

张叔的博客

RSS订阅

0

目录

收藏

评论

微信

微博

QQ

个人资料



弓长步又

关注

原创
27

粉丝
20

喜欢
7

评论
53

等级： 博客 4

访问：13万+

积分：1010

排名：5万+

最新文章

AnyChat实例

ActiveMQ实例

列表数据多行删除

bootstrap-datepicker 与bootstrapValidator同时使用时，选择日期后，无法正常触发校验

JsperReport导出PDF报表中文不显示

个人分类

多线程 5篇

IO 5篇

集合 2篇

反射 1篇

bootstrap 5篇

展开

归档

2017年8月 1篇

2017年7月 1篇

2016年8月 1篇

2016年6月 3篇

2016年4月 1篇

展开

热门文章

bootstrap-datepicker限定可选时间范围
阅读量：44376

zTree实现节点修改的实时刷新

阅读量：18688

动态添加表单元素，并使用bootstrapValidator插件进行动态添加校验

阅读量：13842

bootstrap-datepicker 与bootstrapValidator同时使用时，选择日期后，无法正常触发校

阅读量：10937

最新评论

bootstrap-datepic...

qq_36102602：[reply]maryword[/reply] 设置startView: 2,//月视图 ...

动态添加表单元素，并使用boots...

weixin_40475396：动态添加了HTML的input标签，但就是没法自动验证，官方API都看完了，还是没太明白，刚好找到...

bootstrap-datepic...

cainiao1994：[reply]cainiao1994[/reply] 1

bootstrap-datepic...

cainiao1994：1

bootstrapValidato...

qq_35006986：表单验证的是name=""",而不是id="""...

联系我们



请扫描二维码联系客服

webmaster@csdn.net

400-660-0108

QQ客服  客服论坛

关于 招聘 广告服务  百度

©1999-2018 CSDN版权所有

京ICP证09002463号

经营性网站备案信息

网络110报警服务

中国互联网举报中心

北京互联网违法和不良信息举报中心

原

多线程中多生产多消费问题

2015年10月06日 17:35:32

阅读数：1515

一、多生产多消费实例

多线程中多生产多消费问题的解决有助于更好的理解多线程的使用。现在通过实例来说明多线程中需要注意的地方。

```
[java]
1.  /**
2.   * @author zqx 描述资源
3.   */
4.  public class Resource {
5.      private String name;
6.      private int count = 1;
7.      // 定义flag标记
8.      private boolean flag;
9.
10.     // 提供给消费者获取商品的方法
11.
12.     public synchronized void get() {
13.         /*用while循环判断是为了解决if判断中出现的重复生产，重复消费的问题，经过分析，发现如果使用if判断，被唤醒的线程没有判断标记
14.         * 就开始工作(生产or消费)了。导致了重复的生产和消费的发生。使用while后，所有被唤醒的线程都会首先进行标记的判断。
15.         *
16.         * */
17.         while (!flag) {
18.             try {
19.                 this.wait();
20.             } catch (InterruptedException e) {
21.                 e.printStackTrace();
22.             }
23.         }
```

```
27.         /*
28.          * 唤醒等待线程,用notifyAll()方法是为了解决notify()方法时，本方线程在唤醒时，又唤醒了本文线程，而本文线程循环判断标记
29.          * 又继续等待，而导致所有的线程都等待了，这样就会导致死锁。使用notifyAll()唤醒所有线程，经过循环判断标记
30.          * ，对方线程执行，本文线程等待。
31.          */
32.         this.notifyAll();
33.     }
34.
35.     // 提供给生产者生产商品的方法
36.     public synchronized void set(String name) {
37.         // 如果flag为true,则说明已经有面包，执行等待，如果flag为false，执行生产。
38.         while (flag) {
39.             try {
40.                 this.wait();
41.             } catch (InterruptedException e) {
42.                 e.printStackTrace();
43.             }
44.         }
45.         this.name = name + "--" + count;
46.         count++;
47.         System.out.println(Thread.currentThread().getName() + ".....生产者....." + this.name);
48.         // 生产完毕，将标记置为true，并唤醒消费者
49.         flag = true;
50.         // 唤醒所有等待线程
51.         this.notifyAll();
52.
53.     }
54. }
```

[java]

```
1.  /**
2.   * @author zqx 生产者任务
3.   */
4.  public class Producer implements Runnable {
5.      private Resource resource;
6.
7.      Producer(Resource resource) {
8.          this.resource = resource;
9.      }
10.
11.      @Override
12.      public void run() {
13.          while (true)
14.              resource.set("面包");
15.      }
16.  }
```

[java]

```
1.  /**
2.   * @author zqx 消费者任务
3.   */
4.  public class Customer implements Runnable {
5.      private Resource resource;
6.
7.      Customer(Resource resource) {
8.          this.resource = resource;
9.      }
10.
11.      @Override
12.      public void run() {
13.          while (true)
14.              resource.get();
15.      }
16.  }
```

[java]

```
1.  /**
2.   * @author zqx
3.   * 测试类
4.   */
5.  public class ProducerCustomer {
6.      public static void main(String[] args) {
7.          //1、创建资源对象
8.          Resource resource = new Resource();
9.          //2、创建两个线程，构造函数的方式初始化两个线程，保证操作同一资源
10.         Producer producer = new Producer(resource);
11.         Customer customer = new Customer(resource);
12.         //3、创建线程，创建多个生产者和消费者
13.         Thread t0 = new Thread(producer);
14.         Thread t1 = new Thread(producer);
15.         Thread t2 = new Thread(customer);
16.         Thread t3 = new Thread(customer);
17.         //4、开启线程
18.         t0.start();
19.         t1.start();
20.         t2.start();
21.         t3.start();
22.     }
23. }
```

总结：上述实例已经实现了多生产多消费，但是效率有点低，因为notifyAll也唤醒了本文线程，做了一些不必要的判断，那么有什么更好的方法能够解决上述问题呢？JDK1.5以后出现的Lock接口和Condition接口解决了这一问题

二、多线程Lock接口和Condition接口

1、作用：

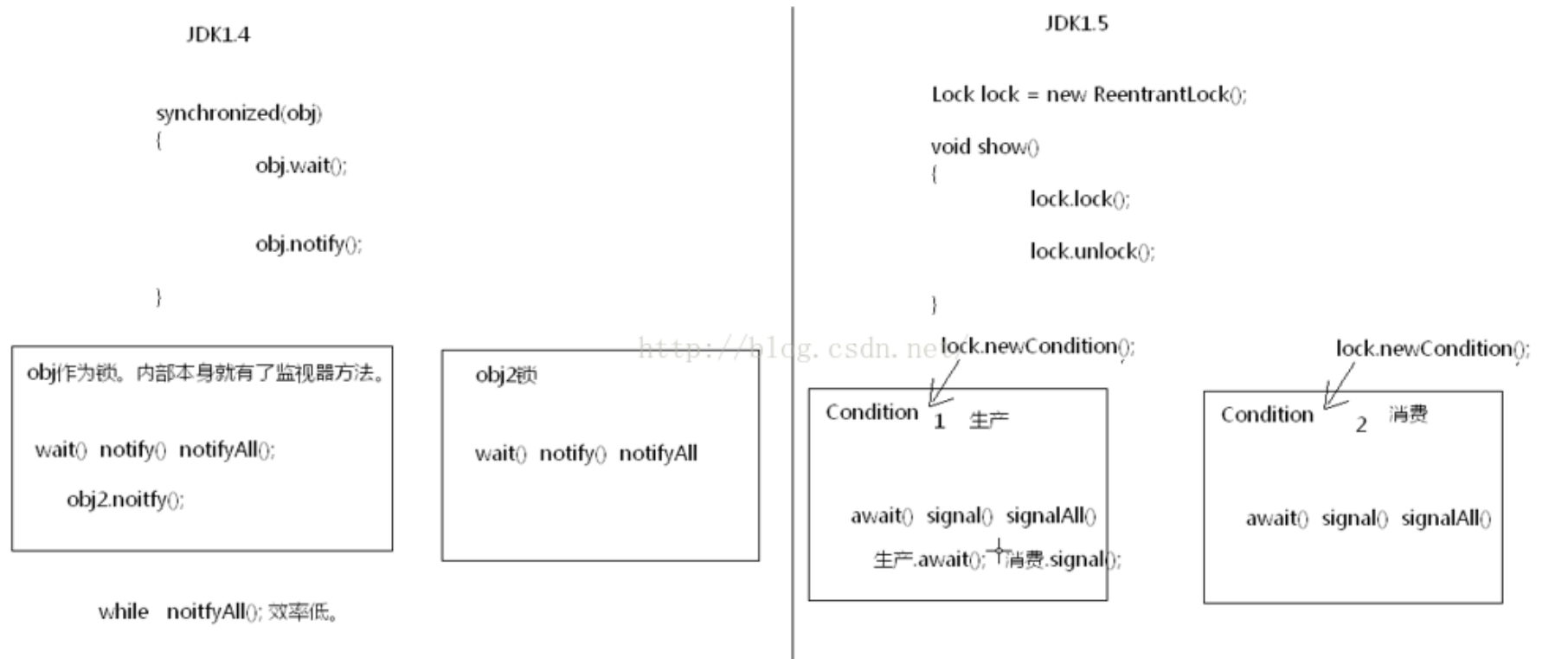
同步函数和同步代码块的锁操作是隐式的，JDK1.5 Lock接口，按照面向对象的思想，将锁单独封装成一个对象，并提供了对锁的显式操作。Lock接口就是同步的替代。condition接口替代了Object中的监视器方法，以前监视器方法封装到Condition对象中。

2、常用方法

- 1、Lock接口中的方法：
lock()：获取锁
unlock()：释放锁
newCondition()：返回绑定到此Lock对象的Condition对象。
- 2、Condition接口中的方法：
await()：使当前线程在接到信号或被中断之前一直处于等待状态。
signal()：唤醒一个等待线程
signalAll()：唤醒所有等待线程。

3、和同步进行比较

JDK1.4中同步中监视器方法都和锁相绑定，而且一个锁上就只能有一组监视器，如果要想实现唤醒对方的一个线程，就要用到锁的嵌套，容易发生死锁。JDK1.5中监视器方法不再和锁绑定，把监视器方法单独封装成对象，而对象可以和锁绑定，一个锁上可以支持多个相关的condition对象。可以创建两个condion对象，一个用来监视生产，一个用来监视消费，由于是同一个锁，不会用到锁的嵌套，也就不会造成同步中的死锁问题，对比简图如下：



4、实例

和上述例子唯一不同的就是描述资源的类。其他都相同。用lock和condition来实现多生产多消费问题，解决了效率低的问题。

```
[java]
1. /**
2.  * @author zqx 描述资源
3.  */
4. public class Resource {
5.     private String name;
6.     private int count = 1;
7.     //1.创建新Lock
8.     private Lock lock = new ReentrantLock();
9.     //2.创建和Lock绑定的监视器对象
10.    //生产者监视器
11.    private Condition pro_condition = lock.newCondition();
12.    //消费者监视器
13.    private Condition cus_condition = lock.newCondition();
14.    // 定义flag标记
15.    private boolean flag;
16.
17.    // 提供给消费者获取商品的方法
18.    public void get() {
19.        // 获取锁
20.        lock.lock();
21.        try {
22.            while (!flag) {
23.                try {
24.                    cus_condition.await();
```

```
28.         }
29.         System.out.println(Thread.currentThread().getName() + ".....消费者....." + this.name);
30.         // 将标记置为false
31.         flag = false;
32.         //消费完毕，要唤醒一个生产者来生产
33.         pro_condition.signal();
34.     } finally {
35.         // 释放锁
36.         lock.unlock();
37.     }
38. }
39.
40. // 提供给生产者生产商品的方法
41. public synchronized void set(String name) {
42.     // 获取锁
43.     lock.lock();
44.     try {
45.         // 如果flag为true,则说明已经有面包，执行等待，如果flag为false，执行生产。
46.         while (flag) {
47.             try {
48.                 pro_condition.await();
49.             } catch (InterruptedException e) {
50.                 e.printStackTrace();
51.             }
52.         }
53.         this.name = name + "--" + count;
54.         count++;
55.         System.out.println(Thread.currentThread().getName() + ".....生产者....." + this.name);
56.         // 生产完毕，将标记置为true，并唤醒消费者
57.         flag = true;
58.         // 生产完毕，应该唤醒一个消费者来消费
59.         cus_condition.signal();
60.     } finally {
61.         // 释放锁
62.         lock.unlock();
63.     }
64. }
65. }
```

三、实际开发中相关的实例

以上实例中，一次只能生产一个，但实际开发中可以要连续生产多个，连续消费多个。JDK1.5中有相关实例，现在一起来感受下吧。以下这段代码相当于描述资源类。有兴趣可以研究下。

```
[java]
1. class BoundedBuffer {
2.     final Lock lock = new ReentrantLock();
3.     final Condition notFull  = lock.newCondition();
4.     final Condition notEmpty = lock.newCondition();
5.
6.     final Object[] items = new Object[100];
7.     int putptr, takeptr, count;
8.
9.     public void put(Object x) throws InterruptedException {
10.         lock.lock();
11.         try {
12.             while (count == items.length)
13.                 notFull.await();
14.             items[putptr] = x;
15.             if (++putptr == items.length) putptr = 0;
16.             ++count;
17.             notEmpty.signal();
18.         } finally {
19.             lock.unlock();
20.         }
21.     }
22.
23.     public Object take() throws InterruptedException {
24.         lock.lock();
25.         try {
26.             while (count == 0)
27.                 notEmpty.await();
28.             Object x = items[takeptr];
29.             if (++takeptr == items.length) takeptr = 0;
30.             --count;
31.             notFull.signal();
32.             return x;
33.         } finally {
34.             lock.unlock();
35.         }
36.     }
37. }
```

[查看更多>>](#)

想对作者说点什么？

我来说一句

多线程_生产者与消费者

创建一个线程有以下两种方式：①线程类继承Thread，重写run()方法，在run()方法中完成此线程所要完成的工作，直接创建线程类的对象，然后调用start()方法启动线程，默认调用run...

 winy_lm 2015-11-28 23:35:10 阅读数：2239

关于java多线程浅析一：简单实现生产消费模式

关于什么是线程，这里就不过多介绍了。但为什么要用多线程呢？原因无他，就是希望更好的利用CPU资源。当然，多线程在模拟很多实际场景下，也是一把利器，就比如生产-消费模式，使用不同的线程来充当生产者和消费...

zhangjunfei12103323 2017-05-11 15:01:05 阅读数：1479

海参的功效，看完你就知道为啥要吃海参了!!

悦盛·顶新

多线程之生产者消费者问题

在多线程编程过程中，为了保证是原子操作，必须处理好线程之间的同步和互斥，生产者消费者问题即是线程间同步和互斥的经典例子。生产者消费者问题的描述：生产者负责生产“产品”，消费者负责消费“产品”，...

miraclewgf 2015-09-09 21:35:36 阅读数：994

线程同步-生产者消费者问题

在进行多线程编程时，难免还要碰到两个问题，那就线程间的互斥与同步：线程同步是指线程之间所具有的一种制约关系，一个线程的执行依赖另一个线程的消息，当它没有得到另一个线程的消息时应等待，直到消息到达时才...

big_bit 2016-05-09 21:26:53 阅读数：8243

java多线程之生产消费模式

/*@author shijin * 生产者与消费者模型中，要保证以下几点： * 1 同一时间内只能有一个生产者生产 生产方法加锁synchronized * 2 同一时间内只能有一...

 BloodyDmusic 2016-07-22 13:01:19 阅读数：961

OS: 生产者消费者问题(多线程+互斥量+条件变量)

一. 引子 用多进程解决生产着消费者问题之后,再尝试多线程方法,才知道多线程多么地方便。多线程方案的易用性,一方面得益于强大的条件变量。赞,太好用了! 二. 思路 互斥量实际上相当于二元信号量, ...

 yaozhiyi 2012-05-13 22:45:03 阅读数：4297

小程序开发，腾讯找来硅谷“独角兽”搞事情

席位有限，立即行动！抢先掌握稀缺技术，成为抢手人才



操作系统中生产者/消费者问题(一)

生产者/消费者是个典型的进程同步问题，只有生产者生产出来东西了，消费者才能消费 也就是说，生产者生成出来一个东西，然后通知消费者，（东西好了，快来拿吧），而消费者则查看缓冲器里面有没有东西，若没有则会...

 best_fiends_zxh 2016-09-29 19:38:03 阅读数：1778

生产者 - 消费者问题

作为操作系统最精华的部分，生产者消费者问题无疑是经典问题中的经典问题。今天终于有空能好好研究一下这类问题了，不对之处还望大家指正。首先，讲解经

生产者与消费者一对一，一对多，多对多

生产者与消费者问题涉及对象 生产者 消费者 生产与消费对象（以下简称目标对象，通常是通过某个容器存放如：ArrayList）类图表示：生产者与消费者问题主要解决的是同步问题：当目标对象已经为...

 qq_24028753 2017-09-02 16:49:25 阅读数：1137

java 实现多生产者多消费问题

2013年01月04日 2KB

下载



多生产多消费的运行图

2014年05月24日 1.29MB

下载



多生产者多消费者问题（Lock接口、Condition接口）

在多生产者多消费者问题中，我们通过while判断和notifyAll()全唤醒方法解决了问题，但是notifyAll()同时也带来了弊端，它要唤醒所有的被等待的线程，意味着既唤醒了对方，也唤醒了本方，...

 syf1970 2016-05-07 17:19:37 阅读数：557

Java中多线程的多生产多消费问题的解决方案

/*创建线程的第二种方式1、实现Runnable接口2、覆盖run方法3、通过Thread类创建线程对象4、将Runnable接口的子类对象作为实参传递给Thread类中的构造函数5、调用start方...

 wangchunlei123 2014-04-16 14:31:31 阅读数：1205

多线程应用例子2—消费者，生产者，多个缓冲区

然后再对这个简单生产者消费者问题加大难度。将消费者改成2个，缓冲池改成拥有4个缓冲区的大缓冲池。 如何来思考了这个问题了？首先根据上面分析的二点，可以知道生产者和消费者由一个变成多个的影...

 xietingcandice 2014-08-19 21:14:03 阅读数：1413

为什么互联网公司都在做小程序开发？

微信请来了硅谷大佬一起开发了门小程序课，了解一下



Docker入门与实战讲解

简述 Docker 是一个开源的应用容器引擎，让开发者可以打包他们的应用以及依赖包到一个可移植的容器中，然后发布到任何流行的 Linux 机器上，也可以实现虚拟化。容器是完全使用沙箱机制，相互之间不...

 relax_hb 2017-04-08 14:53:41 阅读数：18397

Docker入门实战-SSH连接docker容器

什么是DockerDocker 是一个开源项目，诞生于 2013 年初，最初是 dotCloud 公司内部的一个业余项目。它基于 Google 公司推出的 Go 语言实现。项目后来加入了 Linux...

 qq626387 2016-01-12 20:48:53 阅读数：24736

使用gradle对vertx工程的多环境配置和打包

公司日前使用vertx构建rest服务，vertx工程跟普通的工程没有什么区别。我们使用gradle进行构建。后来随着部署发布的频繁，在打包之后再进行配置的修改，会出现很多的问题，很容易出错，然后排查...

 xxssyyyyssxx 2017-06-16 10:21:03 阅读数：1069

宋宝华：Docker 最初的2小时(Docker从入门到入门)

最初的2小时，你会爱上Docker，对原理和使用流程有个最基本的理解，避免满世界无头苍蝇式找资料。本人反对暴风骤雨式多管齐下狂轰滥炸的学习方式，提倡迭代学习法，就是先知道怎么玩，有个感性认识，再深入学...

 21cnbao 2017-02-21 09:42:31 阅读数：45103

Vertx集群部署实例

Vertx集群部署，官方介绍了几种方式，其中有一种方式是使用HazelCast来实现的，下面介绍如何通过HazelCast来实现Vertx集群部署。 HazelCast默认采...

 feinifi 2017-02-12 23:04:31 阅读数：3552

上一篇文章讲述了如何部署kafka集群，而这篇文章则来探讨一下如何使用多线程消费，提高消费能力，保障数据的时效性。而实现多线程消费其实很简单，只需要三步即可： 一：kafka集群配置多线程消费，说白了...

 u011622226 2017-05-18 21:19:51 阅读数：4083

多线程的生产者和消费者问题

多线程的生产者和消费者问题是比较经典的多线程问题，如果知道编码解决生产者和消费者问题，那么对于多线程应该基本算掌握了。我不知道大家的生产者和消费者问题是怎么样的（应该有几个版本吧），这里我说下我的生产...

 YuZhiHui_No1 2015-06-12 14:51:55 阅读数：2563

java 多线程并发系列之 生产者消费者模式的两种实现

生产者消费者模式是并发、多线程编程中经典的设计模式，生产者和消费者通过分离的执行工作解耦，简化了开发模式，生产者和消费者可以以不同的速度生产和消费数据。 真实世界中的生产者消费者模式...

 yujin753 2015-05-14 16:59:48 阅读数：12503

C++多线程学习：生产者消费者问题

多线程相关知识点： C++11 线程库：http://zh.cppreference.com/w/cpp/thread 互斥量和锁 std::unique_lock::lock 和 std::uni...

 quzhongxin 2015-08-19 20:56:01 阅读数：6946

Java多线程系列-之生产消费者问题

来源：博客园 | 作者： 如果天空不死概要本章，会对“生产/消费者问题”进行讨论。涉及到的内容包括：1. 生产/消费者模型2. 生产/消费者实现1. 生产/消费者模型生产/消费者问题是个非常典型的...

 b644ROfP20z37485O35M 2018-04-10 00:00:00 阅读数：18

为什么程序猿都在学习微信小程序开发

小程序开发，腾讯找来了硅谷“独角兽”一起搞事



多线程的经典案例(生产消费问题)

多线程的经典案例(生产消费问题)未完待续。。

 weimeig 2018-03-11 00:29:40 阅读数：84

多线程(生产者-消费者问题)

流程： 1：生产者生产商品，如果有商品，不生产，唤醒消费者消费商品；如果没有商品，生产者生产商品，并唤醒消费者消费商品； 2：消费者消费商品，如果有商品，消费商品；如果没有商品，唤醒生产者生产商品...

 zhou920786312 2017-04-07 01:12:31 阅读数：105

多线程解决生产者与消费者问题

生产者消费者问题是一个很有名的线程同步问题，以前学操作系统时，没怎么搞懂，直到现在学java学到多线程这一块才搞懂。该问题描述的是两个共享固定大小的缓冲区的线程问题。生产者的主要作用是生成一定量的数据...

 weixian52034 2016-09-03 23:07:06 阅读数：403

线程并发控制condition互斥量 多线程写的:生产者、消费者问题

线程并发控制condition 互斥量 多线程写的:生产者、消费者问题 综合评分:0 收藏评论举报 所需: 3积分/C币 下载个数: 2 开通VIP 立即下载 ...

学院 2018年04月23日 00:00

Java多线程模拟实现消费者生产者问题

```
/** * @author Sun 生产者消费者模型 */ public class MultiThreading {     public MultiThreading() { ...
```

 u012835905 2016-03-31 16:39:35 阅读数：355

秒杀多线程第十篇 生产者消费者问题

继经典线程同步问题之后，我们来看看生产者消费者问题及读者写者问题。生产者消费者问题是一个著名的线程同步问题，该问题描述如下：有一个生产者在生产产品，这些产品将提供给若干个消费者去消费，为了使生产者和消...

 liujiayu2 2015-06-08 16:22:59 阅读数：719

多线程之生产者消费者模型

Kafka多线程生产消费

一、kafka生产者 kafka目前在0.9版本后采用java版本实现，生产者KafkaProducer是线程安全对象，所以我们建议KafkaProducer采用单例模式,多个线程共享一个...

 charry_a

2018-03-20 10:00:35

阅读数：78

Java多线程与并发(五)之生产者与消费者案例

android培训——我的java笔记，期待与您交流！生产者与消费者应用案例 多线程的开发中有一个经典的操作案例，就是生产者-消费者，生产者不断生产产品，消费者不断取走产品。 例如： 饭店里...

 u013144863

2016-05-13 09:22:14

阅读数：2939

多线程实验_多生产者多消费者操作一个栈list

//只是很简单的实现了数据集，数据集的操作服务类，多线程模拟多生产者和多消费者。最后一个测试类。 多生产者多消费者。 ...

 JQ_AK47

2016-06-05 20:35:06

阅读数：1086

为什么互联网公司都在做小程序开发？

微信请来了硅谷大佬一起开发了门小程序课，了解一下



C#多线程--生产者和消费者

http://blog.sina.com.cn/s/blog_4ca200e20100oi10.html （ 1 ） Consumer表示消费类，其中定义了一个Array...

 zunguitiancheng

2015-02-28 16:41:01

阅读数：1995

Linux C/C++多线程学习：生产者消费者问题

生产者消费者问题 多个生产者和多个生产者的问题。生产者不断的向仓库放入产品，消费者不断的从仓库取出产品，仓库的容量是有限的。因此，当仓库处于满状态时，生产者必须等待消费者取出 1 个或多个产品后才能...

 ywcpig

2016-09-20 18:55:17

阅读数：1919

Java 多线程 生产者和消费者 队列

wait()和notifyAll()方法以一种非常低级的方式解决了任务互操作问题，即每次交互时都握手。在许多情况下，你可以瞄向更高的抽象级别，使用同步队列来解决任务协作问题，同步队列在任何时刻都只允许...

 liang_henry

2016-12-23 15:07:18

阅读数：926

多线程实现生产者消费者问题 详细注释 事件+临界区 信号量+临界区2种方法

生产者消费者问题 这个我就不解释了 应该都dong

 hnust_xiehonghao

2014-08-03 11:25:33

阅读数：2104

C++11多线程(十六):实战-生产者消费者模型

参考链接：http://www.cnblogs.com/haippy/p/3252092.html 不错的博客 目录 1.单生产者-单消费者模型 2.单生产者-多消费者模型 3.多生产...

 ceasadan

2016-01-06 15:57:14

阅读数：924

Java多线程六:生产者和消费者模型(多对多)

多线程协同任务之:生产者和消费者模型介绍在上一篇中介绍了多线程的等待和唤醒机制，这一篇我们就实践一下，多个生产者和多个消费模型。假如有一个资源类Resouce，其有一个属性name属性和我们赋予的一个...

 nicewuranran

2016-08-10 23:00:22

阅读数：560

java多线程-专题-聊聊并发（十）生产者消费者模式

本文首发于InfoQ 作者：方腾飞 校对：张龙 在并发编程中使用生产者和消费者模式能够解决绝大多数并发问题。该模式通过平衡生产线程和消费线程的工作能力来提高程序的整体处理数据的速度。 为...

 L25000

2015-07-27 10:46:22

阅读数：1162

【Linux】线程总结：线程同步 -互斥锁，条件变量，信号量实现多生产者多消费者模型

学习环境： Centos6.5 Linux 内核 2.6 Linux线程部分总结分为两部分：（ 1 ）线程的使用 ，（ 2 ）线程的同步与互斥。 第一部分线程的使用主要介绍，线程的概念，创建线程...

生产消费模式之数据结构策略

代写

 aazhoukeaa 2016-10-31 10:43:09 阅读数：197

(生产者消费者) 线程池、多线程读写线程

接上篇 非阻塞队列ConcurrentLinkedQueue之容器初步学习地 址: http://blog.csdn.net/RD_moon/article/details/77939147如图所示： ...

 RD_moon 2017-09-20 01:05:30 阅读数：280

小程序开发，腾讯找来硅谷“独角兽”搞事情

席位有限，立即行动！抢先掌握稀缺技术，成为抢手人才



Java多线程之生产者消费者问题<三>：使用阻塞队列更优雅地解决生产者消费者问题

前一篇文章讲了如何使用java5中的重入锁和条件变量优雅地解决生产者消费者问题，本文将继续探究java并发包（ concurrent ），寻求更好的解决方案。 java并发包中提供了阻塞队列（ Block...

 feichenwangyalin 2016-03-17 14:18:21 阅读数：2074

java多线程--ReentrantLock实现生产者与消费者模式

一.本例实现 ：一对一交替打印， 一.生产者逻辑 ：每次只允许一个生产者来进行生产操作（生产者之间互斥访问仓库），必须等消费者取走数据之后，才能进行下一次的生产 二.消费者逻辑 ：每次只允许一个消...

 d06110902002 2017-03-26 21:25:07 阅读数：336

Java多线程 多个生产者和多个消费者实现同步 jdk1.4

程序说明：2个生产者，2个消费者， 生产一个商品，消费一个商品(商品有标号) 特殊：这里生产者、消费者都有多个， 1. 如果生产者、消费者都是1个，那么flag 标记可以用if判断。这里有多个， ...

 nicolas9974 2015-06-23 20:46:20 阅读数：572

java多线程-生产者消费者经典问题 基于BlockingQueue

java多线程-生产者消费者经典问题 基于BlockingQueue

 xinyuan_java 2016-08-04 11:26:38 阅读数：431


JAVA多线程（三）生产者消费者模式及实现方法

介绍了生产者消费者模式以及实现方法（ wait¬ify ，阻塞队列

 antony9118 2016-05-23 15:15:02 阅读数：5701

Java 多线程学习之生产者消费者模型：一个较完善的实现

生产者、消费者模型是学习多线程的时候的一个很好的练习模型。该问题专业的说法应为：有限缓冲问题。该问题描述了两个共享固定大小缓冲区的线程——即所谓的“生产者”和“消费者”——在实际运行时会发生的问题。 ...

 a454042522 2013-02-08 20:31:39 阅读数：1917

没有更多推荐了，[返回首页](#)