# Redis入门：

Redis建议使用linux环境

## Linux安装

1. 下载安装包

l

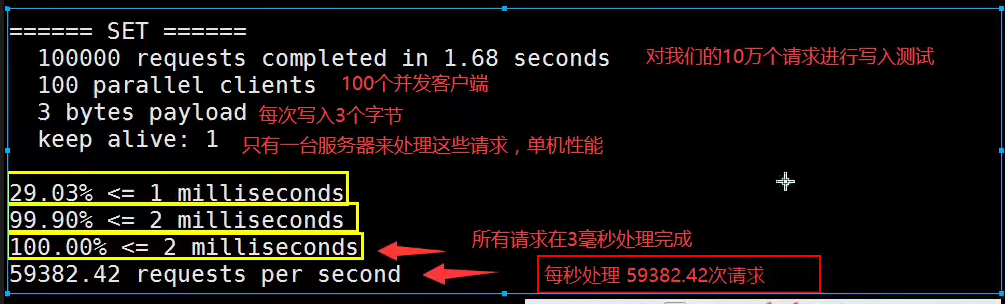
### 测试性能

官方自带的测试工具benchmark

测试：100个并发连接 10000个请求

redis-benchmark –h localhost –p 6379 –c 100 –n 100000

分析测试结果：

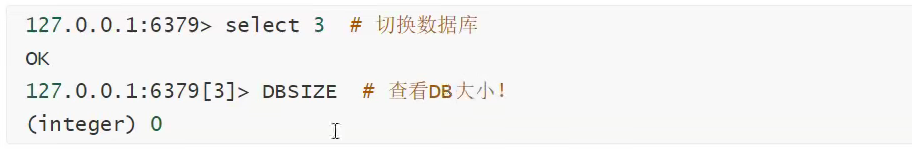


## 基础的知识：

### Redis默认有16个数据库。

默认使用的是0号数据库

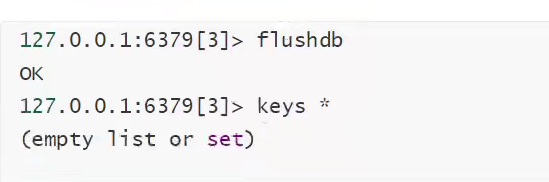
可以使用select进行切换数据库！



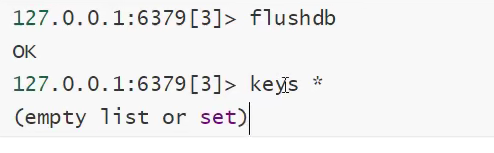
### 查看数据库中所有的key



### 清除当前数据库 flushdb



### 清除所有数据库的内容 flushall



### Redis是单线程的

Redis是很快的，官方表示Redis是基于内存操作，CPU不是REdis性能瓶颈，Redis的瓶颈是根据机器的内存和网络带宽，既然可以使用单线程，就不需要使用多线程！

Redis是C语言写的

### Redis为什么单线程还这么快

1. 误区：高性能的服务器一定是多线程的 ×
2. 误区2：多线程(CPU上下文切换需要资源消耗)一定比单线程效率高 ×

CPU（但是上下文切换是一个耗时的工作）>内存>硬盘

核心：redis是将数据放入到内存中的，所以使用单线程去操作效率就是最高的，多线程（CPU上下文切换会耗时！！），对于内存系统来说，如果没有上下文切换效率就是最高的！（暂时可以这样理解）

# 五大数据类型

Redis 是一个开源（BSD许可）的，内存中的数据结构存储系统，它可以用作数据库、缓存和消息中间件。 它支持多种类型的数据结构，如 [字符串（strings）](http://www.redis.cn/topics/data-types-intro.html#strings)， [散列（hashes）](http://www.redis.cn/topics/data-types-intro.html#hashes)， [列表（lists）](http://www.redis.cn/topics/data-types-intro.html#lists)， [集合（sets）](http://www.redis.cn/topics/data-types-intro.html#sets)， [有序集合（sorted sets）](http://www.redis.cn/topics/data-types-intro.html#sorted-sets) 与范围查询， [bitmaps](http://www.redis.cn/topics/data-types-intro.html#bitmaps)， [hyperloglogs](http://www.redis.cn/topics/data-types-intro.html#hyperloglogs) 和 [地理空间（geospatial）](http://www.redis.cn/commands/geoadd.html) 索引半径查询。 Redis 内置了 [复制（replication）](http://www.redis.cn/topics/replication.html)，[LUA脚本（Lua scripting）](http://www.redis.cn/commands/eval.html)， [LRU驱动事件（LRU eviction）](http://www.redis.cn/topics/lru-cache.html)，[事务（transactions）](http://www.redis.cn/topics/transactions.html) 和不同级别的 [磁盘持久化（persistence）](http://www.redis.cn/topics/persistence.html)， 并通过 [Redis哨兵（Sentinel）](http://www.redis.cn/topics/sentinel.html)和自动 [分区（Cluster）](http://www.redis.cn/topics/cluster-tutorial.html)提供高可用性（high availability）。

## Redis-Key

基本命令：

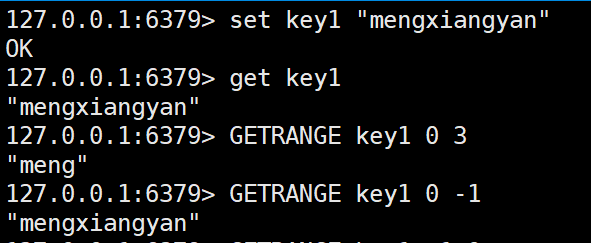
* Keys \*：查看所有的key
* Set key value：设置key的value值
* Get key：获取key的value
* Exists key：判断key是否存在
* Move key 1：从数据库中移除key
* Expire key 数字：设置key的生存时间单位秒
* Ttl key：查看key的剩余生存时间
* Type key：查看key的类型

官网可以查看帮组文档

## String

大部分使用redis只会使用string类型

* Append key value：给key后面增加value：类似于String中的append方法
* Strlen key：key的value的字符串长度
* Incr key：增加key的值+1
* Decr key：减少key的值-1
* Incrby key n：设置key的增加量为数字n
* Decrby key n：设置key的减少量为数字n
* Getrange key n m：得到key 的第n到第m的字符串（闭区间）
* Getrange key 0 -1：得到key的完整字符串



* Setrange key n “str”：替换指定位置开始的字符串为str；
* Setex key num value（set with expire）：设置key的过期时间，单位为秒
* Setnx key value（set if not exist）：如果key不存在则设置...，存在则不变动（分布式锁经常使用！）
* Mset key1 value1 key2 value2 key3 value3：一次性设置多个key和value
* Mget key1 key2 key3：一次性获得多个key
* Msetnx：如果不存在则…，msetnx是一个原子性操作要么一起成功，要么一起失败
* 对象操作：set user:1 {name:meng,age:3}：设置一个user:1对象 值为json字符来保存一个对象

Mset user:1:name meng user:1:age 12：设置一个user:1(对象id为1)对象（属性：name:meng age:12）

* Getset key value：如果不存在key，返回null，并且创建，赋值value

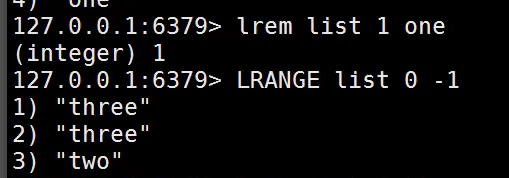
如果key存在，返回当前值，并赋值新的值value

## List

Redis里面，可以把list完成栈、队列、阻塞队列

所有的list命令都是用“l”开头的

* Lpush key value：往key中添加值
* lRange key n m：获取list集合中的【n,m】区间中的值
* Rpush key value：从尾部进行插入值
* LPOP：移除list的第一个元素
* RPOP：移除list 的最后一个元素
* LINDEX key num：获取键为key的list的第num个值
* Llen：返回列表的长度
* Lrem key num value：移除指定的值，num代表移除value值的数量



* Ltrim list n m：截取保留[n,m]区间中的长度
* Rpoplpush list1 list2：移除list1列表的最后一个元素，并将该元素添加到list2中
* Lset list index value：前提是list存在，将下标index的值修改为新的value
* Linsert list before/after value1 value2：想当前list中value1值的前/后插入新的值value2

小结：

Redis中list数据类型实际是一个链表，before node after，left、right都可以插入值

## Set

* Sadd set value：创建集合set中添加值
* Smembers set：查看set中的值
* Sismember set value：判断set中是否含有value
* Scard set：获取set中元素的个数
* Srem set value：移除set中的元素value
* Srandmember set num：获取set集合中随机num个数量元素值
* Spop set：随机删除一些set几个中的元素
* Smove set1 set2 value：将set1集合中的元素移动到set2集合中
* Sdiff set1 set2：查看两个集合中不同的元素
* Sinter set1 set2：查看两个集合中相同的元素
* Sunion set1 set2：查看两集合元素的并集

## Hash

Map集合，key – map（存放的是map集合）！

* Hset hash field value：创建hash，值为field-value键值对，相当于map
* Hget hash field：获取hash中field对应的值
* Hmset hash field1 value1 field2 value2,,,：添加多组键值对
* Hmget hash field1 field2：获取多组值
* Hgetall hash：获取hash中的所有键值对
* Hdel hash field：删除hash中的一个指定键值对
* Hlen hash：获取hash表的字段数量
* Hexists hash filed：判断hash中的filed指定键值对是否存在
* Hkeys ：只获取所有的键field
* Hvals：只获取所有的值
* Hincrby hash field num：给hash中指定值加num（可以正负）
* Hsetnx hash feld value：如果field存在，不改变，不存在，创建

Hash更是个对象的存储，String更适合字符串的存储

## ZSet（有序集合）

在set基础上增加了一个值，set k1 v1 : zset k1 score1 v1(根据score用于排序)

* Zadd set score value：创建set
* Zadd set score1 v1 score2 v2...：添加多个值
* Zrange set n m （withscores limit offset count：带有score以及限制范围大小）：获取区间内的set中的元素
* Zrangebyscore set –inf +inf（负无穷到正无穷：从小到大）：对set中元素排序，输出
* Zreverange set 0 -1：从大到小排序
* Zrem set value：移除set中的value
* Zcard set：获取有序set集合中的元素个数
* Zcount set n m：获取区间n到m中的元素个数，闭区间

# 事务

Redis事务本质：一组命令的集合！一个事务中的所有命令都会被序列化，在事务执行的过程中，会按照顺序执行

一次性、顺序性、排他性！执行一些列的命令

Redis事务没有隔离级别的概念

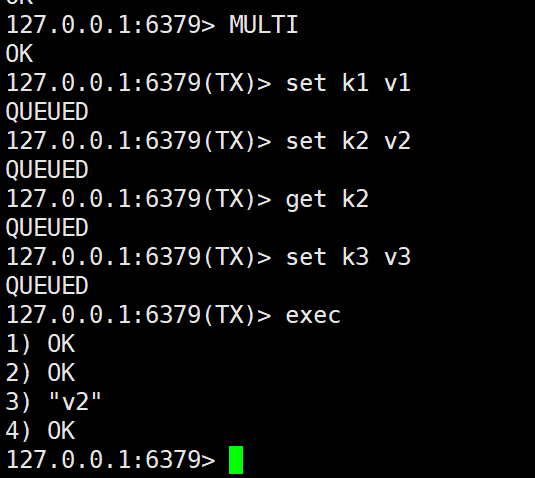
所有命令在事务中，并没有直接的被执行！只有发起执行命令的时候才会执行

Redis单条命令式保存原子性的，但是事务不保证原子性

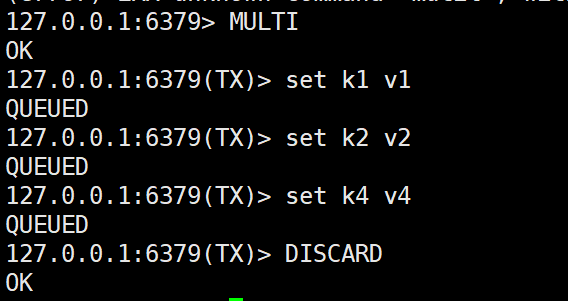
Redis的事务命令

* 开启事务（multi）
* 命令入队（…）
* 执行事务（exec）

## 正常执行事务



## 放弃事务



Discard命令：放弃事务

## 事务错误

编译型异常（命令有错），事务中的所有的命令都不会被执行

运行型异常（1/0），如果事务队列存在语法性，那么执行命令的时候，其他命令是可以正常执行的，错误命令

## 监控（锁）！

悲观锁、乐观锁

## Redis测监视测试

127.0.0.1:6379> set money 100

OK

127.0.0.1:6379> set out 0

OK

127.0.0.1:6379> watch money 监视money对象 watch命令相当于乐观锁 unwatch解锁

OK

127.0.0.1:6379> MULTI 事务正常结束

OK

127.0.0.1:6379(TX)> DECRBY money 20

QUEUED

127.0.0.1:6379(TX)> INCRBY out 20

QUEUED

127.0.0.1:6379(TX)> EXEC

1) (integer) 80

2) (integer) 20

127.0.0.1:6379>

# Jredis

编码测试：

* 连接数据库
* 操作命令
* 断开连接！

# Springboot整合

在springboot 2.x之后，原来使用的jedis被替换成了lettuce

Jedis：采用的直连，多个线程操作是不安全的，想要避免，需要使用jedispool连接池，更像BIO模式

Lettuce：采用netty，实例可以在多个线程中进行共享，不存在线程不安全的情况，可以减少线程数据，更像NIO模式