

大家好，很高兴我们可以继续学习交流Java高频面试题。在前面**Java基础篇章**中，我们分5个小节，依次介绍了**Java基础知识点以及Java中常见的集合知识点**。

为了提高程序的运行效率，我们需要合理运用多线程编程。多线程并发编程是我们必不可少的开发技能。所以，面试中对多线程的考察也是不可或缺的。面试中对多线程并发编程的考察，主要针对多线程出现的问题以及如何保证其线程安全，包括但不限于如下的知识点：

- 单线程和多线程，进程与线程的区别
- 线程活性故障及其解决方法
- 线程调度方式
- 可见性，原子性以及有序性
- synchronized, volatile, Atomic等关键字
- 线程池及阻塞队列
-

从Java 5.0开始，JDK中提供了**java.util.concurrent（简称JUC）包**，在此包中增加了并发编程中常用的工具类，用于定义线程的自定义子系统，包括线程池、异步IO 和轻量级任务框架等。

本节是《**Java进阶 - 高效并发编程（上）**》章节，介绍的知识点包括**线程与进程的区别，线程转换状态以及线程活性故障**等。让我们以面试题为引，开始学习多线程并发编程知识吧。

（1）进程与线程的区别：（重点掌握）

答：进程与线程之间的主要区别可以总结如下。

- 进程是一个“执行中的程序”，是系统进行**资源分配和调度**的一个独立单位
- 线程是进程的一个实体，一个进程中一般拥有多个线程。线程之间**共享地址空间**和其它资源（所以通信和同步等操作,线程比进程更加容易）
- 线程一般不拥有系统资源，但是也有一些必不可少的资源（使用ThreadLocal存储）
- 线程上下文的切换比进程上下文切换要快很多。

线程上下文切换比进程上下文切换快的原因，可以总结如下：

- **进程切换时**，涉及到当前进程的CPU环境的保存和新被调度运行进程的CPU环境的设置
- **线程切换时**，仅需要保存和设置少量的寄存器内容，不涉及存储管理方面的操作

解析：

进程与线程的区别算是一个开场题目，旨在考察大家对进程与线程的理解，因为我们的多线程是指在一个进程中的多个线程。

前面我们说线程之间共享一个进程的资源 and 地址空间，那么**线程可以拥有独属于自己的资源吗？**

答：可以的，通过ThreadLocal可以存储线程的特有对象，也就是属于当前线程的资源。关于ThreadLocal，我们会在后续小节详细介绍。

既然说到了进程，那么来看下进程之间有哪些通信方式吧。

进程之间常见的通信方式：

- 通过使用套接字Socket来实现不同机器间的进程通信
- 通过映射一段可以被多个进程访问的共享内存来进行通信
- 通过写进程和读进程利用管道进行通信

24

(2) 多线程与单线程的关系：（掌握）

答：多线程与单线程之间的关系可以概括如下。

- 多线程是指在一个进程中，并发执行了多个线程，每个线程都实现了不同的功能
- 在单核CPU中，将CPU分为很小的时间片，在每一时刻只能有一个线程在执行，是一种微观上轮流占用CPU的机制。由于CPU轮询的速度非常快，所以看起来像是“同时”在执行一样
- 多线程会存在线程上下文切换，会导致程序执行速度变慢
- 多线程不会提高程序的执行速度，反而会降低速度。但是对于用户来说，可以减少用户的等待响应时间，提高了资源的利用效率

解析：

搞清楚多线程和单线程之间的区别，有助于我们理解为什么要使用多线程并发编程。多线程并发利用了CPU轮询时间片的特点，在一个线程进入阻塞状态时，可以快速切换到其余线程执行其余操作，这有利于提高资源的利用率，最大限度的利用系统提供的处理能力，有效减少了用户的等待响应时间。

但是，多线程并发编程也会带来数据的安全问题，线程之间的竞争也会导致线程死锁和锁死等活性故障。线程之间的上下文切换也会带来额外的开销等问题。

(3) 线程的状态有哪些？（掌握）

答：线程的状态包括新建状态，运行状态，阻塞等待状态和消亡状态。其中阻塞等待状态又分为BLOCKED，WAITING和TIMED_WAITING状态。

解析：


要想准确给出线程的各个状态，我们大可不必看网上的若干个版本的解析。俗话说的好，源码面前没有秘密。我们来看下JDK8中对于Thread状态的枚举定义，所有的状态如下所示：

- NEW
- RUNNABLE
- BLOCKED
- WAITING
- TIMED_WAITING
- TERMINATED

接下来，我们——看看JDK中解释：

(1) NEW:

```
/**
 * Thread state for a thread which has not yet started.
 */
NEW,
```

 牛客@我是祖国的花朵

看得出来，这是属于一个已经创建的线程，但是还没有调用start方法启动的线程所处的状态。

(2) RUNNABLE:


```
/**
 * Thread state for a runnable thread. A thread in the runnable
 * state is executing in the Java virtual machine but it may
 * be waiting for other resources from the operating system
 * such as processor.
 */
RUNNABLE,
```

 牛客@我是祖国的花朵

正如JDK中介绍，该状态包含两种可能。有可能正在运行，或者正在等待CPU资源。总体上就是当我们创建线程并且启动之后，就属于Runnable状态。

(3) BLOCKED:

```
/**
 * Thread state for a thread blocked waiting for a monitor lock.
 * A thread in the blocked state is waiting for a monitor lock
 * to enter a synchronized block/method or
 * reenter a synchronized block/method after calling
 * {@link Object#wait() Object.wait}.
 */
BLOCKED,
```

 牛客@我是祖国的花朵

阻塞状态，当线程准备进入synchronized同步块或同步方法的时候，需要申请一个监视器锁而进行的等待，会使线程进入BLOCKED状态。

(4) WAITING:

```

/**
 * Thread state for a waiting thread.
 * A thread is in the waiting state due to calling one of the
 * following methods:
 * <ul>
 * <li>{@link Object#wait()} Object.wait} with no timeout</li>
 * <li>{@link #join()} Thread.join} with no timeout</li>
 * <li>{@link LockSupport#park()} LockSupport.park}</li>
 * </ul>
 *
 * <p>A thread in the waiting state is waiting for another thread to
 * perform a particular action.
 *
 * For example, a thread that has called <tt>Object.wait()</tt>
 * on an object is waiting for another thread to call
 * <tt>Object.notify()</tt> or <tt>Object.notifyAll()</tt> on
 * that object. A thread that has called <tt>Thread.join()</tt>
 * is waiting for a specified thread to terminate.
 */

```

WAITING,

牛客@我是祖国的花朵

该状态的出现是因为调用了 **Object.wait ()** 或者 **Thread.join ()** 或者 **LockSupport.park ()** 。处于该状态下的线程在等待另一个线程 执行一些其余action来将其唤醒。

(5) TIMED_WAITING:

```

/**
 * Thread state for a waiting thread with a specified waiting time.
 * A thread is in the timed waiting state due to calling one of
 * the following methods with a specified positive waiting time:
 * <ul>
 * <li>{@link #sleep Thread.sleep}</li>
 * <li>{@link Object#wait(long) Object.wait} with timeout</li>
 * <li>{@link #join(long) Thread.join} with timeout</li>
 * <li>{@link LockSupport#parkNanos LockSupport.parkNanos}</li>
 * <li>{@link LockSupport#parkUntil LockSupport.parkUntil}</li>
 * </ul>
 */

```

TIMED_WAITING,

牛客@我是祖国的花朵

该状态和上一个状态其实是一样的，是不过其等待的时间是明确的。

(6) TERMINATED:

```
/**
 * Thread state for a terminated thread.
 * The thread has completed execution.
 */
TERMINATED;
```

牛客@我是祖国的花朵

消亡状态比较容易理解，那就是线程执行结束了，run方法执行结束表示线程处于消亡状态了。

(4) 多线程编程中常用的函数比较：

答：多线程编程中的常用函数的比较和特性总结如下。

sleep 和 wait 的区别：

- **sleep方法：**是Thread类的静态方法，当前线程将睡眠n毫秒，线程进入阻塞状态。当睡眠时间到了，会解除阻塞，进入可运行状态，等待CPU的到来。睡眠不释放锁（如果有的话）。
- **wait方法：**是Object的方法，必须与synchronized关键字一起使用，线程进入阻塞状态，当notify或者notifyall被调用后，会解除阻塞。但是，只有**重新占用互斥锁**之后才会进入可运行状态。睡眠时，会释放互斥锁。

join 方法：当前线程调用，则其它线程全部停止，等待当前线程执行完毕，接着执行。

yield 方法：该方法使得线程放弃当前分得的 CPU 时间。但是不使线程阻塞，即线程仍处于可执行状态，随时可能再次分得 CPU 时间。

解析：

这个题目**主要是考察** sleep和wait方法所处的类是哪个，并且考察其在休眠的时候**对于互斥锁**的处理。

(5) 线程活性故障有哪些？

答：由于资源的稀缺性或者程序自身的问题导致线程**一直处于非Runnable状态**，并且其处理的任务**一直无法完成的现象**被称为是线程活性故障。常见的线程活性故障包括**死锁，锁死，活锁与线程饥饿**。

解析：

每一个线程都有其特定的任务处理逻辑。由于资源的稀缺性或者资源本身的一些特性，导致多个线程需要共享一些排他性资源，比如说处理器，数据库连接等。当出现资源争用的时候，部分线程会进入等待状态。接下来，让我们依次介绍各种形式的线程活性故障吧。

线程死锁：（重点掌握）

死锁是最常见的一种线程活性故障。死锁的起因是多个线程之间相互等待对方而被永远暂停（处于非Runnable）。死锁的产生必须满足如下**四个必要条件**：

- **资源互斥：**一个资源每次只能被一个线程使用
- **请求与保持条件：**一个线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**：线程已经获得的资源，在未使用完之前，不能强行剥夺
- **循环等待条件**：若干线程之间形成一种头尾相接的循环等待资源关系

那么，如何避免死锁的发生？

24

- **粗锁法**：使用一个粒度粗的锁来消除“请求与保持条件”，缺点是会明显降低程序的并发性能并且会导致资源的浪费。
- **锁排序法**：（必须回答出来的点）

指定获取锁的顺序，比如某个线程只有获得A锁和B锁，才能对某资源进行操作，在多线程条件下，如何避免死锁？

通过指定锁的获取顺序，比如规定，只有获得A锁的线程才有资格获取B锁，按顺序获取锁就可以避免死锁。这通常被认为是解决死锁很好的一种方法。

- 使用**显式锁**中的**ReentrantLock.try(long,TimeUnit)**来申请锁

死锁总结：

关于线程活性故障中最常见的死锁，我们必须熟悉其产生的4个必要条件，根据必要条件还应该掌握其避免死锁的方法，锁排序法请大家务必熟练掌握。

线程锁死：

线程锁死是另一种常见的线程活性故障，与线程死锁不可以混为一谈。**线程锁死的定义如下：**

线程锁死是指等待线程由于唤醒其所需的条件永远无法成立，或者其他线程无法唤醒这个线程而一直处于非运行状态（线程并未终止）导致其任务一直无法进展。

线程死锁和线程锁死的外部表现是一致的，即故障线程一直处于非运行状态使得其所执行的任务没有进展。但是锁死的产生条件和线程死锁不一样，即使产生死锁的4个必要条件都没有发生，线程锁死仍然可能已经发生。

线程锁死分为了如下两种：

- **信号丢失锁死：**

信号丢失锁死是因为没有对应的通知线程来将等待线程唤醒，导致等待线程一直处于等待状态。

典型例子是等待线程在执行Object.wait()/Condition.await()前**没有对保护条件进行判断**，而此时保护条件实际上可能**已经成立**，此后可能并无其他线程更新相应保护条件涉及的共享变量使其成立并通知等待线程，这就使得等待线程一直处于等待状态，从而使其任务一直无法进展。

- **嵌套监视器锁死：**

嵌套监视器锁死是由于嵌套锁导致等待线程永远无法被唤醒的一种故障。

比如一个线程，只释放了内层锁Y.wait()，但是没有释放外层锁X；但是通知线程必须先获得外层锁X，才可以通过Y.notifyAll()来唤醒等待线程，这就导致出现了嵌套等待现象。

活锁：

活锁是一种特殊的线程活性故障。当一个线程一直处于运行状态，但是其所执行的任务却没有任何进展称为活锁。比如，一个线程一直在申请其所需要的资源，但是却无法申请成功。

线程饥饿：

线程饥饿是指线程一直无法获得其所需的资源导致任务一直无法运行的情况。线程调度模式有公平调度和非公平调度两种模式。**在线程的非公平调度模式下**，就可能出现线程饥饿的情况。

线程活性故障总结：

- 线程饥饿发生时，如果线程处于可运行状态，也就是其一直在申请资源，那么就会转变为活锁
- 只要存在一个或多个线程因为获取不到其所需的资源而无法进展就是线程饥饿，所以线程死锁其实也算是线程饥饿

总结：

多线程并发编程在Java中占有重要地位，所涉及的知识点也相对较多，限于篇幅，我们在**Java进阶篇章中分三个小节进行学习**。在本小节中，主要讲解了多线程并发编程的一些入门知识点，包括线程与进程的区别，线程的状态以及线程活性故障等。本小节所述均属于**并发编程基础知识点，在面试中出现频率很高**，希望大家可以加强理解与掌握，在面试中务必准确阐述。下一小节中，我们将重点讲述原子性，可见性，有序性以及内部锁与显式锁之间的区别与联系等知识点。

限于作者水平，文章中难免会有不妥之处。大家在学习过程中遇到我没有表达清楚或者表述有误的地方，欢迎随时在文章下边指出，我会及时关注，随时改正。另外，大家有任何话题都可以在下边留言，我们一起交流探讨。

讨论

评论

	<div>offer速来~</div> <div>大概多久能更完呀</div> <div>发表于 2019-11-27 12:55:56</div>	<div>1#</div> <div>赞(1) 回复(0)</div>
	<div>不休的多莉王</div> <div>第七节咋看不到呢</div> <div>发表于 2019-11-28 01:12:03</div>	<div>2#</div> <div>赞(0) 回复(0)</div>
	<div>刘畅201904211606816</div> <div>打卡</div> <div>发表于 2019-12-30 22:04:48</div>	<div>3#</div> <div>赞(0) 回复(0)</div>
	<div>ljin1018</div> <div>打卡</div> <div>发表于 2020-01-02 11:22:18</div>	<div>4#</div> <div>赞(0) 回复(0)</div>
	<div>大王叫我来巡offer</div> <div>打卡</div> <div>发表于 2020-01-03 09:58:21</div>	<div>5#</div> <div>赞(0) 回复(0)</div>

- 

牧水s N

打卡

发表于 2020-01-15 15:15:12

赞(0)

回复(0)
- 

柳杰201905011049420

叮

发表于 2020-01-15 15:53:07

赞(0)

回复(0)
- 

StarHai

kaka

发表于 2020-01-29 19:50:08

赞(0)


回复(0)
- 

小源20190118151956 N

打卡

发表于 2020-02-17 13:48:37

赞(0)

回复(0)
- 

星如月勿忘初心

打卡

发表于 2020-02-17 14:56:59

赞(1)

回复(0)
- 

星如月勿忘初心

打卡

发表于 2020-02-19 15:10:26

赞(1)

回复(0)
- 

牛客248955670号

打卡

发表于 2020-02-25 14:29:56

赞(0)

回复(0)
- 

Gaido

关于避免死锁的部分，使用显式锁中的ReentrantLock.try(long,TimeUnit)的try()是写错了吗？

发表于 2020-02-29 16:02:10

赞(0)

回复(0)
- 

老宋啊啊啊

打卡

发表于 2020-03-05 23:01:28

赞(0)

回复(0)
- 

牛客450070422号

打卡

发表于 2020-03-10 10:48:00

赞(0)

回复(0)

i偏闹




打卡

发表于 2020-04-07 15:00:30

16#

赞(0) 回复(0)



i偏闹

产生死锁的四个条件 资源互斥 请求与保持条件 不剥夺条件 循环等待条件

发表于 2020-04-07 15:02:00

17#

24

赞(1) 回复(0)




怡乐未央

没讲明白啊

发表于 2020-04-13 19:23:20

18#

赞(0) 回复(0)



L201911221558754

一周之前面试官问了我进程与线程的区别，我记得回答了一句引进线程是为了减少时空开销，增加资源利用率，被hr炮轰问进如何减少时空开销的，最后hr说“进程是资源分配的基本单位；线程是程序执行的基本单位”就可，不必回答其他。

发表于 2020-04-15 16:15:09

19#

赞(0) 回复(1)


我是祖国的花朵  作者： 谁家hr这么霸气

2020-05-22 13:15:08

赞(0) 回复(0)

请输入你的观点

回复



L201911221558754

进程通信方式分为低级通信方式和高级通信方式，PV操作为低级通信方式，高级通信方式有管道通信，信息传递，共享存储，共享存储又分为两种，低级方式的共享是基于数据结构的共享，高级方式的共享是基于存储区的共享

发表于 2020-04-15 16:42:09

20#

赞(0) 回复(0)