zookeeper的服务端数据结构类似文件目录，但是不同点可以在任意节点上存储数据

zookeeper的client端可以向服务端进行增删改的操作，同时可以监听服务端的目录结构

重点：zookeeper分为服务端和客户端，服务器只有一个服务端或一组信息同步的服务端集群组

客户端可以有多个，每一个客户端可以对服务端进行监控

数据一致性：

强一致性，必须一致，修改未完成前其他请求会阻塞

弱一致性，不需要完全一致

最终一致性，接受延迟，最终数据会一致

CAP：

C:一致性

A:可用性

P:分区容错性

zk尽量保证了cp一致性 ZAB协议

两阶段提交，发起提议--等待ack--执行提议(2PC)

选举机制：

能力值pk，zxid事务id，myid大小

当从节点回复后，会向主节点从新请求数据并同步。

主节点挂了，未同步到从节点的数据会作为无效数据，回滚

zk的数据类型：

持久化节点

持久化序号目录节点

临时节点

临时序号目录节点

zookeeper客户端建立连接

Zookeeper zk = new Zookeeper(“[ip地址:端口号]” ，[超时时间]，[一个实现Watcher方法的类])；

Watcher接口需要实现process方法。

zookeeper在连接成功时会调用一次process接口。

zk.getData([想要获取的文件夹内容的目录]，[是否继续监听]，[new的Stat对象，在成员变量里])

该方法的接受参数作用，

参数1：定义想要获取的数据在服务中心的位置；

参数2；是否创建一个一次性的监听，该监听会在下次想要获取的文件内容发生改动时调用process方法；

参数3：不知道干啥用的；

返回值：参数1定义的位置的数据值

当zookeeper的服务端数据放生变化时会调用一次process方法，client获取数据是一次性操作，要在getData()方法内定义watch为true才能持续监听

服务器连接成功时，也会调用一次process

process方法：在监听到数据变化时被调用，方法接受参数WatchEvent event

event.getPath();返回值为这次被触发的监听器所监听的数据的目录位置

event.getState();服务器连接状态信息

zookeeper集群管理功能：

1创建一个集群根节点

2让所有集群节点在根目录中创建新的临时目录

3让每个集群节点监听集群的根节点

4当有新的节点加入时，就会在集群根节点目录创建新的临时集群节点，其他节点就会接受到监视的通知，同时重新拉去集群信息

5同理当有节点宕机时也会触发监视

zookeeper集群：

leader节点：可以提供读服务和写服务

follower节点：只能提供读的服务，不能提供写的服务

为了保持数据的一致性，所有对从节点的写操作会被转发到主节点处理

zookeeper客户端：

查询某路径下的节点：ls查询节点

在节点中添加数据：create [-s序号节点] [-e临时节点] [-c 容器节点] [-t ttl定时删除节点] [路径] [添加的数据]

获取节点中的数据：get [路径]

zookeeper不存在文件目录的概念，不能用cd来跳转目录，所有的节点既是目录也是文件

每一个节点都有字节的事务id

zookeeper的节点类型：(3.5版本前有四种类型，3.5之后新增两种)

持久节点：创建节点后，除非手动删除，否则不会丢失，储存在硬盘中

持久序号节点：被创建的节点会在末尾加上排序编号，这个过程是串行的，绝对不会出现相同序号的节点，用于数据争抢时保证唯一性。

临时节点：客户端关闭后就会被删除，不可拥有子节点

临时序号节点：作用等同于持久序号节点。

锁的两种类型：

共享锁：也叫只读锁，当一方获得共享锁之后，其他方也可以获得共享锁，但只允许读。在共享锁全部释放之前，其他方不能获得写锁

排他锁：也叫读写锁。在其释放之前，其他方不能获得任何锁。

分布式锁底层架构和数据结构：

读锁：

1每个客户端在调用资源时会创建临时序号节点

2获取序号比自己小的节点

3判断所有比自己小的节点是否都是读节点

4如果比自己小的节点都是读节点，则可以直接拿到资源信息

5如果有写节点，则添加子节点变更监听

6监听触发后重新判断所有比自己序号小的节点

7所有比自己小的节点都是读结点，拿到资源

写锁：

1客户端创建一个写节点

2判断自己是否为最小事务id节点

3是则直接获取锁，否则进入等待

4