**原子性和可见性**

1,volatile是一个类型修饰符，他是被设计用来修饰被不同线程访问和修改的变量，被volatile定义的变量，系统每次用到他时都是直接从对应的内存当中读取的，而不是利用缓存。

**volatile关键字就是为了保证内存数据的可见性**

**但是volatile和Synchronized具有不同姓：**

**1，相对于synchronized是一种较为轻量级的同步策略 2，synchronized具有互斥性，也就是两个线程同时抢一个锁，当一个线程进去了，另外一个线程就进不去，只有等一个线程出来了，另外一个线程才能进去**

**3，volatile就不具备“互斥性”如果有一个线程在访问共享数据，另外一个线程也可以同时访问共享数据，只不过所有的访问数据都在主存中完成而已**

1. **volatile不能保证变量的原子性<http://www.importnew.com/18126.html>**

如果让volatile保证原子性，必须符合以下两条规则：

运算结果并不依赖于变量的当前值，或者能够确保只有一个线程修改变量的值，2，变量不需要与其他的状态变量共同参与不变约束 3**，synchronized: 具有原子性，有序性和可见性； volatile：具有有序性和可见性**

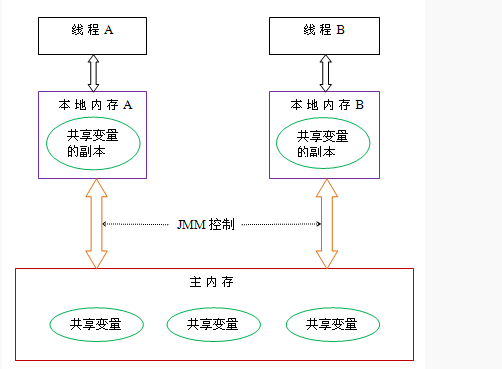
当程序在运行过程中，会将运算需要的数据从主存复制一份到CPU的高速缓存当中，那么CPU进行计算时就可以直接从它的高速缓存读取数据和向其中写入数据，当运算结束之后，再将高速缓存中的数据刷新到主存当中。举个简单的例子，比如下面的这段代码：

|  |  |
| --- | --- |
| 1 | i = i + 1; |

当线程执行这个语句时，会先从主存当中读取i的值，然后复制一份到高速缓存当中，然后CPU执行指令对i进行加1操作，然后将数据写入高速缓存，最后将高速缓存中i最新的值刷新到主存当中。

**5.java内存模型**

java内存模型(JMM)是线程间通信的控制机制.JMM定义了**主内存和线程之间抽象关系。线程之间的共享变量存储在主内存（main memory）中，每个线程都有一个私有的本地内存（local memory），本地内存中存储了该线程以读/写共享变量的副本。**本地内存是JMM的一个抽象概念，并不真实存在。它涵盖了缓存，写缓冲区，寄存器以及其他的硬件和编译器优化。Java内存模型的抽象示意图如下：

线程在工作内存进行操作后何时会写到主内存中？这个时机对普通变量是没有规定的，而针对volatile修饰的变量给java虚拟机特殊的约定，线程对volatile变量的修改会立刻被其他线程所感知，即不会出现数据脏读的现象，从而保证数据的“可见性”。现在我们有了一个大概的印象就是：**被volatile修饰的变量能够保证每个线程能够获取该变量的最新值，从而避免出现数据脏读的现象。**

线程私有的：程序计数器，虚拟机栈，本地方法栈, 线程共享的：堆，方法区