并发编程初级面试题

作者：xiaokedamowang
原文链接：https://ld246.com/article/1629410713127
来源网站：链滴

并发编程初级面试题

一. 什么是进程和什么是线程

进程是操作系统资源分配和调度的基本单位, 也可以说一个程序就是一个进程, 可以看成是程序的实例

线程是操作系统资源分配和调度的最小单位, 他被包含在进程之中, 是进程中的实际运作单位

二. 线程的状态

- new(新建)
- runnable(运行)
- blocked(阻塞)
- waiting(等待)
- timed_waiting(超时等待)
- terminated(终结)

三. wait和sleep的区别

- wait是Object的方法, 调用wait会释放锁
- sleep是Thread的方法, 调用sleep不会释放锁

四. 创建线程的方式

1. 继承Thread, 重写run方法
2. 实现Runnable, 扔到thread里运行
3. 实现Callable, 扔到FutureTask中, 再把FutureTask扔到thread里运行
4. 使用线程池创建

五. 线程池的创建方式和7个参数

1. Executors.newCachedThreadPool()

全是救急线程的线程池

2. Executors.newFixedThreadPool()

可以设置线程数量的线程池

3. Executors.newScheduledThreadPool()

定时执行任务的线程池

4. Executors.newSingleThreadExecutor()

创建只有1个线程的线程池

5. 手动 new ThreadPoolExecutor(...)

根据阿里巴巴开发手册, 不准使用以上方法创建线程池, 所以需要手动创建线程池, 手动创建线程池就要了解创建线程池的7个参数

1. 核心线程数量
2. 最大线程数量
3. 存活时间
4. 时间单位
5. 阻塞队列
6. 线程工厂
7. 拒绝策略(队列满了之后的行为)
    1. 丢弃任务, 并且抛异常
    2. 在调用者线程执行任务, 如果线程池已经shutdown, 则丢弃任务
    3. 丢弃等待最久的任务, 把当前任务加入进去
    4. 丢弃任务, 不抛异常, 什么都不做

六. ForkJoinPool

ForkJoinPool的核心思想是大任务拆成小任务, 最终合并结果

ForkJoinPool使用工作窃取算法, 当某个线程的任务队列中没有可执行任务的时候, 从其他线程的任务列中窃取任务来执行, 可以充分的压榨CPU资源

七. 常用的并发工具类

1. ReentrantLock
2. AtomicInteger
3. LongAdder
4. CopyOnWriteArrayList
5. CopyOnWriteArraySet
6. ConcurrentHashMap
7. CountDownLatch
8. CyclicBarrier
9. Semaphore
10. BlockingQueue
11. 线程池

八. CountDownLatch和CyclicBarrier的区别

CountDownLatch	CyclicBarrier
减数字	加数字

数字为0时释放所有等待的线程	数字加到规定的值
释放所有等待线程, 并且可以执行构造函数传入的任务(**可选**)	
数字无法重置	数字加到规定的值时, 重新从0开始加
调用await()方法只进行阻塞	调用await()方法数
加1, 如果加完数字还是没有达到规定的值, 阻塞等待	
不可重复利用	可重复利用

九. CAS和volatile

CAS是乐观锁, 先比较, 再修改, 内存位置(JAVA中的内存地址, V), 旧的预期值(A)和新值(B), 如果V和A样就改成B, JDK9之前CAS是通过Unsafe对象操作, JDK9之后推荐通过VarHandle, 所以JUC包里面的AS操作都变了

volatile是java的关键字, 作用主要是保证内存可见性和禁止指令重排序

十. synchronized和ReentrantLock的区别

1. synchronized是java关键字, ReentrantLock是Lock接口的实现类
2. synchronized在发生异常时, 会释放锁, ReentrantLock不会
3. ReentrantLock可以打断等待, synchronized不行
4. ReentrantLock可以知道有没有获取锁, synchronized不行
5. ReentrantLock是乐观锁, 而且可以控制是否公平, synchronized严格来说是悲观锁(锁升级之后)
6. ReentrantLock有Condition, 可以唤醒特定的组

十一. ReentrantReadWriteLock读写锁

1. 读读共享, 读写, 写写互斥
2. 可能造成线程饥饿
3. 写锁可以降级到读锁, 读锁不能升级成写锁
4. JDK8加入了新的读写锁, StampLock, 支持乐观读

十二. Java中如何获取到线程dump文件

1. 获取到线程的pid，可以通过使用jps命令，在Linux环境下还可以使用ps -ef | grep java
2. 打印线程堆栈，可以通过使用jstack pid命令，在Linux环境下还可以使用kill -3 pid
3. 使用jconsole查看

十三. Java中用到的线程调度算法是什么

Java虚拟机采用抢占式调度模型

十四. LockSupport

LockSupport.park不需要获取锁资源就能阻塞线程, 而且可以精确控制唤醒的线程, 并且可以在线程

有阻塞之前提前unpark, 这时候线程运行到park方法的时候不会阻塞, JUC工具包里的阻塞等待就是靠ark方法实现

十五. ThreadLocal

每个线程内部有1个ThreadLocalMap, 当调用threadLocal的get, set, remove方法时, 在方法内部拿当前线程, 从线程中获取ThreadLocalMap, 最后以threadLocal(弱引用)为key, 从中取出数据

使用ThreadLocal是线程安全的, 但是使用完需要及时删除, 否则会造成一定程度的内存泄漏或浪费(因ThreadLocalMap的key使用的是弱引用包装的threadLocal)

十六. synchronized锁升级

1. 无锁

无锁就是正常对象状态, 一般新创建的对象状态为无锁可偏向

2. 偏向锁

在对象头的mark word上记录线程ID, 代表当前线程获取了锁

1. 如果发生锁竞争, 会升级成轻量级锁
2. 如果在synchronized之外调用hashcode方法, 锁会升级成轻量级锁
3. 如果在synchronized之内调用hashcode方法, 锁会升级成重量级锁
4. 偏向锁同一线程ID修改次数达到20次后会发生批量重偏向(剩余锁对象的偏向全部修改成该线程I)
5. 偏向锁同一线程ID修改次数达到40次后会关闭整个类的所有对象的偏向锁, 变成不可偏向状态

3. 轻量级锁 + 自旋锁

对象头的mark word上会记录线程栈中的锁记录引用, 如果多次CAS操作失败, 会升级成重量级锁

CAS次数JDK6之前是默认10次, JDK6的时候是自适应次数

4. 重量级锁(管程Monitor)

在对象头的mark word上记录Monitor的引用, 没有获取锁的线程会加入Monitor的EntryList中, 调用ait的线程会加入WaitSet中, Owner记录的是锁的拥有者

5. 锁消除

synchronized的对象是局部变量, 并且JVM发现没有逃出, 会优化成不加锁

6. 锁粗化

当一段代码中有2个或多个synchronized代码块, 并且中间的代码非常少, 而且运行的很快, 会优化成个synchronized

十七. AQS

AbstractQueuedSynchronizer 简称AQS, 是个抽象类, JUC大部分工具类都依赖他实现, 内部拥有1个olatile修饰是state和1个等待队列一对头尾节点(链表实现队列), 通过CAS修改state来判断是否获取锁, 如果没有获取锁, 把当前线程加入等待队列并且park阻塞

AQS支持两种同步方式, 公平和非公平