

Java多线程进阶（五）—— J.U.C之locks框架：LockSupport



Ressimix 发布于 2018-07-07



本文首发于一世流云的专栏：
 <https://segmentfault.com/blog...>

一、LockSupport类简介

LockSupport类，是JUC包中的一个工具类，是用来创建锁和其他同步类的基本线程阻塞原语。（[Basic thread blocking primitives for creating locks and other synchronization classes](#)）

LockSupport类的核心方法其实就两个：`park()`和`unpark()`，其中`park()`方法用来阻塞当前调用线程，`unpark()`方法用于唤醒指定线程。

这其实和Object类的`wait()`和`signal()`方法有些类似，但是LockSupport的这两种方法从语意上讲比Object类的方法更清晰，而且可以针对指定线程进行阻塞和唤醒。

LockSupport类使用了一种名为Permit（许可）的概念来做到阻塞和唤醒线程的功能，可以把许可看成是一种(0,1)信号量（Semaphore），但与 Semaphore 不同的是，许可的累加上限是1。
 初始时，`permit`为0，当调用`unpark()`方法时，线程的`permit`加1，当调用`park()`方法时，如果`permit`为0，则调用线程进入阻塞状态。

1.1 使用示例

来看一个例子：
 假设现在需要实现一种FIFO类型的独占锁，可以把这种锁看成是`ReentrantLock`的公平锁简单版本，且是不可重入的，就是说当一个线程获得锁后，其它等待线程以FIFO的调度方式等待获取锁。

```

public class FIFOMutex {
    private final AtomicBoolean locked = new AtomicBoolean(false);
    private final Queue<Thread> waiters = new ConcurrentLinkedQueue<Thread>();

    public void lock() {
        Thread current = Thread.currentThread();
        waiters.add(current);

        // 如果当前线程不在队首，或锁已被占用，则当前线程阻塞
        // NOTE: 这个判断的意图其实就是：锁必须由队首元素拿到
        while (waiters.peek() != current || !locked.compareAndSet(false, true)) {
            LockSupport.park(this);
        }
        waiters.remove(); // 删除队首元素
    }

    public void unlock() {
        locked.set(false);
        LockSupport.unpark(waiters.peek());
    }
}

```

测试用例：

```

public class Main {
    public static void main(String[] args) throws InterruptedException {
        FIFOMutex mutex = new FIFOMutex();
        MyThread a1 = new MyThread("a1", mutex);
        MyThread a2 = new MyThread("a2", mutex);
        MyThread a3 = new MyThread("a3", mutex);

        a1.start();
        a2.start();
        a3.start();

        a1.join();
        a2.join();
        a3.join();

        assert MyThread.count == 300;
        System.out.print("Finished");
    }
}

class MyThread extends Thread {
    private String name;
    private FIFOMutex mutex;
    public static int count;

    public MyThread(String name, FIFOMutex mutex) {
        this.name = name;
        this.mutex = mutex;
    }

    @Override
    public void run() {
        for (int i = 0; i < 100; i++) {
            mutex.lock();
            count++;
            System.out.println("name:" + name + " count:" + count);
            mutex.unlock();
        }
    }
}

```

上述FIFOMutex 类的实现中，当判断锁已被占用时，会调用`LockSupport.park(this)`方法，将当前调用线程阻塞；当使用完锁时，会调用`LockSupport.unpark(waiters.peek())`方法将等待队列中的队首线程唤醒。

通过LockSupport的这两个方法，可以很方便的阻塞和唤醒线程。但是LockSupport的使用过程中还需要注意以下几点：

1. **park方法的调用一般要方法一个循环判断体里面。**

如上述示例中的：

```

while (waiters.peek() != current || !locked.compareAndSet(false, true)) {
    LockSupport.park(this);
}

```

之所以这样做，是为了防止线程被唤醒后，不进行判断而意外继续向下执行，这其实是一种[Guarded Suspension](#)的多线程设计模式。

2. **park方法是会响应中断的，但是不会抛出异常。（也就是说如果当前调用线程被中断，则会立即返回但不会抛出中断异常）**

中断状态也还在

3. park的重载方法park(Object blocker)，会传入一个blocker对象，所谓Blocker对象，其实就是当前线程调用时所在调用对象（如上述示例中的FIFOMutex对象）。该对象一般供监视、诊断工具确定线程受阻塞的原因时使用。

二、LockSupport类/方法声明

类声明：

```
public class LockSupport
extends Object
```

方法声明：

Modifier and Type	Method and Description
static Object setBlocker私有的	getBlocker(Thread t) Returns the blocker object supplied to the most recent invocation of a park method that has not yet unblocked, or null if not blocked.
static void	park() Disables the current thread for thread scheduling purposes unless the permit is available.
static void	park(Object blocker) Disables the current thread for thread scheduling purposes unless the permit is available.
static void	parkNanos(long nanos) Disables the current thread for thread scheduling purposes, for up to the specified waiting time, unless the permit is available.
static void	parkNanos(Object blocker, long nanos) Disables the current thread for thread scheduling purposes, for up to the specified waiting time, unless the permit is available.
static void	parkUntil(long deadline) Disables the current thread for thread scheduling purposes, until the specified deadline, unless the permit is available.
static void	parkUntil(Object blocker, long deadline) Disables the current thread for thread scheduling purposes, until the specified deadline, unless the permit is available.
static void	unpark(Thread thread) Makes available the permit for the given thread, if it was not already available.

java 多线程

阅读 89.3k · 更新于 2020-03-21

赞 11

收藏 7

分享

本作品系原创，采用《署名-非商业性使用-禁止演绎 4.0 国际》许可协议



透彻理解Java并发编程

Java并发编程是整个Java开发体系中最难以理解但也是最重要的知识点，也是各类开源分布式框架中各...

关注专栏



Ressmix

1.2k 声望 1.3k 粉丝

关注作者

4 条评论

得票数 最新



撰写评论 ...



提交评论



okxuewei: signal(), 文章中这个方法名需要考虑一下

👍 • 回复 • 2018-12-19



okxuewei: unark()方法名

👍 • 回复 • 2018-12-19



只为眼前山间绿水: unark() -> unpark() 纠正一下

👍 • 回复 • 2020-03-21



goller: mark, 感谢分享

👍 • 回复 • 2020-03-28

你知道吗？

测试只能证明程序有错误，而不能证明程序没有错误。

注册登录

继续阅读

浅谈Java并发编程系列（八）—— LockSupport原理剖析

LockSupport是用来创建锁和其他同步类的基本线程阻塞原语。LockSupport 提供park()和unpark()方法实现阻塞线程和解除线程阻...

[codershamo](#) • 阅读 21.5k • 7 赞 • 4 评论

多线程与高并发07-一道有趣的面试题目

一道有趣的面试题 前面学习了多线程中的各种多线程工具类，还是锁的使用，下面来看一道有趣的面试题目，让大家有机会尝试...

[DragonflyDavid](#) • 阅读 660 • 2 赞

面试 LockSupport.park()会释放锁资源吗？

（手机横屏看源码更方便） 引子 大家知道，我最近在招人，今天遇到个同学，他的源码看过一些，然后我就开始了AQS连环问。...

[彤哥读源码](#) • 阅读 563 • 2 赞 • 1 评论

Unsafe类功能之(5): 线程的挂起和恢复

中断当前线程,直到满足以下条件之一返回:(1). 当前线程被别的线程 unpark(2). 当前线程被中断: interrupted(3). isAbsolute true: 绝...

[niewj](#) • 阅读 745 • 1 赞

LockSupport原理分析

LockSupport类可以阻塞当前线程以及唤醒指定被阻塞的线程。主要是通过park()和unpark(thread)方法来实现阻塞和唤醒线程的操...

[wo883721](#) • 阅读 5.1k • 1 赞 • 2 评论

12 多线程

为什么要用多线程单核时代，为了提高cpu和io设备的综合利用率，一个线程执行cpu计算时，另一个线程进行io操作。多核时代，...

[cathy_mu](#) • 阅读 165

多线程交替打印奇偶数

作者：折纸个人博客：[\[链接\]](#)记录菜鸡的成长之旅！LockSupport原理贴一个之前的笔记，字迹比较潦草zz 有空的时候这里会补上...
[折纸](#) · [阅读 722](#)

JUC 包下工具类，它的名字叫 LockSupport！你造么？

前言LockSupport 是 JUC 中常用的一个工具类，主要作用是挂起和唤醒线程。在阅读 JUC 源码中经常看到，所以很有必要了解一...
[程序员小航](#) · [阅读 373](#)

产品

[热门问答](#)

[热门专栏](#)

[热门课程](#)

[最新活动](#)

[技术圈](#)

[酷工作](#)

课程

[Java 开发课程](#)

[PHP 开发课程](#)

[Python 开发课程](#)

[前端开发课程](#)

[移动开发课程](#)

资源

[每周精选](#)

[用户排行榜](#)

[徽章](#)

[帮助中心](#)

[声望与权限](#)

[社区服务中心](#)

[建议反馈](#)

合作

[关于我们](#)

[广告投放](#)

[职位发布](#)

[讲师招募](#)

[联系我们](#)

[合作伙伴](#)

关注

[产品技术日志](#)

[社区运营日志](#)

[市场运营日志](#)

[团队日志](#)

[社区访谈](#)

条款

[服务协议](#)

[隐私政策](#)

[下载 App](#)