多线程

0 线程和进程区别?

进程:操作系统资源分配的基本单位; 程序之间切换有较大开销; 一个进程内有多个线程;

线程:任务调度和执行的基本单位;线程之间切换的开销小;线程是进程的一部分,所以也叫轻量级进程;

1 实际工作应用?

复杂计算会用到多线程;

数据迁移;

任务调度;

2 创建线程的方法? 线程状态?

Start():启动一个线程,线程处于就绪状态,没有运行;

Run():当做普通方法调用,程序中只有主线程跑,等run(执行完毕之后,才可以继续执行下面的代码;

3 线程池原理? 参考:<https://www.cnblogs.com/dongguacai/p/6030187.html>

ThreadPoolExecutor类中属性: core size; max size; Blocking Queue; time; handler(拒绝策略)

拒绝策略:

默认的: 直接抛出异常;

只用调用所在的线程运行任务;

丢弃队列里最近的一个任务,执行当前任务; 不处理,丢弃掉;

4 Lock锁与Synchronized区别?

Synchronized: java关键字,JVM层面实现了对临界资源的同步互斥访问,通过对对象的头文件来操作;从而达到加锁和释放锁;

**代码执行时出现异常,JVM会自动释放锁;**

缺点: 1 不能响应中断 2同一时刻不管是读还是写都只能有一个线程对共享资源操作,其他线程只能等待; 锁的释放由虚拟机来完成,会导致线程阻塞后其他线程会一直等待,性能不高;

Lock: 基于jdk实现的接口; ReentrentLock 和 ReentrentReadWriteLock 都是可重入锁: 当同一线程获取到锁之后,他在不释放锁的情况下,可以再次获取当前已经拿到的锁,只需标记获取到锁的次数加一即可;

**Lock通过代码实现的,要保证锁一定会被释放,就必须将unLock()放到finally{}中;**

通过在读多,写少的高并发下, 使用ReentrantReadWriteLock 获取读锁和写锁提高系统性能;因为读锁是共享锁，即可以同时有多个线程读取共享资源，而写锁则保证了对共享资源的修改只能是单线程的。

公平锁: 当A把锁完全释放后,state恢复为0,然后会通知队列唤醒B线程节点,使得B可以再次竞争锁,B后面有C,C继续休眠,除非B 执行完了,通知C线程;

非公平锁:当A执行完之后,要唤醒线程B是需要时间的,等线程B醒来还要再次竞争锁,假如再来个线程C,那么C获得锁之后,B 就只能休眠了;

5 为什么使用可重入锁?

ReentrantLock是一个可重入的互斥/独占锁;通过自定义队列同步器(AQS-AbstractQueueSychronized)来实现锁的获取与释放;

独占:在同一时间只能有一个线程获取到锁,而其他获取锁的线程只能处于同步队列中等待,只有获取锁的线程释放了锁,后继的线程才能获取锁;

可重入:支持重进入的锁,任意线程在获取锁之后能够再次获取该锁而不被锁阻塞;

两个好处: 中断响应; 锁申请等待限时;

锁的申请默认都是非公平的----非公平锁; 公平锁---按照时间的先后顺序,保证先到先得(不会产生饥饿现象);

应用场景:

1 如果发现操作已经执行,则尝试等待一段时间,等待超时,则不执行;

2如果发现操作已经加锁,则等待一个一个加锁;

3可中断锁;

-------------------------------------------------------------------------------------------------------------------------------

5 CAS 实现原理? 不是很理解;