
redis

序言

> 文章转载自：易百教程 [http://www.yiibai.com]

Redis是一个开源的，先进的key-value存储并用于构建高性能，可扩展Web应用程序的解决方案。 Redis主要的三个特点： Redis数据库完全在内存中，使用磁盘仅用于持久性。 相比许多键值数据存储Redis拥有一套较为丰富的数据类型。 Redis可以将数据复制到任意数量的从服务服务中。

Redis的优势

- 异常快速：Redis速度非常快，每秒能执行约11万集合，约81000条每秒。
- 支持丰富的数据类型：Redis原生支持最大多数开发人员已经知道的，像列表，集合，分类，散列等数据类型。这使得它非常容易解决各种各样的问题，因为我们知道哪些数据类型问题是可以通过它的处理得更好。
- 操作原子性：所有Redis的操作是原子的，这保证了如果两个客户端同时访问的Redis服务器将获得更新后的值。
- MultiUtility工具：Redis是一个多实用的工具，可以在一些像缓存，消息，队列用例中使用（Redis原生支持发布/订阅），在应用程序，如Web应用程序的会话，网络页面点击数短期数据等等。

相对于其他的key-value存储，为什么Redis不同？

- Redis是key-value数据块，其中值可以包含更复杂的数据类型，对这些数据类型定义的原子操作不同的演进路径。
- Redis在内存中，但持续存在磁盘上的数据库，所以它代表了不同的权衡，非常高的读写速度达到了数据集的限制，不能比内存大。在存储器的数据库的另一个优点是，复杂的数据结构存储表示简单得多（相比操作磁盘上相同的数据结构），这样Redis可以做很多，很少有内部的复杂性。

Redis环境安装

在Ubuntu上安装Redis

要安装Redis在Ubuntu上，打开终端，然后键入以下命令：

```
$sudo apt-get update
$sudo apt-get install redis-server
```

这将在您的计算机上安装Redis。

启动Redis

```
$redis-server
```

检查Redis是否在工作？

```
$redis-cli
```

这将打开一个Redis提示，如下图所示：

```
redis 127.0.0.1:6379>
```

在上面的提示127.0.0.1是本机的IP地址，6379是Redis服务器运行的端口。现在输入PING命令，如下图所示。

```
redis 127.0.0.1:6379> ping
PONG
```

这说明你已经成功地在您的机器上安装Redis。

在Ubuntu上安装Redis桌面管理器

在Ubuntu上安装Redis桌面管理，只要从http://redisdesktop.com/download 下载的软件包并安装它。

Redis桌面管理器会给出用户界面来管理Redis的键和数据。

Redis配置

Redis配置

在Redis有配置文件(redis.conf)可在Redis的根目录下找到。可以通过Redis的CONFIG命令设置所有Redis的配置。

语法

Redis的CONFIG命令的基本语法如下所示：

```
redis 127.0.0.1:6379> CONFIG GET CONFIG_SETTING_NAME
```

例如

```
redis 127.0.0.1:6379> CONFIG GET loglevel
1) "loglevel"
2) "notice"
```

让所有的配置使用*代替CONFIG_SETTING_NAME

例子

```
redis 127.0.0.1:6379> CONFIG GET *
 1) "dbfilename"
 2) "dump.rdb"
 3) "requirepass"
 4) ""
 5) "masterauth"
 6) ""
 7) "unixsocket"
 8) ""
 9) "logfile"
10) ""
11) "pidfile"
12) "/var/run/redis.pid"
13) "maxmemory"
14) "0"
15) "maxmemory-samples"
16) "3"
17) "timeout"
18) "0"
19) "tcp-keepalive"
20) "0"
21) "auto-aof-rewrite-percentage"
```

```
22) "100"
23) "auto-aof-rewrite-min-size"
24) "67108864"
25) "hash-max-ziplist-entries"
26) "512"
27) "hash-max-ziplist-value"
28) "64"
29) "list-max-ziplist-entries"
30) "512"
31) "list-max-ziplist-value"
32) "64"
33) "set-max-intset-entries"
34) "512"
35) "zset-max-ziplist-entries"
36) "128"
37) "zset-max-ziplist-value"
38) "64"
39) "hll-sparse-max-bytes"
40) "3000"
41) "lua-time-limit"
42) "5000"
43) "slowlog-log-slower-than"
44) "10000"
45) "latency-monitor-threshold"
46) "0"
47) "slowlog-max-len"
48) "128"
49) "port"
50) "6379"
51) "tcp-backlog"
52) "511"
53) "databases"
54) "16"
55) "repl-ping-slave-period"
56) "10"
57) "repl-timeout"
58) "60"
59) "repl-backlog-size"
60) "1048576"
61) "repl-backlog-ttl"
62) "3600"
63) "maxclients"
64) "4064"
65) "watchdog-period"
66) "0"
67) "slave-priority"
68) "100"
69) "min-slaves-to-write"
70) "0"
71) "min-slaves-max-lag"
72) "10"
73) "hz"
74) "10"
75) "no-appendfsync-on-rewrite"
76) "no"
```

```
 77) "slave-serve-stale-data"
 78) "yes"
 79) "slave-read-only"
 80) "yes"
 81) "stop-writes-on-bgsave-error"
 82) "yes"
 83) "daemonize"
 84) "no"
 85) "rdbcompression"
 86) "yes"
 87) "rdbchecksum"
 88) "yes"
 89) "activerehashing"
 90) "yes"
 91) "repl-disable-tcp-nodelay"
 92) "no"
 93) "aof-rewrite-incremental-fsync"
 94) "yes"
 95) "appendonly"
 96) "no"
 97) "dir"
 98) "/home/deepak/Downloads/redis-2.8.13/src"
 99) "maxmemory-policy"
100) "volatile-lru"
101) "appendfsync"
102) "everysec"
103) "save"
104) "3600 1 300 100 60 10000"
105) "loglevel"
106) "notice"
107) "client-output-buffer-limit"
108) "normal 0 0 0 slave 268435456 67108864 60 pubsub 33554432 8388608 60"
109) "unixsocketperm"
110) "0"
111) "slaveof"
112) ""
113) "notify-keyspace-events"
114) ""
115) "bind"
116) ""
```

编辑配置

要更新配置，可以直接编辑redis.conf文件或更新配置，通过CONFIG set命令

语法

CONFIG SET命令的基本语法如下所示：

```
redis 127.0.0.1:6379> CONFIG SET CONFIG_SETTING_NAME NEW_CONFIG_VALUE
```

例子

```
redis 127.0.0.1:6379> CONFIG SET loglevel "notice"
OK
redis 127.0.0.1:6379> CONFIG GET loglevel
 1) "loglevel"
2) "notice"
```

Redis数据类型

Redis支持5种数据类型，它们描述如下：

Strings - 字符串

Redis的字符串是字节序列。在Redis中字符串是二进制安全的，这意味着他们有一个已知的长度，是没有任何特殊字符终止决定的，所以可以存储任何东西，最大长度可达512兆。

例子

```
redis 127.0.0.1:6379> SET name "yiibai"
OK
redis 127.0.0.1:6379> GET name
"yiibai"
```

在上面的例子使用Redis命令set和get，Redis的名称是yiibai的键存储在Redis的字符串值。

注：字符串值可以存储最大512兆字节的长度。

Hashes - 哈希值

Redis的哈希键值对的集合。 Redis的哈希值是字符串字段和字符串值之间的映射，所以它们被用来表示对象

例子

```
redis 127.0.0.1:6379> HMSET user:1 username yiibai password yiibai points 200
OK
redis 127.0.0.1:6379> HGETALL user:1
 1) "username"
2) "yiibai"
3) "password"
4) "yiibai"
5) "points"
6) "200"
```

在上面的例子中的哈希数据类型，用于存储包含用户的基本信息用户的对象。这里HMSET，HEXTALL对于Redis 命令 user:1 是键。

每个哈希可存储多达232 - 1个 字段 - 值对(超过4十亿)。

Lists - 列表

Redis的列表是简单的字符串列表，排序插入顺序。可以添加元素到Redis列表的头部或尾部。

例子

```
redis 127.0.0.1:6379> lpush tutoriallist redis
(integer) 1
redis 127.0.0.1:6379> lpush tutoriallist mongodb
(integer) 2
redis 127.0.0.1:6379> lpush tutoriallist rabitmq
(integer) 3
redis 127.0.0.1:6379> lrange tutoriallist 0 10
 1) "rabitmq"
2) "mongodb"
3) "redis"
```

列表的最大长度为232- 1元素(4294967295，每个列表中的元素超过4十亿)。

Sets - 集合

Redis集合是字符串的无序集合。在Redis中可以添加，删除和测试文件是否存在在O(1)的时间复杂度的成员。

例子

```
redis 127.0.0.1:6379> sadd tutoriallist redis
(integer) 1
redis 127.0.0.1:6379> sadd tutoriallist mongodb
(integer) 1
redis 127.0.0.1:6379> sadd tutoriallist rabitmq
(integer) 1
redis 127.0.0.1:6379> sadd tutoriallist rabitmq
(integer) 0
redis 127.0.0.1:6379> smembers tutoriallist
 1) "rabitmq"
2) "mongodb"
3) "redis"
```

> 注意：在上面的例子中rabitmq设置属性加两次，但由于唯一性只加一次。

成员中集最大数量为232 - 1(4294967295，集合成员超过4十亿)。

集合排序

Redis的集合排序类似于Redis集合，字符串不重复的集合。不同的是，一个有序集合的每个成员关联分数，用于以便采取有序set命令，从最小的到最大的分数有关。虽然成员都是独一无二的，分数可能会重复。

例子

```
redis 127.0.0.1:6379> zadd tutoriallist 0 redis
(integer) 1
redis 127.0.0.1:6379> zadd tutoriallist 0 mongodb
(integer) 1
redis 127.0.0.1:6379> zadd tutoriallist 0 rabitmq
(integer) 1
redis 127.0.0.1:6379> zadd tutoriallist 0 rabitmq
(integer) 0
redis 127.0.0.1:6379> ZRANGEBYSCORE tutoriallist 0 1000
 1) "redis"
2) "mongodb"
3) "rabitmq"
```

Redis命令

Redis命令用于在redis服务器上执行某些操作。

要在Redis服务器上运行的命令，需要一个Redis客户端。 Redis客户端在Redis的包，这已经我们前面安装使用过了。

语法

Redis客户端的基本语法如下：

```
$redis-cli
```

例子

下面举例说明如何使用Redis客户端。

要启动redis客户端，打开终端，输入命令Redis命令行： `redis-cli` 。这将连接到本地服务器，现在就可以运行各种命令了。

```
$redis-cli
redis 127.0.0.1:6379>
redis 127.0.0.1:6379> PING
 PONG
```

在上面的例子中，我们连接到本地机器上运行的Redis服务器，并且执行ping命令，来检查是否服务器正在运行。

远程服务器上运行命令

要在Redis远程服务器上运行的命令，需要通过同一个客户端redis-cli 连接到服务器

语法

```
$ redis-cli -h host -p port -a password
```

例如

下面的示例演示了如何连接到Redis主机：127.0.0.1，端口：6379 上的远程服务器，并加上验证密码为：mypass。

```
$redis-cli -h 127.0.0.1 -p 6379 -a "mypass"
redis 127.0.0.1:6379>
redis 127.0.0.1:6379> PING
 PONG
```

Redis键

Redis的keys命令用于管理键。使用Redis的keys命令语法如下所示：

语法

```
redis 127.0.0.1:6379> COMMAND KEY_NAME
```

例子

```
redis 127.0.0.1:6379> SET yiibai redis
OK
redis 127.0.0.1:6379> DEL yiibai
(integer) 1
```

在上面的例子中DEL是命令，而yiibai是键。如果键被删除那么输出该命令将是 (integer) 1，否则它是 (integer) 0

Redis的键命令

如下表显示键的一些基本命令：

S.N.	命令 & 描述
1	DEL key此命令删除键，如果存在
2	DUMP key 该命令返回存储在指定键的值的序列化版本。
3	EXISTS key 此命令检查该键是否存在。
4	EXPIRE key seconds指定键的过期时间
5	EXPIREAT key timestamp 指定的键过期时间。在这里，时间是在Unix时间戳格式
6	PEXPIRE key milliseconds 设置键以毫秒为单位到期
7	PEXPIREAT key milliseconds-timestamp 设置键在Unix时间戳指定为毫秒到期
8	KEYS pattern 查找与指定模式匹配的所有键
9	MOVE key db 移动键到另一个数据库
10	PERSIST key 移除过期的键
11	PTTL key 以毫秒为单位获取剩余时间的到期键。
12	TTL key 获取键到期的剩余时间。
13	RANDOMKEY 从Redis返回随机键

S.N.	命令 & 描述
14	RENAME key newkey 更改键的名称
15	RENAMENX key newkey 重命名键，如果新的键不存在
16	TYPE key 返回存储在键的数据类型的值。

Redis DEL命令

Redis的DEL命令用于删除redis中现有键。

返回值

被删除的键的数目。

语法

Redis的DEL命令的基本语法如下所示：

```
redis 127.0.0.1:6379> DEL KEY_NAME
```

例子

首先，在Redis的一个键，并设置一定的值。

```
redis 127.0.0.1:6379> SET yiibai redis
OK
```

现在删除以前生成键

```
redis 127.0.0.1:6379> DEL yiibai
(integer) 1
```

Redis DUMP命令

Redis的DUMP命令用于获取存储在redis的指定键数据的序列化版本。

返回值

序列化的值(字符串)

语法

Redis的DUMP命令的基本语法如下所示：

```
redis 127.0.0.1:6379> DUMP KEY_NAME
```

例子

首先，在Redis创建一个键，并设置一定的值。

```
redis 127.0.0.1:6379> SET yiibai redis
OK
```

现在创建之前创建的键转储

```
redis 127.0.0.1:6379> DUMP yiibai
"\x00\x05redis\x06\x00S\xbd\xc1q\x17z\x81\xb2"
```

Redis EXISTS命令

Redis EXISTS命令被用来检查键是否存在于redis。

返回值

整数值

- 1, 如果键存在。
- 0, 如果键不存在。

语法

Redis exists命令的基本语法，如下所示：

```
redis 127.0.0.1:6379> EXISTS KEY_NAME
```

例子

```
redis 127.0.0.1:6379> EXISTS yiibai-new-key
(integer) 0
```

现在，创建名为yiibai新建一个键并检查是否存在。

```
redis 127.0.0.1:6379> EXISTS yiibai-new-key
(integer) 1
```

Redis Expire命令

Redis Expire命令用于设定键有效期。到期时间后键不会在Redis中使用。

返回值

整数值1或0

- 1, 如果设置的键超时。
- 0, 如果键不存在，或者未设置超时。

语法

Redis expire命令的基本语法如下所示：

```
redis 127.0.0.1:6379> Expire KEY_NAME TIME_IN_SECONDS
```

例子

首先，在Redis创建一个键，并设置一定的值。

```
redis 127.0.0.1:6379> SET yiibai redis
OK
```

现在设置以前创建的键超时

```
redis 127.0.0.1:6379> EXPIRE yiibai 60
(integer) 1
```

在上面的例子中键yiibai被设定一分钟(或者60秒)的时间。 1分钟后，键会自动失效。

Redis Expireat命令

Redis Expireat命令是用来以Unix时间戳格式设置键的到期时间。到期时间键后不会在Redis中使用。

返回值

整数值1或0

- 1, 如果设置的键超时。
- 0, 如果键不存在，或者未设置超时。

语法

Redis Expireat命令的基本语法如下所示：

```
redis 127.0.0.1:6379> Expireat KEY_NAME TIME_IN_UNIX_TIMESTAMP
```

例子

首先，在Redis创建一个键，并设置一定的值。

```
redis 127.0.0.1:6379> SET yiibai redis
OK
```

现在设置以前创建的键超时

```
redis 127.0.0.1:6379> EXPIREAT yiibai 1293840000
(integer) 1
EXISTS yiibai
(integer) 0
```

Redis KEYS命令

Redis KEYS命令用于搜索与键的匹配模式。

返回值

键匹配模式的列表(数组)。

语法

Redis KEYS命令的基本语法如下所示：

```
redis 127.0.0.1:6379> KEYS PATTERN
```

例子

首先，在Redis创建一个键，并设置一定的值。

```
redis 127.0.0.1:6379> SET tutorial1 redis
OK
redis 127.0.0.1:6379> SET tutorial2 mysql
OK
redis 127.0.0.1:6379> SET tutorial3 mongodb
OK
```

现在Redis带有键搜索从关键字教程开始

```
redis 127.0.0.1:6379> KEYS tutorial*
1) "tutorial3"
2) "tutorial1"
3) "tutorial2"
```

Redis要获得所有键的可用列表仅只是使用*

```
redis 127.0.0.1:6379> KEYS *
1) "tutorial3"
2) "tutorial1"
3) "tutorial2"
```

Redis MOVE命令

Redis MOVE命令是用来从当前选择的数据库键移动到指定的目标数据库。

返回值

整数值1或0

- 1, 如果键被移动。
- 0, 如果键没有被移动。

语法

redis MOVE命令的基本语法如下所示：

```
redis 127.0.0.1:6379> MOVE KEY_NAME DESTINATION_DATABASE
```

例子

首先，在Redis创建一个键，并设置一定的值。

```
redis 127.0.0.1:6379> SET tutorial1 redis
OK
```

Redis的默认第0数据库被选中，所以现在我们正朝着第二个数据库生成的键移动。

```
redis 127.0.0.1:6379> MOVE tutorial1 1
1) (integer) 1
```

Redis PERSIST命令

Redis PERSIST命令用于删除到期的键。

返回值

整数值1或0

- 1, 如果超时则删除键。
- 0, 如果key不存在或不具备相关的超时时间。

语法

Redis PERSIST命令基本语法如下所示：

```
redis 127.0.0.1:6379> PERSIST KEY_NAME
```

例子

首先，在Redis创建一个键，并设置一定的值。

```
redis 127.0.0.1:6379> SET tutorial1 redis
OK
```

现在设置键到期，之后直接过期删除。

```
redis 127.0.0.1:6379> EXPIRE tutorial1 60
1) (integer) 1
redis 127.0.0.1:6379> TTL tutorial1
1) (integer) 60
redis 127.0.0.1:6379> PERSIST tutorial1
1) (integer) 1
redis 127.0.0.1:6379> TTL tutorial1
1) (integer) -1
```

Redis PTTL命令

Redis PTTL命令用于获取键以毫秒为单位的到期剩余时间。

返回值

以毫秒为单位的整数值TTL或负值

- TTL 以毫秒为单位。
- -1, 如果key没有到期超时。
- -2, 如果键不存在。

语法

redis PTTL命令的基本语法如下所示：

```
redis 127.0.0.1:6379> PTTL KEY_NAME
```

例如

首先，在Redis创建一个键，并设置一定的值。

```
redis 127.0.0.1:6379> SET tutorialname redis
OK
```

现在设置键到期，之后查看剩余到期时间。

```
redis 127.0.0.1:6379> EXPIRE tutorialname 1
1) (integer) 1
redis 127.0.0.1:6379> PTTL tutorialname
1) (integer) 999
```

Redis TTL命令

Redis TTL命令用于获取键到期的剩余时间(秒)。

返回值

以毫秒为单位的整数值TTL或负值

- TTL以毫秒为单位。
- -1, 如果key没有到期超时。
- -2, 如果键不存在。

语法

Redis TTL命令的基本语法如下所示：

```
redis 127.0.0.1:6379> TTL KEY_NAME
```

例子

首先，在Redis创建一个键，并设置一定的值。

```
redis 127.0.0.1:6379> SET tutorialname redis
OK
```

现在设置键到期，之后只需查看到期的剩余时间。

```
redis 127.0.0.1:6379> EXPIRE tutorialname 60
1) (integer) 1
redis 127.0.0.1:6379> TTL tutorialname
1) (integer) 59
```

Redis RANDOMKEY命令

Redis RANDOMKEY命令用来获取Redis数据库的随机键。

返回值

一个随机键(字符串)或nil，如果数据库是空的。

语法

redis RANDOMKEY命令的基本语法如下所示：

```
redis 127.0.0.1:6379> RANDOMKEY
```

例子

首先，在Redis创建一些键，并在其中设置一些值。

```
redis 127.0.0.1:6379> SET tutorial1 redis
OK
redis 127.0.0.1:6379> SET tutorial2 mysql
OK
redis 127.0.0.1:6379> SET tutorial3 mongodb
OK
```

现在得到Redis的随机键。

```
redis 127.0.0.1:6379> RANDOMKEY
1) tutorial3
```

Redis RENAME命令

Redis RENAME命令是用来改变一个键的名称。

返回值

字符串回复OK或error。

它会返回一个错误，如果旧的key和新的key名称相同或者key不存在。如果新的键已经存在，则覆盖现有的key。

语法

Redis RENAME命令的基本语法如下所示：

```
redis 127.0.0.1:6379> RENAME OLD_KEY_NAME NEW_KEY_NAME
```

例子

首先，在Redis创建一些键，并在其中设置一些值。

```
redis 127.0.0.1:6379> SET tutorial1 redis
OK
```

现在重命名键 tutorial1为new-tutorial。

```
redis 127.0.0.1:6379> RENAME tutorial1 new-tutorial
OK
```

Redis RENAMENX命令

Redis RENAMENX命令是用来改变一个键的名称，如果新的键不存在。

返回值

回复整数1或0。

- 1, 如果键被重命名为新的键。
- 0, 如果新的键已经存在。

Redis RENAMENX命令的基本语法如下所示：

```
redis 127.0.0.1:6379> RENAMENX OLD_KEY_NAME NEW_KEY_NAME
```

例子

首先，在Redis创建一些键，并在其中设置一些值。

```
redis 127.0.0.1:6379> SET tutorial1 redis
OK
redis 127.0.0.1:6379> SET tutorial2 mongodb
OK
```

现在重命名键 tutorial1为new-tutorial。

```
redis 127.0.0.1:6379> RENAMENX tutorial1 new-tutorial
(integer) 1
redis 127.0.0.1:6379> RENAMENX tutorial2 new-tutorial
(integer) 0
```

Redis TYPE命令

Redis TYPE命令用于获取存储在键值的数据类型。

返回值

字符串回复，存储在键值的数据类型或none

语法

Redis TYPE命令的基本语法如下所示：

```
redis 127.0.0.1:6379> TYPE KEY_NAME
```

例子

首先，在Redis创建一些键，并在其中设置一些值。

```
redis 127.0.0.1:6379> SET tutorial1 redis
OK
```

现在检查键的类型。

```
redis 127.0.0.1:6379> TYPE tutorial1
string
```

Redis SET命令

Redis SET命令是用来设置在Redis键部分字符串值。

返回值

简单的字符串回复OK，如果值被设置在键，否则如果值不设置为null。

语法

Redis SET命令的基本语法如下所示：

```
redis 127.0.0.1:6379> SET KEY_NAME VALUE
```

例子

```
redis 127.0.0.1:6379> SET yiibai redis
OK
```

选项

在SET命令有许多可供选项，即修改命令的行为。可用SET命令选项的基本语法如下所示：

```
redis 127.0.0.1:6379> SET KEY VALUE [EX seconds] [PX milliseconds] [NX|XX]
```

- EX seconds - 设置指定的到期时间，单位为秒。
- PX milliseconds - 设置指定到期时间，单位为毫秒。
- NX - 只有设置键，如果它不存在。
- XX - 只有设置键，如果它已经存在。

例子

```
redis 127.0.0.1:6379> SET yiibai redis EX 60 NX
OK
```

上面的例子将设置键yiibai，60秒到期时，如果该键不存在。

Redis GET命令

Redis GET命令用于获取存储在指定的键的值。如果键不存在，那么返回nil。如果返回值不是字符串，则返回错误。

返回值

简单的字符串答复。值或键或零。

语法

Redis GET命令的基本语法如下所示：

```
redis 127.0.0.1:6379> GET KEY_NAME
```

实例

首先，在Redis设置一个键，然后获取它。

```
redis 127.0.0.1:6379> SET yiibai redis
OK
redis 127.0.0.1:6379> GET yiibai
"redis"
```

Redis GETRANGE命令

Redis GETRANGE命令用于获取存储在键字符串值，由偏移确定的子串的开始和结束(两者都包括)。负偏移可以提供从字符串的末尾的偏移开始被使用。

该函数处理超出范围的请求，通过限制所得到的范围内的字符串的实际长度。

返回值

简单的字符串答复。

语法

redis GETRANGE命令的基本语法如下所示：

```
redis 127.0.0.1:6379> GETRANGE KEY_NAME start end
```

例子

首先，在Redis设置一个键，然后得到它的一些组成部分。

```
redis 127.0.0.1:6379> SET mykey "This is my test key"
OK
redis 127.0.0.1:6379> GETRANGE mykey 0 3
"This"
redis 127.0.0.1:6379> GETRANGE mykey 0 -1
"This is my test key"
```

Redis GETSET命令

Redis GETSET命令设置指定的在Redis的键的字符串值，并返回其原来的值。

返回值

回复简单的字符串，键的旧值。如果键不存在，那么返回nil。

语法

redis GETSET命令的基本语法如下所示：

```
redis 127.0.0.1:6379> GETSET KEY_NAME VALUE
```

例子

```
redis 127.0.0.1:6379> GETSET mynewkey "This is my test key"
(nil)
redis 127.0.0.1:6379> GETSET mynewkey "This is my new value to test getset"
"This is my test key"
```

Redis GETBIT命令

Redis GETBIT命令用于获取在存储在键串值偏移的比特值。

返回值

整数，存储在偏移中的位值。

语法

redis GETBIT命令的基本语法如下所示：

```
redis 127.0.0.1:6379> GETBIT KEY_NAME OFFSET
```

示例

```
redis 127.0.0.1:6379> SETBIT mykey 7 1
(integer) 0
redis 127.0.0.1:6379> GETBIT mykey 0
(integer) 0
redis 127.0.0.1:6379> GETBIT mykey 7
(integer) 1
redis 127.0.0.1:6379> GETBIT mykey 100
(integer) 0
```

Redis MGET命令

Redis MGET命令是用来获取所有指定键的值。对于未持有一个字符串值，或者每一个键不存在，返回特殊值为nil。

返回值

数组，在指定键的值列表。

语法

redis MGET命令的基本语法如下所示：

```
redis 127.0.0.1:6379> MGET KEY1 KEY2 .. KEYN
```

语法

```
redis 127.0.0.1:6379> SET key1 "hello"
OK
redis 127.0.0.1:6379> SET key2 "world"
OK
redis 127.0.0.1:6379> MGET key1 key2 someOtherKey
1) "Hello"
2) "World"
3) (nil)
```

Redis SETEX命令

Redis SETEX命令是用来设置一些字符串值，在Redis的键指定的超时时间内。

返回值

简单的字符串回复OK，如果值被设置在键，否则如果值不设置为null。

语法

redis SETEX命令的基本语法如下所示：

```
redis 127.0.0.1:6379> SETEX KEY_NAME TIMEOUT VALUE
```

例子

```
redis 127.0.0.1:6379> SETEX mykey 60 redis
OK
redis 127.0.0.1:6379> TTL mykey
60
redis 127.0.0.1:6379> GET mykey
"redis
```

Redis SETNX命令

Redis SETEX命令是用来设置在Redis的键部分字符串值，如果key没有在Redis的存在。 SETEX表单如果不存在被置位。

返回值

整数回复1或0

- 1, 如果该键设置。
- 0, 如果该键不被设置。

语法

redis SETNX命令的基本语法如下所示：

```
redis 127.0.0.1:6379> SETNX KEY_NAME VALUE
```

例子

```
redis 127.0.0.1:6379> SETNX mykey redis
(integer) 1
redis 127.0.0.1:6379> SETNX mykey mongodb
(integer) 0
redis 127.0.0.1:6379> GET mykey
"redis"
```

Redis SETRANGE命令

Redis SETRANGE命令是用来改写字符串的一部分，在键的指定开始的偏移量。

返回值

整数回复字符串在它被命令修改后的长度。

语法

redis SETRANGE命令的基本语法如下所示：

```
redis 127.0.0.1:6379> SETRANGE KEY_NAME OFFSET VALUE
```

实例

```
redis 127.0.0.1:6379> SET key1 "Hello World"
OK
redis 127.0.0.1:6379> SETRANGE key1 6 "Redis"
(integer) 11
redis 127.0.0.1:6379> GET key1
"Hello Redis"
```

Redis STRLEN命令

Redis STRLEN命令用于获取存储在key字符串值的长度。当key持有非字符串值则返回一个错误。

返回值

回复整数，字符串key长度，或0表示key不存在。

语法

redis SETRANGE命令的基本语法如下所示：

```
redis 127.0.0.1:6379> STRLEN KEY_NAME
```

例子

```
redis 127.0.0.1:6379> SET key1 "Hello World"
OK
redis 127.0.0.1:6379> STRLEN key1
(integer) 11
redis 127.0.0.1:6379> STRLEN key2
(integer) 0
```

Redis MSET命令

Redis MSET命令用于设定多个键，以及多个值。

返回值

回复简单的字符串OK

语法

redis MSET命令的基本语法如下所示：

```
redis 127.0.0.1:6379> MSET key1 value1 key2 value2 .. keyN valueN
```

示例

```
redis 127.0.0.1:6379> MSET key1 "Hello" key2 "World"
OK
redis 127.0.0.1:6379> GET key1
"Hello"
redis 127.0.0.1:6379> GET key2
1) "World"
```

Redis MSETNX命令

Redis MSETNX命令用于设置多个键以及多个值，仅当没有一个已存在。如果从当前操作的任何一个存

在，那么MSETNX不执行任何操作。

返回值

回复整数1或0

- 1, 如果所有的键都在Redis设置
- 0, 如果没有key在Redis设置

语法

Redis MSETNX命令的基本语法如下所示：

```
redis 127.0.0.1:6379> MSETNX key1 value1 key2 value2 .. keyN valueN
```

例如

```
redis 127.0.0.1:6379> MSETNX key1 "Hello" key2 "world"
(integer) 1
redis 127.0.0.1:6379> MSETNX key2 "worlds" key3 "third key"
(integer) 0
redis 127.0.0.1:6379> MGET key1 key2 key3
1) "Hello"
2) "world"
3) (nil)
```

Redis PSETEX命令

Redis PSETEX命令用于设置key的值，随着时间以毫秒为单位过期。

返回值

回复简单的字符串OK

语法

redis PSETEX命令的基本语法如下所示：

```
redis 127.0.0.1:6379> PSETEX key1 EXPIRY_IN_MILLISECONDS value1
```

例子

```
redis 127.0.0.1:6379> PSETEX mykey 1000 "Hello"
OK
redis 127.0.0.1:6379> PTTL mykey
999
redis 127.0.0.1:6379> GET mykey
1) "Hello"
```

Redis INCR命令

Redis INCR命令用于由一个递增key的整数值。如果该key不存在，它被设置为0执行操作之前。如果key包含了错误类型的值或包含不能被表示为整数，字符串，则返回错误。该操作被限制为64位带符号整数。

返回值

回复整数，键增量后的值

语法

Redis INCR命令的基本语法如下所示：

```
redis 127.0.0.1:6379> INCR KEY_NAME
```

例子

```
redis 127.0.0.1:6379> SET visitors 1000
OK
redis 127.0.0.1:6379> INCR visitors
(integer) 1001
redis 127.0.0.1:6379> GET visitors
(integer) 1001
```

Redis INCRBY命令

Redis INCRBY命令用于增加存储在由指定的值key的数量。如果该key不存在时，它被设置为0执行操作之前。如果键包含了错误类型的值或包含不能被表示为整数，字符串，则返回错误。

返回值

回复整数，键增量后的值

语法

redis INCRBY命令的基本语法如下所示：

```
redis 127.0.0.1:6379> INCRBY KEY_NAME INCR_AMOUNT
```

例子

```
redis 127.0.0.1:6379> SET visitors 1000
OK
redis 127.0.0.1:6379> INCRBY visitors 5
(integer) 1005
redis 127.0.0.1:6379> GET visitors
(integer) 1005
```

Redis字符串

Redis字符串命令用于在Redis管理字符串值。使用Redis字符串命令的语法如下所示：

语法

```
redis 127.0.0.1:6379> COMMAND KEY_NAME
```

例子

```
redis 127.0.0.1:6379> SET yiibai redis
OK
redis 127.0.0.1:6379> GET yiibai
"redis"
```

在上面的例子中，set和get是命令，而yiibai是键。

Redis字符串命令

如下表显示一些在Redis管理字符串基本的命令：

S.N.	命令 & 描述
1	SET key value 此命令用于在指定键设置值
2	GET key 键对应的值。
3	GETRANGE key start end 得到字符串的子字符串存放在一个键
4	GETSET key value设置键的字符串值，并返回旧值
5	GETBIT key offset返回存储在键位值的字符串值的偏移
6	MGET key1 [key2..]得到所有的给定键的值
7	SETBIT key offset value设置或清除该位在存储在键的字符串值偏移
8	SETEX key seconds value键到期时设置值
9	SETNX key value设置键的值，只有当该键不存在
10	SETRANGE key offset value覆盖字符串的一部分从指定键的偏移
11	STRLEN key得到存储在键的值的长度
12	MSET key value [key value ...] 设置多个键和多个值
13	MSETNX key value [key value ...] 设置多个键多个值，只有在当没有按键的存在时

S.N.	命令 & 描述
14	PSETEX key milliseconds value 设置键的毫秒值和到期时间
15	INCR key 增加键的整数值一次
16	INCRBY key increment 由给定的数量递增键的整数值
17	INCRBYFLOAT key increment 由给定的数量递增键的浮点值
18	DECR key 递减键一次的整数值
19	DECRBY key decrement 由给定数目递减键的整数值
20	APPEND key value 追加值到一个键

Redis SET命令

Redis GET命令

Redis GETRANGE命令

Redis GETSET命令

Redis GETBIT命令

Redis MGET命令

Redis SETEX命令

Redis SETNX命令

Redis SETRANGE命令

Redis STRLEN命令

Redis MSET命令

Redis MSETNX命令

Redis PSETEX命令

Redis INCR命令

Redis INCRBY命令

Redis INCRBYFLOAT命令

Redis INCRBYFLOAT命令用于递增代表存储在由指定的增量key浮点数的字符串。如果key不存在时，它被设置为0在执行操作之前。如果key包含错误的类型或当前key内容或指定的增量值不是可解析为浮点数，则连接返回错误。

返回值

回复字符串，是增加后key的值。

语法

redis INCRBYFLOAT命令的基本语法如下所示：

```
redis 127.0.0.1:6379> INCRBYFLOAT KEY_NAME INCR_AMOUNT
```

例子

```
redis 127.0.0.1:6379> SET visitors 1000.20
OK
redis 127.0.0.1:6379> INCRBYFLOAT visitors .50
1000.70
redis 127.0.0.1:6379> GET visitors
1000.70
```

Redis DECR命令

Redis DECR命令用于key的整数值减1。如果该键不存在时，它被设置为0执行操作之前。如果键包含了错误类型的值或包含不能被表示为整数，字符串，则返回错误。该操作被限制为64位带符号的整数。

返回值

回复整数，key增量后的值

语法

redis DECR命令的基本语法如下所示：

```
redis 127.0.0.1:6379> DECR KEY_NAME
```

例子

```
redis 127.0.0.1:6379> SET visitors 1000
OK
redis 127.0.0.1:6379> DECR visitors
(integer) 999
redis 127.0.0.1:6379> SET visitors "131313123123123123312rgergerg"
Ok
redis 127.0.0.1:6379> DECR visitors
ERR value is not an integer or out of range
```

Redis DECRBY命令

Redis DECRBY命令用于减小存储在由指定的值的key的数量。如果该键不存在时，它被设置为0在执行操作之前。如果键包含了错误类型的值或包含不能被表示为整数，字符串，则返回错误。

返回值

回复整数，key增量后的值

语法

redis DECRBY命令的基本语法如下所示：

```
redis 127.0.0.1:6379> DECRBY KEY_NAME DECREMENT_AMOUNT
```

例子

```
redis 127.0.0.1:6379> SET visitors 1000
OK
redis 127.0.0.1:6379> DECRBY visitors 5
(integer) 995
```

Redis APPEND命令

Redis APPEND命令用来添加键的一些值。

返回值

答复追加整数操作后的字符串的长度。

语法

Redis APPEND命令的基本语法如下所示：

```
redis 127.0.0.1:6379> APPEND KEY_NAME NEW_VALUE
```

例子

```
redis 127.0.0.1:6379> SET mykey "hello"
OK
redis 127.0.0.1:6379> APPEND mykey " yiibai"
(integer) 20
redis 127.0.0.1:6379> GET mykey
"hello yiibai"
```

Redis哈希

Redis的哈希值是字符串字段和字符串值之间的映射，所以他们是表示对象的完美数据类型

在Redis中的哈希值，可存储超过400十亿键值对。

例子

```
redis 127.0.0.1:6379> HMSET yiibai name "redis tutorial" description "redis basic commands for cachi
ng" likes 20 visitors 23000
OK
redis 127.0.0.1:6379> HGETALL yiibai
 1) "name"
2) "redis tutorial"
3) "description"
4) "redis basic commands for caching"
5) "likes"
6) "20"
7) "visitors"
8) "23000"
```

在上面的例子中，我们已经设置Redis的详细教程(name, description, likes, visitors)在哈希名称为 yiibai

Redis的哈希命令

如下表所示哈希一些基本的命令：

S.N.	命令和说明
1	HDEL key field2 [field2] 删除一个或多个哈希字段
2	HEXISTS key field 判断一个哈希字段存在与否
3	HGET key field 获取存储在指定的键散列字段的值
4	HGETALL key 让所有的字段和值在指定的键存储在一个哈希
5	HINCRBY key field increment 由给定数量增加的哈希字段的整数值
6	HINCRBYFLOAT key field increment 由给定的递增量哈希字段的浮点值
7	HKEYS key 获取所有在哈希字段
8	HLEN key 获取哈希字段数
9	HMGET key field1 [field2] 获得所有给定的哈希字段的值
10	HMSET key field1 value1 [field2 value2] 设置多个哈希字段的多个值
11	HSET key field value 设置哈希字段的字符串值

S.N.	命令和说明
12	HSETNX key field value 设置哈希字段的值，仅当该字段不存在
13	HVALS key 获取在哈希中的所有值
14	HSCAN key cursor [MATCH pattern] [COUNT count] 增量迭代哈希字段及相关值

Redis HDEL命令

Redis HDEL命令用于从存储在键散列删除指定的字段。如果没有这个哈希中存在指定的字段将被忽略。如果键不存在，它将被视为一个空的哈希与此命令将返回0。

返回值

回复整数，从散列中删除的字段的数量，不包括指定的但不是现有字段。

语法

redis HDEL命令的基本语法如下所示：

```
redis 127.0.0.1:6379> HDEL KEY_NAME FIELD1.. FIELDN
```

例子

```
redis 127.0.0.1:6379> HSET myhash field1 "foo"
(integer) 1
redis 127.0.0.1:6379> HDEL myhash field1
(integer) 1
redis 127.0.0.1:6379> HDEL myhash field2
(integer) 1
```

Redis HEXISTS命令

Redis HEXISTS命令被用来检查哈希字段是否存在。

返回值

回复整数，1或0。

- 1, 如果哈希包含字段。

- 0 如果哈希不包含字段，或key不存在。

语法

Redis HEXISTS命令的基本语法如下所示：

```
redis 127.0.0.1:6379> HEXISTS KEY_NAME FIELD_NAME
```

实例

```
redis 127.0.0.1:6379> HSET myhash field1 "foo"
(integer) 1
redis 127.0.0.1:6379> HEXISTS myhash field1
(integer) 1
redis 127.0.0.1:6379> HEXISTS myhash field2
(integer) 0
```

Redis HGET命令

Redis HGET命令用于获取与字段中存储的键哈希相关联的值。

返回值

回复字符串值关联字段，或nil当字段时不存在哈希或键不存在值。

语法

redis HGET命令的基本语法如下所示：

```
redis 127.0.0.1:6379> HGET KEY_NAME FIELD_NAME
```

实例

```
redis 127.0.0.1:6379> HSET myhash field1 "foo"
(integer) 1
redis 127.0.0.1:6379> HGET myhash field1
"foo"
redis 127.0.0.1:6379> HEXISTS myhash field2
(nil)
```

Redis HGETALL命令

Redis HGETALL命令用于获取存储在键的散列的所有字段和值。在返回的值是每一个字段名后跟其值，所以回复的长度是散列值两倍的大小。

返回值

回复数组字段及其值的列表存储在哈希表，或一个空表时，键不存在。

语法

redis HGETALL命令的基本语法如下所示：

```
redis 127.0.0.1:6379> HGETALL KEY_NAME
```

例子

```
redis 127.0.0.1:6379> HSET myhash field1 "foo"
(integer) 1
redis 127.0.0.1:6379> HSET myhash field2 "bar"
(integer) 1
redis 127.0.0.1:6379> HGETALL myhash
1) "field1"
2) "Hello"
3) "field2"
4) "World"
```

Redis HINCRBY命令

Redis HINCRBY命令用于增加存储在字段中存储由增量键哈希的数量。如果键不存在，新的key被哈希创建。如果字段不存在，值被设置为0之前进行操作。

返回值

回复整数，字段的增值操作后的值。

语法

redis HINCRBY命令的基本语法如下所示：

```
redis 127.0.0.1:6379> HINCRBY KEY_NAME FIELD_NAME INCR_BY_NUMBER
```

例子

```
redis 127.0.0.1:6379> HSET myhash field1 20
(integer) 1
redis 127.0.0.1:6379> HINCRBY myhash field 1
(integer) 21
redis 127.0.0.1:6379> HINCRBY myhash field -1
(integer) 20
```

Redis HINCRBYFLOAT命令

Redis HINCRBYFLOAT命令用于增加存储在key的散列值的指定字段中，并且表示为浮点数，由指定的增量。它被设置为0在执行操作之前。如果该字段包含错误类型的值或指定的增量不是可解析为浮点数，那么出错。

返回值

回复字符串，字段的增加后的值。

语法

redis HINCRBYFLOAT命令的基本语法如下所示：

```
redis 127.0.0.1:6379> HINCRBYFLOAT KEY_NAME FIELD_NAME INCR_BY_NUMBER
```

例子

```
redis 127.0.0.1:6379> HSET myhash field 20.50
(integer) 1
redis 127.0.0.1:6379> HINCRBYFLOAT mykey field 0.1
"20.60"
```

Redis HKEYS命令

Redis HKEYS命令是用来获取所有字段名保存在键的哈希值。

返回值

回复数组，哈希字段列表或者当key不存在是为一个空的列表。

语法

Redis HKEYS命令的基本语法如下所示：

```
redis 127.0.0.1:6379> HKEYS KEY_NAME FIELD_NAME INCR_BY_NUMBER
```

实例

```
redis 127.0.0.1:6379> HSET myhash field1 "foo"
(integer) 1
redis 127.0.0.1:6379> HSET myhash field2 "bar"
(integer) 1
redis 127.0.0.1:6379> HKEYS myhash
1) "field1"
2) "field2"
```

Redis HLEN命令

Redis HLEN命令用于获取包含存储于键的散列的字段的数量。

返回值

回复整数哈希字段数或0当键不存在。

语法

Redis HLEN命令的基本语法如下所示：

```
redis 127.0.0.1:6379> HLEN KEY_NAME
```

例子

```
redis 127.0.0.1:6379> HSET myhash field1 "foo"
(integer) 1
redis 127.0.0.1:6379> HSET myhash field2 "bar"
(integer) 1
redis 127.0.0.1:6379> HLEN myhash
(integer) 2
```

Redis HMGET命令

Redis HMGET命令用于获取与存储在键散列指定的字段相关联的值。如果字段中哈希不存在，则nil值被返回。

返回值

回复数组，给定字段相关联的值的列表，与在请求时它们的顺序相同。

语法

redis HMGET命令的基本语法如下所示：

```
redis 127.0.0.1:6379> HMGET KEY_NAME FIELD1...FIELDN
```

例子

```
redis 127.0.0.1:6379> HSET myhash field1 "foo"
(integer) 1
redis 127.0.0.1:6379> HSET myhash field2 "bar"
(integer) 1
redis 127.0.0.1:6379> HMGET myhash field1 field2 nofield
1) "foo"
2) "bar"
3) (nil)
```

Redis HMSET命令

Redis HMSET命令用于设置指定字段各自的值，在存储于键的散列。此命令将覆盖哈希任何现有字段。如果键不存在，新的key由哈希创建。

返回值

一个简单的字符串返回

语法

redis HMSET命令的基本语法如下所示：

```
redis 127.0.0.1:6379> HMSET KEY_NAME FIELD1 VALUE1 ...FIELDN VALUEN
```

实例

```
redis 127.0.0.1:6379> HSET myhash field1 "foo" field2 "bar"
OK
redis 127.0.0.1:6379> HGET myhash field1
"foo"
redis 127.0.0.1:6379> HMGET myhash field2
"bar"
```

Redis HSET命令

Redis HSET命令用于在存储的关键值的散列设置字段。如果键不存在，新的key由哈希创建。如果字段已经存在于哈希值那么将被覆盖。

返回值

返回整数

- 1 如果字段是哈希值和一个新字段被设置。
- 0 如果字段已经存在于哈希并且值被更新。

语法

redis HSET命令的基本语法如下所示：

```
redis 127.0.0.1:6379> HSET KEY_NAME FIELD VALUE
```

实例

```
redis 127.0.0.1:6379> HSET myhash field1 "foo"
OK
redis 127.0.0.1:6379> HGET myhash field1
"foo"
```

Redis HSETNX命令

Redis HSETNX命令用于在存储的关键值的散列设置字段，只有在字段不存在。如果键不存在，新的key会被哈希创建。如果字段已经存在，该操作没有任何影响。

返回值

返回整型

- 1 如果字段是哈希值和一个新字段被设置。
- 0 如果字段已经存在于哈希那么没有执行任何操作。

语法

Redis HSETNX命令的基本语法如下所示：

```
redis 127.0.0.1:6379> HSETNX KEY_NAME FIELD VALUE
```

例子

```
redis 127.0.0.1:6379> HSETNX myhash field1 "foo"
(integer) 1
redis 127.0.0.1:6379> HSETNX myhash field1 "bar"
(integer) 0
redis 127.0.0.1:6379> HGET myhash field1
"foo"
```

Redis HVALS命令

Redis HVALS命令用于获取在存储于 key的散列的所有值。

返回值

回复数组，列表中的散列值，或当key不存在则为一个空的列表。

语法

Redis HVALS命令的基本语法如下所示：

```
redis 127.0.0.1:6379> HVALS KEY_NAME FIELD VALUE
```

例子

```
redis 127.0.0.1:6379> HSET myhash field1 "foo"
(integer) 1
redis 127.0.0.1:6379> HSET myhash field2 "bar"
(integer) 1
redis 127.0.0.1:6379> HVALS myhash
1) "foo"
2) "bar"
```

Redis列表

Redis列表是简单的字符串列表，排序插入顺序。您可以在头部或列表的尾部Redis的列表添加元素。

列表的最大长度为232 - 1 (每个列表超过4十亿元素4294967295)元素。

例子

```
redis 127.0.0.1:6379> LPUSH tutorials redis
(integer) 1
redis 127.0.0.1:6379> LPUSH tutorials mongodb
(integer) 2
redis 127.0.0.1:6379> LPUSH tutorials mysql
(integer) 3
redis 127.0.0.1:6379> LRANGE tutorials 0 10
 1) "mysql"
2) "mongodb"
3) "redis"
```

在上述例子中的三个值被插入到redis的列表命名tutorials 使用LPUSH命令。

Redis的命令列表

如下表所示相关列出了一些基本的命令：

S.N.	命令 & 描述
1	BLPOP key1 [key2] timeout 取出并获取列表中的第一个元素，或阻塞，直到有可用
2	BRPOP key1 [key2] timeout 取出并获取列表中的最后一个元素，或阻塞，直到有可用
3	BRPOPLPUSH source destination timeout 从列表中弹出一个值，它推到另一个列表并返回它;或阻塞，直到有可用
4	LINDEX key index 从一个列表其索引获取对应的元素
5	LINSERT key BEFORE|AFTER pivot value 在列表中的其他元素之后或之前插入一个元素
6	LLEN key 获取列表的长度
7	LPOP key获取并取出列表中的第一个元素
8	LPUSH key value1 [value2] 在前面加上一个或多个值的列表
9	LPUSHX key value在前面加上一个值列表，仅当列表中存在
10	LRANGE key start stop 从一个列表获取各种元素
11	LREM key count value 从列表中删除元素

S.N.	命令 & 描述
12	LSET key index value 在列表中的索引设置一个元素的值
13	LTRIM key start stop 修剪列表到指定的范围内
14	RPOP key 取出并获取列表中的最后一个元素
15	RPOPLPUSH source destination 删除最后一个元素的列表，将其附加到另一个列表并返回它
16	RPUSH key value1 [value2] 添加一个或多个值到列表
17	RPUSHX key value 添加一个值列表，仅当列表中存在

Redis BLPOP命令

Redis BLPOP命令用于删除和获取列表中的第一个元素，或阻塞直到有可用。 BLPOP命令只返回第一个元素(如果有的话)，或阻塞客户端对指定的时间执行任意命令。

返回值

回复字符串，储存在key或nil值

语法

redis BLPOP命令的基本语法如下所示：

```
redis 127.0.0.1:6379> BLPOP LIST1 LIST2 .. LISTN TIMEOUT
```

例子

```
redis 127.0.0.1:6379> BLPOP list1 100
```

上面的例子会阻止客户端100秒来执行任意命令。如果有任何数据来自于指定的键list1则返回，否则后返回百秒nil值。

```
(nil)
(100.06s)
```

Redis BRPOPLPUSH命令

Redis BRPOPLPUSH命令用于从列表中弹出一个值，它推到另一个列表并返回它，或阻塞直到有可用。BRPOPLPUSH命令只返回最后一个元素，并插入到另一个列表中，如果有的话，或阻止客户端对指定的时间执行任意命令。

返回值

回复字符串，储存在key或nil值

语法

redis BRPOPLPUSH命令的基本语法如下所示：

```
redis 127.0.0.1:6379> BRPOPLPUSH LIST1 ANOTHER_LIST TIMEOUT
```

例子

```
redis 127.0.0.1:6379> BRPOPLPUSH list1 list2 100
```

上面的例子会阻止客户端100秒来执行任意命令。如果有任何数据来自于指定的键list1然后它会弹出数据并将其推入，否则的另一个列表百秒后返回nil值。

```
(nil)
(100.06s)
```

Redis LINDEX命令

Redis LINDEX命令用于获取在存储于列表的key索引的元素。索引是从0开始的，所以0表示第一个元素，1第二个元素等等。负指数可用于指定开始在列表的尾部元素。这里，-1表示最后一个元素，-2指倒数第二个等等。

返回值

字符串回复，请求的元素，或者nil当索引超出范围。

语法

redis LINDEX命令的基本语法如下所示：

```
redis 127.0.0.1:6379> LINDEX KEY_NAME INDEX_POSITION
```

实例

```
redis 127.0.0.1:6379> LPUSH list1 "foo"
(integer) 1
redis 127.0.0.1:6379> LPUSH list1 "bar"
(integer) 2
redis 127.0.0.1:6379> LINDEX list1 0
"foo"
redis 127.0.0.1:6379> LINDEX list1 -1
"bar"
redis 127.0.0.1:6379> LINDEX list1 5
nil
```

Redis LINSERT命令

Redis LINSERT命令插入值在存储在key之前或参考值支点后。如果key不存在，它被认为是一个空列表，并没有进行任何操作。当存在key但不持有列表值，则返回一个错误。

返回值

回复整数，列表插入操作后的长度，或-1时没有找到该值枢轴。

语法

redis LINSERT命令的基本语法如下所示：

```
redis 127.0.0.1:6379> LINSERT KEY_NAME BEFORE EXISTING_VALUE NEW_VALUE
```

例子

```
redis 127.0.0.1:6379> RPUSH list1 "foo"
(integer) 1
redis 127.0.0.1:6379> RPUSH list1 "bar"
(integer) 2
redis 127.0.0.1:6379> LINSERT list1 BEFORE "bar" "Yes"
(integer) 3
redis 127.0.0.1:6379> LRANGE mylist 0 -1
1) "foo"
2) "Yes"
3) "bar"
```

Redis LLEN命令

Redis LLEN命令将返回存储在key列表的长度。如果key不存在，它被解释为一个空列表，则返回0。当存储在关key的值不是一个列表，则会返回错误。

返回值

返回整数为列表键长度。

语法

Redis LLEN命令的基本语法如下所示：

```
redis 127.0.0.1:6379> LLEN KEY_NAME
```

例子

```
redis 127.0.0.1:6379> RPUSH list1 "foo"
(integer) 1
redis 127.0.0.1:6379> RPUSH list1 "bar"
(integer) 2
redis 127.0.0.1:6379> LLEN list1
(integer) 2
```