Subject: Re: Use same variable in different threads
Posted by Didier on Thu, 09 Dec 2010 23:26:02 GMT
View Forum Message <> Reply to Message

Hi Koldo,

I would add one thing to Honza's explanation: 'Memory bariers'

Actually this is not a thing that is used directly by people but it is necessary to understand all that is going on.

Volatile allows you to access a shared variable not protected by mutex and you will be sure it always has the wright value (the cache on the processor might be different for each thread, so in extreme cases you can get de synchronised values of one variable from time to time ..... ==> bugs impossible to find).
Of course if the variable is larger than the processor bus ... the R/W process will not be atomic so you need a mutex !

Memory barriers are points where the cpu cache is flushed so all the synchronisation problems just disappear
In mutexes, such barriers are used to avoid such problems, and therefore when using a mutex lock/unlock procedure, you protect the resource (and it's memory) but you also flush the cash at each mutex lock/unlock.
So .... when using a mutex there is no need to use volatile it might even slow down you're app.

Finally, to answer you're question:
* if the variable (and it's context !) is just an int or smaller it can be accessed atomically ==> use this method, it's the fastest one (volatile will work in most cases but to be sure use the dedicated functions)
* otherwise you have to use the good old mutex, no may out !
  - mutex.lock()
  - do you're thing ..R..W....etc
  - mutex.unlock()

NB: the volatile/synchronisation problems mostly has effects on programs compiled in O3.

Hope this helps you !