**一、引言**  
  
        Redis学了一段时间了，基本的东西都没问题了。从今天开始讲写一些redis和lua脚本的相关的东西，lua这个脚本是一个好东西，可以运行在任何平台上，也可以嵌入到大多数语言当中，来扩展其功能。lua脚本是用C语言写的，体积很小，运行速度很快，并且每次的执行都是作为一个原子事务来执行的，我们可以在其中做很多的事情。由于篇幅很多，一次无法概述全部，这个系列可能要通过多篇文章的形式来写，好了，今天我们进入正题吧。  
  
**二、Lua简介**  
      
        Lua 是一个小巧的脚本语言。是巴西里约热内卢天主教大学（Pontifical Catholic University of Rio de Janeiro）里的一个研究小组，由Roberto Ierusalimschy、Waldemar Celes 和 Luiz Henrique de Figueiredo所组成并于1993年开发。 其设计目的是为了嵌入应用程序中，从而为应用程序提供灵活的扩展和定制功能。Lua由标准C编写而成，几乎在所有操作系统和平台上都可以编译，运行。Lua并没有提供强大的库，这是由它的定位决定的。所以Lua不适合作为开发独立应用程序的语言。Lua 有一个同时进行的JIT项目，提供在特定平台上的即时编译功能。  
  
       Lua脚本可以很容易的被C/C++ 代码调用，也可以反过来调用C/C++的函数，这使得Lua在应用程序中可以被广泛应用。不仅仅作为扩展脚本，也可以作为普通的配置文件，代替XML,ini等文件格式，并且更容易理解和维护。 Lua由标准C编写而成，代码简洁优美，几乎在所有操作系统和平台上都可以编译，运行。一个完整的Lua解释器不过200k，在目前所有脚本引擎中，Lua的速度是最快的。这一切都决定了Lua是作为嵌入式脚本的最佳选择。  
  
**三、使用Lua脚本的好处**  
  
**1、减少网络开销：可以将多个请求通过脚本的形式一次发送，减少网络时延和请求次数。**  
  
**2、原子性的操作：Redis会将整个脚本作为一个整体执行，中间不会被其他命令插入。因此在编写脚本的过程中无需担心会出现竞态条件，无需使用事务。**  
  
**3、代码复用：客户端发送的脚步会永久存在redis中，这样，其他客户端可以复用这一脚本来完成相同的逻辑。**  
  
**4、速度快：见 与其它语言的性能比较, 还有一个 JIT编译器可以显著地提高多数任务的性能; 对于那些仍然对性能不满意的人, 可以把关键部分使用C实现, 然后与其集成, 这样还可以享受其它方面的好处。**  
  
**5、可以移植：只要是有ANSI C 编译器的平台都可以编译，你可以看到它可以在几乎所有的平台上运行:从 Windows 到Linux，同样Mac平台也没问题, 再到移动平台、游戏主机，甚至浏览器也可以完美使用 (翻译成JavaScript).**  
  
**6、源码小巧：20000行C代码，可以编译进182K的可执行文件，加载快，运行快。**  
  
**四、redis和lua整合详解**  
  
     **1、调用Lua脚本的语法：**  
              $ redis-cli --eval path/to/redis.lua KEYS[1] KEYS[2] , ARGV[1] ARGV[2] ...  
  
              --eval，告诉redis-cli读取并运行后面的lua脚本  
               path/to/redis.lua，是lua脚本的位置  
               KEYS[1] KEYS[2]，是要操作的键，可以指定多个，在lua脚本中通过KEYS[1], KEYS[2]获取  
               ARGV[1] ARGV[2]，参数，在lua脚本中通过ARGV[1], ARGV[2]获取。  
  
      **注意： KEYS和ARGV中间的 ',' 两边的空格，不能省略。**

**redis支持大部分Lua标准库**

| **库名** | **说明** |
| --- | --- |
| Base | 提供一些基础函数 |
| String | 提供用于字符串操作的函数 |
| Table | 提供用于表操作的函数 |
| Math | 提供数学计算函数 |
| Debug | 提供用于调试的函数 |

       **2、在脚本中调用redis命令**  
               在脚本中可以使用redis.call函数调用Redis命令  
  
              redis.call('set', 'foo', 'bar')  
              local value=redis.call('get', 'foo') --value的值为bar  
  
              redis.call函数的返回值就是Redis命令的执行结果  
  
              Redis命令的返回值有5种类型，redis.call函数会将这5种类型的回复转换成对应的Lua的数据类型，具体的对应规则如下（空结果比较特殊，其对应Lua的false）

**redis返回值类型和Lua数据类型转换规则**

| **redis返回值类型** | **Lua数据类型** |
| --- | --- |
| 整数回复 | 数字类型 |
| 字符串回复 | 字符串类型 |
| 多行字符串回复 | table类型(数组形式) |
| 状态回复 | table类型(只有一个ok字段存储状态信息) |
| 错误回复 | table类型(只有一个err字段存储错误信息) |

                  redis还提供了redis.pcall函数，功能与redis.call相同，唯一的区别是当命令执行出错时，redis.pcall会记录错误并继续执行，而redis.call会直接返回错误，不会继续执行。在脚本中可以使用return语句将值返回给客户端，如果没有执行return语句则默认返回nil

**Lua数据类型和redis返回值类型转换规则**

| **Lua数据类型** | **redis返回值类型** |
| --- | --- |
| 数字类型 | 整数回复(Lua的数字类型会被自动转换成整数) |
| 字符串类型 | 字符串回复 |
| table类型(数组形式) | 多行字符串回复 |
| table类型(只有一个ok字段存储状态信息) | 状态回复 |
| table类型(只有一个err字段存储错误信息) | 错误回复 |

        **3、脚本相关命令**  
             **EVAL语法：** **eval script numkeys key [key ...] arg [arg ...]**  
  
              通过key和arg这两类参数向脚本传递数据，它们的值在脚本中分别使用KEYS和ARGV两个表类型的全局变量访问。

**script**： 是lua脚本

**numkeys**：表示有几个key，分别是KEYS[1],KEYS[2]...，如果有值，从第numkeys+1个开始就是参数值，ARGV[1],ARGV[2]...  
  
             **注意： EVAL命令依据参数numkeys来将其后面的所有参数分别存入脚本中KEYS和ARGV两个table类型的全局变量。当脚本不需要任何参数时，也不能省略这个参数(设为0)**

192.168.127.128:6379>eval "return redis.call('set',KEYS[1],ARGV[1])" 1 name liulei

OK

192.168.127.128:6379>get name

"liulei"

       **4、 EVALSHA命令**  
              在脚本比较长的情况下，如果每次调用脚本都需要将整个脚本传给Redis会占用较多的带宽。为了解决这个问题，Redis提供了EVALSHA命令，允许开发者通过脚本内容的SHA1摘要来执行脚本，该命令的用法和EVAL一样，只不过是将脚本内容替换成脚本内容的SHA1摘要。  
  
             Redis在执行EVAL命令时会计算脚本的SHA1摘要并记录在脚本缓存中，执行EVALSHA命令时Redis会根据提供的摘要从脚本缓存中查找对应的脚本内容，如果找到了则执行脚本，否则会返回错误："NOSCRIPT No matching script. Please use EVAL."  
  
             **在程序中使用EVALSHA命令的一般流程如下。**  
  
**1）、先计算脚本的SHA1摘要，并使用EVALSHA命令执行脚本。**  
  
**2）、获得返回值，如果返回“NOSCRIPT”错误则使用EVAL命令重新执行脚本。**  
  
              虽然这一流程略显麻烦，但值得庆幸的是很多编程语言的Redis客户端都会代替开发者完成这一流程。执行EVAL命令时，先尝试执行EVALSHA命令，如果失败了才会执行EVAL命令。  
  
               SCRIPTLOAD "lua-script"  将脚本加入缓存，但不执行， 返回：脚本的SHA1摘要  
  
               SCRIPT EXISTS lua-script-sha1   判断脚本是否已被缓存  
  
       **5、 SCRIPT FLUSH（该命令不区分大小写）**  
               清空脚本缓存，redis将脚本的SHA1摘要加入到脚本缓存后会永久保留，不会删除，但可以手动使用SCRIPT FLUSH命令情况脚本缓存。

192.168.127.128:6379>script flush

OK

192.168.127.128:6379>SCRIPT FLUSH

OK

      **6、SCRIPT KILL（该命令不区分大小写）**  
              强制终止当前脚本的执行。 但是，如果当前执行的脚步对redis的数据进行了写操作，则SCRIPT KILL命令不会终止脚本的运行，以防止脚本只执行了一部分。脚本中的所有命令，要么都执行，要么都不执行。

[IMG_256](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

192.168.127.128:6379>script kill

(error)NOTBUSY No scripts in execution right now

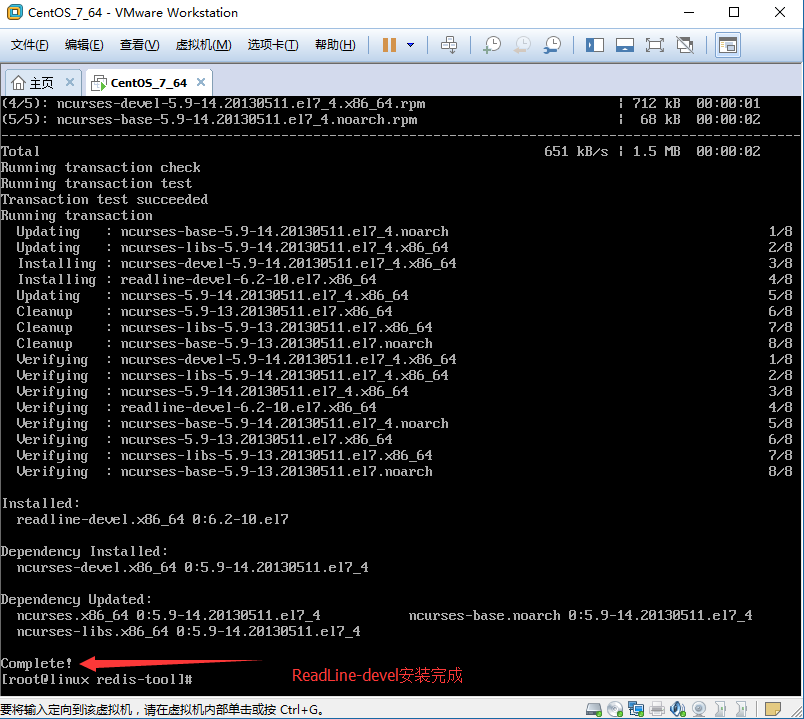
192.168.127.128:6379>SCRIPT KILL

(error)NOTBUSY No scripts in execution right now

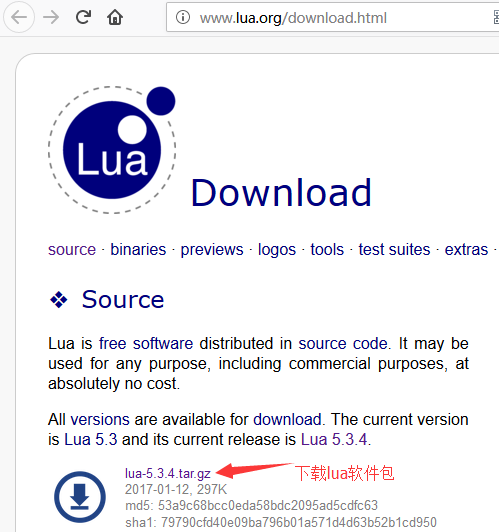
//这是当前没有脚本在执行，所以提示该错误

[IMG_257](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

       **7、lua-time-limit 5000（redis.conf配置文件中）**

              为了防止某个脚本执行时间过长导致Redis无法提供服务（比如陷入死循环），Redis提供了lua-time-limit参数限制脚本的最长运行时间，默认为5秒钟。当脚本运行时间超过这一限制后，Redis将开始接受其他命令但不会执行（以确保脚本的原子性，因为此时脚本并没有被终止），而是会返回“BUSY”错误。  
  
**五、安装和使用Lua脚本**  
  
       **1、安装lua类库环境**  
  
              1.1、yum install -y readline  
                       
              1.2、yum install -y readline-devel  
  
                       
  
        **2、下载lua最新的版本安装**  
  
                **2.1、去官网下载lua，可以直接通过wget下载，地址如下：http://www.lua.org/download.html**

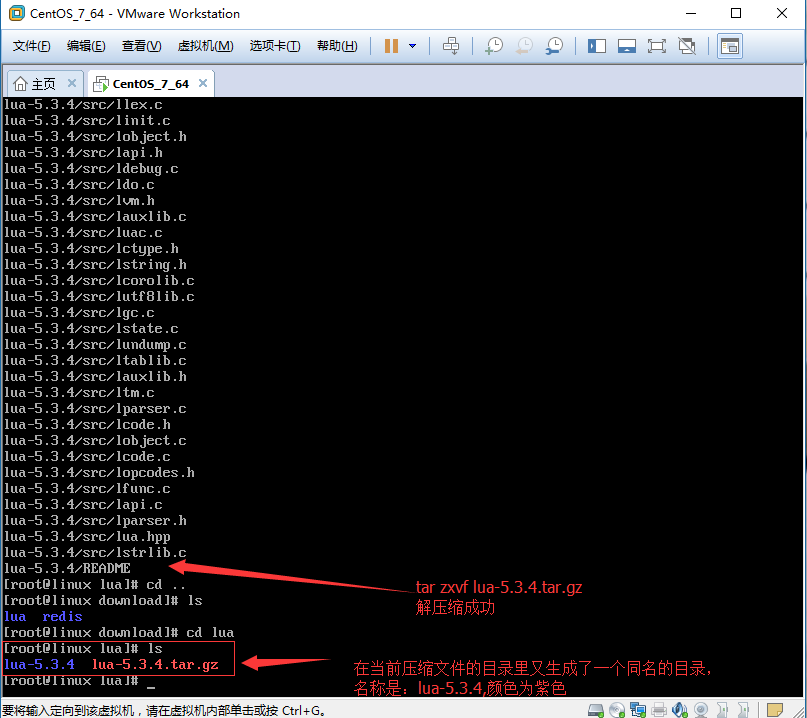
[root@lunux~]# wget http://www.lua.org/ftp/lua-5.3.4.tar.gz /root/software/download/lua/



               **2.2、通过ssh SSH Secure File Transfer Client工具，把软件包上传到Linux服务器上。目录是：/root/software/download/lua/**

[root@linux~]# cd ./software/download/lua/

[root@linux lua]# tar zxvf lua-5.3.4.tar.gz



              **2.3、进入到已经解压的目录lua-5.3.4，准备安装文件。**

[IMG_261](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

[root@linux lua]# ls

[root@linux lua]# lua-5.3.4 lua-5.3.4.tar.gz

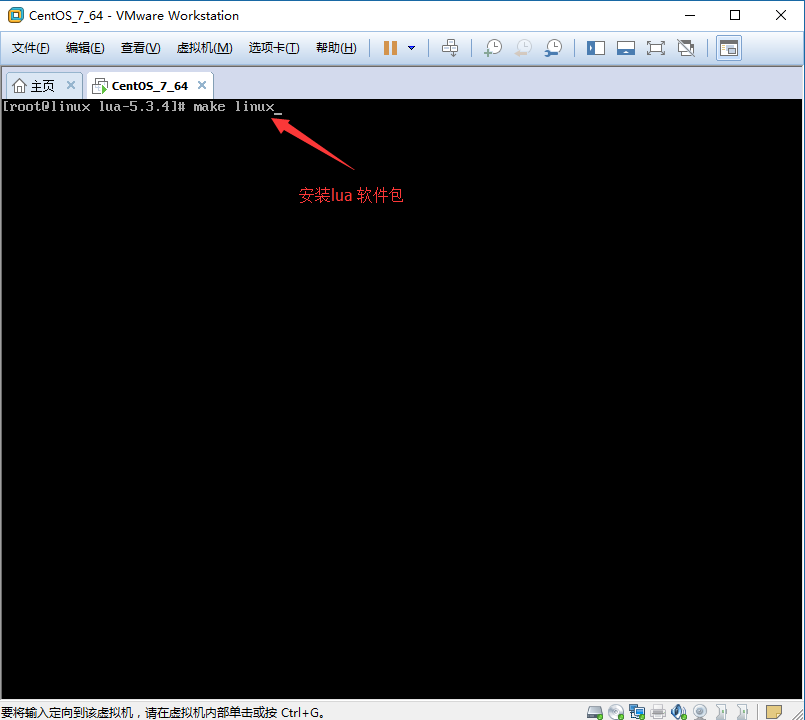
[root@linux lua]# cd lua-5.3.4

[root@linux lua-5.3.4]#

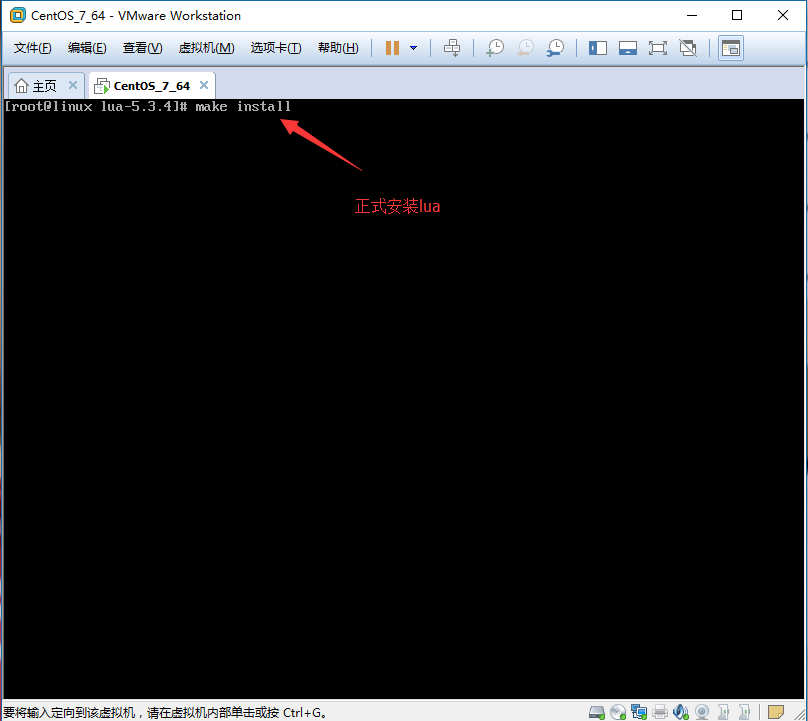
[IMG_262](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

             **2.4、准备安装环境，使用make linux命令，当前也是需要gcc命令的支持，事先必须安装，安装gcc命令：yum install gcc。**

[root@linux lua-5.3.4]# make linux

                          
  
**2.5、开始安装lua软件包，使用make install命令**

[root@linux lua-5.3.4]# make install



**2. 6、最后进行测试，进到Linux的命令行，然后输入lua命令，开始测试。**

[IMG_265](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

[root@linux lua-5.3.4]# lua

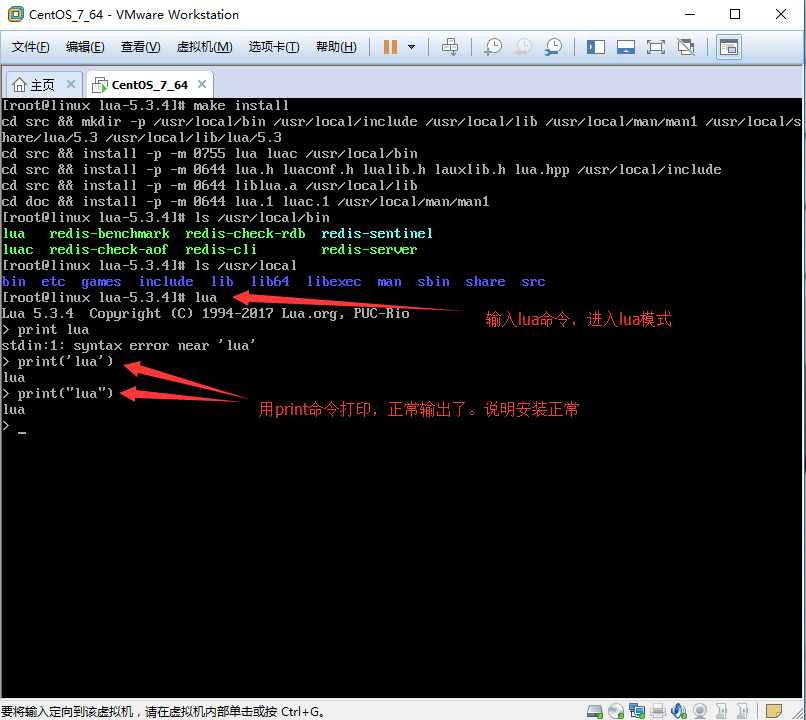
>print('lua')

lua

>print("lua")

lua

[IMG_266](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)



**2.7、按Ctrl+C退出lua命令模式。**

>^C

[root@linux lua-5.3.4]#

**2.8、lua脚本文件名必须以.lua后缀名，如果在Linux命令行执行lua脚本，直接lua 脚本名称。**

[IMG_268](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

[root@linux lua-5.3.4]# cd /root/application/program/ //执行文件都在这个目录里面

[root@linux program]# mkdir luascript //创建luaScript脚本目录，存放lua脚本文件

[root@linux program]# cd luascript

[root@linux luascript]# lua 01.lua //执行01.lua脚本文件

[IMG_269](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

**2.9、redis与lua脚本结合使用，如果在lua脚本里使用了 redis.call命令来操作Redis，执行lua脚步如下面：**

[IMG_270](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

//redis-cli和lua脚本的路径可以是相对路径，也可以是绝对路径

//以下代码就是通过绝对地址来执行

//绝对地址：

[root@linux ~]# /root/application/program/redis-tool/redis-cli -h 192.168.127.128 -p 6379 --eval /root/application/program/luascript/02.lua

//相对地址：

//当前目录

192.168.127.128:6379>pwd

[root@linux redis-tool]/root/application/program/redis-tool/

[root@linux redis-tool]# redis-cli -h 192.168.127.128 -p 6379 --eval /root/application/program/luascript/02.lua

[IMG_271](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

**2.10、Redis客户端执行带有参数的lua脚本，脚本文件的名称是：03.lua。**

[IMG_272](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

//当前redis 数据库中只有name和age两个key，其他数据已经清空。

//当前所在目录

192.168.127.128:6379>keys \*

1)"name"

2)"age"

192.168.127.128:6379>get name

"liulei"

192.168.127.128:6379>get age

"15"

//03.lua脚本代码如下：

local name=redis.call("get",KEYS[1])

local age=redis.call("get",KEYS[2])

if name=="LLL" then

redis.call("set",KEYS[1],ARGV[1])

redis.call("incr",KEYS[2])

end

//执行改脚本的命令，必须在Linux的命令行，不是在Redis的命令行

[root@linux ~]# /root/application/program/redis-tool/redis-cli -h 192.168.127.128 -p 6379 --eval /root/application/program/luascript/03.lua name age , patrickLiu

//执行脚本命令后

192.168.127.128:6379>keys \*

1)"name"

2)"age"

192.168.127.128:6379>get name

"patrickLiu"

192.168.127.128:6379>get age

"16"

//说明带参数的执行Lua脚本成功

[IMG_273](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

**2.11、Redis客户端执行有参数lua，并返回lua的表类型。**

[IMG_274](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

//04.lua文件的源码

local b1=redis.call("hgetall",KEYS[1])

return b1

//代码很简单，话不多说

//清空当前数据库

192.168.127.128:6379>flushdb

192.168.127.128:6379>keys \*

(empty list or set)

192.168.127.128:6379>hmset myhash name zhangsan sex nan address hebeibaoding school laiyuanyizhong

OK

192.168.127.128:6379>hmget myhash name sex address school

1)"zhangsan"

2)"nan"

3)"hebeibaoding"

4)"laiyuanyizhong"

//我们通过redis客户端获取myhash的结果，进入到redis客户端的当前目录

[root@linux redis-tool]# redis-cli -h 192.168.127.128 -p 6379 --eval ../luascript/04.lua myhash

1)"name"

2)"zhangsan"

3)"sex"

4)"nan"

5)"address"

6)"hebeibaoding"

7)"school"

8)"laiyuanyizhong"

//成功获取myhash的列表

[IMG_275](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

**六、总结**  
  
   今天就写到这里，还有很多内容要写，一次也写不完，慢慢来吧，我也需要消化一下。如果有时间，下一篇文章或者下某一篇文章准备来单独写一些关于Lua脚本语法的东西，我也是第一次接触这个东西，大家一起学习吧。