Subject: Re: Use same variable in different threads
Posted by gprentice on Fri, 10 Dec 2010 13:08:20 GMT
View Forum Message <> Reply to Message

No expert here either but when you're sharing data between threads, I think you need both volatile and synchronization.

A mutex ensures that the compiler/linker can't optimise code across the mutex entry, that the cache is flushed and that only one thread can be executing the code the mutex protects (i.e. it's synchronized).

http://msdn.microsoft.com/en-us/library/ms686355(v=VS.85).aspx

volatile ensures that the compiler/linker can't optimise your code and use a cached value. With Microsoft, optimisation can occur at link time. On some platforms you can get away without volatile if you call a global function that the compiler can't see - the compiler has to assume that global function might modify the variable you're sharing so is forced to re-read the variable from memory, but that is't safe with Microsoft.

There's also thread local storage - see thread__

Regarding atomic - on Win32, 32 bits are atomic and on Win64, 64 bits are atomic. On Win32, the atomixXXX functions use the Interlocked... functions that allow you to read/write without being interrupted by another thread etc, and also provide a memory barrier.

Hence I think you need
volatile mySharedData data;
volatile Atomic threadnum;
Anyway, I don't think that using volatile would be wrong, even if it's not always necessary.

Graeme