CAS底层实现

CAS的全称为**Compare-And-Swap** ,它是一条CPU并发原语。它的功能是判断内存某个位置的值是否为预期值,如果是则更新为新的值,这个过程是原子的。

CAS 操作包含三个操作数 —— 内存位置（V）、预期原值（A）和新值(B)。 如果内存位置的值与预期原值相匹配，那么处理器会自动将该位置值更新为新值 。否则，处理器不做任何操作。

- **变量ValueOffset**：它是该变量在内存中的偏移地址,因为UnSafe就是根据内存偏移地址获取数据的
- **变量value**：被volatile修饰,保证了多线程之间的可见性.

CAS通过调用JNI的代码实现的。JNI:Java Native Interface为JAVA本地调用，允许java调用其他语言。

Unsafe类中的compareAndSwapInt，是一个本地方法，该方法的实现位于`unsafe.cpp`中，而compareAndSwapInt就是借助C来调用CPU底层指令实现的。

1. 先想办法拿到变量value在内存中的地址。
2. 通过`Atomic::cmpxchg`实现比较替换，其中参数x是即将更新的值，参数e是原内存的值。

CAS的原子性实际上是CPU实现的. 其实在这一点上还是有排他锁的. 只是比起用synchronized, 这里的排他时间要短的多. 所以在多线程情况下性能会比较好.

缺点：
CAS虽然很高效的解决原子操作，但是CAS仍然存在三大问题。ABA问题，循环时间长开销大和只能保证一个共享变量的原子操作。