# 4、显式锁和AQS

## 显式锁

**Lock接口和核心方法**

**Lock接口和synchronized的比较**

**synchronized 代码简洁，Lock：获取锁可以被中断，超时获取锁，尝试获取锁，读多写少用读写锁**

**可重入锁ReentrantLock、所谓锁的公平和非公平**

如果在时间上，先对锁进行获取的请求，一定先被满足，这个锁就是公平的，不满足，就是非公平的

非公平的效率一般来讲更高

**ReadWriteLock接口和读写锁ReentrantReadWriteLock**

ReentrantLock和Syn关键字，都是排他锁，

读写锁：同一时刻允许多个读线程同时访问，但是写线程访问的时候，所有的读和写都被阻塞，最适宜与读多写少的情况

**Condition接口**

**用Lock和Condition实现等待通知**

## 了解LockSupport工具

**park开头的方法**

负责阻塞线程

**unpark(Thread thread)方法**

负责唤醒线程

## AbstractQueuedSynchronizer深入分析

#### 什么是AQS？学习它的必要性

**AQS使用方式和其中的设计模式**

继承，模板方法设计模式

**了解其中的方法**

模板方法：

独占式获取

accquire

acquireInterruptibly

tryAcquireNanos

共享式获取

acquireShared

acquireSharedInterruptibly

tryAcquireSharedNanos

独占式释放锁

release

共享式释放锁

releaseShared

需要子类覆盖的流程方法

独占式获取 tryAcquire

独占式释放 tryRelease

共享式获取 tryAcquireShared

共享式释放 tryReleaseShared

这个同步器是否处于独占模式 isHeldExclusively

同步状态state：

getState:获取当前的同步状态

setState：设置当前同步状态

compareAndSetState 使用CAS设置状态，保证状态设置的原子性

#### AQS中的数据结构-节点和同步队列

竞争失败的线程会打包成Node放到同步队列，Node可能的状态里：

***CANCELLED：***线程等待超时或者被中断了，需要从队列中移走

***SIGNAL：***后续的节点等待状态，当前节点，通知后面的节点去运行

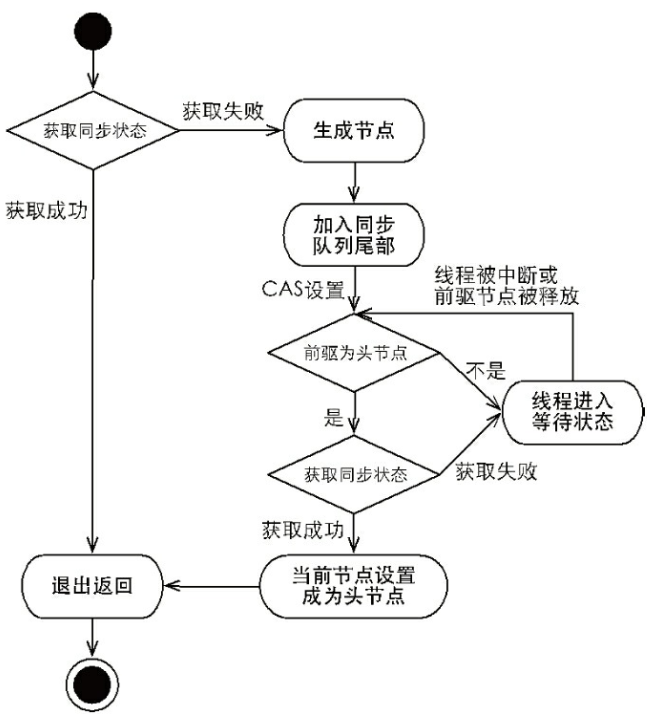
CONDITION :当前节点处于等待队列

***PROPAGATE：共享，表示状态要往后面的节点传播***

1. ***表示初始状态***

#### 节点在同步队列中的增加和移出

#### 独占式同步状态获取与释放



#### 其他同步状态获取与释放

#### Condition分析