

6年的java被刚入门的小弟弟提的问题难住了?

=====

问题

==

有个亲戚的孩子，非要入行计算机学java，这现在还来学java不就是49年入国军吗。非要学，给了几套课程让自学，让有不懂的来问我，这不，带着新的问又来了。

也就是下面这行代码为什么执行之后不能完结，就一直卡在那里？我们来瞅瞅这个代码，基本就是演示了代码演示了`ReentrantLock`的基本使用和线程之间的竞争。

...

```
public class Domain {
    public static void main(String[] args) throws InterruptedException {
        ReentrantLockTest t = new ReentrantLockTest();
        new Thread(t).start();
        new Thread(t).start();
        new Thread(t).start();
    }
}
class ReentrantLockTest implements Runnable {
    ReentrantLock r = new ReentrantLock();
    int t = 2;

    @Override
    public void run() {
        while (true) {
            r.lock();
            if (t > 0) {
                t -= 1;
            } else {
                r.unlock();
                break;
            }
        }
    }
}
```

...

演示

==

执行一下

![image.png](https://p6-juejin.byteimg.com/tos-cn-i-k3u1fbpfcp/ca5e4c2b67ec4a09be0bf2e55dd5b505~tplv-k3u1fbpfcp-jj-mark:3024:0:0:0:q75.awebp#?w=604&h=239&s=11883&e=png&b=25272a)

果然是卡住了，不结束执行也不继续执行。

这得先分析分析，卡哪了？

思路 1 日志大法

=====

先上日志大法：

给run()方法添加日志如下

...

```
@Override
public void run() {
    log.info("Thread.currentThread().getName() = {} 在循环外准备开始循环", Thread.currentThread().getName());
    while (true) {
        log.info("Thread.currentThread().getName() = {} 准备动手拿锁", Thread.currentThread().getName());
        r.lock();
        log.info("Thread.currentThread().getName() = {} 拿到了锁", Thread.currentThread().getName());
        if (t > 0) {
            log.info("Thread.currentThread().getName() = {} 在执行 {}", Thread.currentThread().getName(), t);
            t -= 1;
        } else {
            log.info("Thread.currentThread().getName() = {} 解锁,退出循环",
```

```

Thread.currentThread().getName());
        r.unlock();
        break;
    }
}
log.info("Thread.currentThread().getName() = {} 执行完毕",
Thread.currentThread().getName());
}
...

```

然后执行

![image.png](https://p3-juejin.byteimg.com/tos-cn-i-k3u1fbpfcp/9b6995b03a5349749018e35c2deaaf84~tplv-k3u1fbpfcp-jj-mark:3024:0:0:0:q75.awebp#?w=1090&h=484&s=111262&e=png&b=202124)

可以看到线程0，1，2都准备拿锁，然后线程0拿到了锁，执行完毕之后，线程1和2还是在准备拿锁。这就奇怪了，线程0都释放锁了，线程1，2怎么不去获取锁而是一直卡着呢？因为抢锁打起来，打伤了？？？

思路 2 增加监测线程，检测线程状态

=====

那我们在主线程起起一个检测线程状态的线程看看。

```

...
public static void main(String[] args) throws InterruptedException {
    ReentrantLockTest t = new ReentrantLockTest();
    new Thread(t).start();
    new Thread(t).start();
    new Thread(t).start();
    while (true){
        Thread[] threads = new Thread[Thread.activeCount()];
        Thread.enumerate(threads);
        for (Thread thread : threads) {
            log.info("thread = {}-{}", thread.getName(), thread.getState());
        }
        Thread.sleep(2000);
    }
}
...

```

执行之后：

![image.png](https://p9-juejin.byteimg.com/tos-cn-i-k3u1fbpfcp/f527b45f46134008aa33b2e07145da97~tplv-k3u1fbpfcp-jj-mark:3024:0:0:0:q75.awebp#?w=1092&h=966&s=224728&e=png&b=202124)

可以看到，在线程0拿到锁之后，线程1和线程2的状态一直是`WAITING`状态，ps(这是我多次尝试的结果哈，其实哪个线程拿到锁是不一定的，为了和上边的结果一致，刷了几次刷到了线程0拿锁。)

那么问题来了，为啥一直处于WAITING状态呢。

思路 3 尝试解锁

=====

我们去掉检测线程状态的逻辑，在while循环最后增加一个解锁的操作：

```
...
@Override
public void run() {
    log.info("ThreadName():{} 在循环外准备开始循环",
Thread.currentThread().getName());
    while (true) {
        log.info("ThreadName():{} 准备动手拿锁",
Thread.currentThread().getName());
        r.lock();
        log.info("ThreadName():{} 拿到了锁",
Thread.currentThread().getName());
        if (t > 0) {
            log.info("ThreadName():{} 在执行 {}",
Thread.currentThread().getName(), t);
            t -= 1;
        } else {
            log.info("ThreadName():{} 解锁,退出循环",
Thread.currentThread().getName());
            r.unlock();
            break;
        }
        r.unlock();//增加这个
    }
    log.info("ThreadName():{} 执行完毕",
```

```
Thread.currentThread().getName());  
}
```

...

执行后，可以看到在每次执行完之后解了锁，那么等待拿锁的线程都是有机会拿到锁并且执行的。

![image.png](https://p1-juejin.byteimg.com/tos-cn-i-k3u1fbpfcp/080650c50be04139befacfac4327aa8a~tplv-k3u1fbpfcp-jj-mark:3024:0:0:0:q75.awebp#?w=1124&h=697&s=163673&e=png&b=212225)

那么上面的为啥不行呢

思路 4 粗暴尝试

=====

可以分析到，是加锁多次，然后解锁了一次，应该是这个行不通导致的。

那我们尝试一下。直接粗暴的解锁三次，看看是不是这儿的问题。

![image.png](https://p3-juejin.byteimg.com/tos-cn-i-k3u1fbpfcp/68d821c0af79474eb20506dd320dbdc2~tplv-k3u1fbpfcp-jj-mark:3024:0:0:0:q75.awebp#?w=997&h=766&s=90754&e=png&b=1f2023)

执行，可以看到在线程0执行完解锁之后，其余线程也都执行了，报错可以先忽略，之后会解释下。

![image.png](https://p9-juejin.byteimg.com/tos-cn-i-k3u1fbpfcp/e0adaeff01a74464a75fb47ac17cc5fd~tplv-k3u1fbpfcp-jj-mark:3024:0:0:0:q75.awebp#?w=1267&h=780&s=175788&e=png&b=212225)

那么就可以肯定，就是这里加锁多次，解锁只解了一次导致的。那么就沿着这个方向继续看看。

思路 5 找到方向，查资料

=====

这里肯定就牵扯`ReentrantLock`的一些机制了，查查资料：

> ReentrantLock是一个可重入锁。这意味着同一个线程可以多次获取同一个锁，并且每次获取锁时，锁的重入计数器会增加；每次释放锁时，锁的重入计数器会减少。当重入计数器减为零时，锁才会真正被释放。

思路 6 看源码

=====

哇。还有这个，去看看源码：

![image.png](https://p9-juejin.byteimg.com/tos-cn-i-k3u1fbpfcp/3de4a18592764bf7bd3554cccbd86a19~tplv-k3u1fbpfcp-jj-mark:3024:0:0:0:q75.awebp#?w=884&h=981&s=102553&e=png&b=202124)

这里这个state就是加锁的次数，在每次执行`lock()`的时候，state的值会加1。

然后我们看看解锁的源码：

![image.png](https://p1-juejin.byteimg.com/tos-cn-i-k3u1fbpfcp/3dd66e9cfad346fc93f75b74a31cc594~tplv-k3u1fbpfcp-jj-mark:3024:0:0:0:q75.awebp#?w=1370&h=876&s=130019&e=png&b=020202)

> 懂了懂了，在执行解锁的时候，先给state减1，然后判断是不是减到0了，要是没有到0那就啥都不干。可是要是到0了就检查当前节点是否为空，以及下一个节点是否为空。如果下一个节点的状态不是0（表示有线程在等待），则调用`s.getAndUnsetStatus(WAITING)`来重置节点的等待状态。使用`LockSupport.unpark(s.waiter)`来唤醒等待的线程。

总结

==

就是ReentrantLock在使用的时候，加锁几次，就需要解锁几次。这时候上面那个报错就很好理解了，当前线程不持有锁了，解锁当然是报错的。

相代码这里了：

![image.png](https://p3-juejin.byteimg.com/tos-cn-i-

k3u1fbpfcf/9b85c7ab57df4eb2b199fa6b89612ca4~tplv-k3u1fbpfcf-jj-
mark:3024:0:0:0:q75.awebp#?w=700&h=330&s=35253&e=png&b=1f202
3) 原文链接: <https://juejin.cn/post/7386461612699893770>