二

68 发生死锁必须满足哪 4 个条件?

本课时我将为你介绍发生死锁必须满足哪 4 个条件。

发生死锁的 4 个必要条件

要想发生死锁有 4 个缺一不可的必要条件，我们一个个来看：

- 第 1 个叫**互斥条件**，它的意思是每个资源每次只能被一个线程（或进程，下同）使用，为什么资源不能同时被多个线程或进程使用呢？这是因为如果每个人都可以拿到想要的资源，那就不需要等待，所以是不可能发生死锁的。
- 第 2 个是**请求与保持条件**，它是指当一个线程因请求资源而阻塞时，则需对已获得的资源保持不放。如果在请求资源时阻塞了，并且会自动释放手中资源（例如锁）的话，那别人自然就能拿到我刚才释放的资源，也就不会形成死锁。
- 第 3 个是**不剥夺条件**，它是指线程已获得的资源，在未使用完之前，不会被强行剥夺。比如我们在上一课时中介绍的数据库的例子，它就有可能去强行剥夺某一个事务所持有的资源，这样就不会发生死锁了。所以要想发生死锁，必须满足不剥夺条件，也就是说当现在的线程获得了某一个资源后，别人就不能来剥夺这个资源，这才有可能形成死锁。
- 第 4 个是**循环等待条件**，只有若干线程之间形成一种头尾相接的循环等待资源关系时，才有可能形成死锁，比如在两个线程之间，这种“循环等待”就意味着它们互相持有对方所需的资源、互相等待；而在三个或更多线程中，则需要形成环路，例如依次请求下一个线程已持有的资源等。

案例解析

下面我们回到上一课时中所写的必然死锁的例子中，看看它是否一一满足了这 4 个条件，案例代码如下所示：

```
/**
 * 描述：        必定死锁的情况
```

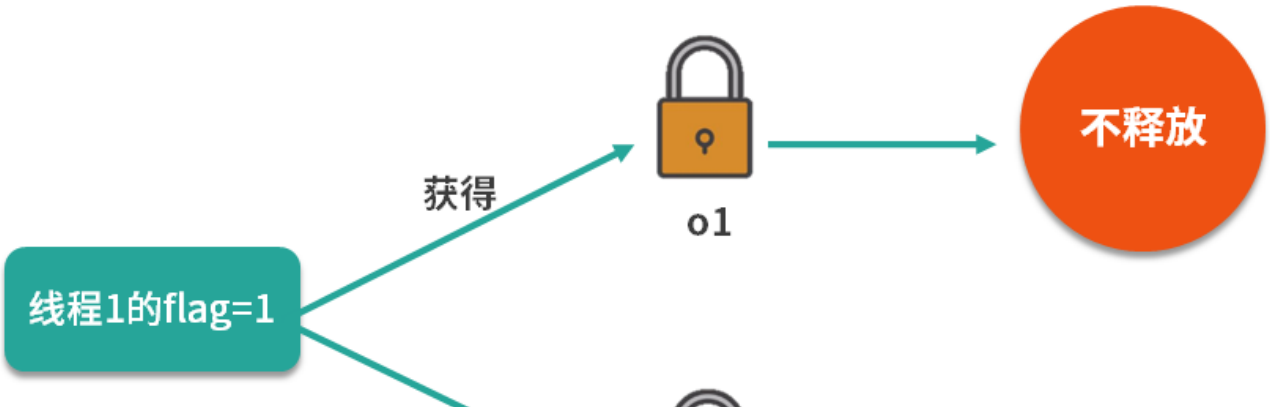
```
 */
public class MustDeadLock implements Runnable {
    public int flag;
    static Object o1 = new Object();
    static Object o2 = new Object();
    public void run() {
        System.out.println("线程"+Thread.currentThread().getName() + "的flag为" + f
        if (flag == 1) {
            synchronized (o1) {
                try {
                    Thread.sleep(500);
                } catch (Exception e) {
                    e.printStackTrace();
                }
                synchronized (o2) {
                    System.out.println("线程1获得了两把锁");
                }
            }
        }
        if (flag == 2) {
            synchronized (o2) {
                try {
                    Thread.sleep(500);
                } catch (Exception e) {
                    e.printStackTrace();
                }
                synchronized (o1) {
                    System.out.println("线程2获得了两把锁");
```

```
                }
            }
        }
    }
    public static void main(String[] argv) {
        MustDeadLock r1 = new MustDeadLock();
        MustDeadLock r2 = new MustDeadLock();
        r1.flag = 1;
        r2.flag = 2;
        Thread t1 = new Thread(r1, "t1");
        Thread t2 = new Thread(r2, "t2");
        t1.start();
        t2.start();
    }
}
```

这个代码的具体分析和执行结果，我们在上一课时中已经介绍过了，这里不重复讲解，下面我们把重点放在对这 4 个必要条件的分析上。

我们先来看一下第 1 个互斥条件，很显然，我们使用的是 synchronized 互斥锁，它的锁对象 o1、o2 只能同时被一个线程所获得，所以是满足互斥条件的。

第 2 个是请求与保持条件，可以看到，同样是满足的。比如，线程 1 在获得 o1 这把锁之后想去尝试获取 o2 这把锁，这时它被阻塞了，但是它并不会自动去释放 o1 这把锁，而是对已获得的资源保持不放。



第 3 个是不剥夺条件，在我们这个代码程序中，JVM 并不会主动把某一个线程所持有的锁剥夺，所以也满足不剥夺条件。

第 4 个是循环等待条件，可以看到在我们的例子中，这两个线程都想获取对方已持有的资源，也就是说线程 1 持有 o1 去等待 o2，而线程 2 则是持有 o2 去等待 o1，这是一个环路，此时就形成了一个循环等待。

可以看出，在我们的例子中确实满足这 4 个必要条件，今后我们就可以从这 4 个发生死锁的必要条件出发，来解决死锁的问题，只要破坏任意一个条件就可以消除死锁，这也是我们后面要讲的解决死锁策略中重点要考虑的内容。

总结

以上就是本课时的内容，我们总结一下，在本课时主要介绍了要想发生死锁，必须满足的 4 个条件，分别是**互斥条件、请求与保持条件、不剥夺条件**和**循环等待条件**；同时还分析了在上一课时中必然发生死锁的例子，可以看到，在这个例子中确实满足了这 4 个条件。

上一页 下一页