1 LockSupport 方法介绍

Method	Description
park()	挂起当前线程。唤醒条件：1、unpark 2、被中断 **3、莫名其妙的醒来**
parkNanos(long)	唤醒的条件增加一条：当过了一定的时间后
parkUntil(long)	唤醒的条件增加一条：当到达指定的时间后
park(obj)	obj：挂起当前线程所驻留的对象（同 synchronized）
parkNanos(obj,long)	同上
parkUntil(obj,long)	同上
unpark(thread)	唤醒 thread 线程
getBlocker(thread)	如果 thread 线程被 park，**返回驻留的对象**。否则返回 null
nextSecondarySeed()	返回 伪随机数

2 LockSupport 的适用场景

- 由于 park() 方法会莫名其妙醒来，需要通过 while 循环条件包装它

```
while (condition()) {
    LockSupport.park(this);
}
```

- 由于线程被中断时，也会被唤醒，可能要进行是否被中断的判断

3 LockSupport 在 AQS 中的应用

- 以下是线程获取独占锁失败后，尝试进入等待队列的代码
- 注意：park() 方法是在 for() 循环中的，可以看出，即使线程醒来，它还是需要重新 tryAcquire() 才能正在获取到锁的，而不是直接返回。这样就避免了 线程被无故唤醒的场景

```
final boolean acquireQueued(final Node node, int arg) {
    boolean failed = true;
    try {
        boolean interrupted = false;
```

```
        for (;;) {
            final Node p = node.predecessor();
            if (p == head && tryAcquire(arg)) {
                setHead(node);
                p.next = null; // help GC
                failed = false;
                return interrupted;
            }
            if (shouldParkAfterFailedAcquire(p, node) &&
                parkAndCheckInterrupt())
                interrupted = true;
        }
    } finally {
        if (failed)
            cancelAcquire(node);
    }
}
//如果shouldParkAfterFailedAcquire(p, node)返回true：即线程需要被挂起，则执行以下方法
private final boolean parkAndCheckInterrupt() {
    LockSupport.park(this);//这里调用了 park() 方法来挂起线程
    return Thread.interrupted();
}
```

- 以下是线程释放锁的流程

```
public final boolean release(int arg) {
    if (tryRelease(arg)) {
        Node h = head;
        if (h != null && h.waitStatus != 0)
            //释放锁成功后，调用这个方法来唤醒后继节点，内部调用 unpark() 方法，unpark() 当然不用
while 包围了
            unparkSuccessor(h);
        return true;
    }
    return false;
}
```

4 LockSupport 与 Synchronized 比较

- LockSupport 的 park() 方法可以直接挂起一个线程，而 Synchronized 挂起的是没有抢到监视器的线程，具体哪个线程控制不了
- LockSupport 中的 park() 方法所挂起的线程会意外醒来，需要用 while(condition) 修饰，因 Synchronized 而挂起的线程不会
- LockSupport 的 park()、unpark() 方法更类似于 wait()/notify() 方法，主要用于线程通信。尤其是 wait() 方法也会意外醒来，也需要被 while(condition) 包围
- 区别于 wait() 方法，park() 方法不能被 Synchronized 包围，否则会产生死锁，而 wait() 则必须被 Synchronized 包围，否则会报非法锁状态异常

参考

jdk 1.8u171