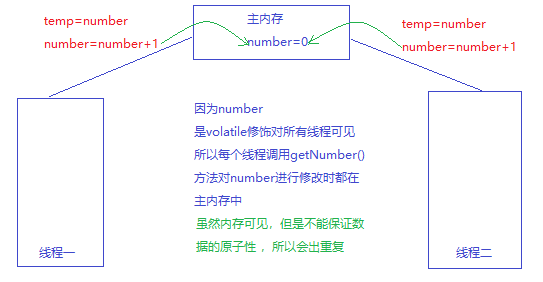
**原子变量-CAS算法**

* **原子变量-CAS算法**



以上程序代码虽然保证了内存的可见性，但是当多个线程对同一数据的更改，并没有保证数据的安全性，会出现重复的值，原因：



CAS (Compare-And-Swap) 是一种硬件对并发的支持，针对多处理器操作而设计的处理器中的一种特殊指令，用于管理对共享数据的并发访问。

CAS 是一种无锁的非阻塞算法的实现。

CAS 包含了 3 个操作数：

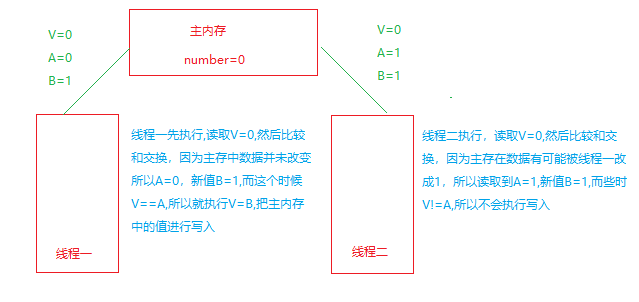
需要读写的内存值 V

进行比较的值 A

拟写入的新值 B

当且仅当 V 的值等于 A 时，CAS 通过原子方式用新值 B 来更新V的值，否则不会执行任何操作。





原子变量

类的小工具包，支持在单个变量上解除锁的线程安全编程。事实上，此包中的类可将 volatile 值、字段和数组元素的概念扩展到那些也提供原子条件更新操作的类。

类 AtomicBoolean、AtomicInteger、AtomicLong 和 AtomicReference 的实例各自提供对相应类型单个变量的访问和更新。每个类也为该类型提供适当的实用工具方法。

AtomicIntegerArray、AtomicLongArray 和 AtomicReferenceArray 类进一步扩展了原子操作，对这些类型的数组提供了支持。这些类在为其数组元素提供 volatile 访问语义方面也引人注目，这对于普通数组来说是不受支持的。

核心方法：boolean compareAndSet(expectedValue, updateValue)

java.util.concurrent.atomic 包下提供了一些原子操作的常用类:

AtomicBoolean 、AtomicInteger 、AtomicLong 、 AtomicReference

AtomicIntegerArray 、AtomicLongArray

AtomicMarkableReference

AtomicReferenceArray

AtomicStampedReference