# 入门

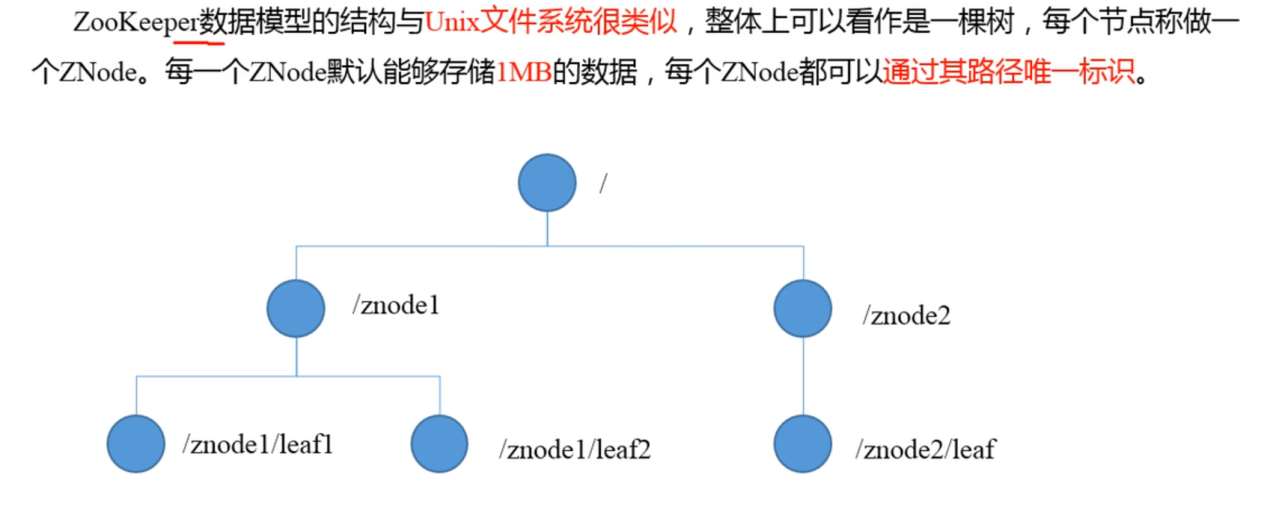
Zookeeper是开源分布式，apache项目

其从设计角度来看是观察者模式设计的分布式服务管理框架

## 特点

1. 由一个leader，多个follower组成的集群
2. 集群中半数以上节点存活，才能正常工作
3. 全局数据一致，每个server数据一样
4. 客户端提交的任务顺序执行
5. 数据更新原子性

## 数据结构



## 参数配置

tickTime：心跳时间

initTime：启动时候心跳帧（多少个心跳时间）

syncLimit：运行时候心跳帧

dataDir：持久化路径

# 分布式

## CAP定理和BASE理论

### CAP

指出一个系统不可能同时满足下面C一致性，A可用性、P分区容错性，最多满足两个

因为是分布式，所以P是不能去掉的，不然就是单机了，没有意思。

所以只能在C和A中选择

### BASE理论

是基本可用、软状态、最终一致性的简称。

核心思想：无法达到强一致性，也可以通过适当方式来使系统保证最终一致性

## 一致性协议

# zookeeper原理

## 选举机制

1. 半数机制：半数以上节点存活才能运行，适合安装奇数台服务器
2. 每个server会先给自己投票，并且其他server交换信息查看，如果没有结果，会给id大的投票。选取出leader后，之后进入的server只能成为follower

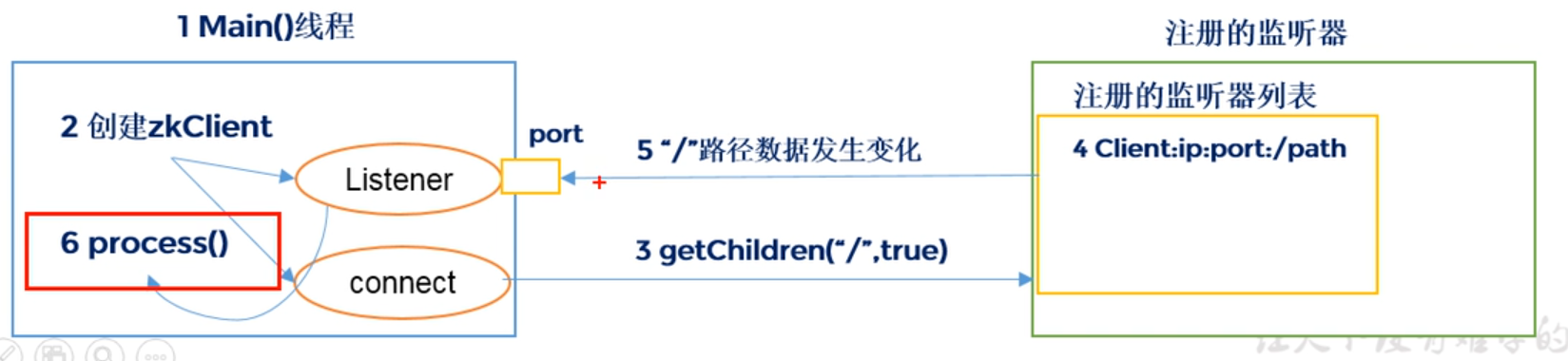
## 节点类型

持久：客户端与服务器断开连接，创建的节点不删除

短暂：客户端与服务器断开连接，节点会删除

## 监听器原理

1. main线程创建zookeeper客户端，也就创建listener线程和connect线程
2. connect线程把要监听的路径发送到zookeeper服务器
3. 注册到监听器列表中
4. 对应路径发生变化，就返回给listener线程
5. 最后内部调用process方法



个人理解客户端注册watcher，向服务器发出请求，服务器响应后，客户端在watchermanageer注册它。服务器那边也相应注册其到watchermanager，当相应节点发生变化，服务器查找对应watcher，并通过watcher.process方法向对应客户端发出消息，客户端收到后封装一个watcherEvent并交给EventThread线程处理，EventThread线程会获得对应watcher，并调用其process方法。

## 写数据流程

1. client修改数据，向server1发送写请求
2. 如果server1不是leader，则需要向leader发请求
3. leader需要向其他follower广播请求
4. 各个follower收到后修改成功就通知leader
5. 收到大多数请求成功后，leader通知server1写成功
6. Server1通知client写成功

