Subject: Re: MT again (SOLVED!)
Posted by mdelfede on Fri, 20 Aug 2010 09:45:31 GMT
View Forum Message <> Reply to Message

Well, just for reference I'll post a bug that drove me crazy for long time
Here the sample code :

```
void ThreadCb(void)
{
   INTERLOCKED_(mutex) {
      inThread = true;
   }

   << DO SOME TIME CONSUMING STUFFS>

   INTERLOCKED_(mutex) {
      inThread = false;
}

bool StartThread(void)
{
   bool inTh;
   INTERLOCKED_(mutex) {
      inTh = inThread;
   }
   if(inTh)
      return false;
   myThread.Run(ThreadCb);
   return true;
}
```

Purpose is clear, just start the thread *ONLY* if not already running, return true if success or false if thread was already running.
Where's the caveat ? It's simply in the time BETWEEN the start of ThreadCb() callback and the setting of 'inThread' variable to true. If code calls fast enough the StartThread() routine, it can happen that the Thread callback is in the middle of startup code BUT has still NOT SET the 'inThread' flag; so the StartThread() routine believes thread isn't running and calls it again.
The solution :

```
void ThreadCb(void)
{
   << DO SOME TIME CONSUMING STUFFS>

   INTERLOCKED_(mutex) {
       inThread = false;
}

bool StartThread(void)
{
   bool inTh;
   INTERLOCKED_(mutex) {
       inTh = inThread;
   }

   if(inTh)
       return false;
   inThread = true;
   myThread.Run(ThreadCb);
   return true;
}
```

So, I've moved the 'inThread' flag SETTING from the thread routine to its caller; the reset still belong to thread routine.
Now it's the caller that has the responsibility of setting the variable, so it can't happen that thread routine performs some stuffs between check and launch.

Ciao

Max