或者 C/C++ 语言基础比较薄弱，下面的内容可能需要花一点时间消化，我也会竭尽所能解释清楚代码的意思，但如果你已经是个 C/C++ 程序员，那么下面的代码对你来说应该不会太难。

我们需要使用 lua_register，我们先看它里面有什么参数。第一个是**字符串**，也就是 **char***；第二个是**函数指针**，也就是 **int ()(lua_State)** 这种形式的。

那么，我们需要定义一个 struct 结构，这个结构可以这么写：


 复制代码

```
1  #define _max 256
2  typedef struct _ph_func
3  {
4      char ph_name[_max];
5      int (*ph_p_func)(lua_State*);
6  } ph_func;
7
```

我们定义了一个 struct 结构，这个结构的名字叫 `_ph_func`，名字叫什么并没有关系，但是最开始有一个 `typedef`，这说明在这个结构声明完后，接下来最后一行 `ph_func` 就是替代最初定义的那个 `_ph_func` 的名字，替代的结果是，**`ph_func` 等同于 `struct _ph_func`**，这在很多 C 语言的代码里面经常能见到。

接下来，我们看到 `char ph_name[_max]`。其中 `_max` 的值为 256。我相信你应该能理解这句话。第二个变量就是我们所看到的函数指针，其中 `ph_p_func` 是函数指针，其中函数指针指向的内容目前暂时还没有确定，我们将在后续初始化这个结构变量的时候进行赋值。

我们来仔细看一下这两段宏的内容。

 复制代码

```
1 #define func_reg(fname) #fname, &ph_##fname
2 #define func_lua(fname) int ph_##fname(lua_State* L)
```

其中 `func_reg` 是在给前面那个结构体初始化赋值的时候使用的，因为我们知道，如果我们需要给这个结构体赋值，看起来的代码是这样：

 复制代码

```
1 ph_func pobj = {"test", &test_func};
```

那么由于我们有大量的函数需要注册，所以我们将之拆分为宏，其中 `#fname` 的意思是，将 `fname` 变为字符串，而 `ph_##fname` 的意思是使用 `##` 字符，将前后内容连接起来。

通过这个宏，比如我们输入一个 `a` 赋值给 `fname`，那么 `#fname` 就变成字符串 `"a"`，通过 `ph_##fname`，结果就是 `ph_a`。

接下来的代码，是方便在代码中编写一个一个 lua 注册函数用的，所以很明显，和上述的宏一样，我们只需要输入 `a`，那么这个函数就变成了 `int ph_a(lua_State* L)`；

定义好了这两个宏，我们怎么来应用呢？


 复制代码

```

1 func_lua(test_func);
2
3 ph_func p_funcs[] =
4 {
5     { func_reg(test_func) },
6 };
7 func_lua(test_func)
8 {
9     const char *p1 = lua_tostring(L, 1);
10    const char *p2 = lua_tostring(L, 2);
11    // .... do something
12    lua_pushstring(L, "something");
13    return 1;
14 }
15 void register_func(lua_State* L)
16 {
17     int i;
18     for(i=0; i<sizeof(p_funcs)/sizeof(p_funcs[0]); i++)
19         lua_register(L, p_funcs[i].ph_name, p_funcs[i].ph_p_func);
20 }

```

首先，联系上面的宏，第一行代码是使用 `func_lua`，所以 `func_lua` 输入的宏参数是 `test_func`。于是，通过这个宏，我们最终得到的函数名字是 `int ph_test_func(lua_State* L);`。

 复制代码


```

1 ph_func p_funcs[] =
2 {
3     { func_reg(test_func) },
4 };

```

这段代码，使用的是 `func_reg` 的宏。`test_func` 最终在宏里面，变成了 “`test_func`”，以及 `&ph_test_func` 函数指针。

最后我们来看一个重要的函数，**`register_func`**，这个函数在后续将会输入一个 Lua 虚拟机指针，但是我们要知道它在函数内部它做了什么东西。

 复制代码

```

1 int i;
2 for(i=0; i<sizeof(p_funcs)/sizeof(p_funcs[0]); i++)

```

在循环里面，我们计算 p_funcs 的结构数组的长度，怎么计算的呢？

首先，我们使用 sizeof 编译器内置函数，来取得 p_funcs 整个数组的长度，这个长度等于 sizeof(ph_func) 的值乘以数组长度。而 ph_func 结构体拥有一个字符串数组，每个数组长度是 256，加上一个函数指针为 4 字节长，所以长度是 260。而如果有两个数组元素，那就是 520 的长度。

以此类推，/sizeof(p_funcs[0]) 的意思是，我们取出第一个数组的长度作为被除数。事实上就是结构体本身的长度，所以就是结构体数组总长度除以结构体长度，就是一共有多少数组元素，随后进行循环。

在循环的过程中，我们看到，我们填入了结构体里面的两个变量 ph_name 以及 ph_p_func，这样一来，我们只需要通过宏加上一些小技巧，就可以把 Lua 的函数都注册到 C 程序里面，我们假设这个 C 程序就是游戏的话，那么我们很容易就可以和 Lua 进行互通了。

小结

我总结一下今天所讲的内容。

在 Lua 与 C 的结合过程中，C 语言需要新建一个 Lua 虚拟机，然后使用虚拟机的指针来操作 Lua 函数。

在程序的应用中，使用 C 语言中的一些宏的技巧，可以使代码能够便利地应用在程序里。

最后，给你留一个小问题。

如果使用 Lua 往 C 语言传递一些内容，比如从 C 语言获取 Lua 脚本中某个变量的值，应该怎么做？

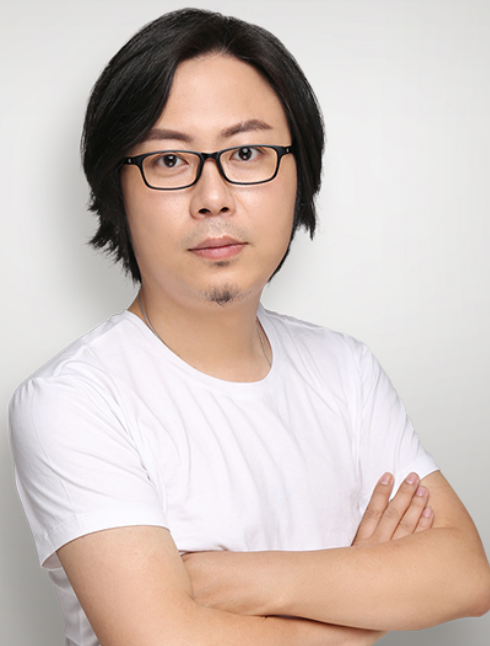
欢迎留言说出你的看法。我在下一节的挑战中等你！

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。

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 第25讲 | 热点剖析（六）：AR和人工智能在游戏领域有哪些应用？

下一篇 第27讲 | 如何使用脚本语言编写周边工具？

精选留言 (1)

写留言



云学

2018-07-26



test_fun里的dosomething是在调用lua脚本中的函数吗？p1 .p2是入参？

展开