http://www.cnblogs.com/shiyanch/archive/2011/04/04/2005233.html

java.util.concurrent.CountDownLatch,

一个同步辅助类，在完成一组正在其他线程中执行的操作之前，它允许一个或多个线程一直等待。

用给定的计数初始化 CountDownLatch。当调用countDown()方法时，所有在当前计数到达0之前，await 方法会一直受阻塞。计数到0时，会释放所有等待的线程，await的所有后续调用都将立即返回。

CountDownLatch如其所写，是一个倒计数的锁存器，当计数减至0时触发特定的事件。利用这种特性，可以让主线程等待子线程的结束。下面以一个模拟运动员比赛的例子加以说明。

```
package com.hexun.hxt.admin;
import java.util.concurrent.CountDownLatch;
import java.util.concurrent.Executor;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

public class CountDownLatchDemo {
    private static final int PLAYER_AMOUNT = 5;
    public CountDownLatchDemo() {
   }
    public static void main(String[] args) {

        //对于每位运动员，CountDownLatch减1后即结束比赛
        CountDownLatch begin = new CountDownLatch(1);

        //对于整个比赛，所有运动员结束后才算结束
        CountDownLatch end = new CountDownLatch(PLAYER_AMOUNT);

        Player[] plays = new Player[PLAYER_AMOUNT];

         for(int i=0;i<PLAYER_AMOUNT;i++) {
             plays[i] = new Player(i+1,begin,end);
         }
```

```
        //设置特定的线程池，大小为5
        ExecutorService exe =
Executors.newFixedThreadPool(PLAYER_AMOUNT);

        //分配5个线程到线程池执行,[执行一行代码后发现此时begin处于await状态,
        // 需要等待调用begin.countDown(),线程才能往下执行]
         for(Player p:plays)
           exe.execute(p);
        System.out.println("Race begins!");

        //递减锁存器的计数,如果计数到达零,则释放所有等待的线程
        //begin只有1, 所以递减一次就能达到0
        begin.countDown();
         try{
           end.await(); //等待end状态变为0，即为比赛结束
            }catch (InterruptedException e) {
           e.printStackTrace();
        }finally{
           System.out.println("Race ends!");
        }
        exe.shutdown();
    }
}

class Player implements Runnable {

    private int id;
    private CountDownLatch begin;
    private CountDownLatch end;
    public Player(int i, CountDownLatch begin, CountDownLatch end) {
        super();
        this.id = i;
        this.begin = begin;
        this.end = end;
     }

     @Override
      public void run() {
        try{
```

```
                begin.await(); //等待begin的状态为0
                Thread.sleep((long)(Math.random()*100)); //随机分配时间，即运动员
完成时间
                System.out.println("Play"+id+" arrived.");
            }catch (InterruptedException e) {
                e.printStackTrace();
            }finally{
                //每个线程都会使end的状态减1,五个线程使其最终减至0
                end.countDown();
        }
    }
}
```