# Zookeeper基础

## ZooKeeper用途

Zookeeper的用途包括统一配置管理、统一命名服务、分布式锁、集群管理。

**统一配置管理**：

我们可以将common.yml这份配置放在ZooKeeper的Znode节点中，系统A、B、C监听着这个Znode节点有无变更，如果变更了，及时响应。

**统一命名服务**：

将几个服务的ip存于一个Znode下

**分布式锁**：

系统A、B、C都去访问/locks节点。访问的时候会创建带顺序号的临时/短暂(EPHEMERAL\_SEQUENTIAL)节点，比如，系统A创建了id\_000000节点，系统B创建了id\_000002节点，系统C创建了id\_000001节点。

接着，拿到/locks节点下的所有子节点(id\_000000,id\_000001,id\_000002)，判断自己创建的是不是最小的那个节点。如果是，则拿到锁（释放锁：执行完操作后，把创建的节点给删掉）。如果不是，则监听比自己要小1的节点变化。

**集群管理**：

以我们三个系统A、B、C为例，在ZooKeeper中创建临时节点即可（/groupMember/A，/groupMember/B，/groupMember/C）, 只要系统A挂了，那/groupMember/A这个节点就会删除，通过监听groupMember下的子节点，系统B和C就能够感知到系统A已经挂了。(新增也是同理)

## Zookeeper实现

ZooKeeper通过Znode的节点类型+监听机制来实现以上功能

ZooKeeper的数据结构，跟Unix文件系统非常类似，可以看做是一颗树，每个节点叫做ZNode。每一个节点可以通过路径来标识，Znode分为两种类型：

短暂/临时(Ephemeral)：当客户端和服务端断开连接后，所创建的Znode(节点)会自动删除

持久(Persistent)：当客户端和服务端断开连接后，所创建的Znode(节点)不会删除

ZooKeeper还配合了监听器才能够做那么多事的。常见的监听场景有以下两项：

监听Znode节点的数据变化

监听子节点的增减变化