### 想必大家平时都见过volatile关键字，可是大家知道什么时候需要使用volatile关键字吗？

直接看下面代码：

Int a = 100;

While(a == 100){

//code

}

这段程序编译时，如果编译器发现程序始终没有企图改变a的值，那它可能就会优化这段代码，变成while(true)的死循环使得程序执行的更快，然而这种优化有时候会变成过度优化，编译器有时候可能没有意识到程序会改变a的值，却做了这种优化导致程序没有产生预期的行为。

这里为了产生预期的行为，需要阻止编译器做这种优化，可以使用volatile关键字修饰

volatile int a = 100;

volatile关键字和const关键字相对应，const关键字告诉编译器其修饰的变量是只读的，编译器根据只读属性做一些操作，而volatile关键字告诉编译器其修饰的变量是易变的，同理编译器根据易变属性也会做一些操作。它会确保修饰的变量每次都读操作都从内存里读取，每次写操作都将值写到内存里。volatile关键字就是给编译器做个提示，告诉编译器不要对修饰的变量做过度的优化，提示编译器该变量的值可能会以其它形式被改变。

volatile修饰结构体时，结构体的成员也是volatile的吗？

答案是结构体内所有的都是volatile。

这里大体可以理解为一个对象是volatile，那对象里所有的成员也都是volatile。其实const和volatile可以理解为是硬币的两面，我们经常听到看到传说中的CV修饰词就是const和volatile关键字。

**volatile可以保证原子性吗**

想必大家都知道答案，volatile只保证内存可见性，不能保证操作是原子的，拿i++举例：

volatile int i = 0;

i++; // i = i + 1

i++ 相当于i=i+1，而i=i+1其实可以分解为好几步：

先读取i的值到tmp

增加tmp的值

把tmp的值写回到i的地址里

而volatile只能保证内存可见，可以理解为上述三步中的每一步都是原子的，但是三步合起来却不一定是原子的，因为在多线程中三步中间可能插入一些其它操作改变了预期的行为，所以volatile不能用在多线程中，多线程中的原子操作还是需要使用atomic。单例模式的double check方法中instance变量为什么需要使用volatile修饰也是这个原理。

**小总结**

**tips：**volatile不能解决多线程安全问题，针对特种内存才需要使用volatile，它和atomic的特点如下：

• std::atomic用于多线程访问的数据，且不用互斥量，用于并发编程中• volatile用于读写操作不可以被优化掉的内存，用于特种内存中

#### **参考资料**

https://stackoverflow.com/questions/4437527/why-do-we-use-volatile-keyword

https://stackoverflow.com/questions/72552/why-does-volatile-exist

https://stackoverflow.com/questions/4479597/does-making-a-struct-volatile-make-all-its-members-volatile/4479652

https://stackoverflow.com/questions/19744508/volatile-vs-atomic

https://blog.csdn.net/bdss58/article/details/44813597