Java并发编程相关面试问题

基础概念

1.什么是原子操作？在Java Concurrency API中有哪些原子类(atomic classes)?

原子操作（atomic operation）意为"不可被中断的一个或一系列操作" 。

处理器使用基于对缓存加锁或总线加锁的方式来实现多处理器之间的原子操作。

在Java中可以通过**锁和循环CAS的方式**来实现原子操作。 **CAS操作——Compare & Set**，或是Compare & Swap，现在几乎所有的CPU指令都支持CAS的原子操作。

原子操作是指一个不受其他操作影响的操作任务单元。**原子操作是在多线程环境下避免数据不一致必须的手段。**

int++并不是一个原子操作，所以当一个线程读取它的值并加1时，另外一个线程有可能会读到之前的值，这就会引发错误。

为了解决这个问题，必须保证增加操作是原子的，在JDK1.5之前我们可以使用同步技术来做到这一点。

到JDK1.5，java.util.concurrent.atomic包提供了int和long类型的原子包装类，它们可以自动的保证对于他们的操作是原子的并且不需要使用同步。

java.util.concurrent这个包里面提供了一组**原子类**。其基本的特性就是在多线程环境下，当有多个线程同时执行这些类的实例包含的方法时，具有排他性，即当某个线程进入方法，执行其中的指令时，不会被其他线程打断，而别的线程就像自旋锁一样，一直等到该方法执行完成，才由JVM从等待队列中选择一个另一个线程进入，这只是一种逻辑上的理解。

AtomicBoolean，AtomicInteger，AtomicLong，AtomicReference
AtomicIntegerArray，AtomicLongArray
AtomicLongFieldUpdater，AtomicIntegerFieldUpdater，
AtomicReferenceFieldUpdater
AtomicMarkableReference，AtomicStampedReference，AtomicReferenceArray

2.Java Concurrency API中的Lock接口(Lock interface)是什么？对比同步它有什么优势?

Lock接口比同步方法和同步块提供了**更具扩展性的锁操作**。

他们允许更灵活的结构，可以具有完全不同的性质，并且可以支持多个相关类的条件对象。

它的优势有：

可以使锁更公平
可以使线程在等待锁的时候响应中断
可以让线程尝试获取锁，并在无法获取锁的时候立即返回或者等待一段时间
可以在不同的范围，以不同的顺序获取和释放锁

3.什么是Executors框架?

Executor框架是一个根据一组执行策略调用，调度，执行和控制的异步任务的框架。

无限制的创建线程会引起应用程序内存溢出。**所以创建一个线程池是个更好的的解决方案，因为可以限制线程的数量并且可以回收再利用这些线程。利用Executors框架可以非常方便的创建一个线程池。**

4.什么是阻塞队列？阻塞队列的实现原理是什么？如何使用阻塞队列来实现生产者-消费者模型?

阻塞队列（BlockingQueue）是一个支持两个附加操作的队列。

这两个附加的操作是：在队列为空时，获取元素的线程会等待队列变为非空。当队列满时，存储元素的线程会等待队列可用。

阻塞队列常用于生产者和消费者的场景，生产者是往队列里添加元素的线程，消费者是从队列里拿元素的线程。阻塞队列就是生产者存放元素的容器，而消费者也只从容器里拿元素。

JDK7提供了7个阻塞队列。分别是：

ArrayBlockingQueue ： 一个由数组结构组成的有界阻塞队列。
LinkedBlockingQueue ： 一个由链表结构组成的有界阻塞队列。
PriorityBlockingQueue ： 一个支持优先级排序的无界阻塞队列。
DelayQueue： 一个使用优先级队列实现的无界阻塞队列。

SynchronousQueue：一个不存储元素的阻塞队列。
LinkedTransferQueue：一个由链表结构组成的无界阻塞队列。
LinkedBlockingDeque：一个由链表结构组成的双向阻塞队列。

Java 5之前实现同步存取时，可以使用普通的一个集合，然后在使用线程的协作和线程同步可以实现生产者，消费者模式，主要的技术就是用好，wait ,notify,notifyAll,sychronized这些关键字。而在java 5之后，可以使用阻塞队列来实现，此方式大大简少了代码量，使得多线程编程更加容易，安全方面也有保障。
BlockingQueue接口是Queue的子接口，它的主要用途并不是作为容器，而是作为线程同步的的工具，因此他具有一个很明显的特性，当生产者线程试图向BlockingQueue放入元素时，如果队列已满，则线程被阻塞，当消费者线程试图从中取出一个元素时，如果队列为空，则该线程会被阻塞，正是因为它所具有这个特性，所以在程序中多个线程交替向BlockingQueue中放入元素，取出元素，它可以很好的控制线程之间的通信。

阻塞队列使用**最经典的场景就是socket客户端数据的读取和解析，读取数据的线程不断将数据放入队列，然后解析线程不断从队列取数据解析。**

5.什么是Callable和Future?

Callable接口类似于Runnable，从名字就可以看出来了，

但是Runnable不会返回结果，并且无法抛出返回结果的异常，而Callable功能更强大一些，被线程执行后，可以返回值，这个返回值可以被Future拿到，也就是说，Future可以拿到异步执行任务的返回值。

可以认为是带有回调的Runnable。

Callable接口代表一段可以调用并返回结果的代码;Future接口表示异步任务，是还没有完成的任务给出的未来结果。所以说Callable用于产生结果，Future用于获取结果。

6.什么是FutureTask?使用ExecutorService启动任务。

在Java并发程序中FutureTask表示一个可以取消的异步运算。它有启动和取消运算、查询运算是否完成和取回运算结果等方法。只有当运算完成的时候结果才能取回，如果运算尚未完成get方法将会阻塞。一个FutureTask对象可以对调用了Callable和Runnable的对象进行包装，由于FutureTask也是调用了Runnable接口所以它可以提交给Executor来执行。

7.什么是并发容器的实现?

何为同步容器：可以简单地理解为通过synchronized来实现同步的容器，如果有多个线程调用同步容器的方法，它们将会串行执行。比如Vector，Hashtable，以及Collections.synchronizedSet，synchronizedList等方法返回的容器。
可以通过查看Vector，Hashtable等这些同步容器的实现代码，可以看到这些容器实现线程安全的方式就是将它们的状态封装起来，并在需要同步的方法上加上关键字synchronized。

并发容器使用了与同步容器完全不同的加锁策略来提供更高的并发性和伸缩性，例如在ConcurrentHashMap中采用了一种粒度更细的加锁机制，可以称为分段锁，在这种锁机制下，允许任意数量的读线程并发地访问map，并且执行读操作的线程和写操作的线程也可以并发的访问map，同时允许一定数量的写操作线程并发地修改map，所以它可以在并发环境下实现更高的吞吐量。

8.多线程同步和互斥有几种实现方法，都是什么?

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，直到消息到达时才被唤醒。
线程互斥是指对于共享的进程系统资源，在各单个线程访问时的排它性。当有若干个线程都要使用某一共享资源时，任何时刻最多只允许一个线程去使用，其它要使用该资源的线程必须等待，直到占用资源者释放该资源。线程互斥可以看成是一种特殊的线程同步。

线程间的同步方法大体可分为两类：用户模式和内核模式。顾名思义，内核模式就是指利用系统内核对象的单一性来进行同步，使用时需要切换内核态与用户态，而用户模式就是不需要切换到内核态，只在用户态完成操作。
用户模式下的方法有：原子操作（例如一个单一的全局变量），临界区。内核模式下的方法有：事件，信号量，互斥量。

9.什么是竞争条件？你怎样发现和解决竞争?

当多个进程都企图对共享数据进行某种处理，而最后的结果又取决于进程运行的顺序时，则我们认为这发生了竞争条件（race condition）。

10.你将如何使用thread dump？你将如何分析Thread dump?

java.lang.Thread.State枚举中定义了线程的几种状态：
NEW：线程刚被创建，但是还没有被处理。
RUNNABLE：线程占用了 CPU 并且处理了一个任务。（或是是在等待状态由于操作系统的资源分配）
BLOCKED：该线程正在等待另外的不同的线程释放锁，以便获取监视器锁
WAITING：该线程正在等待，通过使用了 wait, join 或者是 park 方法

搜索

我的标签

Java (12) 读书笔记 (9)
设计模式 (9) 数据库 (4)
算法 (1) 数据结构 (1)
ACM (1) Git (1)
JavaScript (1)
Java集合框架 (1) 更多

随笔分类

Dcoker容器化实践(1)
Dubbo与分布式服务框架(6)
ELK日志收集(1)
Flume应用
Git最佳实践(6)
Go语言编程(1)
Hibernate应用(2)
Java应用开发(23)
JVM虚拟机深入(11)
LeetCode题解(4)
Linux与Shell脚本(21)
Maven笔记(8)
Memcached应用
MongoDB实践(1)
Python笔记(1)
Redis应用(11)
Solr索引应用(4)
Spring应用开发(6)
SQL应用(5)
TCP/IP协议(12)
阿里开源系列(1)
并发编程实战(15)
测试及性能优化(3)
产品设计(1)
大数据与数据分析(2)
大型网站系统架构(10)
读书笔记(13)
分布式一致性(5)
高性能数据库(18)
工程师随笔(4)
公开课笔记(6)
机器学习
面试经验(10)
前端开发(5)
区块链技术
设计模式系列(9)
数据结构(7)
算法与程序设计(11)
网络编程实践(6)
项目管理和敏捷开发(1)
消息中间件应用(7)
源码学习(8)

相册

博文图床(2)

积分与排名

积分 - 191336
排名 - 1137

阅读排行榜

1. 菜鸟学Linux命令:ssh命...

TIMED_WAITING：该线程正在等待，通过使用了 sleep, wait, join 或者是 park 方法。（这个与 WAITING 不同是通过方法参数指定了最大等待时间，WAITING 可以通过时间或者是外部的变化解除）

11.为什么我们调用start()方法时会执行run()方法，为什么我们不能直接调用run()方法?

当你调用start()方法时你将创建新的线程，并且执行在run()方法里的代码。但是如果你直接调用run()方法，它不会创建新的线程也不会执行调用线程的代码。

12.Java中你怎样唤醒一个阻塞的线程?

在Java发展史上曾经使用suspend()、resume()方法对于线程进行阻塞唤醒，但随之出现很多问题，比较典型的还是死锁问题。
解决方案可以使用以对象为目标的阻塞，即利用Object类的wait()和notify()方法实现线程阻塞。
首先，wait、notify方法是针对对象的，调用任意对象的wait()方法都将导致线程阻塞，阻塞的同时也将释放该对象的锁，相应地，调用任意对象的notify()方法则将随机解除该对象阻塞的线程，但它需要重新获取改对象的锁，直到获取成功才能往下执行；其次，wait、notify方法必须在synchronized块或方法中被调用，并且要保证同步块或方法的锁对象与调用wait、notify方法的对象是同一个，如此一来在调用wait之前当前线程就已经成功获取某对象的锁，执行wait阻塞后当前线程就将之前获取的对象锁释放。

14.在Java中CycliBarriar和CountdownLatch有什么区别?

CyclicBarrier可以重复使用，而CountdownLatch不能重复使用。
Java的concurrent包里面的CountDownLatch其实可以把它看作一个计数器，只不过这个计数器的操作是原子操作，同时只能有一个线程去操作这个计数器，也就是同时只能有一个线程去减这个计数器里面的值。
你可以向CountDownLatch对象设置一个初始的数字作为计数值，任何调用这个对象上的await()方法都会阻塞，直到这个计数器的计数值被其他的线程减为0为止。
所以在当前计数到达零之前，await 方法会一直受阻塞。之后，会释放所有等待的线程，await的所有后续调用都将立即返回。这种现象只出现一次——计数无法被重置。如果需要重置计数，请考虑使用CyclicBarrier。
CountDownLatch的一个非常典型的应用场景是：有一个任务想要往下执行，但必须要等到其他的任务执行完毕后才可以继续往下执行。假如我们这个想要继续往下执行的任务调用一个CountDownLatch对象的await()方法，其他的任务执行完自己的任务后调用同一个CountDownLatch对象上的countDown()方法，这个调用await()方法的任务将一直阻塞等待，直到这个CountDownLatch对象的计数值减到0为止。

CyclicBarrier一个同步辅助类，它允许一组线程互相等待，直到到达某个公共屏障点 (common barrier point)。在涉及一组固定大小的线程的程序中，这些线程必须不时地互相等待，此时 CyclicBarrier 很有用。因为该 barrier 在释放等待线程后可以重用，所以称它为循环 的 barrier。

14.什么是不可变对象，它对写并发应用有什么帮助?

不可变对象(Immutable Objects)即对象一旦被创建它的状态（对象的数据，也即对象属性值）就不能改变，反之即为可变对象(Mutable Objects)。
不可变对象的类即为不可变类(Immutable Class)。Java平台类库中包含许多不可变类，如String、基本类型的包装类、BigInteger和BigDecimal等。
不可变对象天生是线程安全的。它们的常量（域）是在构造函数中创建的。既然它们的状态无法修改，这些常量永远不会变。

不可变对象永远是线程安全的。
只有满足如下状态，一个对象才是不可变的；
它的状态不能在创建后再被修改；
所有域都是final类型；并且，
它被正确创建（创建期间没有发生this引用的逸出）。

15.什么是多线程中的上下文切换?

在上下文切换过程中，CPU会停止处理当前运行的程序，并保存当前程序运行的具体位置以便之后继续运行。从这个角度来看，上下文切换有点像我们同时阅读几本书，在来回切换书本的同时我们需要记住每本书当前读到的页码。在程序中，上下文切换过程中的"页码"信息是保存在进程控制块（PCB）中的。PCB还经常被称作"切换桢"（switchframe）。"页码"信息会一直保存到CPU的内存中，直到他们被再次使用。
上下文切换是存储和恢复CPU状态的过程，它使得线程执行能够从中断点恢复执行。上下文切换是多任务操作系统和多线程环境的基本特征。

16.Java中用到的线程调度算法是什么?

计算机通常只有一个CPU,在任意时刻只能执行一条机器指令,每个线程只有获得CPU的使用权才能执行指令.所谓多线程的并发运行,其实是指从宏观上看,各个线程轮流获得CPU的使用权,分别执行各自的任务.在运行池中,会有多个处于就绪状态的线程在等待CPU,JAVA虚拟机的一项任务就是负责线程的调度,线程调度是指按照特定机制为多个线程分配CPU的使用权.

有两种调度模型：分时调度模型和抢占式调度模型。
分时调度模型是指让所有的线程轮流获得cpu的使用权,并且平均分配每个线程占用的CPU的时间片这个也比较好理解。
java虚拟机采用抢占式调度模型，是指优先让可运行池中优先级高的线程占用CPU，如果可运行池中的线

2. 阿里、百度、搜狐等公司...
3. Dubbo超时和重连机制(1...
4. Java Eclipse进行断点调...
5. 使用Jenkins配置Git和Ma...

评论排行榜

1. 阿里、百度、搜狐等公司...
2. 回顾和总结2016年，展望...
3. 开卷有益，工程师进阶推...
4. 大型网站压力测试及优化...
5. 使用Json Web Token设...

打赏

程优先级相同，那么就随机选择一个线程，使其占用CPU。处于运行状态的线程会一直运行，直至它不得不放弃CPU。

17.什么是线程组，为什么在Java中不推荐使用?

线程组和线程池是两个不同的概念，他们的作用完全不同，前者是为了方便线程的管理，后者是为了管理线程的生命周期，复用线程，减少创建销毁线程的开销。

18.为什么使用Executor框架比使用应用创建和管理线程好?

19.java中有几种方法可以实现一个线程?

20.如何停止一个正在运行的线程?

21.notify()和notifyAll()有什么区别?
22.什么是Daemon线程？它有什么意义?
23.java如何实现多线程之间的通讯和协作?
24.什么是可重入锁（ReentrantLock）？
25.当一个线程进入某个对象的一个synchronized的实例方法后，其它线程是否可进入此对象的其它方法?

26.synchronized和java.util.concurrent.locks.Lock的异同?
27.乐观锁和悲观锁的理解及如何实现，有哪些实现方式?
28.SynchronizedMap和ConcurrentHashMap有什么区别?
29.CopyOnWriteArrayList可以用于什么应用场景?
30.什么叫线程安全？servlet是线程安全吗?

31.volatile有什么用？能否用一句话说明下volatile的应用场景?
32.请说明下java的内存模型及其工作流程。
33.为什么代码会重排序?
34.现在有T1、T2、T3三个线程，你怎样保证T2在T1执行完后执行，T3在T2执行完后执行?
可以用join方法实现。
35.在Java中Lock接口比synchronized块的优势是什么？你需要实现一个高效的缓存，它允许多个用户读，但只允许一个用户写，以此来保持它的完整性，你会怎样去实现它?
lock接口在多线程和并发编程中最大的优势是它们为读和写分别提供了锁，它能满足你写像ConcurrentHashMap这样的高性能数据结构和有条件的阻塞。

36.在java中wait和sleep方法的不同?
最大的不同是在等待时wait会释放锁，而sleep一直持有锁。Wait通常被用于线程间交互，sleep通常被用于暂停执行。
37.用Java实现阻塞队列。
38.用Java实现代码来解决生产者——消费者问题。
39.用Java编程一个会导致死锁的程序，你将怎么解决?
40.什么是原子操作，Java中的原子操作是什么?

41.Java中的volatile关键是什么作用？怎样使用它？在Java中它跟synchronized方法有什么不同?

42.一个线程运行时发生异常会怎样?
如果异常没有被捕获该线程将会停止执行。Thread.UncaughtExceptionHandler是用于处理未捕获异常造成线程突然中断情况的一个内嵌接口。当一个未捕获异常将造成线程中断的时候JVM会使用Thread.getUncaughtExceptionHandler()来查询线程的UncaughtExceptionHandler并将线程和异常作为参数传递给handler的uncaughtException()方法进行处理。

43.如何在两个线程间共享数据?

44.Java中notify 和 notifyAll有什么区别?
notify() 方法不能唤醒某个具体的线程，所以只有一个线程在等待的时候它才有用武之地。而notifyAll()唤醒所有线程并允许他们争夺锁确保了至少有一个线程能继续运行。

45.为什么wait, notify 和 notifyAll这些方法不在thread类里面?
一个很明显的原因是JAVA提供的锁是对象级的而不是线程级的，每个对象都有锁，通过线程获得。由于wait，notify和notifyAll都是锁级别的操作，所以把他们定义在Object类中因为锁属于对象。

46.什么是ThreadLocal变量?

ThreadLocal是Java里一种特殊的变量。每个线程都有一个ThreadLocal就是每个线程都拥有了自己独立的一个变量，竞争条件被彻底消除了。它是为创建代价高昂的对象获取线程安全的好方法，比如你可以用ThreadLocal让SimpleDateFormat变成线程安全的，因为那个类创建代价高昂且每次调用都需要创建不同的实例所以不值得在局部范围使用它，如果为每个线程提供一个自己独有的变量拷贝，将大大提高效率。首先，通过复用减少了代价高昂的对象的创建个数。其次，你在没有使用高代价的同步或者不变性的情况下获得了线程安全。

47.Java中interrupted 和 isInterruptedd方法的区别?

48.为什么wait和notify方法要在同步块中调用?

Java API强制要求这样做，如果你不这么做，你的代码会抛出IllegalMonitorStateException异常。还有一个原因是为了避免wait和notify之间产生竞态条件。

49.为什么你应该在循环中检查等待条件?

处于等待状态的线程可能会收到错误警报和伪唤醒，如果不在循环中检查等待条件，程序就会在没有满足结束条件的情况下退出。

50.Java中的同步集合与并发集合有什么区别?

同步集合与并发集合都为多线程和并发提供了合适的线程安全的集合，不过并发集合的可扩展性更高。在Java1.5之前程序员们只有同步集合来用且在多线程并发的时候会导致争用，阻碍了系统的扩展性。Java5介绍了并发集合像ConcurrentHashMap，不仅提供线程安全还用锁分离和内部分区等现代技术提高了可扩展性。

51.Java中堆和栈有什么不同?

52.什么是线程池？ 为什么要使用它?

创建线程要花费昂贵的资源和时间，如果任务来了才创建线程那么响应时间会变长，而且一个进程能创建的线程数有限。为了避免这些问题，在程序启动的时候就创建若干线程来响应处理，它们被称为线程池，里面的线程叫工作线程。从JDK1.5开始，Java API提供了Executor框架让你可以创建不同的线程池。

53.如何避免死锁?

死锁的发生必须满足以下四个条件：
互斥条件：一个资源每次只能被一个进程使用。
请求与保持条件：一个进程因请求资源而阻塞时，对已获得的资源保持不放。
不剥夺条件：进程已获得的资源，在末使用完之前，不能强行剥夺。
循环等待条件：若干进程之间形成一种头尾相接的循环等待资源关系。
避免死锁最简单的方法就是阻止循环等待条件，将系统中所有的资源设置标志位、排序，规定所有的进程申请资源必须以一定的顺序（升序或降序）做操作来避免死锁。

54.Java中活锁和死锁有什么区别?

活锁和死锁类似，不同之处在于处于活锁的线程或进程的状态是不断改变的，活锁可以认为是一种特殊的饥饿。一个现实的活锁例子是两个人在狭小的走廊碰到，两个人都试着避让对方好让彼此通过，但是因为避让的方向都一样导致最后谁都不能通过走廊。简单的说就是，活锁和死锁的主要区别是前者进程的状态可以改变但是却不能继续执行。

55.怎么检测一个线程是否拥有锁?

在java.lang.Thread中有一个方法叫holdsLock()，它返回true如果当且仅当当前线程拥有某个具体对象的锁。

56.你如何在Java中获取线程堆栈?

57.JVM中哪个参数是用来控制线程的栈堆栈小的?

58.Java中synchronized 和 ReentrantLock 有什么不同?

59.Thread类中的yield方法有什么作用?

60.Java中ConcurrentHashMap的并发度是什么?

ConcurrentHashMap把实际map划分成若干部分来实现它的可扩展性和线程安全。这种划分是使用并发度获得的，它是ConcurrentHashMap类构造函数的一个可选参数，默认值为16，这样在多线程情况下就能避免争用。

61.Java中Semaphore是什么?

Java中的Semaphore是一种新的同步类，它是一个计数信号。从概念上讲，从概念上讲，信号量维护了一个许可集合。如有必要，在许可可用前会阻塞每一个 acquire()，然后再获取该许可。每个 release() 添加一个许可，从而可能释放一个正在阻塞的获取者。但是，不使用实际的许可对象，Semaphore只对可用许可的号码进行计数，并采取相应的行动。信号量常常用于多线程的代码中，比如数据库连接池。

62.Java线程池中submit() 和 execute()方法有什么区别?

两个方法都可以向线程池提交任务，execute()方法的返回类型是void，它定义在Executor接口中，而submit()方法可以返回持有计算结果的Future对象，它定义在ExecutorService接口中，它扩展了Executor接口，其它线程池类像ThreadPoolExecutor和ScheduledThreadPoolExecutor都有这些方法。

63.如何在Java中创建Immutable对象?

64.什么是阻塞式方法?

阻塞式方法是指程序会一直等待该方法完成期间不做其他事情，ServerSocket的accept()方法就是一直等待客户端连接。这里的阻塞是指调用结果返回之前，当前线程会被挂起，直到得到结果之后才会返回。此外，还有异步和非阻塞式方法在任务完成前就返回。

65.Java中的ReadWriteLock是什么?

读写锁是用来提升并发程序性能的锁分离技术的成果。

66.volatile 变量和 atomic 变量有什么不同?

Volatile变量可以确保先行关系，即写操作会发生在后续的读操作之前，但它并不能保证原子性。例如用volatile修饰count变量那么 count++ 操作就不是原子性的。而AtomicInteger类提供的atomic方法可以让这种操作具有原子性如getAndIncrement()方法会原子性的进行增量操作把当前值加一，其它数据类型和引用变量也可以进行相似操作。

67.用户线程和守护线程有什么区别?

当我们在Java程序中创建一个线程，它就被称为用户线程。一个守护线程是在后台执行并且不会阻止JVM终止的线程。当没有用户线程在运行的时候，JVM关闭程序并且退出。一个守护线程创建的子线程依然是守护线程。

68.线程生命周期有哪些阶段?

当我们在Java程序中新建一个线程时，它的状态是New。当我们调用线程的start()方法时，状态被改变为Runnable。线程调度器会为Runnable线程池中的线程分配CPU时间并且将它们的状态改变为Running。其他的线程状态还有Waiting，Blicked 和Dead。

69.可以直接调用Thread类的run ()方法么?

当然可以。但是如果我们调用了Thread的run()方法，它的行为就会和普通的方法一样，会在当前线程中执行。为了在新的线程中执行我们的代码，必须使用Thread.start()方法。

70.如何让正在运行的线程暂停一段时间?

我们可以使用Thread类的Sleep()方法让线程暂停一段时间。需要注意的是，这并不会让线程终止，一旦从休眠中唤醒线程，线程的状态将会被改变为Runnable，并且根据线程调度，它将得到执行。

71.你对线程优先级的理解是什么?

每一个线程都是有优先级的，一般来说，高优先级的线程在运行时会具有优先权，但这依赖于线程调度的实现，这个实现是和操作系统相关的(OS dependent)。我们可以定义线程的优先级，但是这并不能保证高优先级的线程会在低优先级的线程前执行。线程优先级是一个int变量(从1-10)，1代表最低优先级，10代表最高优先级。

72.什么是线程调度器(Thread Scheduler)和时间分片(Time Slicing)?

线程调度器是一个操作系统服务，它负责为Runnable状态的线程分配CPU时间。一旦我们创建一个线程并启动它，它的执行便依赖于线程调度器的实现。时间分片是指将可用的CPU时间分配给可用的Runnable线程的过程。分配CPU时间可以基于线程优先级或者线程等待的时间。线程调度并不受到Java虚拟机控制，所以由应用程序来控制它是更好的选择（也就是说不要让你的程序依赖于线程的优先级）。

73.你如何确保main()方法所在的线程是Java 程序最后结束的线程?

我们可以使用Thread类的joint()方法来确保所有程序创建的线程在main()方法退出前结束。

74.线程之间是如何通信的?

当线程间是可以共享资源时，线程间通信是协调它们的重要的手段。Object类中wait()\notify()\notifyAll()方法可以用于线程间通信关于资源的锁的状态。

75. 为什么线程通信的方法wait(), notify()和notifyAll()被定义在Object 类里?

Java的每个对象中都有一个锁(monitor，也可以成为监视器) 并且wait()，notify()等方法用于等待对象的锁或者通知其他线程对象的监视器可用。在Java的线程中并没有可供任何对象使用的锁和同步器。这就是为什么这些方法是Object类的一部分，这样Java的每一个类都有用于线程间通信的基本方法。

76.为什么wait(), notify()和notifyAll ()必须在同步方法或者同步块中被调用?

当一个线程需要调用对象的wait()方法的时候，这个线程必须拥有该对象的锁，接着它就会释放这个对象锁并进入等待状态直到其他线程调用这个对象上的notify()方法。同样的，当一个线程需要调用对象的notify()方法时，它会释放这个对象的锁，以便其他在等待的线程就可以得到这个对象锁。由于所有的这些方法都需要线程持有对象的锁，这样就只能通过同步来实现，所以他们只能在同步方法或者同步块中被调用。

77.为什么Thread类的sleep()和yield ()方法是静态的?

Thread类的sleep()和yield()方法将在当前正在执行的线程上运行。所以在其他处于等待状态的线程上调用这些方法是没有意义的。这就是为什么这些方法是静态的。它们可以在当前正在执行的线程中工作，并避免程序员错误的认为可以在其他非运行线程调用这些方法。

78.如何确保线程安全?

在Java中可以有很多方法来保证线程安全——同步，使用原子类(atomic concurrent classes)，实现并发锁，使用volatile关键字，使用不变类和线程安全类。

79.volatile关键字在Java 中有什么作用?

当我们使用volatile关键字去修饰变量的时候，所以线程都会直接读取该变量并且不缓存它。这就确保了线程读取到的变量是同内存中是一致的。

80.同步方法和同步块，哪个是更好的选择?

同步块是更好的选择，因为它不会锁住整个对象（当然你也可以让它锁住整个对象）。同步方法会锁住整个对象，哪怕这个类中有多个不相关联的同步块，这通常会导致他们停止执行并需要等待获得这个对象上的锁。

81.如何创建守护线程?

使用Thread类的setDaemon(true)方法可以将线程设置为守护线程，需要注意的是，需要在调用start()方法前调用这个方法，否则会抛出IllegalThreadStateException异常。

82. 什么是ThreadLocal?

ThreadLocal用于创建线程的本地变量，我们知道一个对象的所有线程会共享它的全局变量，所以这些变量不是线程安全的，我们可以使用同步技术。但是当我们不想使用同步的时候，我们可以选择ThreadLocal变量。
每个线程都会拥有他们自己的Thread变量，它们可以使用get()\set()方法去获取他们的默认值或者在线程内部改变他们的值。ThreadLocal实例通常是希望它们同线程状态关联起来是private static属性。在ThreadLocal例子这篇文章中你可以看到一个关于ThreadLocal的小程序。

83.什么是Thread Group ？ 为什么建议使用它?

ThreadGroup是一个类，它的目的是提供关于线程组的信息。
ThreadGroup API比较薄弱，它并没有比Thread提供了更多的功能。它有两个主要的功能：一是获取线程组中处于活跃状态线程的列表；二是设置为线程设置未捕获异常处理器(ncaught exception handler)。但在Java 1.5中Thread类也添加了setUncaughtExceptionHandler(UncaughtExceptionHandler eh) 方法，所以ThreadGroup是已经过时的，不建议继续使用。

```
t1.setUncaughtExceptionHandler(new UncaughtExceptionHandler(){

@Override
public void uncaughtException(Thread t, Throwable e) {
System.out.println("exception occured:"+e.getMessage());
}

});
```

84. 什么是Java线程转储(Thread Dump)，如何得到它?

线程转储是一个JVM活动线程的列表，它对于分析系统瓶颈和死锁非常有用。有很多方法可以获取线程转储——使用Profiler，Kill -3命令，jstack工具等等。我更喜欢jstack工具，因为它容易使用并且是JDK自带的。由于它是一个基于终端的工具，所以我们可以编写一些脚本去定时的产生线程转储以待分析。

85.什么是Java Timer 类？如何创建一个有特定时间间隔的任务?

java.util.Timer是一个工具类，可以用于安排一个线程在未来的某个特定时间执行。Timer类可以用安排一次性任务或者周期任务。
java.util.TimerTask是一个实现了Runnable接口的抽象类，我们需要去继承这个类来创建我们自己的定时任务并使用Timer去安排它的执行。
目前有开源的Qurtz可以用来创建定时任务。

应用场景

(一)并发容器和框架

1.如何让一段程序并发的执行，并最终汇总结果?

2.如何合理的配置java线程池？如CPU密集型的任务，基本线程池应该配置多大？IO密集型的任务，基本线程池应该配置多大？用有界队列好还是无界队列好？任务非常多的时候，使用什么阻塞队列能获取最好的吞吐量?

3.如何使用阻塞队列实现一个生产者和消费者模型？请写代码。

4.多读少写的场景应该使用哪个并发容器，为什么使用它?

比如你做了一个搜索引擎，搜索引擎每次搜索前需要判断搜索关键词是否在黑名单里，黑名单每天更新一次。

(二)Java中的锁

1.如何实现乐观锁（CAS）？如何避免ABA问题?

2.读写锁可以用于什么应用场景?

3.什么时候应该使用可重入锁?

4.什么场景下可以使用volatile替换synchronized?

(三)并发工具

1.如何实现一个流控程序，用于控制请求的调用次数?

几道笔试题目

1.（百度笔试题）以下多线程对int型变量x的操作，哪几个不需要进行同步：

A. x=y; B. x++; C. ++x; D. x=1;

2.（阿里巴巴笔试题）多线程中栈与堆是公有的还是私有的

A：栈公有, 堆私有

B：栈公有，堆公有

C：栈私有, 堆公有

D：栈私有，堆私有

3.一个全局变量tally，两个线程并发执行（代码段都是ThreadProc)，问两个线程都结束后，tally取值范围。

```
inttally = 0;//glable
voidThreadProc()
{
for(inti = 1; i <= 50; i++)
tally += 1;
}
```

4.子线程循环 10 次，接着主线程循环 100 次，接着又回到子线程循环 10 次，接着再回到主线程又循环 100 次，如此循环50次，试写出代码。

5.（迅雷笔试题）：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。

6.（Google面试题）有四个线程1、2、3、4。线程1的功能就是输出1，线程2的功能就是输出2，以此类推.....现在有四个文件ABCD。初始都为空。现要让四个文件呈如下格式：

A：1 2 3 4 1 2....

B：2 3 4 1 2 3....

C：3 4 1 2 3 4....

D：4 1 2 3 4 1....

请设计程序。

7.启动3个线程打印递增的数字, 线程1先打印1,2,3,4,5, 然后是线程2打印6,7,8,9,10, 然后是线程3打印11,12,13,14,15. 接着再由线程1打印16,17,18,19,20....以此类推, 直到打印到75. 程序的输出结果应该为:

```
线程1: 1
线程1: 2
线程1: 3
线程1: 4
线程1: 5

线程2: 6
线程2: 7
线程2: 8
线程2: 9
线程2: 10
...

线程3: 71
线程3: 72
线程3: 73
线程3: 74
```