使用Lua编写可嵌入式脚本之一 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/144/2021_2022__E4_BD_BF_E7_94_A8Lua_E7_c103_144250.htm 虽然编译性编程语言和脚本语言各自具有自己独特的优点，但是如果我们使用这两种类型的语言来编写大型的应用程序会是什么样子呢？ Lua 是一种嵌入式脚本语言，它非常小，速度很快，功能却非常强大。在创建其他配置文件或资源格式（以及与之对应的解析器）之前，请尝试一下 Lua。 尽管诸如 Perl、Python、PHP 和 Ruby 之类的解释性编程语言日益被 Web 应用程序广泛地采纳 它们已经长期用来实现自动化系统管理任务 但是诸如 C、C 之类的编译性编程语言依然是必需的。编译性编程语言的性能是脚本语言所无法企及的（只有手工调优的汇编程序的性能才能超过它），有些软件 包括操作系统和设备驱动程序 只能使用编译代码来高效地实现。实际上，当软件和硬件需要进行无缝地连接操作时，程序员本能地就会选择 C 编译器：C 非常基础，距离“ 原始金属材料非常近 ” 即可以操作硬件的很多特性 并且 C 的表现力非常强大，可以提供高级编程结构，例如结构、循环、命名变量和作用域。 然而，脚本语言也有自己独特的优点。例如，当某种语言的解释器被成功移植到一种平台上以后，使用这种语言编写的大量脚本就可以不加任何修改在这种新平台上运行 它们没有诸如系统特定的函数库之类的依赖限制。（我们可以考虑一下 Microsoftreg. 操作系统上的许多 DLL 文件和 UNIXreg. 上的很多 libcs）。另外，脚本语言通常都还会提供高级编程构造和便利的操作，程序员可以使用这些功能来提高生产效率和灵活性。另外，

使用解释语言来编程的程序员工作的速度更快，因为这不需要编译和链接的步骤。C 及其类似语言中的“ 编码、编译、链接、运行 ”周期缩减成了更为紧凑的“ 编写脚本、运行 ”。 Lua 新特性 与其他脚本语言一样，Lua 也有自己的一些特性：* Lua 类型。在 Lua 中，值可以有类型，但是变量的类型都是动态决定的。nil、布尔型、数字 和 字符串 类型的工作方式与我们期望的一样。 o Nil 是值为 nil 的一种特殊类型，用来表示没有值。 o 布尔型的值可以是 true 和 false 常量。（Nil 也可以表示 false，任何非 nil 的值都表示 true。） o Lua 中所有的数字都是双精度的（不过我们可以非常简便地编写一些代码来实现其他数字类型）。 o 字符串是定长字符数组。（因此，要在一个字符串后面附加上字符，必须对其进行拷贝。）* 表、函数 和线程 类型都是引用。每个都可以赋值给一个变量，作为参数传递，或作为返回值从函数中返回。例如，下面是一个存储函数的例子：-- example of an anonymous function-- returned as a value-- see http://www.tecgraf.puc-rio.br/~lhf/ftp/doc/hopl.pdffunction add(x) return function (y) return (x y) endendf = add(2)print(type(f), f(10))function 12 * Lua 线程。线程是通过调用内嵌函数 coroutine.create(f) 创建的一个协同例程 (co-routine)，其中 f 是一个 Lua 函数。线程不会在创建时启动；相反，它是在创建之后使用 coroutine.resume(t) 启动的，其中 t 就是一个线程。每个协同例程都必须使用 coroutine.yield() 偶尔获得其他协同例程的处理器。 * 赋值语句。Lua 允许使用多种赋值语句，可以先对表达式进行求值，然后再进行赋值。例如，下面的语句：i = 3a = {1, 3, 5, 7, 9}i, a[i], a[i 1], b = i 1, a[i 1], a[i]print (i,

a[3], a[4], b, I) 会生成 4 7 5 nil nil。如果变量列表的个数大于值列表的个数，那么多出的变量都被赋值为 nil；因此，b 就是 nil。如果值的个数多于变量的个数，那么多出的值部分就会简单地丢弃。在 Lua 中，变量名是大小写敏感的，这可以解释为什么 I 的值是 nil。 * 块（Chunk）。 块 可以是任何 Lua 语句序列。块可以保存到文件中，或者保存到 Lua 程序中的字符串中。每个块都是作为一个匿名函数体来执行的。因此，块可以定义局部变量和返回值。 * 更酷的东西。Lua 具有一个标记-清理垃圾收集器。在 Lua 5.1 中，垃圾收集器是以增量方式工作的。Lua 具有完整的词法闭包（这与 Scheme 类似，而与 Python 不同）。Lua 具有可靠的尾部调用语义（同样，这也与 Scheme 类似，而与 Python 不同）。 在 Programming in Lua 和 Lua-users wiki （链接请参见后面的 参考资料 部分）中可以找到更多 Lua 代码的例子。 在所有的工程任务中，要在编译性语言和解释性语言之间作出选择，就意味着要在这种环境中对每种语言的优缺点、权重和折中进行评测，并接受所带来的风险。 