## AbstractQueuedSynchronizer

## ReentrantLock

跟synchronized相同，是可重入的重量级锁。但是其用法则相当不同，首先ReentrantLock要显式的调用lock方法表示接下来的这段代码已经被当前线程锁住，其他线程需要执行时需要拿到这个锁才能执行，而当前线程在执行完之后要显式的释放锁。

### 2.1、ReentrantLock()

方法作用：无参构造方法，默认新建非公平同步器

①调用内部类NonfairSync无参构造赋值给成员变量sync；

②依次调用父类的默认构造函数，各级父类无参构造函数没有做任何事情；

### 2.2、ReentrantLock(boolean fair)

方法作用：带布尔参数构造方法，fair为true，创建公平同步器，否则创建非公平同步器

### 2.3、void lock()

方法作用：获得锁

假如创建的是非公平同步器：

①调用成员变量sync的lock()方法（及调用NonfairSync的lock()方法）；

②调用AbstractQueuedSynchronizer（AQS）的compareAndSetState(int expect, int update)方法，expect默认为0，update默认为1；

③调用成员变量unsafe的compareAndSwapInt(this, stateOffset, expect, update)，this代表本非公平同步器sync，stateOffset默认值为0，expect默认为0，update默认为1；此方法希作用：当成员变量state的值为0时，则将state设置为1并返回true，否则state值保持不变并返回false；

④如果第③步返回true，则调用Thread.currentThread()获取当前线程，然后调用AbstractOwnableSynchronizer的setExclusiveOwnerThread(Thread thread)方法将当前线程赋值给成员变量exclusiveOwnerThread（此线程就是独占所有者线程）；到此本方法全部执行完毕；

⑤如果第③步返回false，则调用AQS的acquire(int arg)方法，参数arg默认为1；

⑥