1.几个重要概念

1.同步(synchronous)和异步(asynchronous)　　形容一次方法调用

2.并发(Concurrency)和并行(Parallelism)

3.临界区

4.阻塞(Blocking)和非阻塞(Non-Blocking),　　形容线程间的影响

5.死锁(Deadlock),饥饿(Starvation)和活锁(Livelock)　　多线程的活跃性问题

6.并发的级别

-阻塞

-无饥饿

-无障碍　非阻塞

-无锁　　非阻塞

-无等待　非阻塞

阻塞:

-当一个线程进入临界区后,其他线程必须等待,synchronized关键字或者重入锁

无饥饿:

-如果线程之间有优先级,那么线程调度的时候总是会倾向于先满足高优先级的线程,资源分配不公平

无障碍:

-无障碍是一种最弱的非阻塞调度

-自由出入临界区　(乐观态度)

-无竞争,有限步内完成操作

-有竞争时,回滚数据

无锁(Lock-Free): 必须是无障碍的(基于无障碍)

-是无障碍的

-保证有一个线程可以胜出

例子:

```
while(!atomicVar.compareAndSet(localVar,localVar++){
   localVar = atomicVar.get()
}
```

无等待(Wait-Free):(基于无锁的)

-无锁的

-要求所有的线程都必须在有限步内完成

-无饥饿的

-一种典型的无等待结构:RCU(Read Copy Update)

2.并行的2个重要定律

Amdahl定律(阿姆达尔定律)

定义了加速比,串行改并行后加速比

Gustafson定律(古斯塔夫森)

3.回到Java:JMM

JMM的关键技术点都是围绕着多线程的原子性,可见性和有序性来建立的

3.1 原子性

指一个操作是不可中断的.即使是在多个线程一起执行的时候,一个操作一旦开始,就不会被其他线程干扰.

3.2 可见性

可见性是指当一个线程修改了某一个共享变量的值时,其他线程是否能够立即知道这个修改,由于编译器优化或者硬件优化的缘故,变量缓存在cache中或者寄存器里,这样,两个线程就无法意识到变量已经改变了,

会导致可见性问题的优化:缓存优化,硬件优化,指令重排,编辑器的优化

3.3 有序性

对于一个线程的执行代码而言,我们总是习惯地认为代码是从前向后依次执行的,但是,在并发时,会出现乱序,有序性问题地原因是程序在执行时,可能会进行指令重排,重排的指令与原指令顺序未必一致

那些指令不能重排:.Happen-Before规则

1. 程序顺序原则:一个线程内保证语义的串行性
2. volatile原则:volatile变量的写先于读发生,这保证了volatile变量的可见性
3. 锁规则:解锁(unlock)必然发生在随后的加锁(lock)
4. 传递性:A先于B,B先于C,那么A必然先于C
5. 线程的start()方法先于它的每一个动作
6. 线程的所有操作先于线程的终结(Thread.join())
7. 线程的中断(interrupt())先于被中断线程的代码
8. 对象的构造函数的执行,结束先于finalize()方法