**一、介绍**  
  
       redis学了有一段时间了，以前都是看视频，看教程，很少看官方的东西。现在redis的东西要看的都差不多看完了。网上的东西也不多了。剩下来就看看官网的东西吧，一遍翻译，一遍测试。不错的使用体验，今天开始就写一下redis的客户端的使用，也就是redis-cli工具类，这个文件的内容真不少，一次写完有点吃力。分多次写吧。  
  
**二、使用详解**  
  
   **1、redis-cli，Redis命令行界面**  
  
           redis-cli是Redis命令行界面，它是一个允许向Redis发送命令、并直接从终端读取服务器发送的回复的简单的程序。  
  
           它有两种主要模式：一种交互模式，其中有一个REPL（读取评估打印循环），用户输入命令并获取回复；另一种模式是将命令作为redis-cli的参数发送，执行并打印在标准输出中。  
  
           在交互模式下，redis-cli具有基本的行编辑功能，可以提供良好的打字体验。  
  
           然而，redis-cli不仅仅具有以上功能，还有更高级的用法。有些选项可用来启动程序并将其置于特殊模式之下，以便redis-cli可以完成更复杂的任务，如模拟从站并打印从主站接收到的复制流，检查Redis服务器的延迟，并显示统计数据，甚至是延时采样和频率的ASCII-art频谱图，以及其他许多事情。  
  
           本指南将涵盖redis-cli的不同方面，从最简单到最高级的结尾。  
  
           如果您要广泛使用Redis，或者您已经这么做了，那么很可能会碰巧使用redis-cli。 花一些时间熟悉它可能是一个非常好的主意，一旦你知道了命令行界面的所有技巧，你就会更有效地使用Redis。  
  
  
  **2、命令行的用法**  
  
           只需运行一个命令并在标准输出上打印其答复就如同输入要作为redis-cli的分隔参数执行的命令一样简单：

$ redis-cli -h 192.168.127.130 -p 6379 incr mycounter

(integer) 7

            该命令的结果是“7”。 由于Redis的结果是强类型的（它们可以是字符串，数组，整数，NULL，错误等等），您可以看到括号之间的就是结果的类型。 但是，当redis-cli的输出必须用作另一个命令的输入，或者我们希望将它重定向到一个文件时，这并不是一个好的做法。  
  
           实际上，redis-cli当检测到标准输出是一个tty设备时就会来显示这些附加信息以帮助提高人们的可读性。 否则，它将通过 auto-enable 启用原始输出模式，如下例所示：

$ redis-cli -h 192.168.127.130 -p 6379 incr mycounter > /tmp/output.txt

$ cat /tmp/output.txt

8

           由于CLI检测到输出不再写入终端，因此这次输出中才省略了（整数）这个显示。 你甚至可以在终端上用--raw选项强制原始输出：

$ redis-cli -h 192.168.127.130 -p 6379 --raw incr mycounter

9

           同样，通过使用--no-raw，可以在写入文件或将管道传送到其他命令时强制读取可读的输出。  
  
  
   **3、主机，端口，密码和数据库**  
  
           默认情况下，redis-cli在127.0.0.1端口6379连接到服务器。正如您所猜测的，您可以使用命令行选项轻松更改此设置。 要指定不同的主机名或IP地址，请使用-h。 为了设置不同的端口，请使用-p。

$ redis-cli -h 192.168.127.130 -p 6379 ping

PONG

           如果您的实例受密码保护，-a <密码>选项将执行身份验证，以保证明确使用AUTH命令的需要：

$ redis-cli -h 192.168.127.130 -p 6379 -a myUnguessablePazzzzzword123 ping

PONG

           最后，可以通过使用-n <dbnum>选项发送一个命令，该命令对除默认数字零以外的数据库号进行操作：

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

$ redis-cli -h 192.168.127.130 -p 6379 flushall

OK

$ redis-cli -h 192.168.127.130 -p 6379 -n 1 incr a

(integer) 1

$ redis-cli -h 192.168.127.130 -p 6379 -n 1 incr a

(integer) 2

$ redis-cli -h 192.168.127.130 -p 6379 -n 2 incr a

(integer) 1

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

           部分或全部信息也可以通过使用-u <uri>选项和一个有效的URI来提供：

$ redis-cli -u redis://p%40ssw0rd@redis-16379.hosted.com:16379/0 ping

PONG

    **4、从其他程序获取输入**  
  
           有两种方法可以使用redis-cli来获取来自其他命令的输入（基本上来自标准输入）。 一种是使用我们从标准输入读取的有效数据作为最后一个参数。 例如，假设在我的电脑里，为了将Redis的key设置到文件/etc/services的内容里，我可以使用-x选项：

$ redis-cli -h 192.168.127.130 -p 6379 -x set foo < /etc/services

OK

$ redis-cli -h 192.168.127.130 -p 6379 getrange foo 0 50

"#\n# Network services, Internet style\n#\n# Note that "

           正如您在上述会话的第一行中看到的那样，SET命令的最后一个参数未被指定。 参数只是SET foo，并没有为我希望的key设置的实际的值。  
  
           相反，指定了-x选项，并将文件重定向到CLI的标准输入。所以输入被读取，并被用作命令的最后一个参数。这对编写脚本很有用。  
  
           另一种不同的方法是为redis-cli提供一个写在文本文件中的命令序列：

[IMG_258](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

$ cat /tmp/commands.txt

set foo 100

incr foo

append foo xxx

get foo

$ cat /tmp/commands.txt | redis-cli -h 192.168.127.130 -p 6379

OK

(integer) 101

(integer) 6

"101xxx"

[IMG_259](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

           command.txt中的所有命令都由redis-cli一个接一个地执行，就好像它们是由用户交互式键入的一样。 如果需要，可以在文件内使用双引号来标识字符串，这样在字符串中间就可以包含空格或换行符或其他特殊字符，但是此字符串是作为一个参数使用的：

[IMG_260](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

$ cat /tmp/commands.txt

set foo "This is a single argument"

strlen foo

$ cat /tmp/commands.txt | redis-cli -h 192.168.127.130 -p 6379

OK

(integer) 25

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

    **5、连续运行相同的命令**  
  
           在执行过程中，当用户选择暂停的情况下执行相同命令的指定次数是可行的。这在不同情况下很有用，例如，当我们想要持续监视一些 key 的内容或INFO字段输出时，或者当我们想模拟一些重复写入事件时（比如每5秒将一个新项目推入列表）。  
  
           该功能由两个选项控制：-r <count>和-i <delay>。 第一种选项表示运行命令的次数，第二种表示不同命令调用之间的延迟，以秒为单位（能够指定十进制数字，如0.1表示100毫秒）。  
  
           默认情况下，间隔（或延迟）被设置为0，所以命令只是尽快执行：

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

$ redis-cli -h 192.168.127.130 -p 6379 -r 5 incr foo

(integer) 1

(integer) 2

(integer) 3

(integer) 4

(integer) 5

[IMG_263](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

           要长时间运行相同的命令，请使用-1作为计数。 因此，为了随时监控RSS存储器的大小，可以使用如下的命令：

$ redis-cli -h 192.168.127.130 -p 6379 -r -1 -i 1 INFO | grep rss\_human

used\_memory\_rss\_human:1.38M

used\_memory\_rss\_human:1.38M

used\_memory\_rss\_human:1.38M

... a new line will be printed each second ...

**6、使用redis-cli大量插入数据**  
  
          使用redis-cli进行大数据量插入会使用单独的章节来说，因为它本身就是一个有价值的话题。请参阅我们的《[批量插入指南](https://redis.io/topics/mass-insert" \t "http://www.cnblogs.com/PatrickLiu/p/_blank)》。  
  
  
**7、CSV输出**  
  
            有时候您可能想要使用redis-cli来快速将数据从Redis导出到外部程序。 这可以使用CSV（逗号分隔值）输出功能来完成：

$ redis-cli -h 192.168.127.130 -p 6379 lpush mylist a b c d

(integer) 4

$ redis-cli -h 192.168.127.130 -p 6379 --csv lrange mylist 0 -1

"d","c","b","a"

             目前无法像这样导出整个数据库，但只能用CSV输出运行单个命令。  
  
  
**8、运行Lua脚本**  
  
           redis-cli对使用Lua脚本编写的新Lua调试工具提供了广泛的支持，稳定版本是从Redis 3.2开始。有关此功能，请参阅《Redis Lua调试器文档》https://redis.io/topics/ldb。  
  
           但是，即使不使用调试器，与以交互方式将脚本输入到shell或作为参数相比，您可以使用redis-cli从文件运行脚本，这种方式更加舒适：

$ cat /tmp/script.lua return redis.call('set',KEYS[1],ARGV[1])

$ redis-cli -h 192.168.127.130 -p 6379 --eval /tmp/script.lua foo , bar

OK

            Redis EVAL命令将脚本使用的键列表和其他非键参数作为不同的数组。在调用EVAL命令时，您可以将key的数量作为一个数字提供。 但是，使用redis-cli并使用上面的--eval选项，并没有明确指定key的数量。相反，它使用用逗号分隔键和参数。这就是为什么在上面的调用中你看到foo，bar作为参数。  
  
            所以foo会填充KEYS数组，bar会填充ARGV数组。  
  
            编写简单脚本时，--eval选项很有用。对于更复杂的工作，使用Lua调试器肯定更适合。可以混合使用这两种方法，因为调试器也使用来自外部文件的执行脚本。  
  
  
**9、交互模式**  
  
            到目前为止，我们探讨了如何使用Redis CLI作为命令行程序。 这对于脚本和某些类型的测试非常有用，但是大多数人的大部分时间将会使用redis-cli的交互模式。  
  
            在交互模式下，用户在提示符下键入Redis命令。该命令被发送到服务器，进行处理，回复被解析回来并呈现为更简单的形式来阅读。  
  
           在交互模式下运行CLI不需要什么特别的东西 - 在没有任何参数的情况下运行即可，并且您处于：

$ redis-cli -h 192.168.127.139 -p 6379

192.168.127.130:6379> ping

PONG

            字符串192.168.127.130:6379>是提示符。它提醒您，您已连接到给定的Redis实例。  
          
            当连接的服务器发生更改时，或者当您在与数据库编号0不同的数据库上运行时，提示会发生的更改：

[IMG_264](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

192.168.127.130:6379> select 2

OK

192.168.127.130:6379[2]> dbsize

(integer) 1

192.168.127.130:6379[2]> select 0

OK

192.168.127.130:6379> dbsize

(integer) 503

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

     **9.1、处理连接和重新连接**  
  
                在交互模式下使用connect命令可以通过指定我们要连接的主机名和端口来连接到不同的实例：

192.168.127.130:6379> connect metal 6379

metal:6379> ping

PONG

               正如你所看到的以下的提示也会有相应的改变。如果用户尝试连接到无法访问的实例，则redis-cli将进入断开连接模式，并且准备使用新命令尝试重新连接：

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

192.168.127.130:6379> connect 192.168.127.130 9999

Could not connect to Redis at 192.168.127.130:9999: Connection refused

not connected> ping

Could not connect to Redis at 192.168.127.130:9999: Connection refused

not connected> ping

Could not connect to Redis at 192.168.127.130:9999: Connection refused

[IMG_267](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

               通常在检测到断开连接后，CLI始终尝试以透明方式重新连接：如果尝试失败，则会显示错误并进入断开连接状态。 以下是断开和重新连接的示例：

192.168.127.130:6379> debug restart

Could not connect to Redis at 192.168.127.130:6379: Connection refused

not connected> ping

PONG

192.168.127.130:6379> (now we are connected again)

                 当执行重新连接时，redis-cli会自动重新选择所选的上一个数据库编号。然而，关于连接的所有其他状态都会丢失，例如，事务的状态信息就会丢失，当我们处于事务的执行过程中的时候：

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

$ redis-cli -h 192.168.127.130 -p 6379

192.168.127.130:6379> multi

OK

192.168.127.130:6379> ping

QUEUED

( here the server is manually restarted )

192.168.127.130:6379> exec

(error) ERR EXEC without MULTI

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

               在交互模式下使用CLI进行测试时，这通常不是问题，但您应该了解这一限制。  
  
  
   **9.2、编辑，历史和完成**  
  
                由于redis-cli使用《linenoise行编辑库》，因此它一直具有行编辑功能，而不依赖于libreadline或其他可选库。  
  
                您可以访问已执行的命令的历史记录，以便通过按方向键（上和下）来避免重复键入它们。在用户主目录内的一个名为.rediscli\_history的文件中，指定HOME环境变量，在重新启动CLI的时候保留历史记录。  可以通过设置 REDISCLI\_HISTFILE 环境变量来使用不同的历史文件名，并通过将其设置为 /dev/null 来禁用它。  
  
                CLI客户端还能够通过按TAB键执行命令名称的补全操作，<TAB>表示按TAB键，如下例所示：

192.168.127.130:6379> Z<TAB>

192.168.127.130:6379> ZADD<TAB>

192.168.127.130:6379> ZCARD<TAB>

       **9.3、运行相同的命令N次**  
  
                可以通过在命令名前添加一个数字来多次运行相同的命令：

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

192.168.127.130:6379> 5 incr mycounter

(integer) 1

(integer) 2

(integer) 3

(integer) 4

(integer) 5

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

        **9.4、显示有关Redis命令的帮助**  
  
                Redis有很多命令，有时候，当你测试的时候，你可能不记得参数的确切顺序。redis-cli使用help命令为大多数Redis命令提供联机帮助。 该命令可以以两种形式使用：  
  
                **1、help @ <category>显示关于给定类别的所有命令。 类别包括：@generic，@list，@set，@sorted\_set，@hash，@pubsub，@transactions，@connection，@server，@scripting，@hyperloglog。**  
  
                 **2、help <commandname>显示作为参数给出的命令的特定帮助。**  
  
                例如，为了显示PFADD命令的帮助，使用：

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

192.168.127.130:6379> help PFADD

PFADD key element [element ...]

summary: Adds the specified elements to the specified HyperLogLog.

since: 2.8.9

group: hyperloglog

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

                  PFADD key element [element ...]  
                  摘要：将指定的元素添加到指定的HyperLogLog。   
                  从版本：2.8.9  
  
                  请注意，帮助也支持TAB补全功能。  
  
  
        **9.5、清除终端屏幕**  
  
            在交互模式下使用 clear 命令将清除终端屏幕。  
  
　    **后面还有很多高级内容，就放在下篇文章里面了。**  
  
**三、总结**  
  
         好了，今天就写到这里了，剩下还有很多高级的内容需要些，慢慢来，一口不能吃一个胖子，而且翻译起来也挺耗时间的。剩下的内容就留到下一篇文章吧。继续努力，不能松懈。如果想看原文，地址如下：[https://redis.io/topics/rediscli。](https://redis.io/topics/rediscli" \t "http://www.cnblogs.com/PatrickLiu/p/_blank)