学习Lua需要的知识

知识图谱

扎实的C语言功底 一定的编译原理知识（词法分析、语法分析、递归下降分析、BNF规则等）

后面我会针对知识图谱中的知识点，类似LeetCode题库，设计一些编程实例，确保一件事：如果读者能够把这个小题库刷一遍再学Lua内核，便能够无阻力地进行下去。

Lua特点总结（要求每一点都能举至少一个切实的例子）

1. 如果只使用静态语言，比如C++，在写大型项目时，比如游戏后台，每次更新，都要编译，关服务器，刷程序，如果测试中遇到问题，又要从新进行上述过程。使用纯静态语言开发的情况，有相当一部分时间并没有用在业务逻辑的开发中。如果核心层用C/C++实现，暴露接口给Lua虚拟机，这样所有业务逻辑的开发和修改都可以动态修改了。热更新。

2. 在设计上非常克制，核心部分的代码也就一万行左右，这样一门广泛使用的工业级脚本语言，只需要吃透一万行代码就能明白其核心原理，学习的性价比很高，诱惑力很大。

3. 简洁、高效、可移植、可嵌入、可扩展，做得很好。

4. Lua，专注一个配角，胶水语言，C/C++攻城拔寨，Lua在后方完成辅助。在现在大部分主流语言都走大而全的路线时，在号称学会某一门语言就能成为所谓的“全栈工程师”的年代，Lua始终恪守自己做好胶水语言的本职工作。“上善若水，水善利万物而不争”，简单、极致、强大的可扩展性，Lua语言设计哲学。

5. 适用对象：Lua程序开发人员，对Lua内部实现感兴趣的人，对程序语言设计感兴趣的人。

前世今生

自己查去（书、官网、作者采访、作者演讲等）

一个Lua的用法案例 游戏服务器

（1）C/C++实现服务器的引擎内核，包括所有核心的功能，比如网络收发、数据库操作、游戏主逻辑循环。引擎层。

（2）向引擎层注册一个Lua主逻辑脚本，当接收到用户数据时，将数据包放到Lua脚本中进行处里，主逻辑脚本主要是一个大的函数表，可以根据接收到的协议包的类型，调用相关的函数进行处里。脚本层。

（3）引擎层向脚本层提供很多的API，能方便地调用引擎层的操作。

引擎层的变动相对不那么频繁，而游戏的逻辑、玩法是变动很频繁的，这部分使用脚本来完成。这个组合架构的优势在于：

（1）编码效率高：脚本经常要变，不需要编译就能直接运行，省去了很多编译的时间。

（2）更新效率高：Lua支持热更新，不用停服务器就能调试新的脚本代码，省去了服务重启的时间，比如加载数据库数据、静态配置文件。

（3）对游戏开发人员的素质要求相对低：主程级别的人把控引擎层，其它成员编写脚本玩法逻辑，即使出错，大部分时候也不会导致服务器宕机等严重问题。

1和2 Lua语言基础

（1）print “heihei” 和 print( “heihei” ) 一样

（2）从控制台读取一个整数：

a = io.read(“\*n”)

（3）chunk 解释器进行解释执行的单元

（4）stand-alone interpreter interactive mode

（5）退出交互环境：Ctrl+Z 或 os.exit()

（6）以往，交互模式下，用户只能输入命令，不能直接输入表达式，比如a，必须要加符号=。但5.3版本开始，用户可以了。lua会输出表达式的值，例如：

> math.pi/4 -->3.1415926535898

> a = 15

>a -->15

>a^2 -->25

与MATLAB的控制台一样。

要以代码段的形式运行代码，直接用表达式，是不会输出相关内容的，要用print。

在交互模式下，lua解释器会把我们输入的一行，当做程序块来解释执行，赋值表达式，表达式。

如果我们输入的一行，作为代码块或表达式，不完整，那么解释器会等输入完整后再进行解释执行。

（7）交互模式，是测试程序的一个很好的手段，交互模式下，可以调用dofile(“path”)进行函数和变量的加载，然后可以逐个调用它们，测试它们。

（7）另一种运行程序的方式：加载文件（加载库）

dofile( “lib1.lua” )

这种方式可以热更新库文件，更新虚拟机内部变量（包括函数）

（8）词法规范

变量名 下划线+大写字母（\_VERSION）被Lua用作特殊用途，应该避免使用这种变量名。

下划线+小写字母，用作哑变量（dummy variable）。

Lua的保留字：in、local、until、repeat

（9）Lua的if else

if then-elseif-else-end

Lua的 for

for i=1, 3 do end

（10）Lua代码块的注释

行注释 --

多行注释 –[[ ]]

代码段注释技巧 ---[[ --]]

（11）全局变量无需声明即可使用。使用未经声明的全局变量，不会导致错误，其值为nil。

> b --> nil

> b = 10

> b --> 10

当把nil赋值给全局变量时，Lua会回收该变量，仿佛从未出现过一样

> b = nil

> b --> nil

（12）变量类型

Lua是动态类型语言。没有type definition，与MATLAB一样。每个变量内部自带类型信息。不需要类型转换语句。

8种基本类型：

nil, boolean, number, string, userdata, function, thread, table.

可以使用type(变量) 获取变量的类型。

（13）nil

nil，nil类型，值只有nil。它的主要作用是与其他所有值进行区分。

将nil赋值给变量，相当于删除这个变量。

（14）boolean

这个类型，只有两个值，true和false。

boolean值并非是条件测试的唯一方式，任何值都可以表示条件。Lua语言的条件测试，将除boolean的false和nil外的所有值视为真，特别的是，Lua语言会把0和空字符串也都视为真。逻辑运算同理。

可以这样记忆：

“false” --> false 或者 nil

“true” --> 非 “false”

（15）常见的逻辑运算符 and、or、not

and和or遵循短路求值原则（short-circuit evaluation），即只在必要时才对第二个操作数进行求值。

如果一个and或or的表达式中有错误的表达式，但是程序在执行时，由于短路原则而没有执行到错误的表达式语句，那么就不会出现运行时报错的情况。

Lua中，and、or逻辑运算的结果并非boolean类型的值。not和关系运算的结果是boolean。

and和or的运算结果为一个操作数，根据短路原则，计算到了哪个操作数处，就返回那个操作数。

> 4 and 5 -->5

> nil and 13 -->nil

> 0 or 5 -->0

Lua语言的惯用写法：x = x or v

if not x then x = v end （假设x不是boolean的false）

Lua语言的惯用写法：(a and b or c)，and运算的优先级高于or

相当于C语言中的三目运算符（a?b:c）

（x>y）and x or y

not运算符永远返回boolean类型的值

> not nil --> true

> not not 1 --> true

表达式 （x and y and (not z)）or ((not y) and x) 中括号不是必须的，但是从代码可读性的角度来讲，建议使用括号。

（16）在交互模式下，如果用户输入表达式或者函数，Lua5.3以后的版本会输出结果，在早期的版本中要加=号才输出结果，如果不想输出结果，可以加分号； 和MATLAB一样。分号可以让表达式成为无效表达式，但依然是有效的命令。

>io.flush() -->true

>io.flush();

（17）命令：lua -i test.lua 1 2 “he”

会去读取test.lua文件，并执行。在解释执行前，Lua会把其后的参数，作为字符串，赋给arg变量，test.lua脚本中可以使用它们。类似main(int argc, char \*argv[ ])。

其中arg[0] = test.lua

arg[1] = 1

arg[2] = 2

arg[3] = nil

arg[-1] = -i

arg[-2] = lua

arg[-3] = nil

3 数值

（1）从Lua5.3开始，数值有两种类型，integer 64位整型，float 双精度浮点类型，对于资源受限的平台，可以将Lua5.3编译为Small Lua，LUA\_32BITS，使用32位整型和单精度浮点类型。在Lua 5.3版本之前，数值都以双精度浮点格式表示。

（2）对于Lua5.3，Small Lua和标准Lua，除了数值表示占用的字节大小不一样，其余都一样。

（3）数值类型，number。可以相互转换，具有相同算数值的整型和浮点型值在Lua语言中是相等的。

> 1 == 1.0 -> true

（4）type和math.type

> type (1) --> number

> type (0.1) --> number

> math.type(3) -->integer

> math.type(3.0) -->float

（5）科学计数法表示常量，用e/E

5e12 5e-12 5E12 5E+12

（6）可用0x开头表示16进制的常量和浮点数。Lua 5.2 中引入的：16进制表示浮点。

> 0xff --> 255

> 0x0.2 --> 0.125

> 0x1p-1 --> 0.5

> 0xa.bp2 -->42.75

> string.format ( “%a”, 419 ) --> 0x1.a3p+8

> string.format ( “%a”, 0.1 ) --> 0x1.999999999999ap-4

这种格式很难阅读，但可以保留所有浮点数的精度，并且比十进制的转换速度更快。？？

（7）除了加减乘除、取负数的常见运算外，Lua还支持取整除法（floor）、取模（求余）和指数运算。对于加减乘、取负数、取模，floor，如果两个操作数都是整型值，那么结果就是整型值，否则，结果就是浮点型值。对于除法、指数运算，操作数始终会被作为浮点型。

> 13.0+25 --> 38.0

> 4%2.0 --> 0.0

> 3//2 --> 1

> 3.0//2 --> 1.0

> -9//2 --> -5

a % b = a – ((a//b) \* b)

整型取模运算结果的符号，与第二个操作数保持一致。

> 5%-2 --> -1

> -5%2 --> 1

> x = math.pi

> x – x % 0.01 --> 3.14

> x – x % 0.001 --> 3.141

指数运算，x^0.5，求平方根。3^-1

（8）关系运算。六种，< > <= >= == ~= 运算结果为boolean。当==或者~=用于，两个不同的类型时，Lua认为他们是不相等的。

比较数值时，会忽略子类型，只与算数值有关。

（9）Lua提供了标准数学库math，由一组标准的数学函数组成，三角函数（sin、cos、tan、asin等）、指数函数、取整函数、max、min、伪随机函数random、常量pi、最大可表示数值huge（在大多数平台上代表inf）

三角函数以弧度为单位，可通过deg和rad函数进行角度和弧度的转换。

随机数发生器，math.random，生成伪随机数。不带参数调用，返回[0, 1) 范围内的数。当传入参数n，返回一个在[1, n]范围内的整数。random(6) 模拟掷骰子。当传入两个整数l和u，返回[l， u] 范围内的整数。

randomseed用于设置伪随机数发生器的种子。应传入一个数值类型。程序启动，系统固定使用1作为种子初始化发生器。如果不设置成其他种子，每次程序运行都会生成相同的伪随机数序列。对于游戏而言，会导致相同的场景重复不断地出现。为了解决这个问题，math.randomseed(os.time())。

数学库提供了3个取整函数。

> math.floor (3.3) --> 3

> math.floor (-3.3) --> -4

> math.ceil (3.3) --> 4

> math.ceil (-3.3) --> -3

> math.modf (3.3) --> 3 0.3

> math.modf (-3.3) --> -3 -0.3

Lua支持一个函数返回多个值。

nearest integer：floor (x+0.5)。这个式子可能会引入bug，当x是个很大的整数时，大整数化作浮点数时，是不精确的。这种情况下，整数可以单独处理。判断x为整数：if math.floor(x) == x。

原因：足够大的整数，2^52+1在转成double时，会出现问题。

（10）表示范围

请阅读教材的P24. 对于浮点运算，我还是一无所知。

根3.6一起看。

（11）Lua运算符优先级

优先级表 P27

在二元运算符中，除了幂运算和连接操作符是右结合，其它都是左结合。

-x^2 <--> -(x^2)

-x+y <--> (-x)+y

x^y^z <--> x^(y^z)

（12）Lua5.3 与之前版本的兼容性

P27

4 字符串

Lua的字符串标准库默认处理1Byte的字符，但是同样也可以优雅地处理UTF-8字符串。此外，从Lua5.3开始还提供了一个帮助使用UTF-8编码的函数库。

Lua语言中的字符串是不可变值immutable value，我们只能通过创建新的字符串的方式来达到修改的目的，不像C语言。

a = “one string”

b = string.gsub(a, “one”, another”);

print(a) --> one string

print(b) --> another string

> a = “Hello”

> a .. “ World” --> Hello World

> a --> Hello

Lua语言的字符串同其他对象一样，是自动内存管理的对象之一。Lua语言会负责字符串内存的分配的释放，开发人员无须关注。

长度操作符#，可以获取字符串的长度

a = “hello”

print(#a) -->5

print(#”good bye”) -->8

可以使用连接符..来进行字符串的连接，如果操作数中存在数值，那么Lua语言会先把数值转换成字符串：

> “Hello “..”World” --> Hello World

> “result is “..3 --> result is 3

（1）字符串常量 string literal

a = ‘a line’

a = “another line”

‘和“是一样的，但使用‘时，字符串中的“可以不用转义，使用”时，字符串中’可以不用转义。

这样设计的好处是，比如，要引用一段XML，作为字符串，其中有大量双引号，如果用双引号引用他，需要将其中的双引号全部转义表示，而如果用单引号引用它，则很方便。

别的编程语言，表示字符串，有的用’,有的用”，都可以不做修改转到lua中使用。

Lua语言支持下列C语言风格的转义字符：

\a \b \f \n \r \t \v \\ \” \’

> print ( “one line\nnext line\n\”in quotes\”, ‘in quotes’ ” )

字符串中，可以通过转义序列\ddd和\xhh来声明字符。其中ddd是最多3个十进制数字，hh是两个16进制数字。举个例子，在一个使用ASCII编码的系统中，“AL0\n123\””和”\x41L0\10\04923”是一样的（想想为什么是\049而不是\49）。

从Lua5.3开始，也可以使用转义序列\u{h..h}来声明UTF-8字符，花括号中支持任意十六进制数字：

> “\u{3b1}\u{3b2}\u{3b3}” -->

上例中假定终端使用的是UTF-8编码。

长字符串 多行字符串 P33 4.2

[[ ]] 表示长字符串。

它在声明包含大段代码的字符串时非常方便。

字符串中可能有：a=b[c[i]]，这种情况。怎么才能在不修改原字符串的情况下解决它？P33

对注释而言，也是。--[[ ]] 表示长注释。如果内部包含注释代码，该怎么办？

这是Lua词法特征中，比较有特色的一点，不知lexer源码中是怎么实现这一点的。

在用引号声明字符串时，如果字符串过长，需要换行，可用\z，它可以跳过[ \t\r\n]

a = “a

b” 🡪 报错

a = “a\z

b” 🡪a = “ab”

强制类型转换 coercion

Lua在运行时会尝试数值与字符串之间的自动转换。在需要数值的地方如果传入了字符串，则。。。在需要字符串的地方如果传入了数值，则。。。

需要数值，有可能传入字符串的地方：

（1）算术运算，+-\*/ 等等。

（2）函数传参

需要字符串，有可能传入数值的地方：

（1）..

（2）函数传参

比较操作符，不会触发自动转换。

“0”==0，为flase。 2<”15”，报错。

2<15 “2”>”15”为true。

数字后面如果紧跟运算符..，必须有空格，否则lua会将第一个点识别为小数点，从而报错。

任何有字符串参与的算术运算，都会被当做浮点运算。

“1”+2 🡪 3.0

显示地将一个字符串转换成数值：tonumber

* tonumber(“ -3 “) 🡪 -3
* tonumber( “ 10e4”) 🡪 100000.0
* tonumber( “10e”) 🡪 nil
* tonumber(“0x1.3p-4”) 🡪 0.07421875

使用任意进制

* tonumber(“10”, 2) 🡪 2
* tonumber(“fff”, 16) 🡪 4095
* tonumber(“ZZ”,36) 🡪 -1295
* tonumber(“987”, 8) 🡪 nil

显示的将数值转为字符串：tostring

* a = tostring(10).

这种方法不能控制输出的字符串的格式。可通过string.format去控制。