链滴

CountDownLatch 和 CyclicBarrier 的区别

作者：huihui
原文链接：https://ld246.com/article/1500975633807
来源网站：链滴

CountDownLatch是什么

CountDownLatch是在java1.5被引入的，跟它一起被引入的并发工具类还有**CyclicBarrier、Semaphore、ConcurrentHashMap和BlockingQueue**，它们都存在于java.util.concurrent包下。CountDownLatch这个类能够使**一个线程等待其他线程完成各自的工作后再执行**。例如，应用程序的主线程希在负责启动框架服务的线程已经启动所有的框架服务之后再执行。

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如何工作

CountDownLatch.java类中定义的构造函数：

//Constructs a CountDownLatch initialized with the given count.

public void CountDownLatch(int count) {...}

构造器中的计数值（count）实际上就是闭锁需要等待的线程数量。这个值只能被设置一次，而且CoutDownLatch没有提供任何机制去重新设置这个计数值。

与CountDownLatch的第一次交互是主线程等待其他线程。主线程必须在启动其他线程后立即调用**ountDownLatch.await()**方法。这样主线程的操作就会在这个方法上阻塞，直到其他线程完成各自任务。

其他N 个线程必须引用闭锁对象，因为他们需要通知CountDownLatch对象，他们已经完成了各自任务。这种通知机制是通过**CountDownLatch.countDown()**方法来完成的；每调用一次这个方，在构造函数中初始化的count值就减1。所以当N个线程都调用了这个方法，count的值等于0，然主线程就能通过await()方法，恢复执行自己的任务。

在实时系统中的使用场景

1.实现最大的并行性：有时我们想同时启动多个线程，实现最大程度的并行性。例如，我们想测试一单例类。如果我们创建一个初始计数为1的CountDownLatch，并让所有线程都在这个锁上等待，那我们可以很轻松地完成测试。我们只需调用 一次countDown()方法就可以让所有的等待线程同时恢执行。

2.开始执行前等待n个线程完成各自任务：例如应用程序启动类要确保在处理用户请求前，所有N个外系统已经启动和运行了。

3.死锁检测：一个非常方便的使用场景是，你可以使用n个线程访问共享资源，在每次测试阶段的线数目是不同的，并尝试产生死锁。

常见面试题

可以为你的下次面试准备以下一些CountDownLatch相关的问题：

解释一下CountDownLatch概念?

CountDownLatch 和CyclicBarrier的不同之处?

给出一些CountDownLatch使用的例子?

CountDownLatch 类中主要的方法?

CountDownLatch(int)

await()

await(long,TimeUnit)

CountDown()

getCount()

toString

在网上看到很多人对于CountDownLatch和CyclicBarrier的区别简单理解为**CountDownLatch是一性的，而CyclicBarrier在调用reset之后还可以继续使用。**

那如果只是这么简单的话，我觉得CyclicBarrier简单命名为ResetableCountDownLatch好了，显然是的。

我的理解是，要从他们的设计目的去看这两个类。javadoc里面的描述是这样的。

CountDownLatch: A synchronization aid that allows one or more threads to wait until a set of operations being performed in other threads completes.

CyclicBarrier : A synchronization aid that allows a set of threads to all wait for each other to re ch a common barrier point.

CountDownLatch和CyclicBarrier本质是一样的,都是在每个Thread 设置 CountDownLatch或Cyclic arrier 的屏障点point， 每个Thread 计入的Condition 中队列， 执行到屏障点point时候，回调 Coun DownLatch或CyclicBarrier 的await方法，在Condition 中队列遍历是否所有的Thread 都已经执行屏障点point。但是并没有释放Thread ， Thread 一致是运行状态（可以理解成时等待其他Thread全完成屏障点point）。

区别是:

每个Thread 到达 CountDownLatch的屏障点point 可以调用 countDown（）计数-1，到全部为计0的时候，全部其他Thread通过，继续执行或结束。

所有到达 CyclicBarrier 的屏障点point 时候，该屏障点point 结束，继续执行。如果不能全部到达（断、失败或者超时等原因），设置 BrokenBarrierException引起所有调用await方法的“Thread”全报错，全部要么全不 (all-or-none) 的破坏模式。

reset也是抛出BrokenBarrierException。

转自：http://www.cnblogs.com/lsx1993/p/4631733.html

【CountDownLatch、CyclicBarrier和Semaphore】

http://www.cnblogs.com/dolphin0520/p/3920397.html

【CountDownLatch同步工具类】

http://www.importnew.com/15731.html