java认证辅导:Java多线程问题及处理(笔记)Java认证考试 PDF 转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/601/2021_2022_java_E8_AE_A4_E8_AF_81_c104_601946.htm 1.死锁 多线程编程在实际的网络程序开发中，在客户端程序实现中使用的比较简单，但是在服务器端程序实现中却不仅是大量使用，而且会出现比客户端更多的问题。 另外一个容易在服务器端出现的多线程问题是死锁。死锁指两个或两个以上的线程为了使用某个临界资源而无限制的等待下去。还是以前面卫生间的例子来说明死锁，例如两个人都同时到达卫生间，而且两个人都比较礼貌，第一个人和第二个人说：你先吧，第二个人和第一个人说：你先吧。这两个人就这样一直在互相礼让，谁也不进入，这种现象就是死锁。这里的两个人就好比是线程，而卫生间在这里就是临界资源，而由于这两个线程在一直谦让，谁也不使用临界资源。 死锁不仅使程序无法达到预期实现的功能，而且浪费系统的资源，所以在服务器端程序中危害比较大，在实际的服务器端程序开发中，需要注意避免死锁。 而死锁的检测比较麻烦，而且不一定每次都出现，这就需要在测试服务器端程序时，有足够的耐心，仔细观察程序执行时的性能检测，如果发现执行的性能显著降低，则很可能是发生了死锁，然后再具体的查找死锁出现的原因，并解决死锁的问题。 死锁出现的最本质原因还是逻辑处理不够严谨，在考虑时不是很周全，所以一般需要修改程序逻辑才能够很好的解决死锁。 2. 线程优先级 在日常生活中，例如火车售票窗口等经常可以看到“ XXX优先 ”，那么多线程编程中每个线程是否也可以设置优先级呢？ 在多线程编程中，支持为每个

线程设置优先级。优先级高的线程在排队执行时会获得更多的CPU执行时间，得到更快的响应。在实际程序中，可以根据逻辑的需要，将需要得到及时处理的线程设置成较高的优先级，而把对时间要求不高的线程设置成比较低的优先级。在Thread类中，总计规定了三个优先级，分别为：l MAX_PRIORITY最高优先级 l NORM_PRIORITY普通优先级，也是默认优先级 l MIN_PRIORITY最低优先级 在前面创建的线程对象中，由于没有设置线程的优先级，则线程默认的优先级是NORM_PRIORITY，在实际使用时，也可以根据需要使用Thread类中的setPriority方法设置线程的优先级，该方法的声明为： public final void setPriority(int newPriority) 假设t是一个初始化过的线程对象，需要设置t的优先级为最高，则实现的代码为： t. setPriority(Thread. MAX_PRIORITY). 这样，在该线程执行时将获得更多的执行机会，也就是优先执行。如果由于安全等原因，不允许设置线程的优先级，则会抛出SecurityException异常。 下面使用一个简单的输出数字的线程演示线程优先级的使用，实现的示例代码如下： package priority. /** * 测试线程优先级 */ public class TestPriority { public static void main(String[] args) { PrintNumberThread p1 = new PrintNumberThread("高优先级"). PrintNumberThread p2 = new PrintNumberThread("普通优先级"). PrintNumberThread p3 = new PrintNumberThread("低优先级"). p1.setPriority(Thread.MAX_PRIORITY). p2.setPriority(Thread.NORM_PRIORITY). p3.setPriority(Thread.MIN_PRIORITY). p1.start(). p2.start(). p3.start(). } } package priority. /** * 输出数字的线程 */ public class

PrintNumberThread extends Thread { String name. public PrintNumberThread(String name){ this.name = name. } public void run(){ try{ for(int i = 0.i &lt. 10.i){ System.out.println(name ":" i). } }catch(Exception e){} } } 程序的一种执行结果为： 高优先级：0 高优先级：1 高优先级：2 普通优先级：0 高优先级：3 普通优先级：1 高优先级：4 普通优先级：2 高优先级：5 高优先级：6 高优先级：7 高优先级：8 高优先级：9 普通优先级：3 普通优先级：4 普通优先级：5 普通优先级：6 普通优先级：7 普通优先级：8 普通优先级：9 低优先级：0 低优先级：1 低优先级：2 低优先级：3 低优先级：4 低优先级：5 低优先级：6 低优先级：7 低优先级：8 低优先级：9 在该示例程序，PrintNumberThread线程实现的功能是输出数字，每次数字输出之间没有设置时间延迟，在测试类TestPriority中创建三个PrintNumberThread类型的线程对象，然后分别设置线程优先级是最高、普通和最低，接着启动线程执行程序。从执行结果可以看出高优先级的线程获得了更多的执行时间，首先执行完成，而低优先级的线程由于优先级较低，所以最后一个执行结束。 其实，对于线程优先级的管理主要由系统的线程调度实现，较高优先级的线程优先执行，所以可以通过设置线程的优先级影响线程的执行。 3 总结 关于多线程的基础知识就介绍这么多，在本章中介绍了线程的概念、线程的实现方式以及使用多线程时会遇到的问题以及解决办法，而需要建立多线程的概念，也就是并发编程的概念还需要进行比较多的练习，理解多线程的概念并熟悉多线程的编程。 而关于多线程编程的高级知识，如线程组等则可以在熟悉了线程的基本概念以后再进行更加深入的学习。 更多优质资料尽在