# redis安装&启动

## 安装&配置

redis是C语言开发的，安装需要进行编译，编译需要gcc环境，所以需要先安装gcc，执行命令：yum install gcc-c++。

①先去官网下载，然后用rz命令上传

②解压redis的压缩包



③解压后会有一个redis的目录，切换到该目录，用make命令编译

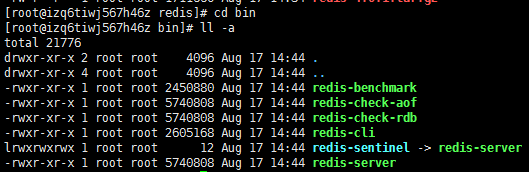


④安装，使用命令 make PREFIX=/usr/redis install，将编译成功后的

redis，安装到PREFIX参数指定的目录中



安装成功后，在指定目录下会出现一个bin目录，里面的内容有：



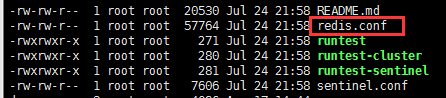
redis-benchmark ------性能测试工具

redis-check-aof ------AOF文件修复工具

redis-cli ------命令行客户端（重要）

redis-server ------redis服务器启动命令（重要）

⑤配置，在redis的解压目录中，有一个redis.conf配置文件：



需要把它复制到与bin目录一起：



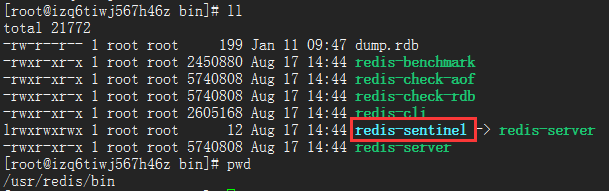
**一、额外配置：配置环境变量**

将redis/bin下面的命令配置到环境变量，就跟在windows配置Java环境变量一样，方便直接启动/关停redis服务。有如下2个步骤：

①拷贝redis源码包的哨兵脚本拷贝到redis的安装目录

cd /<redis源码目录>/src

cp redis-sentinel /<redis安装目录>

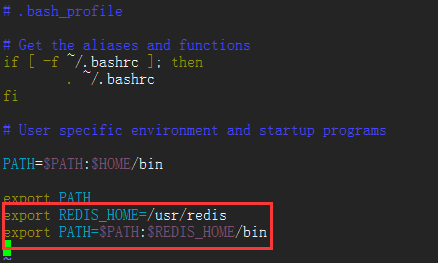


②配置Linux系统的环境变量

vim ~/.bash\_profile，打开环境变量文件后，加入下面这两句：

export REDIS\_HOME=/<redis安装目录>/bin

export PATH=$PATH:$REDOS\_HOME/bin



③刷新环境变量，让redis配置生效：source ~/.bash\_profile

**二、额外配置：配置服务**

可以将Redis配置成一个服务，然后使用service <服务名> start来启动redis就像在window下，使用net start mysql就能启动Mysql数据库一样

配置需要2个步骤：

①安装源码包下面的server.sh

cd <reids源码包目录>/utils

./install server.sh

执行上面两条命令，把编译好的redis作为一个服务，把6379.conf放到

/etc/init.d/redis\_6379

②重命名redis服务

cd /etc/init.d

mv redis\_6379 redisd

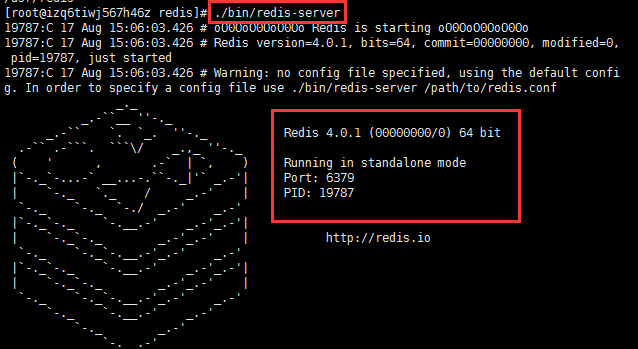
然后就可以使用service redisd start启动redis服务，以这种方式启动的redis，修改配置文件需要到/etc/redis/6379.conf去修改

## 启动方式

**①前端启动方式**

前端启动方式，非常简单，就是无法部署集群，首先，进入到redis的目录，

用 ./bin/redis-server命令启动



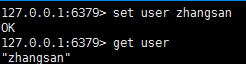
启动后可以看到redis的版本和位数，并且可以看到它的端口是6379，另起

一个会话，启动redis客户端，先进入redis的目录,用./bin/redis-cli

命令： redis-cli -h IP地址 -p 端口号(不写参数默认连本机)



然后就可以用set和get（类似java的map操作）来设值和取值



**②后端启动方式**

使用后端启动方式，需要修改redis.conf配置文件：把daemonize no改为

daemonize yes



切换到redis目录，使用命令：./bin/redis-server ./redis.conf



可以看到并不会像前端启动方式那样，占用界面，便于集群部署。后端启动后

关闭redis不同于前端启动那样，直接ctrl+c就可以结束服务：

./bin/redis-cli shutdown

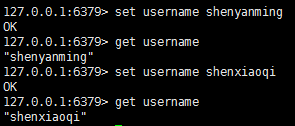


# 字符串命令

## set

set key value ：设定key持有的特定的字符串value，如果该Key存在则进

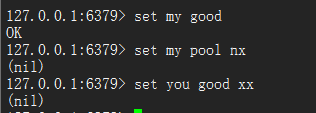
行覆盖操作，总是返回ok



set命令后面可以跟上2个后缀：[nx]、[xx]

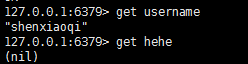
[nx]表示：键不存在时才可以设置，否则返回nil，防止覆盖键值

[xx]表示：键不存在时不能设置，否则返回nil，为了覆盖键值



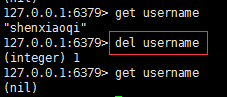
## get

get key ：获取Key对应的value，如果与该Key关联的value不是String类型，redis将返回错误信息，因为get命令只能用于获取String类型的key；如果该key不存在，则返回（nil）



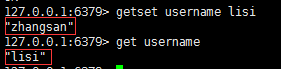
## del

del key ：删除指定的Key，返回值是一个integer类型，表示删除几条数据



## getset

getset key value ：先获取key的value值，然后再设置该key的value

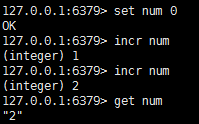


## incr

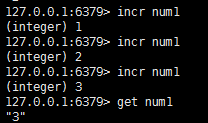
incr key ：将key对应的value值原子性的递增+1，相当于java中的++i，执行一次，就加1，然后将加完的结果返回。如果key不存在，则会自动创建并将其初始值设为0，在执行incr后加1，即结果返回1。如果value值不能转成整型，如hello，此操作会执行失败并返回相应的错误信息

**例子：**

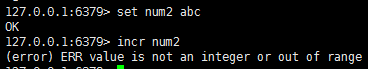
①设置num的value为0，每执行一次，num的值就+1...以此类推



②num1不先设置，直接incr num1，会帮我们自动执行set，并赋初始值为0

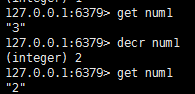


③设置num2的value为abc（非整形），会报错

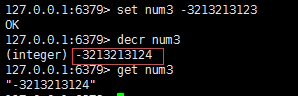


## decr

decr的作用和incr的作用相反，它是递减，相当于java中的--i



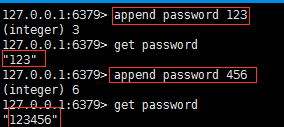
而且，decr是可以一直减到负数的



## append

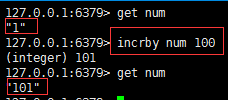
append key value：拼凑字符串，如果该key存在，则在原有的valye后追加改值；如果key不存在，新创建一个key/value。与set命令不一样，set如果是同样的key，它会覆盖掉，而append不会覆盖，只是拼接在原有value的屁股后面

如：password不存在，直接append password 123，会帮我们自动set一个，然后再执行append password 456，就会在原有123的基础上加上456

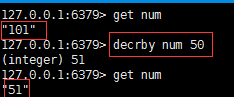


## incrby、decrby

incrby是跟incr一样，但是它可以指定加多少：incrby key 值



decrby是跟decr一样，但是它可以指定减多少：decrby key 值



# 散列命令

## hset/hmset

①hset key field value ：为key设置键值对（field/value），即本身

key—field/value是一个键值对，而field-value又是一个键值对



(为hash1设置{name:zhangsan}，重复或多次执行，会覆盖原来的值)

②hmset key field1 value1 field2 value2 ... ... ：

为key设置多个键值对

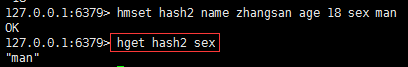


(为hash2设置{name:"zhangsan",age:18,sex:"man"})

## hget/hmget

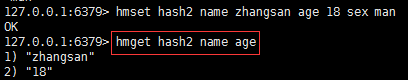
①hget key field ：先取key对应的value键值对，再从value键值对中

取field对应的值



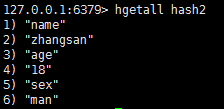
②hmget key field1 field2... ... ：取key对应的键值对中的多个field

对应的值



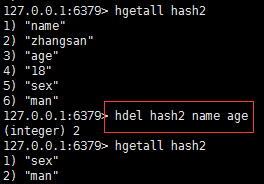
③hgetall key ：获取key中对应的所有field-value值，不仅获取属性名

还获取属性值



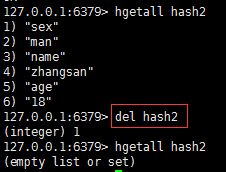
## hdel/del

①hdel key field1 field2...：删除散列内的若干个属性，返回删除个数



**PS**：删掉field后，若此hash数据没有任何字段，会自动把整个key删掉

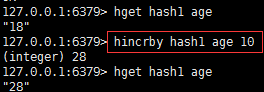
②del key ：删除整个hash键值对



## hincrby

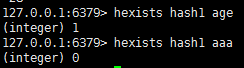
hincrby key field 数值：增加hash数据某个字段的值，注意只能是数字

返回增加后的值



## hexists

hexists key field ：判断指定的key中的field是否存在，存在返回1，不存在返回0



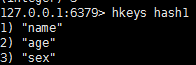
## hlen

hlen key ：获取key中所包含的field数量，有多少就返回多少



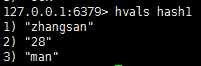
## hkeys

hkeys key ：获取key中所有的field字段



## hvals

hvals key ：获取key中所有field对应的值



# 列表命令

列表是有头和尾的，在Redis中：头指左端，尾指右端

## lpush/rpush

①lpush key value1 value2... ：在指定的Key所关联的List的头部(左端)插入所有的values，如果该Key不存在，该命令在插入之前会先创建一个与该Key关联的空链表，之后再向该链表的头部(左端)插入数据。插入成功返回元素的个数

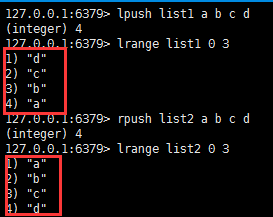
②rpush key val1 val2...：在该key关联的list的尾部(右端)添加数据

**注意：**

lpush命令每次都会从左侧添加数据；rpush命令，每次都会从右侧添加数据

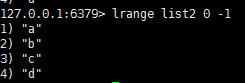
所以同样的数据：lpush list1 a b c d和rpush list2 a b c d

执行完的效果： lpush：d-->c-->b-->a，rpush：a-->b-->c-->d



## lrange

lrange key start end ：获取链表中从start到end的元素的值，start、end从0开始计数；也可以为负数，若为-1表示链表尾部的元素，-2表示倒数第二个元素，以此类推... start=0，end=-1，就可以将全部的值取出

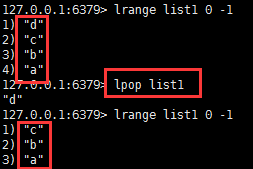


注：只有lrange没有rrange，换句话说都是从左---->右取值。

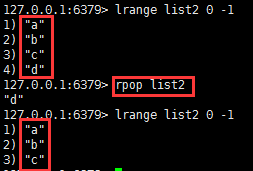
## lpop

①lpop key ：返回并弹出指定key关联的链表中的第一个元素，即头部元素。

如果该key不存在，返回nil；若key存在，则返回删除的元素



②rpop key ：从尾部弹出元素，同上



## llen

llen key ：返回key关联的链表中元素的数量

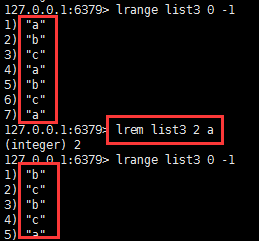


## lrem(效率极低)

lrem key count value ：删除count个值为value的元素，如果count大于0，从头到尾遍历并删除count个值为value的元素；如果count小于0，则从尾到头遍历删除count个值为value的元素。如果count=0，删除全部值为value的元素。

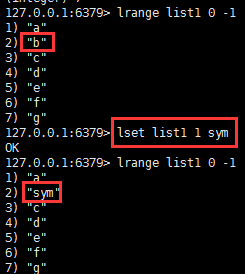
虽然此命令可以删除指定的元素，但是由于他在删除的时候会赋索引，所以如果在大数据量的前提，仅仅赋索引就会消耗大量的内存

**例子：**lrem list3 2 a ：表示从头到尾数，删除2个a值，第3个a值就没删



## lset(效率不高)

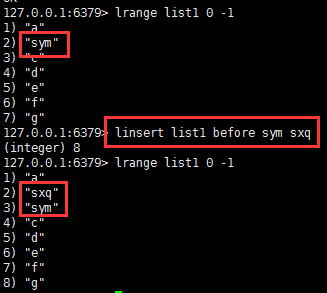
lset key index value ：设置key对应链表，下标为Index的元素的值，index=0,表示链表头部。index=-1表示链表尾部，下标不存在会抛异常。此命令跟lrem命令一样，都会赋索引，一旦赋索引，就会拉低redis的效率



## linsert(效率不高)

linsert key before/after pivot value：在pivot元素前/后插入value

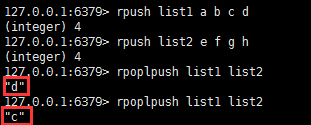
元素，before指前面插入，after指后面插入，pivot指元素的值，由于此命令也是要建立索引，所以效率也不高。



## rpoplpush

rpoplpush key1 key2 ：将key1的尾部元素弹出插入到key2的头部，此

命令可以分开看，rpop是右侧弹出，lpush是左侧插入。如果key1==key2，那就是循环队列，将该队列的尾部一直弹出并插入到头部中



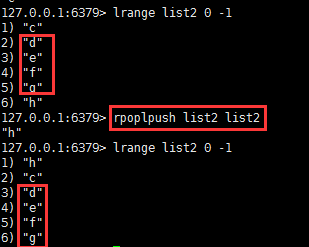
第一次执行，list1的尾部元素是d，会把d弹出然后插到list2的头部：

此时list1 ：a b c，此时list2 ：d e f g h

第二次执行，list1的尾部元素是c，会把c弹出然后插到list2的头部：

此时list1 ：a b，此时list2 ：c d e f g h

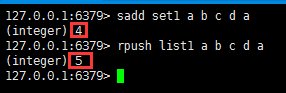
但是，如果两个key都是同一个List本身，就会循环地将自己的尾部弹出并左插到自己的头部中：



# 集合命令

## sadd

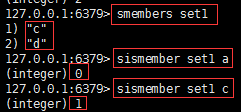
sadd key value1 valu2 ... ：可以指定多个value，但是不能重复，就算添加了n个一样的value，也只保留一个value



## smembers

①smembers key ：获取set中所有的成员

②sismenber key member ：判断参数中指定的成员是否在该set中，1表示存在，0表示不存在或者该key本身就不存在，无论集合中有多少元素都可以极速地返回结果



## srem

srem key value1 valu2 ... ：删除key关联的set集合中指定的值为

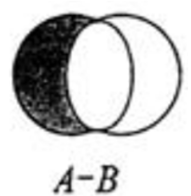
value1、value2...若干个数据



## 差集运算

sdiff key1 key2：返回key1与key2中相差的成员，且与key的顺序有关，

即谁先写，就以谁为基础，key1先写，就获取key1有且key2没有的元素集合

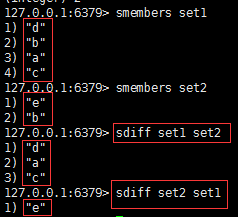
（属于A且不属于B的元素构成的集合）

set1: [a b c d]

set2: [b e]

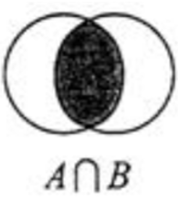
sdiff set1 set2：set1有但set2没有的，扣去b，就是acd

sdiff set2 set1：set2有但set1没有的，扣去b，就是e

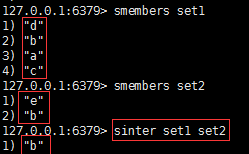


## 交集运算

sinter key1 key2 key3... ：返回2个或2个以上集合的交集

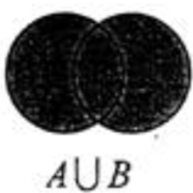
（属于A且属于B的元素构成的集合）

交集运算没有差集运算对顺序的要求，谁先写谁后写都一样

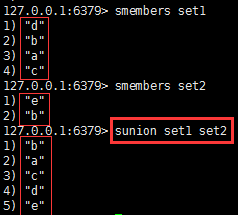


## 并集运算

sunion key1 key2 key3 ... ：返回2个或2个以上集合的并集

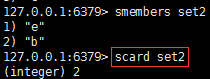


**注意：**即使是并集运算，结果也是set集合，所以也就不会有重复的元素



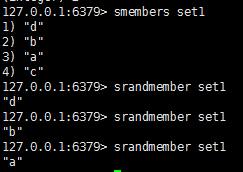
## scard

scard key ：获取set中成员的数量，返回set中的元素总个数



## srandmember

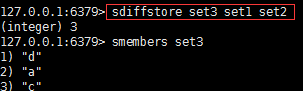
srandmember key ：随机返回set中的一个成员



## sdiffstore

sdiffstore destination key1 key2：将key1与key2的差集储存在

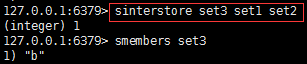
destination上，返回key1和key2差集的元素个数



## sinterstore

sinterstore destination key1 key2：将key1与key2的交集储存在

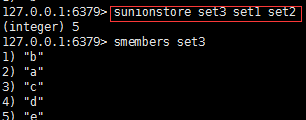
destination上，返回交集的元素个数



## sunionstore

sunionstore destination key1 key2 ... ：将返回的并集储存在

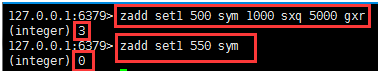
destination上,返回并集的个数



# 有序集合命令

## zadd

zadd key score1 member1 score2 member2... ：将所有成员和它们的分数保存到sorted-set中，如果该元素已经存在则会用新的分数替换原有的分数。返回值是新加入到集合中的元素个数，不包含之前已经存在的元素。新添加的元素才会被返回，已存在的元素只会被修改不会被返回



如果新增的元素的分数不是数字，会报错：

C:\Users\user\Desktop\{2F41B704-5F8C-43DB-9463-32838D7199A1}.bmp

## zscore/zcard/zrange

①zscore key member：返回指定成员的分数

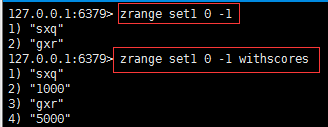


②zcard key：获取集合中的成员数量



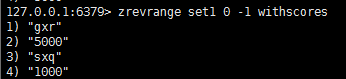
③zrange key start end [withscores]：**正序**获取集合中下标从start

到end的元素集合，[withscores]表示参数可选，是否将其分数也返回

（由小到大）

④zrevrange key start end [withscores]：**倒序**获取集合中下标从start

到end的元素集合，[withscores]表示参数可选，是否将其分数也返回

（由大到小）

## zrem

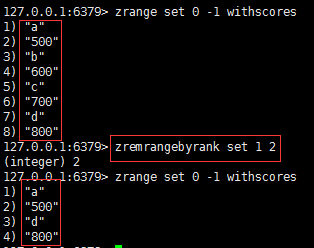
zrem key member1 member2...：移除集合中指定的成员，可以指定多个

成员移除几个，就返回几个



## zremrangebyrank

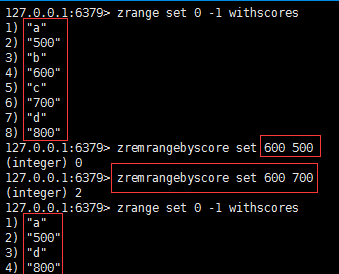
zremrangebyrank key start end：按照排名范围删除元素，排名第1的为0，排名最后的为-1，例子中意思是删除排名为2和3的元素，说是排名其实就是集合下标，从0开始算的集合下标



## zremrangebyscore

zremrangebyscore key min max：按照分数范围删除元素，注意这个分数

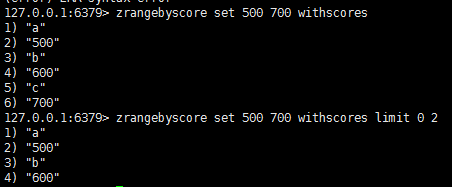
范围前后都是闭区间。删除几条数据就会返回几条，没有删除数据就返回0，



## zrangebyscore

zrangebyscore key min max [withscores] [limit offset count]：

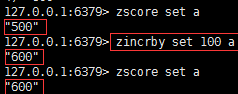
返回分数在[min,max]之间的成员并按照分数从低到高排序。[withscores]：是否显示分数；[limit offset count]类似mysql的分页查询，表示从下标为offset的元素开始并返回count个成员，如limit 0 1：从第一个元素开始，返回1个元素，其实就是返回第一个元素



## zincrby

zincrby key increment member：为member指定成员的分数增加

increment的值。返回值是更改后的分数



## zcount

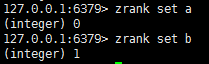
zcount key min max：获取分数在[min,max]之间的成员数量，有几个就返回几，注意这只是返回数量值而已



## zrank

zrank key member：返回成员在集合中的排名。索引(从小到大)

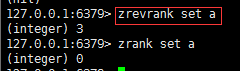
第一名返回0，以此类推，如果集合中没有此元素，返回nil



## zrevrank

zrevrank key member ：返回成员在集合中的排名。索引(从大到小)

第一名返回0，以此类推，如果集合中没有此元素，返回nil

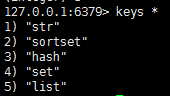


# 通用命令

通用命令是每种数据类型都可以使用

## keys

keys pattern ：获取所有与pattern匹配的键值，\*表示任意0个或多个字符，?表示任意一个字符

 (???表示3个字符)

## del

del key1 key2 ... ：删除指定的key(可以多尔衮)

## exists

exists key ：判断key是否存在，1表示存在，0表示不存在

## rename

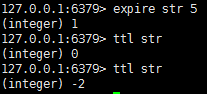
rename Key newKey ：为当前的Key重命名

## expire

expire key time ：为key设置过期时间，单位是秒

## ttl

ttl key ：获取该Key所剩的超时时间，如果没有设置，返回-1，如果返回-2表示该key已经超时，自动被redis删除，不存在了



## type

type key ：获取指定key值的类型，以字符串的格式返回。返回的字符串为string、list、set、hash和zset，如果该Key不存在则返回none

## info

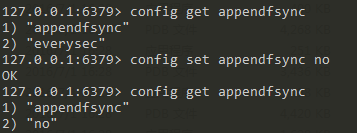
info ：查看当前redis数据库的所有信息

## config

config get：获取redis的某项配置

config set：设置redis的某项配置

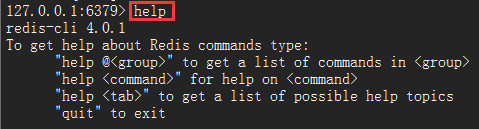
（注意这样设置只是暂时的，在服务器重启后就会失效）



## help

help命令可以帮助我们在只知道命令用途，不知道命令的语法时，给我们提示

help命令提供的部分功能如下：



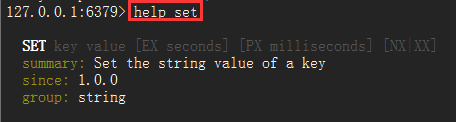
①help @<group>给出一种数据类型的命令列表

<group> 通常为string，hash，set，sortet\_set,list

②help <command>给出一个命令的提示

比如想知道set命令的用法和格式可以这样：

redis会详细告诉我们语法、用法...

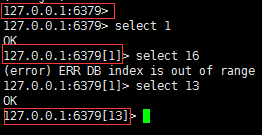


# 数据库命令

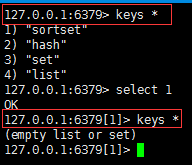
在redis中，也是有数据库的概念，而且redis已经自己创建好了：redis默认有16个数据库，0,1,2,3... ...15，它是以阿拉伯数字作为数据库名称，一般默认操作的是0号数据库，注：数据库与数据库之间，不能共享键值对

## select

使用命令：select [数据库编号]，可以切换数据库，除了0号数据库不会加任何修饰外，切换到其他数据库，会在主机地址后面加一个[数据库编号]

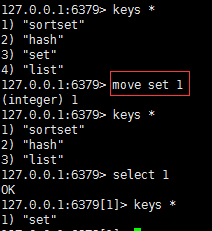


数据库与数据库之间的键值对不会共享，并不会因为切换了数据库而将键值对也复制一份过去



## move

使用命令：move [key名] [数据库编号]，将当前数据库的key移植到指定的数据库中。例如：将0号数据库“set”这个key移植到1号数据库中，移植后0号数据库就没有"set"这个key，而1号数据库多了一个"set"的key



## flush

①flushdb ：清空当前数据库

②flushall ：redis服务器所有数据清空

注：这两个命令慎用，一旦删除了，数据就不可挽回的没了