历年阿里面试题汇总（2017年不断更新中） - CSDN博客

原 历年阿里面试题汇总（2017年不断更新中）

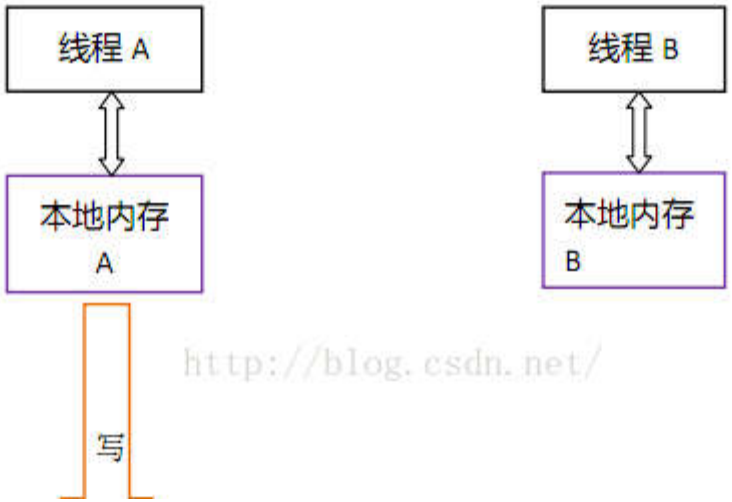
置顶 2017年03月06日 09:15:48 阅读数：37709

Volatile的特征：

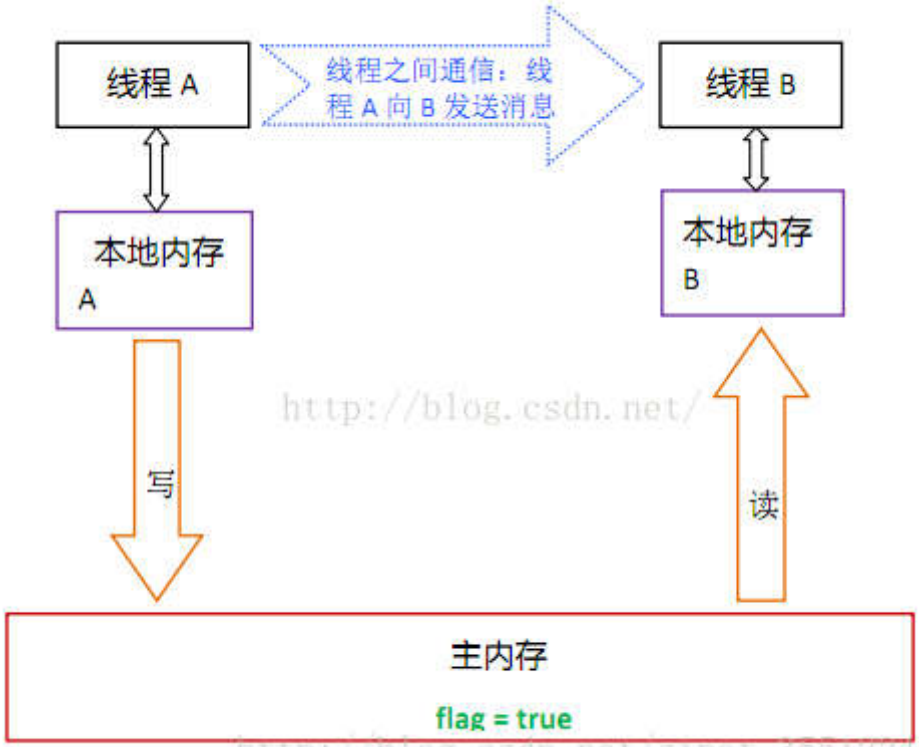
A、禁止指令重排（有例外）
B、可见性

Volatile的内存语义：

当写一个volatile变量时，JMM会把线程对应的本地内存中的共享变量值刷新到主内存。



当读一个volatile变量时，JMM会把线程对应的本地内存置为无效，线程接下来将从主内存中读取共享变量。



Volatile的重排序

1、当第二个操作为volatile写操做时,不管第一个操作是什么(普通读写或者volatile读写),都不能进行重排序。这个规则确保volatile写之前的所有操作都不会被重排序到volatile之后;

2、当第一个操作为volatile读操作时,不管第二个操作是什么,都不能进行重排序。这个规则确保volatile读之后的所有操作都不会被重排序到volatile之前;

3、当第一个操作是volatile写操作时,第二个操作是volatile读操作,不能进行重排序。

这个规则和前面两个规则一起构成了:两个volatile变量操作不能够进行重排序;

除以上三种情况以外可以进行重排序。

比如:

1、第一个操作是普通变量读/写,第二个是volatile变量的读;
2、第一个操作是volatile变量的写,第二个是普通变量的读/写;

内存屏障/内存栅栏

内存屏障（Memory Barrier，或有时叫做内存栅栏，Memory Fence）是一种CPU指令，用于控制特定条件下的重排序和内存可见性问题。Java编译器也会根据内存屏障的规则禁止重排序。（也就是让一个CPU处理单元中的内存状态对其它处理单元可见的一项技术。）

内存屏障可以被分为以下几种类型:

LoadLoad屏障：对于这样的语句Load1; LoadLoad; Load2，在Load2及后续读取操作要读取的数据被访问前，保证Load1要读取的数据被读取完毕。

StoreStore屏障：对于这样的语句Store1; StoreStore; Store2，在Store2及后续写入操作执行前，保证Store1的写入操作对其它处理器可见。

LoadStore屏障：对于这样的语句Load1; LoadStore; Store2，在Store2及后续写入操作被刷出前，保证Load1要读取的数据被读取完毕。

StoreLoad屏障：对于这样的语句Store1; StoreLoad; Load2，在Load2及后续所有读取操作执行前，保证Store1的写入对所有处理器可见。它的开销是四种屏障中最大的。

在大多数处理器的实现中，这个屏障是个万能屏障，兼具其它三种内存屏障的功能。

内存屏障阻碍了CPU采用优化技术来降低内存操作延迟，必须考虑因此带来的性能损失。为了达到最佳性能，最好是把要解决的问题模块化，这样处理器可以按单元执行任务，然后在任务单元的边界放上所有需要的内存屏障。采用这个方法可以让处理器不受限的执行一个任务单元。合理的内存屏障组合还有一个好处是：缓冲区在第一次被刷后开销会减少，因为再填充改缓冲区不需要额外工作了。

---

happens-before原则

Java是如何实现跨平台的?

跨平台是怎样实现的呢？这就要谈及Java虚拟机（Java Virtual Machine，简称 JVM）。

JVM也是一个软件，不同的平台有不同的版本。我们编写的Java源码，编译后会生成一种 .class 文件，称为字节码文件。Java虚拟机就是负责将字节码文件翻译成特定平台下的机器码然后运行。也就是说，只要在不同平台上安装对应的JVM，就可以运行字节码文件，运行我们编写的Java程序。

而这个过程中，我们编写的Java程序没有做任何改变，仅仅是通过JVM这一”中间层“，就能在不同平台上运行，真正实现了”一次编译，到处运行“的目的。

JVM是一个”桥梁“，是一个”中间件“，是实现跨平台的关键，Java代码首先被编译成字节码文件，再由JVM将字节码文件翻译成机器语言，从而达到运行Java程序的目的。

注意：编译的结果不是生成机器码，而是生成字节码，字节码不能直接运行，必须通过JVM翻译成机器码才能运行。不同平台下编译生成的字节码是一样的，但是由JVM翻译成的机器码却不一样。

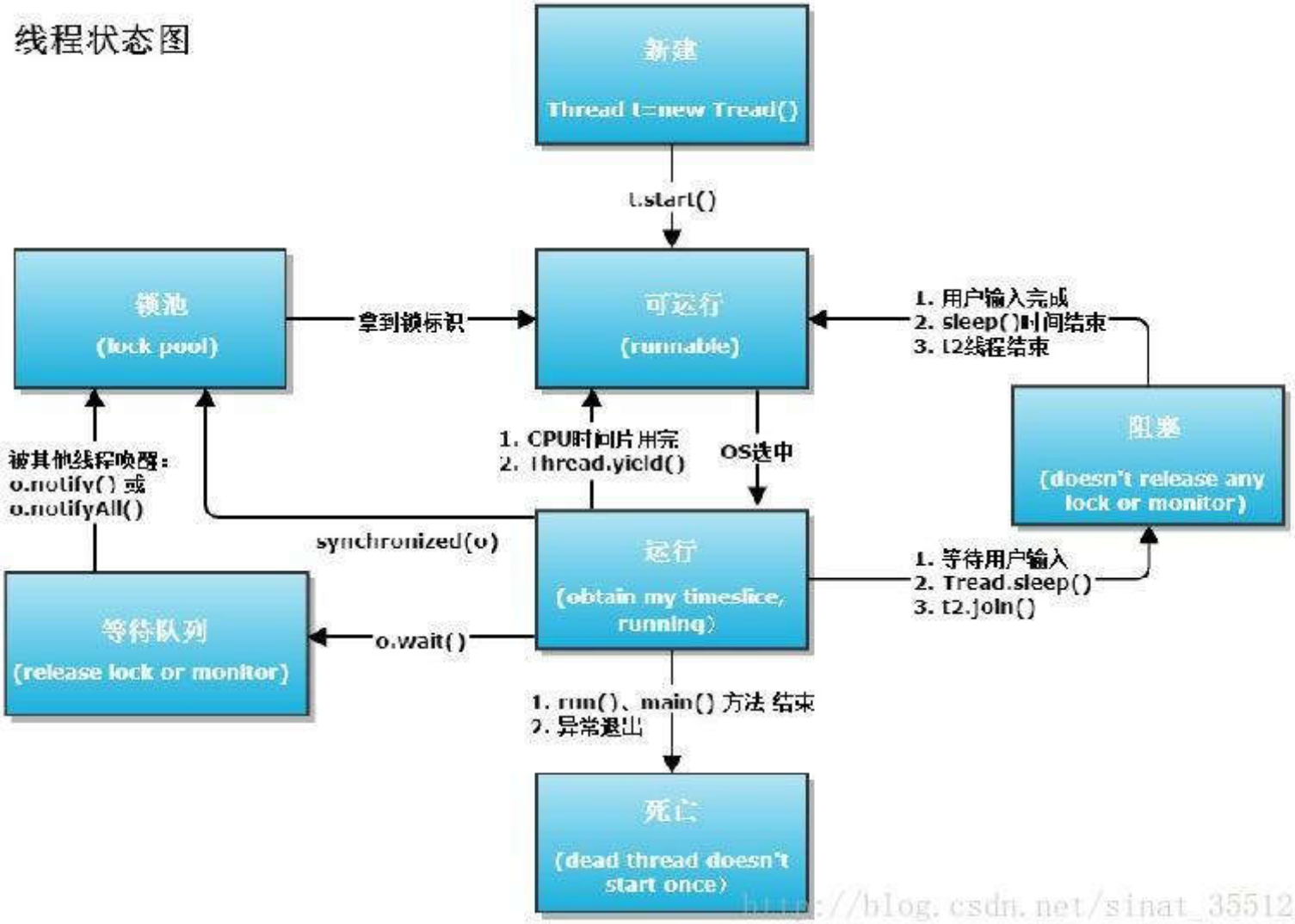
所以，运行Java程序必须有JVM的支持，因为编译的结果不是机器码，必须要经过JVM的再次翻译才能执行。即使你将Java程序打包成可执行文件（例如 .exe），仍然需要JVM的支持。

注意：跨平台的是Java程序，不是JVM。JVM是用C/C++开发的，是编译后的机器码，不能跨平台，不同平台下需要安装不同版本的JVM。

手机扫二维码登录是怎么实现的?

友情链接：扫码登录是如何实现的?

Java 线程有哪些状态，这些状态之间是如何转化的?



1. 新建(new)：新创建了一个线程对象。

2. 可运行(runnable)：线程对象创建后，其他线程(比如main线程）调用了该对象的start()方法。该状态的线程位于可运行线程池中，等待被线程调度选中，获取cpu 的使用权 。

3. 运行(running)：可运行状态(runnable)的线程获得了cpu 时间片（timeslice） ，执行程序代码。

4. 阻塞(block)：阻塞状态是指线程因为某种原因放弃了cpu 使用权，也即让出了cpu timeslice，暂时停止运行。直到线程进入可运行(runnable)状态，才有机会再次获得cpu timeslice 转到运行(running)状态。阻塞的情况分三种：

(一). 等待阻塞：运行(running)的线程执行o.wait()方法，JVM会把该线程放入等待队列(waitting queue)中。

(二). 同步阻塞：运行(running)的线程在获取对象的同步锁时，若该同步锁被别的线程占用，则JVM会把该线程放入锁池(lock pool)中。

(三). 其他阻塞：运行(running)的线程执行Thread.sleep(long ms)或t.join()方法，或者发出了I/O请求时，JVM会把该线程置为阻塞状态。当sleep()状态超时、join()等待线程终止或者超时、或者I/O处理完毕时，线程重新转入可运行(runnable)状态。

1. 死亡(dead)：线程run()、main() 方法执行结束，或者因异常退出了run()方法，则该线程结束生命周期。死亡的线程不可再次复生。

List接口、Set接口和Map接口的区别

友情链接：List接口、Set接口和Map接口的区别

Cookie和Session的区别?

友情链接：Cookies 和 Session的区别

Java中的equals和hashCode方法详解

友情链接：Java提高篇——equals()与hashCode()方法详解

友情链接：Java提高篇 —— equals()与hashCode()方法详解

Java中CAS算法

友情链接：乐观的并发策略——基于CAS的自旋

TimSort原理

友情链接：TimSort原理

---

comparable与comparator的区别?

友情链接：Comparable和Comparator的区别

手写单例模式（线程安全）

友情链接：快速理解Java中的五种单例模式

Java线程间的通信方式?

友情链接：Java 多线程（七） 线程间的通信——wait及notify方法
友情链接：Java线程间的通信方式详解

Java8的内存分代改进

友情链接：Java7、Java8的堆内存有啥变化?

对Java内存模型的理解以及其在并发当中的作用?

友情链接：对Java内存模型的理解以及其在并发当中的作用?

Arrays和Collections 对于sort的不同实现原理?

1、Arrays.sort()
该算法是一个经过调优的快速排序，此算法在很多数据集上提供N*log(N)的性能，这导致其他快速排序会降低二次型性能。

2、Collections.sort()
该算法是一个经过修改的合并排序算法（其中，如果低子列表中的最高元素效益高子列表中的最低元素，则忽略合并）。此算法可提供保证的N*log(N)的性能，此实现将指定列表转储到一个数组中，然后再对数组进行排序，在重置数组中相应位置处每个元素的列表上进行迭代。这避免了由于试图原地对链接列表进行排序而产生的$n^2log(n)$性能。

Java中object常用方法

1、clone()
2、equals()
3、finalize()
4、getclass()
5、hashcode()
6、notify()
7、notifyAll()
8、toString()

对于Java中多态的理解

所谓多态就是指程序中定义的引用变量所指向的具体类型和通过该引用变量发出的方法调用在编程时并不确定，而是在程序运行期间才确定，即一个引用变量到底会指向哪个类的实例对象，该引用变量发出的方法调用到底是哪个类中实现的方法，必须在由程序运行期间才能决定。因为在程序运行时才确定具体的类，这样，不用修改源程序代码，就可以让引用变量绑定到各种不同的类实现上，从而导致该引用调用的具体方法随之改变，即不修改程序代码就可以改变程序运行时所绑定的具体代码，让程序可以选择多个运行状态，这就是多态性。

多态的定义：指允许不同类的对象对同一消息做出响应。即同一消息可以根据发送对象的不同而采用多种不同的行为方式。（发送消息就是函数调用）

Java实现多态有三个必要条件：继承、重写、父类引用指向子类对象。

继承：在多态中必须存在有继承关系的子类和父类。

重写：子类对父类中某些方法进行重新定义，在调用这些方法时就会调用子类的方法。

父类引用指向子类对象：在多态中需要将子类的引用赋给父类对象，只有这样该引用才能够具备技能调用父类的方法和子类的方法。

实现多态的技术称为：动态绑定（dynamic binding），是指在执行期间判断所引用对象的实际类型，根据其实际的类型调用其相应的方法。

多态的作用：消除类型之间的耦合关系。

Session机制?

友情链接： Session机制详解

Java序列化与反序列化是什么？为什么需要序列化与反序列化？如何实现Java序列化与反序列化?

友情链接： Java序列化与反序列化

Spring AOP 实现原理?

友情链接： Spring AOP 实现原理

Servlet 工作原理?

友情链接： Servlet 工作原理解析

Java NIO和IO的区别?

友情链接： Java NIO和IO的区别

Java中堆内存和栈内存区别?

友情链接： Java中堆内存和栈内存详解

反射讲一讲，主要是概念,都在哪需要反射机制，反射的性能，如何优化?

反射机制的定义：

是在运行状态中，对于任意的一个类，都能够知道这个类的所有属性和方法，对任意一个对象都能够通过反射机制调用一个类的任意方法，这种动态获取类信息及动态调用类对象方法的功能称为java的反射机制。

反射的作用：

1、动态地创建类的实例，将类绑定到现有的对象中，或从现有的对象中获取类型。

2、应用程序需要在运行时从某个特定的程序集中载入一个特定的类。

如何保证RESTful API安全性？

友情链接： 如何设计好的RESTful API之安全性

如何预防Mysql注入?

友情链接：MySQL 及 SQL 注入与防范方法

欢迎加入QQ学习交流群（内有干货）：

群　号：677585877

文章标签： 面试题 阿里 互联网 春招 Java

个人分类： 笔试题集合

所属专栏： Java面试题总结