

排头兵 @ Talk

follow

用技术来驱动业务和产品体验

[首页](#) [关于我](#) [产品|观点](#) [思考&团队管理](#) [数据结构&算法](#) [网站优化](#) [项目 & 开源](#)
[微博](#)

Redis指令手册中文版

五月 19th, 2010 in NOSQL

[Skip to comments \(0\)](#) ↓

Redis指令手册中文版——powered by shjuto@gmail.com

连接控制

QUIT:退出,关闭连接

代码实例:连接 退出

```
telnet localhost 6379
QUIT
```

• AUTH: 密码验证

举例说明

- 1、首先需要在redis的配置文件redis.conf中requirepass注释掉的内容, 设置需要密码连接, 否则auth任何密码都通过
- 2、重新启动redis
- 3、验证 auth testpassword,testpassword是我在配置文件中设置的requirepass testpassword
- 4、redis 服务器的速度众所周知, 因此官方文件中 提醒设置比较复杂的密码,防止机器破解

```
telnet localhost 6379
Escape character is '^]'.
auth dsddsd
-ERR invalid password
keys global*
-ERR operation not premitted
auth ddddd
-ERR invalid password
auth testpassword
+OK
```

管理数据操作

- EXISTS:判断一个键是否存在;存在返回 1;否则返回0;

举例:

```
EXISTS burce
:0
SET bruce 10
```

排头兵
Evangelist
PHP / Blogger
Live in Shanghai
Work@SDO.com
Email@shjuto(at)gmail.com
[More](#)

最近文章

[国内的开放平台就是一个玩笑](#)
[分享会-高性能nosql数据库redis](#)
[盛大在线跨站攻击分享会](#)
[Bambook 知识和文化传承的载体](#)
[加入盛大在线](#)

最近评论

efish 在 [国内的开放平台就是一个玩笑](#) 上的评论
最弱网 在 [加入盛大在线](#) 上的评论
大大的小蜗牛 在 [国内的开放平台就是一个玩笑](#) 上的评论
youstar 在 [国内的开放平台就是一个玩笑](#) 上的评论
wss8848 在 [国内的开放平台就是一个玩笑](#) 上的评论

分类目录

[CSS](#)
[FreeBSD](#)
[jquery](#)
[memcached](#)
[MySQL](#)
[nginx](#)
[NOSQL](#)
[PHP](#)
[SEO技术](#)
[SEO新闻](#)
[人在江湖](#)
[团队管理](#)
[我看互联网](#)

```
paitoubing
+OK
SET test 5
paitoubing
+OK
-ERR unknown command 'ing'
EXISTS bruce
:1
```

上面的程序

EXISTS bruce 是否存在, 结果是不存在, 然后 *set* 一个 *key* 为 *bruce* 数据长度为 10 的数据, 如果数据长度操作设置的值, 多余的字节会当作 *redis* 命令来处理

DEL :删除某个 *key*, 或是一系列 *key*; *DEL key1 key2 key3 key4*

TYPE: 返回某个 *key* 元素的数据类型 (*none*:不存在, *string*: 字符, *list*, *set*, *zset*, *hash*)

KEYS: 返回匹配的 *key* 列表 (*KEYS foo**: 查找 *foo* 开头的 *keys*)

RANDOMKEY: 随机获得已经存在的 *key*

RENAME : 更改 *key* 的名字, 如果名字存在则更改失败

DBSIZE: 返回当前数据库的 *key* 的总数

EXPIRE: 设置某个 *key* 的过期时间 (秒), (*EXPIRE bruce 1000*: 设置 *bruce* 这个 *key* 1000 秒后系统自动删除)

TTL: 查找某个 *key* 还有多长时间过期, 返回时间秒

SELECT: 选择数据库

MOVE: 把 *key* 从一个数据库转移到另外一个库

FLUSHDB: 清空当前数据库数据

FLUSHALL: 清空所有数据库数据

字符串类型的数据操作

SET 存一个数据到数据库 *SET keyname datalength data* (*SET bruce 10 paitoubing*: 保存 *key* 为 *burce*, 字符串长度为 10 的一个字符串 *paitoubing* 到数据库)

GET: 获取某个 *key* 的 *value* 值

GETSET *GETSET* 可以理解成获得的 *key* 的值然后 *SET* 这个值, 更加方便的操作 (*SET bruce 10 paitoubing*, 这个时候需要修改 *bruce* 变成 1234567890 并获取这个以前的数据 *paitoubing*, *GETSET bruce 10 1234567890*)

MGET 一次性获得多个 *key* 的数据 (*MGET uid:1:name uid:1:email uid:1:ciy*)

SETNX *SETNX* 与 *SET* 的区别是 *SET* 可以创建与更新 *key* 的 *value*, 而 *SETNX* 是如果 *key* 不存在, 则创建 *key* 与 *value* 数据

SETEX *SETEX* = *SET* + *EXPIRE*, 貌似我的这个版本没有办法测试

MSET 一次性设置多个参数的值 (*MSET uid:1:name shjuto uid:1:email*

碎言碎语

网站设计

网站运营

网络营销

文章索引模板

2010年十一月

2010年九月

2010年八月

2010年七月

2010年六月

2010年五月

2010年三月

2010年二月

2010年一月

2009年十二月

2009年六月

2009年五月

2009年四月

2009年三月

2009年一月

2008年十二月

2008年十一月

2008年十月

2008年九月

2008年八月

2008年七月

2008年六月

2008年五月

2008年四月

2008年三月

2007年九月

2007年八月

2007年四月

2006年七月

链接

Jackzou

omiga

PHPPAN

Reco Lee

Show Framework

suppermen

Tino Web 开发

一亩三分地

五四陈科学院

伊人莫公

哥学社

恋上E人

甘！的！恒！自！址！

shjuto@gmail.com uid:1:city 8 回车 nanchang)最后一个值需要回车输入, 和 SET一样, 不知为啥。

MSETNX 如果设置的key不存在的话;或是叫做新key的话;一次性设置多个参数的值(MSET uid:1:name shjuto uid:1:email shjuto@gmail.com uid:1:city 8 回车 nanchang)最后一个值需要回车输入, 和SET一样, 不知为啥。

INCR 自增, 有点类是mysql incr.(INCR global:uid)

INCRBY 自增 +length ,(INCRBY uid 5)原来的基础+5=result

DECR 自减

* **DECRBY** 自减 -length

APPEND 一个例子足以说明

```
redis> exists mykey
(integer) 0
redis> append mykey "Hello "
(integer) 6
redis> append mykey "World"
(integer) 11
redis> get mykey
"Hello World"
```

SUBSTR 一个例子足以说明一切,LIKE PHP 'S STYLE

```
redis> set s "This is a string"
OK
redis> substr s 0 3
"This"
redis> substr s -3 -1
"ing"
redis> substr s 0 -1
"This is a string"
redis> substr s 9 100000
" string"
```

LISTS (无索引序列,head位置是0,.....)

RPUSH 追加数据到系列的尾部 (RPUSH listtest 10 \n 1111111122)

LPUSH 追加数据到序列的头部 (LPUSH listtest 10 \n 2222222222)

LLEN 一个序列的长度; (LLEN listtest)

LRANGE 从自定的范围内返回序列的元素 (LRANGE testlist 0 2;返回序列 testlist前0 1 2元素)

LTRIM 修剪某个范围之外的数据 (LTRIM testlist 0 2;保留0 1 2元素, 其余的删除)

LINDEX 返回某个位置的序列值(LINDEX testlist 0;返回序列testlist位置为零的元素)

LSET 更新某个位置元素的值 (LSET testlist 0 5 \n 55555;)

LPOP key Return and remove (atomically) the first element of the List at key

RPOP key Return and remove (atomically) the last element of the List at key

LREM 根据值删除序列元素 (LREM testlist 0 5 \n 33333;删除序列中所有的等于33333的元素,为何不是REMOVE BY KEY?不知道何故, 可能对删除重复数据有用吧)

BLPOP key1 key2 ... keyN timeout Blocking LPOP >1.31, 后续更新

未来的回忆地

武林

精神鸦片

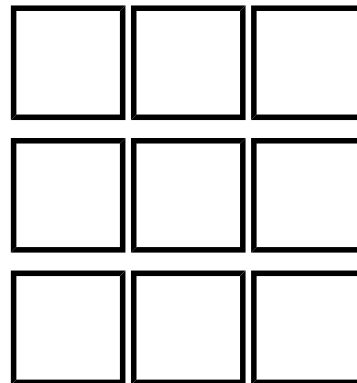
赖文辉

野兔窝

鸡尾吧

黄铭

Flickr



BRPOP key1 key2 ... keyN timeout Blocking RPOP >1.31
RPOPLPUSH srckey dstkey Return and remove (atomically) the last element of the source List stored at _srckey_ and push the same element to the destination List stored at _dstkey_

SETS (有索引无序序列)

SADD 增加元素到SETS序列,如果元素不存在则添加成功 1, 否则失败 0;
(SADD testlist 3 \n one)

SREM 删除SETS序列的某个元素, 如果元素不存则失败0, 否则成功 1
(SREM testlist 3 \N one)

SPOP 随机删除某个元素 (SPOP testlist)

SMOVE 把一个SETS序列的某个元素 移动到 另外一个SETS序列 (SMOVE testlist test 3\n two;从序列testlist移动元素two到 test中, —testlist中将不存在 two元素)

SCARD 统计某个SETS的序列的元素数量 (SCARD testlist)

SISMEMBER 产看某个数据是否在序列中,(SISMEMBER testlist 3 \n two)

SINTER 几个SETS序列的交集 SINTER key1 key2 ... keyN (SINTER test testlist),牛B呀

SINTERSTORE 把计算出来的交集 记录到一个新的序列 SINTERSTORE dstkey key1 key2 ... keyN (SINTERSTORE resultlist testlist test;把testlist test的交集 记录到resultlist)

SUNION 几个SETS序列的并集 SUNION key1 key2 ... keyN (SUNION test testlist)

SUNIONSTORE 把计算出来的并集 记录到一个新的序列 SUNIONSTORE dstkey key1 key2 ... keyN (SUNIONSTORE resultlist testlist test;把testlist test的交集 记录到resultlist)

SDIFF key1 key2 ... keyN,求出某几个序列的并集 与 某个序列 求出差集, 请看官方例子:
key1 = x,a,b,c
key2 = c
key3 = a,d
SDIFF key1,key2,key3 => x,b

SDIFFSTORE dstkey key1 key2 ... keyN ,和前面的SINTERSTORE SUNIONSTORE差不多, 对比

SMEMBERS KEY 返回某个序列的所有元素

SRANDMEMBER KEY 随机返回某个序列的元素

Commands operating on sorted sets (zsets, Redis version >= 1.1)

- **ZADD** key score member Add the specified member to the Sorted Set value at key or update the score if it already exist

- **ZREM** *key member* Remove the specified member from the Sorted Set value at key
- **ZINCRBY** *key increment member* If the member already exists increment its score by `_increment_`, otherwise add the member setting `_increment_` as score
- **ZRANK** *key member* Return the rank (or index) or `_member_` in the sorted set at `_key_`, with scores being ordered from low to high
- **ZREVRANK** *key member* Return the rank (or index) or `_member_` in the sorted set at `_key_`, with scores being ordered from high to low
- **ZRANGE** *key start end* Return a range of elements from the sorted set at key
- **ZREVRANGE** *key start end* Return a range of elements from the sorted set at key, exactly like ZRANGE, but the sorted set is ordered in traversed in reverse order, from the greatest to the smallest score
- **ZRANGEBYSCORE** *key min max* Return all the elements with score \geq min and score \leq max (a range query) from the sorted set
- **ZCARD** *key* Return the cardinality (number of elements) of the sorted set at key
- **ZSCORE** *key element* Return the score associated with the specified element of the sorted set at key
- **ZREMRANGEBYRANK** *key min max* Remove all the elements with rank \geq min and rank \leq max from the sorted set
- **ZREMRANGEBYSCORE** *key min max* Remove all the elements with score \geq min and score \leq max from the sorted set
- **ZUNIONSTORE / ZINTERSTORE** *dstkey N key1 ... keyN WEIGHTS w1 ... wN AGGREGATE SUM|MIN|MAX* Perform a union or intersection over a number of sorted sets with optional weight and aggregate

Commands operating on hashes

- **HSET** *key field value* Set the hash field to the specified value. Creates the hash if needed.
- **HGET** *key field* Retrieve the value of the specified hash field.
- **HMSET** *key field1 value1 ... fieldN valueN* Set the hash fields to their respective values.
- **HINCRBY** *key field integer* Increment the integer value of the hash at `_key_` on `_field_` with `_integer_`.
- **HEXISTS** *key field* Test for existence of a specified field in a hash
- **HDEL** *key field* Remove the specified field from a hash
- **HLEN** *key* Return the number of items in a hash.
- **HKEYS** *key* Return all the fields in a hash.
- **HVALS** *key* Return all the values in a hash.
- **HGETALL** *key* Return all the fields and associated values in a hash.

Sorting

- **SORT** *key BY pattern LIMIT start end GET pattern ASC|DESC ALPHA* Sort a

Set or a List accordingly to the specified parameters

Transactions

- [MULTI/EXEC/DISCARD](#) Redis atomic transactions

Publish/Subscribe

- [SUBSCRIBE/UNSUBSCRIBE/PUBLISH](#) Redis Public/Subscribe messaging paradigm implementation

Persistence control commands

- [SAVE](#) Synchronously save the DB on disk
- [BGSAVE](#) Asynchronously save the DB on disk
- [LASTSAVE](#) Return the UNIX time stamp of the last successfully saving of the dataset on disk
- [SHUTDOWN](#) Synchronously save the DB on disk, then shutdown the server
- [BGREWRITEAOF](#) Rewrite the append only file in background when it gets too big

Remote server control commands

- [INFO](#) Provide information and statistics about the server
- [MONITOR](#) Dump all the received requests in real time
- [SLAVEOF](#) Change the replication settings
- [CONFIG](#) Configure a Redis server at runtime

update:2010-05-21,my redis version 1.26

Tags: [redis](#)

This entry was posted on 星期三, 五月 19th, 2010 at 11:12 下午 and is filed under [NOSQL](#). You can follow any responses to this entry through the [RSS 2.0](#) feed. You can [leave a response](#), or [trackback](#) from your own site.

Comments

No comments so far.

Leave a Reply

Your Name

Name *

Email *

Website





Submit Comment

About This Website

Lamp development & SEO &
Plan of Website & Project
Managment

[Learn more »](#)

Follow Us (SNS)

 [Twitter](#)
 [Facebook](#)
 [Flickr](#)
 [Wealink](#)

Help & Support

[more about Bruce.xu »](#)

Get in touch

QQ: +252339382

Email: [shjuto @ gmail.com](mailto:shjuto@gmail.com)

[Online contact form »](#)