watch机制

一次watch只能收到一次通知。如需要次序监控，需要收到通知后再watch。

客户端先收到watch通知，然后才能感知数据变化。

数据监控，子节点监控

3.各种watch触发的情况总结

可以注册watcher的方法：getData、exists、getChildren。

       可以触发watcher的方法：create、delete、setData。连接断开的情况下触发的watcher会丢失。

       一个Watcher实例是一个回调函数，被回调一次后就被移除了。如果还需要关注数据的变化，需要再次注册watcher。

         New ZooKeeper时注册的watcher叫default watcher，它不是一次性的，只对client的连接状态变化作出反应。

watch为什么不是永久的？

为什么不是永久的，举个例子，如果服务端变动频繁，而监听的客户端很多情况下，每次变动都要通知到所有的客户端，给网络和服务器造成很大压力。  
一般是客户端执行getData(“/节点A”,true)，如果节点A发生了变更或删除，客户端会得到它的watch事件，但是在之后节点A又发生了变更，而客户端又没有设置watch事件，就不再给客户端发送。  
在实际应用中，很多情况下，我们的客户端不需要知道服务端的每一次变动，我只要最新的数据即可。

面试文章：

<https://www.cnblogs.com/lanqiu5ge/p/9405601.html>

多层级节点命名空间，这些节点可以设置关联的数据。而文件系统只有文件可以存放数据，目录节点不行。

为了保证低延迟和吞吐量，zookeeper在内存中维护了整个数据树。这使得zookeeper不能用于存放大量的数据，每个节点存放数据量的上限1m

4种类型的节点。

persistent-持久节点

除非手动删除，否则节点一直存在于zookeeper上

ephemeral-临时节点

临时节点的生命周期与客户端回话绑定，客户端失效，客户端创建的临时节点都会被移除。

persistent-sequential-顺序持久节点

基本特性同持久节点，增加了顺序属性，节点名后边会追加一个由父节点维护的自增整型数字。

ephemeral-sequential-临时顺序节点

基本特性同临时节点，增加了顺序属性，节点名后边会追加一个由父节点维护的自增整型数字。

服务器角色

**Leader**

* 事务请求的唯一调度和处理者，保证集群事务处理的顺序性
* 集群内部各服务的调度者

**Follower**

* 处理客户端的非事务请求，转发事务请求给Leader服务器
* 参与事务请求Proposal的投票
* 参与Leader选举投票

**Observer**

3.3.0版本以后引入的一个服务器角色，在不影响集群事务处理能力的基础上提升集群的非事务处理能力

* 处理客户端的非事务请求，转发事务请求给Leader服务器
* 不参与任何形式的投票

zookeeper节点状态

服务器具有四种状态，分别是LOOKING、FOLLOWING、LEADING、OBSERVING。

* LOOKING：寻找Leader状态。当服务器处于该状态时，它会认为当前集群中没有Leader，因此需要进入Leader选举状态。
* FOLLOWING：跟随者状态。表明当前服务器角色是Follower。
* LEADING：领导者状态。表明当前服务器角色是Leader。
* OBSERVING：观察者状态。表明当前服务器角色是Observer。

zookeeper节点数量 奇数。

zookeeper集群部署模式

部署模式：单机模式、伪集群模式、集群模式。

zookeeper集群节点挂掉

Zookeeper本身也是集群，推荐配置不少于3个服务器。Zookeeper自身也要保证当一个节点宕机时，其他节点会继续提供服务。  
如果是一个Follower宕机，还有2台服务器提供访问，因为Zookeeper上的数据是有多个副本的，数据并不会丢失；  
如果是一个Leader宕机，Zookeeper会选举出新的Leader。  
ZK集群的机制是只要超过半数的节点正常，集群就能正常提供服务。只有在ZK节点挂得太多，只剩一半或不到一半节点能工作，集群才失效。  
所以  
3个节点的cluster可以挂掉1个节点(leader可以得到2票>1.5)  
2个节点的cluster就不能挂掉任何1个节点了(leader可以得到1票<=1)

zookeeper如何保证事务的顺序性？

事务id。是一个64位的数字，高32位是epoch，低32位是自增id。epoch用来标leader周期，如果有新的leader产生，epoch会自增。

两阶段提交。