853769620

1、面试题

AQS的实现原理是什么？哪些场景会用到AQS？

2、面试官心里分析

这个AQS说实在的，是比较高级的并发这块的面试题了，但是确实有些公司会问，所以我这边还是给大家来讲一下

3、面试题剖析

AQS，AbstractQueuedSynchronizer，抽象的队列式同步器，AQS其实主要定义了一套对共享资源访问的同步框架，比如ReentrantLock、Semaphore、CountDownLatch都是依赖这个AQS的。AQS是java用来构建锁和其他同步组件的一个基础框架。

内部有一个FIFO队列，如果某个线程获取同步状态失败了，那么AQS就会把这个线程构造成一个node，塞入FIFO队列的尾部，然后如果同步状态释放的时候，就会从FIFO队列的头部唤醒一个node。其实说白了，就是让线程获取一个共享资源的时候，自动进行排队。

这就是所谓java中的最核心的同步这块的底层逻辑。

AQS里是有一个volatile修饰的state的，就代表了某个共享资源的同步状态，或者更加方便的理解，可以认为是一把锁，比如多个线程访问某段被锁的代码，会去尝试获取锁，其实就是在这里去尝试获取同步状态。

一个AQS，你可以认为是代表了用于获取一把锁的这么一个控制器

AQS里面，有一个FIFO队列，用来排队的；还有一个state，代表了锁的这么一个状态

Lock.lock()

// 干活儿

Lock.unlock()

获取一把锁的时候，尝试去获取AQS里的state状态（锁的状态）

刚开始state的值是0（代表没有人获取这把锁），线程1是可以获取到锁的，然后就会将state设置为1；接着线程2来尝试获取state的值，但是此时state = 1，表示被别人锁了，所以线程2的信息构造成一个node放入FIFO队列中来排队，同时阻塞住线程2

如果线程1释放锁的时候，就会将state设置为0，此时会唤醒FIFO队列中的排在队头的那个线程2，然后线程2发现state = 0，就会获取锁，将state设置为1，同时线程2的node从FIFO队列里面出队。

其实这块我不打算在这里太过于深入的分析，因为反正是应付面试题，你能理解到这里就ok了，简单来说AQS就是一个基础并发框架，ReentractLock之类的都是基于AQS来的，比如ReentractLock里面，某个线程想要获取锁，那么就会走上面AQS定义好的基础逻辑，获取state值啥的。

包括多个线程争用一把ReentractLock锁的时候，就会出现排队，然后某个线程释放锁之后，按照FIFO的顺序后面的线程自动获取锁。就这个意思。所以AQS就是个基础框架，提供了获取锁，FIFO排队，释放锁的基础逻辑，让ReentractLock、Semaphore、CountDownLatch之类的并发包下的类来使用。

暂时理解到这个程度就ok了。