**总体来看redis大致了解特性和内容之后，与mysql一样需要不断练习**

-----------------------------Redis安装和使用-----------------------------------

1. C语言编写、支持网络、可基于内存亦可持久化的日志型、Key-Value数据库，是高性能键值对数据库，支持的键值对有：
   1. 字符串类型
   2. 列表类型
   3. 有序集合类型
   4. 散列类型
   5. 集合类型

应用场景：缓存（数据查询、新闻、商品内容）、任务队列（秒杀、抢购）、网站访问统计、数据过期处理、分布式集群架构中session的分离

1. ubuntu下安装redis：
   1. 下载redis的压缩文件，解压缩并在解压后的目录（类似于redis-5.0.4）下编译：make
   2. 编译完成后，采用make PREFEX=/usr/.... install 进行安装
   3. 将编译目录（redis-5.0.4）下的redis.conf拷贝到安装目录下

**启动服务**

* 1. 进入bin，运行redis（./redis-server）这种方式采用的是前端运行方式。
  2. 后端运行：在redis.conf文件中找到：

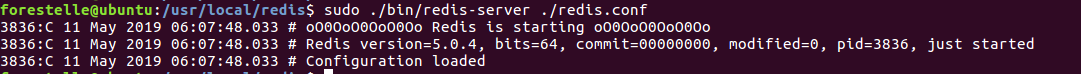
# By default Redis does not run as a daemon. Use 'yes' if you need it.

# Note that Redis will write a pid file in /var/run/redis.pid when daemonized.

daemonize no

将其更改成yes，然后采用 “./redis-server /path/to/redis.conf”将redis后端运行

如图：



过程中出现如下报错：Failed opening the RDB file dump.rdb (in server root dir /usr/local/redis/bin) for saving: Permission denied

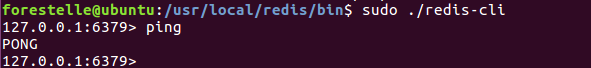
f）停止redis：调用客户端指令shutdown：

使用”./server-cli shutdown”即可。（注意可能因为没权限要加sudo）

可以采用sudo chown 770 [fileName]更改所有者，或者使用sudo的方式运行redis

1. 执行语句：

调用客户端



1. 使用jedis

jedis是redis的客户端，使用方式也很容易，官网上拥有jedis的maven依赖，注入以后就很好使用了。小demo查看仓库内的代码。

1. jedis连接服务器失败的问题（Jedis jedis = new Jedis("192.168.145.130",6379);）：
   1. 由于linux防火墙未打开6379端口，所以首先需要打开防火墙端口。
   2. 打开防火墙之后，如果依旧无法ping通6379端口（telnet可以ping指定端口）：(参考：<https://www.cnblogs.com/bestmystery/p/6371229.html>)

打开cmd 然后使用 “telnet ip 端口”的格式 来ping 配置的redis（要保证redis已启动），发现无法ping通。

这是因为在redis.conf中有个配置 **bind 127.0.0.1** 这个是默认只有本机访问，把这个注释掉就好了

注释以后查看redis进程就变为下面这样：

[root@localhost redis]# ps -ef | grep redis

root 5655 1 0 11:40 ? 00:00:23 ./redis-server \*:6379

root 21184 18040 0 17:33 pts/1 00:00:00 grep --color=auto redis

这个\*号就表示允许其它用户访问了。然后在用打开本机的 cmd使用 telnet ip 端口 就能ping通了。

* 1. 如果出现报错（依旧无法ping通）DENIED Redis is running in protected mode because protected mode is enabled：

这是说Redis服务处于保护模式，我们需要修改配置文件redis.conf。将NETWORK下的protected-mode yes修改为protected-mode no，然后重启服务

可以使用redis-cli [--raw]方式运行其客户端，在raw下，可以显示中文，get的数据没有引号，结果前面不提供数据类型......

-----------------------------Redis数据结构-------------------------------------

**以下内容皆可参考：**[**https://www.runoob.com/redis/redis-tutorial.html**](https://www.runoob.com/redis/redis-tutorial.html) **下面内容太冗余**

1. 类型
   1. 字符串String
   2. 字符串有序集合sorted set
   3. 字符串列表list
   4. 哈希hash
   5. 字符串集合 set
2. key定义的注意点
   1. 不要过长（最好不要超过1024个字节）、不要过短：
   2. 要有统一的命名的规范
3. **String类型**基本操作：（ps：操作过程中出现中文时，cli表示比较无能为力，会出现各种问题（即使使用—raw方式打开cli，不过可能是命令行的编码问题），但是jedis对象可以很好地处理中文）
   1. 赋值：set key value(写不写引号都行)
   2. 取值：get key
   3. 先取值再赋值：getset
   4. 删除：del key（删除成功会返回1，失败则返回0）
   5. 数值增减：incr key（将指定key增加1，如果key不存在，在初始化成0并加1） decr key
   6. 增减指定值：incrby key value，decrby key value
   7. 拼接字符串：append key value（返回字符串长度）
4. **Hash类型**基本操作：在基本操作的前面加一个h（适合存储对象信息：用户名、密码、年龄）参考 <https://www.runoob.com/redis/redis-hashes.html>

String key 和 String value 的 map 容器，是一个string类型的field和value的映射表

* 1. 赋值：
     1. hset hash key value(hset myhash username estelle) 每个hash可以存储约40亿的键值对。
     2. hmset hash key1 value1 [key2 value2 ...]一次存储多个键值对到一个hash中
  2. 取值：
     1. hget hash key(hget myhash username)
     2. hmget hash key1 [key2 key3 ...]
     3. hgetall hash
  3. 删除
     1. hdel hash key1 [key2 key3 ...]（可以多个一起删除）
     2. del hash （删除整个hash）
  4. 增加：hincrby hash key number
  5. 判断存在：hexists hash key（1：存在，0：不存在）
  6. 获得所有的key/value：hkeys hash hvalues hash

1. **List类型**基本操作
   1. Redis列表是简单的字符串**链表**，按照插入顺序排序。你可以添加一个元素到列表的头部（左边）或者尾部（右边）
   2. 效率：如果插入删除是在链表头尾进行，操作会在常量时间里完成。如果插入删除是链表中间，那效率会低。
   3. ArrayList使用数组方式存储：根据索引查询速度快、新增和删除元素涉及到位移。LinkedList使用双向链表方式：新增和删除速度快
   4. LPUSH key value1 [value2] 将一个或多个值插入到列表头部。如果 key 不存在，一个空列表会被创建并执行 LPUSH 操作。 当 key 存在但不是列表类型时，返回一个错误。注意，value2比较靠近头部。
   5. RPUSH key value1 [value2] 将一个或多个值插入到列表的尾部(最右边)。如果列表不存在，一个空列表会被创建并执行 RPUSH 操作。 当列表存在但不是列表类型时，返回一个错误。注意，value2更靠近尾部。
   6. RPUSHX key value 用于将一个值插入到已存在的列表尾部(最右边)。如果列表不存在，操作无效。
   7. LRANGE KEY START END 返回列表中指定区间内的元素，区间以偏移量 START 和 END 指定。 其中 0 表示列表的第一个元素， 1 表示列表的第二个元素，以此类推。 你也可以使用负数下标，以 -1 表示列表的最后一个元素， -2 表示列表的倒数第二个元素，以此类推。
   8. LPOP key 用于移除并返回列表的第一个元素。当列表不存在时，返回 nil 。
   9. RPOP key 用于移除并返回列表的最后一个元素。当列表不存在时，返回 nil 。
   10. LLEN key 命令用于返回列表的长度。 如果列表 key 不存在，则 key 被解释为一个空列表，返回 0 。 如果 key 不是列表类型，返回一个错误。
   11. LREM key count value 根据参数count的值，移除列表中与参数value相等的元素。
       * count > 0 : 从表头开始向表尾搜索，移除与value相等的元素，数量为 count。
       * count < 0 : 从表尾开始向表头搜索，移除与value相等的元素，数量为count的绝对值。
       * count = 0 : 移除表中所有与 VALUE 相等的值。
   12. LSET key index value 通过索引来设置元素的值（直接**替换**索引位置上的值）。当索引参数超出范围，或对一个空列表进行 LSET 时，返回一个错误。
   13. LINSERT key BEFORE|AFTER pivot value 命令用于在列表的元素前或者后插入元素。当指定元素不存在于列表中时，不执行任何操作。pivot是元素的值。
   14. RPOPLPUSH source destination 用于移除列表source的最后一个元素，并将该元素添加到另一个列表destination并返回。（使用场景：redis链表经常用于消息队列的服务，完成多个程序之间的消息交互。假设生产者插入数据到链表中，消费者从链表中取出数据，消费者在取出数据时崩溃，而为进行操作，那数据便丢失了。使用rpoplpush将数据添加到备份队列中，直到消费者完成逻辑处理再删除备份队列信息）
2. **set类型**基本操作：
   1. Set 是 String 类型的无序集合。集合成员是唯一的，这就意味着集合中不能出现重复的数据。Redis 中集合是通过哈希表实现的，所以添加，删除，查找的复杂度都是 O(1)。set具有一个重要的功能就是在服务器端完成多个set之间的聚合计算操作：union，difference...由于实在服务器端完成的，可以节省大量开销。
   2. SADD key member1 [member2] 将一个或多个成员元素加入到集合中，已经存在于集合的成员元素将被忽略。
   3. SREM key member1 [member2] 移除集合中的一个或多个成员元素，不存在的成员元素会被忽略。
   4. SMEMBERS key 返回集合中的所有的成员。 不存在的集合 key 被视为空集合。
   5. SISMEMBER key member 命令判断成员元素是否是集合的成员。如果成员元素是集合的成员，返回 1 。 如果成员元素不是集合的成员，或 key 不存在，返回 0 。

下面的集合间操作不改变集合

* 1. SDIFF key1 [key2] 返回给定集合之间的差集。不存在的集合 key 将视为空集。差集的结果来自前面的key1 ,而不是后面的key2
  2. SINTER key1 [key2] 返回给定所有给定集合的交集，不存在的集合key被视为空集。
  3. SUNION key1 [key2] 返回给定集合的并集。不存在的集合 key 被视为空集。
  4. SCARD key 返回集合中元素的数量。
  5. SRANDMEMBER key [count] 返回集合中的一个随机元素。从 Redis 2.6 版本开始， Srandmember 命令接受可选的 count 参数：
     1. 如果 count 为正数，且小于集合基数，那么命令返回一个包含 count 个元素的数组，数组中的元素各不相同。如果 count 大于等于集合基数，那么返回整个集合。
     2. 如果 count 为负数，那么命令返回一个数组，数组中的元素可能会重复出现多次，而数组的长度为 count 的绝对值。

该操作和 SPOP 相似，但 SPOP 将随机元素从集合中移除并返回，而 Srandmember 则仅仅返回随机元素，而不对集合进行任何改动。

* 1. SPOP key 用于移除集合中的一个或多个随机元素，移除后会返回移除的元素。
  2. SDIFFSTORE destination key1 [key2] 将给定集合之间的差集存储在指定的集合中。如果指定的集合 key 已存在，则会被覆盖。SINTERSTORE destination key1 [key2] 和SUNIONSTORE destination key1 [key2]类似

set使用场景：跟踪一些具有唯一性的数据（访问某网站的ip，set会保证唯一性。且set数据的聚合操作在服务器上完成，就比较高效，可以用来维护关联关系。如，所有购买某一商品的客户id可以放在一个set中，购买另一种商品的id放在另一个set中，想获取同时购买两个商品的客户id只需要做交集）

1. **sorted set有序集合**基本操作：

**原文中说，集合是通过哈希表实现的，所以添加，删除，查找的复杂度都是O(1)其实不太准确。其实在redis sorted sets里面当items内容大于64的时候同时使用了hash和skiplist两种设计实现。这也会为了排序和查找性能做的优化。所以如上可知：添加和删除都需要修改skiplist，所以复杂度为O(log(n))。但是如果仅仅是查找元素的话可以直接使用hash，其复杂度为O(1)其他的range操作复杂度一般为O(log(n))当然如果是小于64的时候，因为是采用了ziplist的设计，其时间复杂度为O(n)。**

* 1. 有序集合和集合一样也是string类型元素的集合,且不允许重复的成员。不同的是每个元素都会关联一个double类型的分数。redis正是通过分数来为集合中的成员进行从小到大的排序。有序集合的成员是唯一的,但分数(score)却可以重复。集合是通过哈希表实现的，所以添加，删除，查找的复杂度都是O(1)。由于具有顺序，所以，即便对集合中间的元素进行操作也是高效的。
  2. 应用场景：游戏排名，微博热搜
  3. ZADD key score1 member1 [score2 member2] 用于将一个或多个成员元素及其分数值加入到有序集当中。如果某个成员已经是有序集的成员，那么更新这个成员的分数值，并通过重新插入这个成员元素，来保证该成员在正确的位置上。
  4. ZCARD key获取有序集合的成员数
  5. ZSCORE key member返回有序集中，成员的分数值。 如果成员元素不是有序集 key 的成员，或 key 不存在，返回 nil 。
  6. ZREM key member [member ...] 移除有序集合中的一个或多个成员
  7. ZRANGE key start stop [WITHSCORES] 返回有序集中，指定区间内的成员。下标参数start和stop都以 0 为底，也就是说，以0表示有序集第一个成员，以 1 表示有序集第二个成员，以此类推。你也可以使用负数下标，以 -1表示最后一个成员， -2表示倒数第二个成员，以此类推。如果带有withscores那会将分数一起返回。
  8. ZREVRANGE key start stop [WITHSCORES] 返回有序集中指定区间内的成员，通过索引，分数从高到底。
  9. ZREMRANGEBYRANK key start stop和ZREMRANGEBYSCORE key min max分别是 移除有序集合中给定的排名区间的所有成员和移除有序集合中给定的分数区间的所有成员。

使用场景：大型在线游戏积分排行榜（玩家分数发生变化时使用zadd进行更新，在通过zrange获取积分）、用于构建索引数据

1. Keys的通用操作
   1. 获取所有key：keys \* 、 keys wi? 、 key w\*
   2. 删除：del key1 [key2...] 存在： exists key 重命名：rename oldkey newkey
   3. 设置过期时间（过期删除，单位：秒）：expire 查看剩余过期时间：ttl key
   4. 查看key的类型：type key
2. **Redis特性**：
   1. 多数据库：
      1. 一个redis实例可以包含多个数据库，客户端可以指定连接，指定实例的指定数据库的。一个redis实例最多提供16个数据库，默认连接第0号数据库
      2. 数据库间数据交换：move key number 将key移动到第number个数据库
   2. Redis事务：事务可以一次执行多个命令， 并且带有以下两个重要的保证：批量操作在发送 EXEC 命令前被放入队列缓存。**收到 EXEC 命令后进入事务执行，事务中任意命令执行失败，其余的命令依然被执行（和关系型数据库不同）。在事务执行过程，其他客户端提交的命令请求不会插入到事务执行命令序列中。**
      1. multi 开启事务，开启后输入各种待执行指令
      2. exec 执行事务
      3. discard 回滚

事务在jedis中：

1、开启采用transaction对象并调用multi方法。2、执行事务之前不能使用jedis类中的方法。（Cannot use Jedis when in Multi. Please use Transaction or reset jedis state）3、使用exec执行事务。4、回滚：discard。（ps：教程中写到事务可以理解为一个打包的批量执行脚本，但批量指令并非原子化的操作，中间某条指令的失败不会导致前面已做指令的回滚，也不会造成后续的指令不做。在测试中已被证伪(error) EXECABORT Transaction discarded because of previous errors.）

1. Redis持久化：

Redis的高性能是由于其所有数据都存储在内存当中，为了保证redis重启以后数据不丢失，需要将数据持久化。

参考<https://baijiahao.baidu.com/s?id=1611955931705092609&wfr=spider&for=pc>

**RDB**：优点（fork子进程完成，可以避免服务器进程进行大量的I/O操作）缺点：快照定期产生，一定会出现数据丢失

**AOF：**Redis重启时，会把AOF文件中记录的所有写操作顺序执行一遍，确保数据恢复到最新。有三种同步策略：appendfsync no（不同步）、appendfsync always（每修改同步）appendfsync everysec（每秒同步）