







环境准备

58和59 的Redis服务停止

把redis集群中的主机恢复为独立数据库服务器：

51 - 56 做如下操作

停止redis服务

注释配置文件里的集群功能配置项

清空数据库目录

启动redis服务

查看进程 netstat -utnlp | grep redis-server 只有redis服务端口

查看数据 keys \* 没有数据

1 主从复制

分为主master数据库服务器 和 从slave数据库服务器

master数据库服务器:客户端存取数据时连接的服务器

slave数据库服务器：连接主服务器 同步主服务器的数据到本机

配置redis主从结构，涉及到的命令

注意 : redis服务启动后，默认就是master角色

命令行配置（立即生效，但重启redis服务或操作系统会还原）

> slaveof 主服务器ip 主服务器端口

> info replication # 查看复制信息

> slaveof no one #临时恢复为主服务器

永久配置 vim /etc/redis/6379.conf

slaveof 主服务器ip 主服务器端口

主从复制结构： 一主一从 一主多从 主从从

配置一主一从

要求：把52配置为51的从服务器

52的配置如下

redis-cli -h 192.168.4.52 -p 6352

> slaveof 192.168.4.51 6351

> info replication

> exit

vim /etc/redis/6379.conf

282 slaveof 192.168.4.51 6351

:wq

配置一主多从

要求;把53主机 也配置为51 的从服务器

53主机的配置，如下 时间3分钟到14:17

vim /etc/redis/6379.conf

282 slaveof 192.168.4.51 6351

:wq

]#redis-cli -h 192.168.4.53 -p 6353 shutdown

]# /etc/init.d/redis\_6379 start

]# redis-cli -h 192.168.4.53 -p 6353 查看到 和 51一样的数据

51 主机查看复制信息

192.168.4.51:6351> INFO replication

# Replication

role:master

connected\_slaves:2

slave0:ip=192.168.4.52,port=6352,state=online,offset=11645,lag=0

slave1:ip=192.168.4.53,port=6353,state=online,offset=11645,lag=1

配置主从从 练习3分钟 到 14:31

要求 把主机54 配置为53的slave服务器

54主机的配置如下：

192.168.4.54:6354> slaveof 192.168.4.53 6353

192.168.4.54:6354> config rewrite #把配置保存的/etc/redis/6379.conf文件

在53主机查看复制信息

192.168.4.53:6353> info replication #是51的slave 又是54的master

在51主机 存储新数据 52 53 54 三台主机都会有数据

配置带验证的主从复制（主服务器有连接密码）

51主机 设置连接密码

[root@localhost ~]# vim +501 /etc/redis/6379.conf

requirepass 654321

:wq

[root@localhost ~]# redis-cli -h 192.168.4.51 -p 6351 shutdown

[root@localhost ~]# /etc/init.d/redis\_6379 start

Starting Redis server...

[root@localhost ~]# redis-cli -h 192.168.4.51 -p 6351 -a 654321

192.168.4.51:6351> keys \*

53 he 52 主机 指定连接密码

]# vim /etc/redis/6379.com

289 masterauth 654321

:wq

]# redis-cli -h 192.168.4.52 -p 6352 shutdown

]# /etc/init.d/redis\_6379 start

]# redis-cli -h 192.168.4.52 -p 6352

>info replication

哨兵服务：监视主从复制结构中的主服务器，把对应的从升级为master 服务器。

在主机57 配置哨兵服务 监视主数据库服务器51 会自动把对应的从升级为主服务器。

时间到15:20

51 和 52 为一主一从结构 （让53不做51的 slave服务器）

在主机57 配置哨兵服务 步骤如下

安装Redis软件（不需要做初始配置） 时间3分钟到 15:33

yum -y install gcc

tar -zxvf redis-4.0.8.tar.gz

cd redis-4.0.8

make && make install

创建并编辑哨兵服务的主配置文件

vim /etc/sentinel.conf

bind 192.168.4.57

sentinel monitor redisser 192.168.4.51 6351 1

sentinel auth-pass redisser 654321

:wq

启动配置文件运行哨兵服务

~]# redis-sentinel /etc/sentinel.conf

测试 ：

停止51主机的redis服务

在 52主机 查看复制信息（发现52的角色自动变为master）

查57的哨兵配置文件，自动监视当前的主服务器52

2 持久化（理论 大于 操作）

（redis服务 先把客户端的数据存储在内存里， 在把数据保存到硬盘）

2.1 RDB 指定的就是 数据库库目录下的dump.rdb文件

#使用dump.rdb文件恢复数据

50主机备份文件

[root@host50 ~]# cp /var/lib/redis/6379/dump.rdb /root/

[root@host50 ~]# ls /root/\*.rdb

/root/dump.rdb

[root@host50 ~]# scp /root/dump.rdb root@192.168.4.55:/opt/

55 主机恢复数据，步骤如下：

停止redis服务 ]# redis-cli -h 192.168.4.55 -p 6355 shutdown

移动备份文件到数据库目录下

[root@localhost ~]# ls /var/lib/redis/6379/

dump.rdb

[root@localhost ~]# mv /opt/dump.rdb /var/lib/redis/6379/

mv：是否覆盖"/var/lib/redis/6379/dump.rdb"？ y

[root@localhost ~]#

启动服务 /etc/init.d/redis\_6379 start

查看数据redis-cli -h 192.168.4.55 -p 6355

> keys \*

验证存盘频率

192.168.4.50:6350> exit

[root@host50 ~]# redis-cli -h 192.168.4.50 -p 6350 -a 123456 shutdown

[root@host50 ~]# ls /var/lib/redis/6379/

dump.rdb

[root@host50 ~]# rm -rf /var/lib/redis/6379/dump.rdb

[root@host50 ~]# vim /etc/redis/6379.conf

save 900 1

#save 300 10

save 120 10

save 60 10000

:wq

[root@host50 ~]# /etc/init.d/redis\_6379 start

测试 连接服务 在120秒内 存储10或10以上的变量 ，查看数据库目录下是否有dump.rdb文件

2.2 AOF: （Append Only File）

追加方式记录，在服务器上执行的写命令的 文件 （默认没有启用）

默认每隔1秒记录一次

启用aof

[root@host50 ~]# redis-cli -h 192.168.4.50 -p 6350 -a 123456

192.168.4.50:6350> config set appendonly yes

OK

192.168.4.50:6350> config rewrite

OK

192.168.4.50:6350> exit

[root@host50 ~]# grep -n 'appendonly' /etc/redis/6379.conf

673:appendonly yes

675:# The name of the append only file (default: "appendonly.aof")

677:appendfilename "appendonly.aof"

[root@host50 ~]#

[root@host50 ~]# ls /var/lib/redis/6379/

appendonly.aof dump.rdb

[root@host50 ~]#

[root@host50 ~]# wc -l /var/lib/redis/6379/appendonly.aof

82 /var/lib/redis/6379/appendonly.aof

[root@host50 ~]#

使用备份的AOF文件恢复数据

在50主机备份

[root@host50 ~]# cp /var/lib/redis/6379/appendonly.aof /root/

[root@host50 ~]# ls /root/appendonly.aof

/root/appendonly.aof

[root@host50 ~]# scp /root/appendonly.aof root@192.168.4.55:/opt/

在55 主机恢复 ，步骤如下

1 停止Redis服务

2 清空数据库目录

3 修改主配置文件启用AOF

vim /etc/redis/6379.conf

appendonly yes

:wq

4 把备份的aof文件移动到数据库目录下

mv /opt/appendonly.aof /var/lib/redis/6379/

5 启动服务

6 连接服务查看数据

3 数据类型

3.1 字符类型 set mset get mget setrange strlen append

setbit bitcount decr decrby incr incrby incrbyfloat

setbit 使用二进制数给变量赋值 0 或 1 使用二进数存储数据的目的是为节省内存存储空间

bitcount 统计二进制数中1的个数

有1G的内存存储空间 可以存储下多少个0 或 1

1G=1024M

1M=1024K 1M 1024\*1024\*8

1K=1024bytes 1k 1024\*8

1bytes=8bit

1bit=0或1

setbit nb 1 1

setbit nb 2 0

setbit nb 3 1

setbit nb 4 1

5 0

6 0

7 1

setbit nb 8 0

nb 10110010 365bit /8 = 45bytes \* 150 = 6750bytes / 1024 = 6k

3.2 列表类型 list 把多个值存储到一个变量里，

lpush llen lrange lpop lindex lset rpush rpop

3.3 hash类型 也叫做hash表

一个变量可以存储多列，每列都各自的值 节省存储空间

hset hmset hget hmget hkeys hvals hgetall hdel

**NSD NOSQL DAY03**

1. [案例1：redis主从复制](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN202001/NOSQL/DAY03/CASE/01/index.html#case1)
2. [案例2：配置带验证的主从复制](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN202001/NOSQL/DAY03/CASE/01/index.html#case2)
3. [案例3：哨兵服务](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN202001/NOSQL/DAY03/CASE/01/index.html#case3)
4. [案例2 ：使用RDB文件恢复数据](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN202001/NOSQL/DAY03/CASE/01/index.html#case4)
5. [案例5 ：使用AOF文件恢复数据](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN202001/NOSQL/DAY03/CASE/01/index.html#case5)
6. [案例6：string 字符串](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN202001/NOSQL/DAY03/CASE/01/index.html#case6)
7. [案例7：list 列表](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN202001/NOSQL/DAY03/CASE/01/index.html#case7)
8. [案例8：hash表](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN202001/NOSQL/DAY03/CASE/01/index.html#case8)

**1 案例1：redis主从复制**

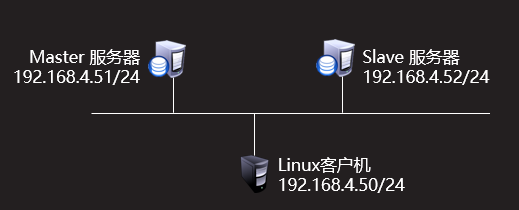
**1.1 问题**

具体要求如下：

* 将主机192.168.4.51配置为主服务器
* 将主机192.168.4.52配置为192.168.4.51的从服务器
* 测试配置

**1.2 方案**

部署redis一主一从复制结构，主机角色，如图-1所示：



图－1

**1.3 步骤**

实现此案例需要按照如下步骤进行。

步骤一：将主机192.168.4.51配置为主服务器

1） 每台redis服务器，默认都是主服务器;所以主服务器不需要配置。

1. [root@redisA ~]# redis-cli -h 192.168.4.51 –p 6351
2. 192.168.4.51:6351> info replication        //查看复制信息
3. # Replication
4. role:master            //是master 服务器
5. connected\_slaves:0 //从服务器个数零台
6. master\_replid:eaa14478158a71c41f947eaea036658c2087e8f2
7. master\_replid2:0000000000000000000000000000000000000000
8. master\_repl\_offset:0
9. second\_repl\_offset:-1
10. repl\_backlog\_active:0
11. repl\_backlog\_size:1048576
12. repl\_backlog\_first\_byte\_offset:0
13. repl\_backlog\_histlen:0
14. 192.168.4.51:6351>

步骤二：将主机192.168.4.52配置为192.168.4.51的从服务器

1）命令行配置（马上生效）

1. [root@redisB ~]# redis-cli -h 192.168.4.52 –p 6352
2. 192.168.4.52:6352> slaveof 192.168.4.51 6351 //指定主服务器ip地址与端口
3. OK
4. 192.168.4.52:6352> info replication //查看复制信息
5. # Replication
6. role:slave //从服务器
7. master\_host:192.168.4.51        //主服务器ip地址
8. master\_port:6351 //主服务器端口
9. master\_link\_status:up //连接状态开启
10. master\_last\_io\_seconds\_ago:3
11. master\_sync\_in\_progress:0

2）永久配置（重新redis服务后，依然有效）

1. [root@redisB ~]# vim /etc/redis/6379.conf
2. slaveof 192.168.4.51 6351 //在文件末尾添加或在原有配置项上修改都可以
3. :wq

3）在主服务器查看复制信息

1. [root@redisA ~]# redis-cli -h 192.168.4.51 –p 6351
2. 192.168.4.51:6351> info replication //查看复制信息
3. # Replication
4. role:master
5. connected\_slaves:1 //从服务器个数 1台
6. slave0:ip=192.168.4.52,port=6352,state=online,offset=14,lag=1    //从服务器信息
7. master\_replid:db7932eb0ea4302bddbebd395efa174fb079319f
8. master\_replid2:0000000000000000000000000000000000000000
9. master\_repl\_offset:14
10. second\_repl\_offset:-1
11. repl\_backlog\_active:1
12. repl\_backlog\_size:1048576
13. repl\_backlog\_first\_byte\_offset:1
14. repl\_backlog\_histlen:14
15. 192.168.4.51:6351>

步骤三：测试配置

1）客户端连接主服务器存储数据

1. [root@client50 ~]# redis-cli -h 192.168.4.51 –p 6351
2. 192.168.4.51:6351> set x 9
3. OK
4. 192.168.4.51:6351> set y 8
5. OK
6. 192.168.4.51:6351> set z 7
7. OK
8. 192.168.4.51:6351>

2）在从服务器本机登录，查看数据（与主服务器数据一致）

1. [root@redisB ~]#     redis-cli -h 192.168.4.52 –p 6352
2. 192.168.4.52:6352> keys \*
3. 1）“x”
4. 2）“y”
5. 3）“z”
6. 192.168.4.52:6352>

**2 案例2：配置带验证的主从复制**

**2.1 问题**

* 具体要求如下：
* 基于案例1的配置
* 设置主服务器192.168.4.51 设置连接密码123456
* 配置从服务器192.168.4.52

**2.2 步骤**

实现此案例需要按照如下步骤进行。

步骤一：设置主服务器192.168.4.51 设置连接密码123456

1） 修改主服务器的配置文件，设置密码。

1. [root@redisA ~]# vim +501 /etc/redis/6379.conf
2. requirepass 123456 //设置密码
3. :wq
4. [root@redisA ~]# vim +43 /etc/init.d/redis\_6379 //修改脚本
5. $CLIEXEC -h 192.168.4.51 -p 6351 -a 123456 shutdown //添加密码
6. :wq
7. [root@redisA ~]# /etc/init.d/redis\_6379 stop //停止服务
8. [root@redisA ~]# /etc/init.d/redis\_6379 start //启动服务
9. Starting Redis server...
10. [root@redisA ~]#
11. [root@redisA ~]# netstat -utnlp | grep :6351 //查看端口
12. tcp 0 0 192.168.4.51:6351 0.0.0.0:\* LISTEN 11523/redis-server

步骤二：配置从服务器192.168.4.52

1） 修改配置文件，设置主服务器连接密码。

1. [root@redisB ~]# /etc/init.d/redis\_6379 stop //停止服务
2. [root@redisB ~]# vim +289 /etc/redis/6379.conf
3. masterauth 123456 //设置密码
4. :wq
5. [root@redisA ~]# /etc/init.d/redis\_6379 start //启动服务
6. Starting Redis server...
7. [root@redisA ~]#
8. [root@redisA ~]# netstat -utnlp | grep :6351 //查看端口
9. tcp 0 0 192.168.4.51:6351 0.0.0.0:\* LISTEN 11523/redis-server

2） 在从服务器本机连接服务，查看复制信息

1. [root@redisB ~]# redis-cli -h 192.168.4.52 –p 6352
2. 192.168.4.52:6352> info replication //查看复制信息
3. # Replication
4. role:slave //从服务器
5. master\_host:192.168.4.51        //主服务器ip地址
6. master\_port:6351 //主服务器端口
7. master\_link\_status:up //连接状态开启
8. master\_last\_io\_seconds\_ago:3
9. master\_sync\_in\_progress:0
10. ……
11. ……
12. 192.168.4.52:6352>

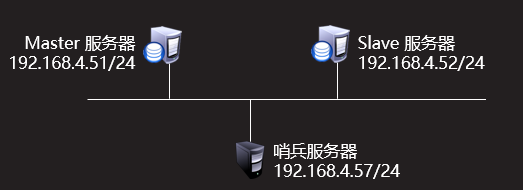
**3 案例3：哨兵服务**

**3.1 问题**

* 具体要求如下：
* 基于案例2配置
* 配置哨兵服务
* 测试配置

**3.2 方案**

角色规划如图-1所示：



图－1

**3.3 步骤**

实现此案例需要按照如下步骤进行。

步骤一：配置哨兵服务（192.168.4.57）

1） 安装源码软件redis ，无需做初始化配置。

1. [root@redis57 redis]# yum -y install gcc
2. [root@redis57 redis]# tar -zxf redis-4.0.8.tar.gz
3. [root@redis57 redis]# cd redis-4.0.8/
4. [root@redis1 redis-4.0.8]# make
5. [root@redis1 redis-4.0.8]# make install

2）编辑主配置文件

1. [root@redis57 redis]# vim /etc/sentinel.conf //创建主配置文件
2. sentinel monitor server51 192.168.4.51 6351 1 //监视主服务器
3. bind 0.0.0.0 //哨兵服务地址（表示本机所有网络接口）
4. sentinel auth-pass server51 123456 //主服务器密码
5. :wq

3）启动哨兵服务

1. [root@redis57 redis]# redis-sentinel /etc/sentinel.conf    //启动哨兵服务
2. 25371:X 28 Sep 11:16:54.993 # +sdown master redis51 192.168.4.51 6351
3. 25371:X 28 Sep 11:16:54.993 # +odown master redis51 192.168.4.51 6351 #quorum 1/1
4. 25371:X 28 Sep 11:16:54.993 # +new-epoch 3
5. 25371:X 28 Sep 11:16:54.993 # +try-failover master redis51 192.168.4.51 6351
6. 25371:X 28 Sep 11:16:54.994 # +vote-for-leader be035801d4d48eb63d8420a72796f52fc5cec047 3
7. ...
8. 25371:X 28 Sep 11:16:55.287 \* +slave slave 192.168.4.51:6351 192.168.4.51 6351 @ redis51 192.168.4.52 6351
9. 25371:X 28 Sep 11:17:25.316 # +sdown slave 192.168.4.51:6379 192.168.4.51 6379 @ redis51 192.168.4.52 6352

步骤二：测试配置

1）停止主服务器51的redis服务

1. [root@redisA ~]#     /etc/init.d/redis\_6379 stop
2. Stopping ...
3. Waiting for Redis to shutdown ...
4. Redis stopped
5. [root@redisA ~]#

2）在服务器52主机，查看复制信息

1. [root@redisB ~]# redis-cli -h 192.168.4.52 -p 6352
2. 192.168.4.52:6352> info replication
3. # Replication
4. role:master //角色是master
5. connected\_slaves:0
6. ……
7. ……

**4 案例2 ：使用RDB文件恢复数据**

**4.1 问题**

* 要求如下：
* 启用RDB
* 设置存盘间隔为120秒且10个key改变数据自动存盘
* 备份RDB文件
* 删除数据
* 使用RDB文件恢复数据

**4.2 步骤**

实现此案例需要按照如下步骤进行。

步骤一：使用RDB文件恢复数据

RDB介绍：

Redis数据库文件，全称Reids DataBase

数据持久化方式之一

在指定时间间隔内，将内存中的数据集快照写入硬盘

术语叫Snapshot快照

恢复时，将快照文件直接读到内存里

相关配置参数

文件名

dbfilename “dump.rdb” 文件名

数据从内存保存到硬盘的频率

save 900 1 900秒内且有1个key改变

save 300 10 300秒内且有10个key改变

save 60 10000 60秒内且有10000个key改变

1. [root@redisA ~]# vim /etc/redis/6379.conf
2. dbfilename dump.rdb
3. save 900 1
4. #save 300 10 //注释原有设置
5. save 120 10 //时间修改为 120秒
6. save 60 10000
7. ：wq
8. [root@redisA ~]#     /etc/init.d/redis\_6379 stop //停止服务
9. Stopping ...
10. Waiting for Redis to shutdown ...
11. Redis stopped
12. [root@redisA ~]#
13. [root@redisA ~]# rm –rf /var/lib/redis/6379/\* //清空数据库目录
14. [[root@redisA](mailto:root@redisA) ~]# /etc/init.d/redis\_6379 start //启动服务
15. Starting Redis server...
16. [[root@redisA](mailto:root@redisA) ~]#
17. [[root@redisA](mailto:root@redisA) ~]# ls /var/lib/redis/6379 //此时，查看数据库目录下没有dump.rdb文件
18. [[root@redisA](mailto:root@redisA) ~]#
19. [[root@redisA](mailto:root@redisA) ~]# redis-cli -h 192.168.4.51 -p 6351 -a 123456 //连接服务，在200秒内存储10个变量，就会自动在数据库目录下创建dump.rdb 文件
20. 192.168.4.51:6351> set v1 k1
21. OK
22. 192.168.4.51:6351> set v2 k1
23. OK
24. 192.168.4.51:6351> set v3 k1
25. OK
26. 192.168.4.51:6351> set v4 k1
27. OK
28. 192.168.4.51:6351> set v45 k1
29. OK
30. 192.168.4.51:6351> set v46 k1
31. OK
32. 192.168.4.51:6351> set v7 k1
33. OK
34. 192.168.4.51:6351> set v8 k1
35. OK
36. 192.168.4.51:6351> set v9 k1
37. OK
38. 192.168.4.51:6351> set v10 k1
39. OK
40. 192.168.4.51:6351> keys \*
41. 1) "v2"
42. 2) "v9"
43. 3) "v10"
44. 4) "v45"
45. 5) "v4"
46. 6) "v1"
47. 7) "v46"
48. 8) "v8"
49. 9) "v7"
50. 10) "v3"
51. 192.168.4.51:6351>exit
52. [[root@redisA](mailto:root@redisA) ~]# ls /var/lib/redis/6379 //此时，查看数据库目录下有dump.rdb文件
53. dump.rdb
54. [[root@redisA](mailto:root@redisA) ~]#

备份数据

1. [root@redisA ~]# cd /var/lib/redis/6379/
2. [root@redisA 6379]# ls
3. dump.rdb
4. [root@redisA 6379]# cp dump.rdb /tmp/dump.rdb    //备份dump.rdb文件
5. [root@redisA 6379]# scp /tmp/dump.rdb root@192.168.4.56:/root/ //传递备份文件给目标主机

删除数据 （56主机模拟误删除数据）

1. [root@redis56 ~]# redis-cli -h 192.168.4.56 –p 6356        //连接服务
2. 192.168.4.56:6356> flushall
3. OK
4. 192.168.4.51:6379> keys \*        //已经没有数据
5. (empty list or set)
6. 192.168.4.56:6356> exit
7. [root@redis56 ~]#

恢复数据(56主机使用备份文件恢复数据)

1. [root@redis56 ~]# /etc/init.d/redis\_6379 stop //停止服务
2. Stopping ...
3. Waiting for Redis to shutdown ...
4. Redis stopped
5. [root@redis56 ~]#
6. [root@redis56 ~]# rm -rf /var/lib/redis/6379/\* //清空数据库目录
7. [[root@redis56](mailto:root@redis56) ~]# cp /tmp/dump.rdb /var/lib/redis/6379/ //拷贝备份文件到数据库目录下
8. [[root@redis56](mailto:root@redis56) ~]# /etc/init.d/redis\_6379 start //    启动服务
9. Starting Redis server...
10. [[root@redis56](mailto:root@redis56) ~]# redis-cli -h 192.168.4.56 –p 6356 //访问服务
11. 192.168.4.56:6356> keys \* //查看数据
12. 1) "v7"
13. 2) "v46"
14. 3) "v45"
15. 4) "v8"
16. 5) "v4"
17. 6) "v2"
18. 7) "v1"
19. 8) "v3"
20. 9) "v9"
21. 10) "v10"
22. 192.168.4.56:6356>

**5 案例5 ：使用AOF文件恢复数据**

**5.1 问题**

* 具体要求如下：
* 启用AOF
* 备份AOF文件
* 删除数据
* 使用AOF文件恢复数据

**5.2 步骤**

实现此案例需要按照如下步骤进行。

步骤一：使用AOF文件恢复数据

1）修改配置文件

1. [root@redisA ~]# redis-cli -h 192.168.4.51 –p 6351 -a 123456 //连接服务
2. 192.168.4.51:6351>config set appendonly yes     //启用aof，默认no
3. 192.168.4.51:6351> config rewrite //写进配置文件
4. 192.168.4.51:6351> save
5. 192.168.4.51:6351> exit
6. [root@redisA ~]# ls /var/lib/redis/6379/ //会出现appendonly.aof文件
7. appendonly.aof dump.rdb
8. [root@redisA ~ ]#

2）备份AOF文件

1. [root@redisA ~]# cd /var/lib/redis/6379/
2. [root@redisA 6379]# cp appendonly.aof /tmp/appendonly.aof
3. [root@redisA 6379]# scp /tmp/appendonly.aof root@192.168.4.57:/root/ //传递备份文件给目标主机

3）删除数据（在57主机 默认数据误删除）

1. [root@redis57 ~]# redis-cli -h 192.168.4.57 -p 6357 //连接服务
2. 192.168.4.57:6357> flushall //清除数据
3. OK
4. 192.168.4.57:6357> keys \* //查看数据
5. (empty list or set)
6. 192.168.4.57:6357> exit
7. [root@redis57 ~ ]#

4) 使用AOF文件恢复数据

1. [root@redis57 ~]# vim +673 /etc/redis/6379.conf
2. appendonly yes //启用AOF
3. :wq
4. [root@redis57 ~]#
5. [root@redis57 ~]# /etc/init.d/redis\_6379 stop //停止服务
6. Stopping ...
7. Waiting for Redis to shutdown ...
8. Redis stopped
9. [root@redis57 ~]#
10. [root@redis57 ~]# /etc/init.d/redis\_6379 start //启动服务
11. Starting Redis server...
12. [root@redis57 ~]#
13. [root@redis57 ~]# rm -rf /var/lib/redis/6379/\* //删除没有数据的文件
14. [[root@redis57](mailto:root@redis57) ~]# cp /root/appendolny.aof /var/lib/redis/6379/ //拷贝文件
15. [[root@redis57](mailto:root@redis57) ~]# /etc/init.d/redis\_6379 start //启动服务
16. Starting Redis server...
17. [[root@redis57](mailto:root@redis57) ~]# redis-cli -h 192.168.4.57 -p 6357 //连接服务
18. 192.168.4.57:6357> keys \* //查看数据
19. 1) "v9"
20. 2) "v5"
21. 3) "v8"
22. 4) "v2"
23. 5) "v1"
24. 6) "v4"
25. 7) "v10"
26. 8) "v6"
27. 9) "v7"
28. 10) "v3"
29. 192.168.4.57:6357>

**6 案例6：string 字符串**

**6.1 问题**

* 练习命令的使用，具体命令如下：
* set getrange strlen append setbit bitcount
* decr decrby incr incrby incrbyfloat

**6.2 步骤**

实现此案例需要按照如下步骤进行。

步骤一：string 字符串

设置key及值，过期时间可以使用秒或毫秒为单位

setrange key offset value

1. 192.168.4.56:6356> set x 9 ex 10 //单位秒
2. OK
3. 192.168.4.56:6356>
4. 192.168.4.56:6356> set y 29 px 10 //单位毫秒
5. OK
6. 192.168.4.56:6356>
7. 192.168.4.56:6356> set y 39 NX //不存在赋值
8. OK
9. 192.168.4.56:6356> get y //变量值没变
10. "39"
11. 192.168.4.56:6356>
12. 192.168.4.56:6356> set y 49 xx //变量存在赋值
13. OK
14. 192.168.4.56:6356> get y //变量变了
15. "49"
16. 192.168.4.56:6356>

2) 从偏移量开始复写key的特定位的值

1. 192.168.4.51:6351> set first "hello world"
2. OK
3. 192.168.4.51:6351> setrange first 6 "Redis"     //改写为hello Redis
4. (integer) 11
5. 192.168.4.51:6351> get first
6. "hello Redis"

3) strlen key，统计字串长度

1. 192.168.4.51:6379> strlen first
2. (integer) 11

4) append key value 存在则追加，不存在则创建key及value，返回key长度

1. 192.168.4.51:6379> append myname jacob
2. (integer) 5

5) setbit key offset value 对key所存储字串，设置或清除特定偏移量上的位(bit)，value值可以为1或0，offset为0~2^32之间，key不存在，则创建新key

1. 192.168.4.51:6379> setbit bit 0 1         //设置bit第0位为1
2. (integer) 0
3. 192.168.4.51:6379> setbit bit 1 0         //设置bit第1位为0
4. (integer) 0

6) bitcount key 统计字串中被设置为1的比特位数量

1. 192.168.4.51:6379> setbit bits 0 1        //0001
2. (integer) 0
3. 192.168.4.51:6379> setbit bits 3 1        //1001
4. (integer) 0
5. 192.168.4.51:6379> bitcount bits            //结果为2
6. (integer) 2

记录网站用户上线频率，如用户A上线了多少天等类似的数据，如用户在某天上线，则使用setbit，以用户名为key，将网站上线日为offset，并在该offset上设置1，最后计算用户总上线次数时，使用bitcount用户名即可，这样即使网站运行10年，每个用户仅占用10\*365比特位即456字节

1. 192.168.4.51:6379> setbit peter 100 1        //网站上线100天用户登录了一次
2. (integer) 0
3. 192.168.4.51:6379> setbit peter 105 1        //网站上线105天用户登录了一次
4. (integer) 0
5. 192.168.4.51:6379> bitcount peter
6. (integer) 2

7) decr key 将key中的值减1，key不存在则先初始化为0，再减1

1. 192.168.4.51:6379> set z 10
2. OK
3. 192.168.4.51:6379> decr z
4. (integer) 9
5. 192.168.4.51:6379> decr z
6. (integer) 8
7. 192.168.4.51:6379> decr bb
8. (integer) -1
9. 192.168.4.51:6379> decr bb
10. (integer) -2

8) decrby key decrement 将key中的值，减去decrement

1. 192.168.4.51:6379> set count 100
2. OK
3. 192.168.4.51:6379> DECRBY cc 20    //定义每次减少20（步长）
4. (integer) -20
5. 192.168.4.51:6379> DECRBY cc 20
6. (integer) -40

9) getrange key start end 返回字串值中的子字串，截取范围为start和end，负数偏移量表示从末尾开始计数，-1表示最后一个字符，-2表示倒数第二个字符

1. 192.168.4.51:6379> set x 123456789
2. OK
3. 192.168.4.51:6379> getrange x -5 -1
4. "56789"
5. 192.168.4.51:6379> getrange x 0 4
6. "12345"

10) incr key 将key的值加1，如果key不存在，则初始为0后再加1，主要应用为计数器

1. 192.168.4.51:6379> set page 20
2. OK
3. 192.168.4.51:6379> incr page
4. (integer) 21

11) incrby key increment 将key的值增加increment

1. 192.168.4.51:6379> set x 10
2. OK
3. 192.168.4.51:6379> incr x
4. (integer) 11
5. 192.168.4.51:6379> incr x
6. (integer) 12

12) incrbyfloat key increment 为key中所储存的值加上浮点数增量 increment

1. 192.168.4.51:6379> set num 16.1
2. OK
3. 192.168.4.51:6379> incrbyfloat num 1.1
4. "17.2"

**7 案例7：list 列表**

**7.1 问题**

* 练习命令使用，具体如下：
* lpush llen lrange lpop
* lindex lset rpush rpop

**7.2 步骤**

实现此案例需要按照如下步骤进行。

步骤一：list 列表

1) lpush key value [value…] 将一个或多个值value插入到列表key的表头，Key不存在，则创建key

1. 192.168.4.51:6379> lpush list a b c        //list值依次为c b a
2. (integer) 3

2) lrange key start stop 从开始位置读取key的值到stop结束

1. 192.168.4.51:6379> lrange list 0 2        //从0位开始，读到2位为止
2. 1) "c"
3. 2) "b"
4. 3) "a"
5. 192.168.4.51:6379> lrange list 0 -1    //从开始读到结束为止
6. 1) "c"
7. 2) "b"
8. 3) "a"
9. 192.168.4.51:6379> lrange list 0 -2        //从开始读到倒数第2位值
10. 1) "c"
11. 2) "b"

3) lpop key 移除并返回列表头元素数据，key不存在则返回nil

1. 192.168.4.51:6379> lpop list        //删除表头元素，可以多次执行
2. "c"
3. 192.168.4.51:6379> LPOP list
4. "b"

4) llen key 返回列表key的长度

1. 192.168.4.51:6379> llen list
2. (integer) 1

5) lindex key index 返回列表中第index个值

1. 192.168.4.51:6379> lindex list 1
2. "c"

6) lset key index value 将key中index位置的值修改为value

1. 192.168.4.51:6379> lpush list a b c d
2. (integer) 5
3. 192.168.4.51:6379> lset list 3 test        //将list中第3个值修改为test
4. OK

7) rpush key value [value…] 将value插入到key的末尾

1. 192.168.4.51:6379> rpush list3 a b c    //list3值为a b c
2. (integer) 3
3. 192.168.4.51:6379> rpush list3 d    //末尾插入d
4. (integer) 4

8) rpop key 删除并返回key末尾的值

1. 192.168.4.51:6379> RPOP list3
2. "d"

**8 案例8：hash表**

**8.1 问题**

* 练习命令使用，具体如下：
* hset hmset hgetall hkeys hvals
* hget hmget hdel

**8.2 步骤**

实现此案例需要按照如下步骤进行。

步骤一：hash表

1）hset key field value 将hash表中field值设置为value

1. 192.168.4.51:6379> hset site google 'www.g.cn'
2. (integer) 1
3. 192.168.4.51:6379> hset site baidu 'www.baidu.com'
4. (integer) 1

2) hget key filed 获取hash表中field的值

1. 192.168.4.51:6379> hget site google
2. "www.g.cn"

3) hmset key field value [field value…] 同时给hash表中的多个field赋值

1. 192.168.4.51:6379> hmset site google www.g.cn baidu www.baidu.com
2. OK

4) hmget key field [field…] 返回hash表中多个field的值

1. 192.168.4.51:6379> hmget site google baidu
2. 1) "www.g.cn"
3. 2) "www.baidu.com"

5) hkeys key 返回hash表中所有field名称

1. 192.168.4.51:6379> hmset site google www.g.cn baidu www.baidu.com
2. OK
3. 192.168.4.51:6379> hkeys site
4. 1) "google"
5. 2) "baidu"

6) hgetall key 返回hash表中所有key名和对应的值列表

1. 192.168.4.51:6379> hgetall site
2. 1) "google"
3. 2) "www.g.cn"
4. 3) "baidu"
5. 4) "www.baidu.com"

7) hvals key 返回hash表中所有key的值

1. 192.168.4.51:6379> hvals site
2. 1) "www.g.cn"
3. 2) "www.baidu.com"

8) hdel key field [field…] 删除hash表中多个field的值，不存在则忽略

1. 192.168.4.51:6379> hdel site google baidu
2. (integer) 2