.volatile的本质：

1> 编译器的优化

在本次线程内, 当读取一个变量时，为提高存取速度，编译器优化时有时会先把变量读取到一个寄存器中；以后，再取变量值时，就直接从寄存器中取值；当变量值在本线程里改变时，会同时把变量的新值copy到该寄存器中，以便保持一致。

当变量在因别的线程而改变了值，该寄存器的值不会相应改变，从而造成应用程序读取的值和实际的变量值不一致。

当该寄存器在因别的线程而改变了值，原变量的值不会改变，从而造成应用程序读取的值和实际的变量值不一致。

2>volatile应该解释为“直接存取原始内存地址”比较合适，“易变的”这种解释简直有点误导人。

volatile 影响编译器编译的结果,指出，volatile 变量是随时可能发生变化的，与volatile变量有关的运算，不要进行编译优化，以免出错，（VC++ 在产生release版可执行码时会进行编译优化，加volatile关键字的变量有关的运算，将不进行编译优化。）。   
例如：   
volatile int i=10;   
int j = i;   
...   
int k = i;   
volatile 告诉编译器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，因而编译器生成的可执行码会重新从i的地址读取数据放在k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k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，不会出错。

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

一个定义为volatile的变量是说这变量可能会被意想不到地改变，这样，编译器就不会去假设这个变量的值了。精确地说就是，优化器在用到这个变量时必须每次都小心地重新读取这个变量的值，而不是使用保存在寄存器里的备份。下面是volatile变量的几个例子：   
1) 并行设备的硬件寄存器（如：状态寄存器）   
2) 一个中断服务子程序中会访问到的非自动变量(Non-automatic variables)   
3) 多线程应用中被几个任务共享的变量   
回 答不出这个问题的人是不会被雇佣的。我认为这是区分C程序员和嵌入式系统程序员的最基本的问题。搞嵌入式的家伙们经常同硬件、中断、RTOS等等打交道， 所有这些都要求用到volatile变量。不懂得volatile的内容将会带来灾难。假设被面试者正确地回答了这是问题（嗯，怀疑是否会是这样），我将 稍微深究一下，看一下这家伙是不是直正懂得volatile完全的重要性。   
1)一个参数既可以是const还可以是volatile吗？解释为什么。   
2); 一个指针可以是volatile 吗？解释为什么。   
3); 下面的函数有什么错误：   
int square(volatile int \*ptr)   
{   
return \*ptr \* \*ptr;   
}   
下面是答案：   
1)是的。一个例子是只读的状态寄存器。它是volatile因为它可能被意想不到地改变。它是const因为程序不应该试图去修改它。   
2); 是的。尽管这并不很常见。一个例子是当一个中服务子程序修改一个指向一个buffer的指针时。   
3) 这段代码有点变态。这段代码的目的是用来返指针\*ptr指向值的平方，但是，由于\*ptr指向一个volatile型参数，编译器将产生类似下面的代码：   
int square(volatile int \*ptr)   
{   
int a,b;   
a = \*ptr;   
b = \*ptr;   
return a \* b;   
}   
由于\*ptr的值可能被意想不到地该变，因此a和b可能是不同的。结果，这段代码可能返不是你所期望的平方值！正确的代码如下：   
long square(volatile int \*ptr)   
{   
int a;   
a = \*ptr;   
return a \* a;   
}   
位操作（Bit manipulation）

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

嵌入式编程中经常用到 volatile这个关键字，在网上查了下他的用法可以归结为以下两点：

**一：告诉compiler不能做任何优化**

   比如要往某一地址送两指令：   
   int \*ip =...; //设备地址   
   \*ip = 1; //第一个指令   
   \*ip = 2; //第二个指令   
   以上程序compiler可能做优化而成：   
   int \*ip = ...;   
   \*ip = 2;   
   结果第一个指令丢失。如果用volatile, compiler就不允许做任何的优化，从而保证程序的原意：   
   volatile int \*ip = ...;   
   \*ip = 1;   
   \*ip = 2;   
   即使你要compiler做优化，它也不会把两次付值语句间化为一。它只能做其它的优化。这对device driver程序员很有用。

**二：表示用volatile定义的变量会在程序外被改变,每次都必须从内存中读取，而不能把他放在cache或寄存器中重复使用。**

   如   volatile char a;     
        a=0;   
       while(!a){   
//do some things;     
       }     
       doother();   
   如果没有 volatile doother()不会被执行

分类: [c语言](https://www.cnblogs.com/reality-soul/category/917830.html)