

Lua 教程

Lua 教程

Lua 环境安装

Lua 基本语法

Lua 数据类型

Lua 变量

Lua 循环

Lua 流程控制

Lua 函数

Lua 运算符

Lua 字符串

Lua 数组

Lua 迭代器

Lua table(表)

Lua 模块与包

Lua 元表(Metatable)

Lua 协同程序(coroutine)

Lua 文件 I/O

Lua 错误处理

Lua 调试(Debug)

Lua 垃圾回收

Lua 面向对象

Lua 数据库访问

Lua5.3 参考手册

← Lua 协同程序(coroutine)

Lua 错误处理 →

Lua 文件 I/O

Lua I/O 库用于读取和处理文件。分为简单模式（和C一样）、完全模式。

- 简单模式（simple model）拥有一个当前输入文件和一个当前输出文件，并且提供针对这些文件相关的操作。
- 完全模式（complete model）使用外部的文件句柄来实现。它以一种面对对象的形式，将所有的文件操作定义为文件句柄的方法

简单模式在做一些简单的文件操作时较为合适。但是在进行一些高级的文件操作的时候，简单模式就显得力不从心。例如同时读取多个文件这样的操作，使用完全模式则较为合适。

打开文件操作语句如下：

```
file = io.open (filename [, mode])
```

mode 的值有：

模式	描述
r	以只读方式打开文件，该文件必须存在。
w	打开只写文件，若文件存在则文件长度清为0，即该文件内容会消失。若文件不存在则建立该文件。
a	以附加的方式打开只写文件。若文件不存在，则会建立该文件，如果文件存在，写入的数据会被加到文件尾，即文件原先的内容会被保留。（EOF符保留）
r+	以可读写方式打开文件，该文件必须存在。
w+	打开可读写文件，若文件存在则文件长度清为零，即该文件内容会消失。若文件不存在则建立该文件。
a+	与a类似，但此文件可读可写
b	二进制模式，如果文件是二进制文件，可以加上b
+	号表示对文件既可以读也可以写

简单模式

简单模式使用标准的 I/O 或使用一个当前输入文件和一个当前输出文件。

以下为 file.lua 文件代码，操作的文件为test.lua(如果没有你需要创建该文件)，代码如下：

```
-- 以只读方式打开文件
file = io.open("test.lua", "r")

-- 设置默认输入文件为 test.lua
io.input(file)

-- 输出文件第一行
print(io.read())

-- 关闭打开的文件
io.close(file)

-- 以附加的方式打开只写文件
file = io.open("test.lua", "a")

-- 设置默认输出文件为 test.lua
io.output(file)

-- 在文件最后一行添加 Lua 注释
io.write("-- test.lua 文件末尾注释")

-- 关闭打开的文件
io.close(file)
```

执行以上代码，你会发现，输出了 test.ua 文件的第一行信息，并在该文件最后一行添加了 lua 的注释。如我这边输出的是：

```
-- test.lua 文件
```

在以上实例中我们使用了 io."x" 方法，其中 io.read() 中我们没有带参数，参数可以是下表中的一个：

模式	描述
"n"	读取一个数字并返回它。例：file.read("n")
"a"	从当前位置读取整个文件。例：file.read("a")
"l"（默认）	读取下一行，在文件尾 (EOF) 处返回 nil。例：file.read("l")
number	返回一个指定字符个数的字符串，或在 EOF 时返回 nil。例：file.read(5)

其他的 io 方法有：

- io.tmpfile():返回一个临时文件句柄，该文件以更新模式打开，程序结束时自动删除
- io.type(file): 检测obj是否一个可用的文件句柄
- io.flush(): 向文件写入缓冲中的所有数据
- io.lines(optional file name): 返回一个迭代函数,每次调用将获得文件中的一行内容,当到文件尾时，将返回nil,但不关闭文件

完全模式

通常我们需要在同一时间处理多个文件。我们需要使用 file.function_name 来代替 io.function_name 方法。以下实例演示了如何同时处理同一个文件:

```
-- 以只读方式打开文件
file = io.open("test.lua", "r")

-- 输出文件第一行
print(file:read())

-- 关闭打开的文件
file:close()

-- 以附加的方式打开只写文件
file = io.open("test.lua", "a")

-- 在文件最后一行添加 Lua 注释
file:write("--test")

-- 关闭打开的文件
file:close()
```

执行以上代码，你会发现，输出了 test.ua 文件的第一行信息，并在该文件最后一行添加了 lua 的注释。如我这边输出的是：

```
-- test.lua 文件
```

read 的参数与简单模式一致。

其他方法

- file:seek(optional whence, optional offset): 设置和获取当前文件位置,成功则返回最终的文件位置(按字节),失败则返回nil加错误信息。参数 whence 值可以是：
 - "set": 从文件头开始
 - "cur": 从当前位置开始[默认]
 - "end": 从文件尾开始
 - offset 默认为0不带参数file:seek()则返回当前位置, file:seek("set")则定位到文件头, file:seek("end")则定位到文件尾并返回文件大小
- file:flush(): 向文件写入缓冲中的所有数据
- io.lines(optional file name): 打开指定的文件filename为读模式并返回一个迭代函数, 每次调用将获得文件中的一行内容, 当到文件尾时，将返回nil,并自动关闭文件。
若不带参数时io.lines() <=> io.input():lines(), 读取默认输入设备的内容，但结束时不关闭文件,如：

```
for line in io.lines("main.lua") do

    print(line)

end
```

以下实例使用了 seek 方法，定位到文件倒数第 25 个位置并使用 read 方法的 'a' 参数，即从当期位置(倒数第 25 个位置)读取整个文件。

```
-- 以只读方式打开文件
file = io.open("test.lua", "r")

file:seek("end",-25)
print(file:read("a"))

-- 关闭打开的文件
file:close()
</pre>
<p>我这边输出的结果是: </p>
<pre>
st.lua 文件末尾--test
--test.lua 文件
```

← Lua 协同程序(coroutine)

Lua 错误处理 →

+

4 篇笔记

写笔记

python免费公开课

授课模式：在线直播+课后视频，从零基础到中高级开发工程师 编程学习网

查看详情

分类导航

HTML / CSS

JavaScript

服务端

数据库

移动端

XML 教程

ASP.NET

Web Service

开发工具

网站建设

Advertisement

CSS HTML JS

你要的WEB前端的都在这里都有学习资料

我要听课

在线实例

- HTML 实例
- CSS 实例
- JavaScript 实例
- Ajax 实例
- JQuery 实例
- XML 实例
- Java 实例

字符集&工具

- HTML 字符集设置
- HTML ASCII 字符集
- HTML ISO-8859-1
- HTML 实体符号
- HTML 拾色器
- JSON 格式化工具

最新更新

- PostgreSQL 选择...
- PostgreSQL 创建...
- PostgreSQL 数据...
- C 语言整数与李...
- Vue.js Ajax(ax...
- PostgreSQL 语法
- Mac OS 上安装 P...

站点信息

- 意见反馈
- 合作联系
- 免责声明
- 关于我们
- 文章归档

关注微信

