链滴

多线程之倒计时计数器 CountDownLatch

作者：MaidongAndYida

原文链接：https://ld246.com/article/1555142789482

来源网站：链滴

上篇回顾：多线程回环栅栏CyclicBarrier

使用场景

假设有一个列表，多选之后，点击压缩。此时后台肯定是多线程跑压缩任务，那我们就可以使用CountDownLatch来在全部压缩完成之后处理一些事情。

示例代码

```
public class testMain {

    public static void main(String[] args) {
        //初始化计数
        AtomicInteger count = new AtomicInteger(0);
        //创建10个线程
        ExecutorService es = Executors.newFixedThreadPool(10);
        //设置CountDownLatch为10
        CountDownLatch countDownLatch = new CountDownLatch(10);

        for (int i = 0; i < 10; i++) {
            es.execute(() -> {
                //每个线程累计增加1000次
                for (int j = 0; j < 1000; j++) {
                    count.addAndGet(1);
                }
                //线程最后执行CountDownLatch
                countDownLatch.countDown();
            });
        }
        try {
            countDownLatch.await();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        ;
        System.out.println(count);

    }
}
```

可以看到，CountDownLatch与CyclicBarrier非常类似，由此引出几个问题：

解释一下CountDownLatch概念?

CountDownLatch是通过一个计数器来实现的，计数器的初始值为线程的数量。每当一个线程完成了己的任务后，计数器的值就会减1。当计数器值到达0时，它表示所有的线程已经完成了任务，然后在锁上等待的线程就可以恢复执行任务。

CountDownLatch 和CyclicBarrier的不同之处?

CyclicBarrier可以重用，CountDownLatch 不可以重用。

CountDownLatch的下一步的动作实施者是主线程
CyclicBarrier的下一步动作实施者还是“其他线程”本身

原文链接：多线程之倒计时计数器 CountDownLatch