第17章 模块与包

一个模块就是一些代码，要么是Lua编写，要么是C语言编写，这些代码可以通过函数require加载，然后创建和返回一个表。

独立解释器会使用跟如下代码等价的方式提前加载所有标准库：

math = require "math" string = require "string"

17.1 函数require

函数require在表package.loaded中检查模块是否已被加载。如果已经加载，require就返回相应的值。

如果模块尚未加载，那么函数require则搜索具有指定模块名的Lua文件，搜索路径由package.path指定。

如果require找到相应的文件，那么就用函数loadfile将其进行加载，结果是一个称为加载器的函数。

如果require找不到指定模块名的lua文件，那么它就搜索相应名称的C标准库。这种情况，搜索路径由package.cpath指定。

如果找到C标准库，则使用底层函数package.loadlib进行加载，这个底层函数会查找名为luaopen\_modname的函数。

要强制函数require加载同一模块两次，可以先将模块从package.loaded中删除：package.loaded.modname = nil（热更新？）

17.1.1 模块重命名

17.1.2 搜索路径

当package模块被初始化后，它就把变量package.path设置成环境变量LUA\_PATH\_5\_3，如果这个环境变量没有定义，那么Lua会尝试另一个环境变量LUA\_PATH。

如果这两个环境变量都没有被定义，那么Lua语言则使用一个编译时定义的默认路径。

在使用一个环境变量的值时，Lua会将其中所有的";;"替换成默认路径。

搜索C标准库的路径的逻辑与此相同，只不过C标准库的路径来自变量package.cpath而不是package.path。

函数package.searchpath中实现了搜索库的所有规则，该函数的参数包括模块名和路径，然后遵循上述规则来搜索文件。

package.searchpath要么返回第一个存在的文件的文件名，要么返回nil外加描述所有文件都无法成功打开的错误信息。

17.1.3 搜索器

17.2 Lua语言中编写模块的基本方法

创建模块的最简单的方法是，创建一个表并将所有需要导出的函数放入其中，并返回这个表。

一种将return语句省略的方式是直接把模块对应的表放到package.loaded中：

local M = {} package.loaded[...] = M

另一种编写模块的方法是把所有的函数定义为局部变量，然后在最后构造返回的表。

17.3 子模块和包