# 2、线程的并发工具类

## Fork-Join

**什么是分而治之？**

规模为N的问题，N<阈值，直接解决，N>阈值，将N分解为K个小规模子问题，子问题互相对立，与原问题形式相同，将子问题的解合并得到原问题的解

动态规范

**工作密取**

workStealing

**Fork/Join使用的标准范式**

## 常用的并发工具类

**CountDownLatch**

作用：是一组线程等待其他的线程完成工作以后在执行，加强版join

await用来等待，countDown负责计数器的减一

**CyclicBarrier**

让一组线程达到某个屏障，被阻塞，一直到组内最后一个线程达到屏障时，屏障开放，所有被阻塞的线程会继续运行CyclicBarrier(int parties)

CyclicBarrier(int parties, Runnable barrierAction)，屏障开放，barrierAction定义的任务会执行

CountDownLatch和CyclicBarrier辨析

1、countdownlatch放行由第三者控制，CyclicBarrier放行由一组线程本身控制  
2、countdownlatch放行条件》=线程数，CyclicBarrier放行条件=线程数

**Semaphore**

控制同时访问某个特定资源的线程数量，用在流量控制

**Exchange**

两个线程间的数据交换，

**Callable、Future和FutureTask**

isDone，结束，正常还是异常结束，或者自己取消，返回true；

isCancelled 任务完成前被取消，返回true；

cancel（boolean）：

1. 任务还没开始，返回false
2. 任务已经启动，cancel（true），中断正在运行的任务，中断成功，返回true，cancel（false），不会去中断已经运行的任务
3. 任务已经结束，返回false

包含图片和文字的文档的处理：图片（云上），可以用future去取图片，主线程继续解析文字。