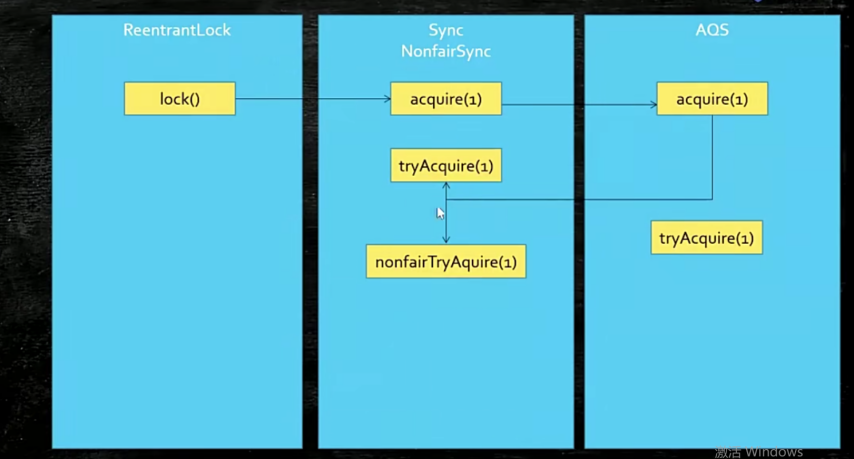
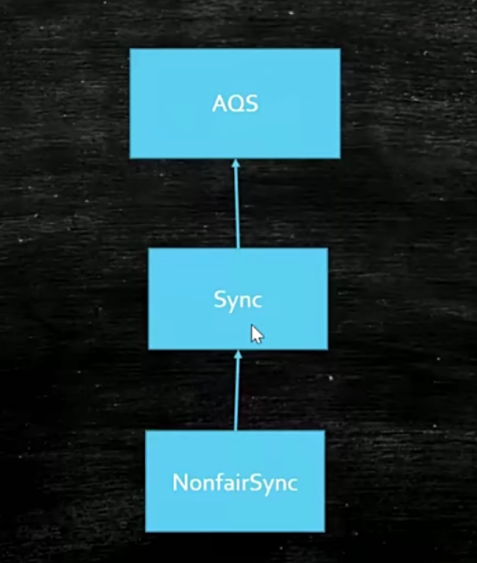
1. 公平锁、非公平锁

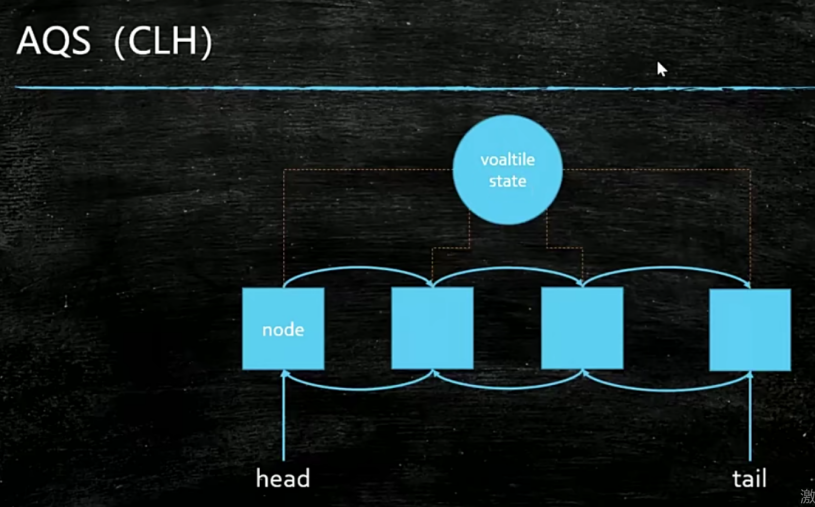
公平锁：线程想要获得锁，需要在等待队列排队，等待前面线程释放锁，才可以获得锁

非公平锁：线程想要获得锁，可以直接抢占，没抢占才进入等待队列排队

1. AQS







1. AQS结构

A）state

state是volatile，保证线程间可见，线程共享数据。

含义（不同子类，不同含义）

* ReentrantLock：1代表锁被抢占了，0代表锁未被抢占
* CountDownLatch：数值代表门闩数

B）双向链表

监控state的双向链表，每个节点存储线程Thread

1. AQS= volatile + CAS

volatile：state，线程共享可见

CAS：修改state值；将节点接入队列尾部

1. 非公平锁的加锁过程：
2. 线程尝试获得锁（tryAquire）

* 如果state为0，则通过CAS操作尝试将state改为1，修改改成功则表示当前线程获得锁。
* 否则如果当前线程持有锁，则将state加1，表示锁为可重入锁。

1. 线程未获得锁，线程加入等待队列，等待获得锁（aquireQueued）

* 将线程通过CAS操作（将tail修改为当前线程Node）加入等待队列（addwaiter）
* 如果当前线程为等待队列第二个，则tryaqire去获得锁，获得锁则返回
* 否则对于其他情况，需要park等待前面线程释放锁

1. 非公平锁的释放锁过程：
2. 线程释放锁

如果state减完之后为0，则将state置为0，并设置当前线程不占用锁（可重入锁需要减为0才可以释放）

1. 当前线程唤醒下一个线程获得锁

从前往后遍历等待队列，则通过unpark唤醒第一个未获得锁的线程