Volatile

【基本作用】

volatile是修饰变量的关键字，被volatile修饰的变量说明该变量会被编译器的其他因素而改变值，比如操作系统，硬件，线程等。编译器对访问该变量的代码不做优化，变量的访问只能从内存中访问，从而提供对特殊地址的稳定访问。

优化编译器的做法是，对于需要重复访问的变量优化成只访问一次。比如：

```
volatile int i=10;
int a = i;
    ...
// 其他代码，并未明确告诉编译器，对 i 进行过操作
int b = i;
```

volatile 指出 i 是随时可能发生变化的，每次使用它的时候必须从 i的地址中读取，因而编译器生成的汇编代码会重新从i的地址读取数据放在 b 中。而优化做法是，由于编译器发现两次从 i读数据的代码之间的代码没有对 i 进行过操作，它会自动把上次读的数据放在 b 中。而不是重新从 i 里面读。这样以来，如果 i是一个寄存器变量或者表示一个端口数据就容易出错，所以说 volatile 可以保证对特殊地址的稳定访问。注意，在 VC 6 中，一般调试模式没有进行代码优化，所以这个关键字的作用看不出来

【声明】

int volatiole vInt;

【volatile指针】

与const修饰词一样，const有常量指针和指针常量的说法，volatile也有相应的概念。

注意：

(1) 可以把一个非**volatile int**赋给**volatile int**，但是不能把非**volatile**对象赋给一个**volatile**对象。

(2) 除了基本类型外，对用户定义类型也可以用**volatile**类型进行修饰。

(3) C++中一个有**volatile**标识符的类只能访问它接口的子集，一个由类的实现者控制的子集。用户只能用**const_cast**来获得对类型接口的完全访问。此外，**volatile**向**const**一样会从类传递到它的成员。

【多线程下的volatile指针】

有些变量是用volatile关键字声明的。当两个线程都要用到某一个变量且该变量的值会被改变时，应该用volatile声明，该关键字的作用是防止优化编译器把变量从内存装入CPU寄存器中。如果变量被装入寄存器，那么两个线程有可能一个使用内存中的变量，一个使用寄存器中的变量，这会造成程序的错误执行。volatile的意思是让编译器每次操作该变量时一定要从内存中真正取出，而不是使用已经存在寄存器中的值，如下：

volatile BOOL bStop = FALSE;
(1) 在一个线程中：
while(!bStop) { ... }
bStop = FALSE;
return;

(2) 在另外一个线程中，要终止上面的线程循环：
bStop = TRUE;
while(bStop); //等待上面的线程终止，如果bStop不使用volatile申明，那么这个循环将是一个死循环，因为bStop已经读取到了寄存器中，寄存器中bStop的值永远不会变成FALSE，加上volatile，程序在执行时，每次均从内存中读出bStop的值，就不会死循环了。 这个关键字是用来设定某个对象的存储位置在内存中，而不是寄存器中。因为一般的对象编译器可能会将其的拷贝放在寄存器中用以加快指令的执行速度