**Java并发编程 之 Condition与ReentrantLock的使用**

先来看一道笔试题（迅雷的笔试题）：编写一个程序，开启3个线程，这3个线程的ID分别为A、B、C，每个线程将自己的ID在屏幕上打印10遍，要求输出结果必须按ABC的顺序显示；如：ABCABC….依次递推。

很明显考虑这题的时候，需要想到使线程之间能够的进行消息传递。这题如果想要用Object自带的wait和notify，相比于Condition感觉会更麻烦。wait和notify更容易实现两个线程之间的通信（生产和消费），而Condition用于多个线程之间也可以处理的游刃有余。

这篇只是针对Condition和ReentrantLock的如何使用，实现原理在下一篇博客

import java.util.concurrent.locks.Condition;

import java.util.concurrent.locks.ReentrantLock;

/\*\*

\* Created by jintx on 2017/9/29.

\*/

public class CondtionTest {

public static ReentrantLock lock = new ReentrantLock();

public static Condition conditionA = lock.newCondition();

public static Condition conditionB = lock.newCondition();

public static Condition conditionC = lock.newCondition();

public static int index = 0;

public static void main(String[] args){

ThreadA threadA = new ThreadA();

ThreadB threadB = new ThreadB();

ThreadC threadC = new ThreadC();

threadA.start();

threadB.start();

threadC.start();

}

public static class ThreadA extends Thread{

@Override

public void run(){

try{

lock.lock();

for(int i = 1 ; i <= 5; i++) {

for (int j = 1; j <= 5; j++) {

/\*不需要担心++index同步的问题，因为我们已经保证了每次只能有一个线程在++index\*/

System.out.println("A进程输出" + " : " + ++index);

}

conditionB.signal();//第一次调用该方法没什么意义，因为线程B还没等待队列中

conditionA.await();//本质是释放了对lock的同步控制

}

}catch (Exception e){

e.printStackTrace();

}finally {

lock.unlock();

}

}

}

public static class ThreadB extends Thread{

@Override

public void run(){

try{

lock.lock();

for(int i = 1 ; i <= 5; i++) {

for (int j = 1; j <= 5; j++) {

System.out.println("B进程输出" + " : " + ++index);

}

conditionC.signal();

conditionB.await();

}

}catch (Exception e){

e.printStackTrace();

}finally {

lock.unlock();

}

}

}

public static class ThreadC extends Thread{

@Override

public void run(){

try{

lock.lock();

for(int i = 1 ; i <= 5; i++) {

for (int j = 1; j <= 5; j++) {

System.out.println("C进程输出" + " : " + ++index);

}

conditionA.signal();

conditionC.await();

}

}catch (Exception e){

e.printStackTrace();

}finally {

lock.unlock();

}

}

}

}

---------------------

作者：嘎里给给

来源：CSDN

原文：https://blog.csdn.net/MakeContral/article/details/78133717

版权声明：本文为博主原创文章，转载请附上博文链接！