Subject: MT again
Posted by mdelfede on Fri, 18 Dec 2009 11:03:01 GMT
View Forum Message <> Reply to Message

```
// calculation thread function
void CalcPage::DoCalculation(void)
{
 int mod;

 INTERLOCKED_(mutex)  {
  modified = false;
 }

 do
 {
  LOG(__FUNCTION__ << " Start calculation");
  // gets the status bar
  StatusBar &sb = ((Lamell *)(GetMainWindow()))->GetStatusBar();
  {
   // signals calculation in progress
   GuiLock __;
   sb.Set("Calcolo in corso");
  }

  calcAborted = !Calculate();
  {
   // signals end of calculation
   GuiLock __;
   if(calcAborted)
    sb.Set("Errore");
   else
    sb.Set("Pronto");
  }
  INTERLOCKED_(mutex) {
   mod = modified;
  }
  LOG(__FUNCTION__ << " End calculation, modified = " << mod);
 }
 while(mod);
}

// modify handler -- triggers page calculation
// on keypress on every (editable) ctrl on page
void CalcPage::ModifyCb(void)
{
 LOG(__FUNCTION__ << " ModifyCb called");
 if(!calcThread.IsOpen())
```

```
 {
  calcThread.Run(THISBACK(DoCalculation));
  return;
 }
 INTERLOCKED_(mutex) {
  modified = true;
 }
}
```

CalcPage::ModifyCb is called on every change on editables ctrls on page (editfields mostly)
DoCalculation thread should take in account changes on page that happens between calculations, and so it should recalc the whole stuff.

My results :
Sometimes it starts just once, then it's never run again (the calc thread), so I guess it's hanging somewhere
Sometimes it keeps recalculating, but ModifyCb is not reentered, so I'm quite sure that the modify flag is not recursively set because of calc routine.

First case happens mostly

Max

Subject: Re: MT again (SOLVED!)
Posted by mdelfede on Fri, 18 Dec 2009 11:51:12 GMT
View Forum Message <> Reply to Message

```
// calculation thread function
void CalcPage::DoCalculation(void)
{
 bool mod;

 INTERLOCKED_(mutex)  {
  inThread = true;
 }
 do
 {
  INTERLOCKED_(mutex)  {
   modified = false;
  }

  // gets the status bar
  StatusBar &sb = ((Lamell *)(GetMainWindow()))->GetStatusBar();
  {
```

```
   // signals calculation in progress
   GuiLock __;
   sb.Set("Calcolo in corso");
  }
  calcAborted = !Calculate();
  {
   // signals end of calculation
   GuiLock __;
   if(calcAborted)
    sb.Set("Errore");
   else
    sb.Set("Pronto");
  }
  INTERLOCKED_(mutex) {
   mod = modified;
  }
 }
 while(mod);
 INTERLOCKED_(mutex) {
  inThread = false;
 }
}

// modify handler -- triggers page calculation
// on keypress on every (editable) ctrl on page
void CalcPage::ModifyCb(void)
{
 bool inTh;
 INTERLOCKED_(mutex) {
  inTh = inThread;
 }
 if(!inTh)
 {
  calcThread.Run(THISBACK(DoCalculation));
  return;
 }
 INTERLOCKED_(mutex) {
  modified = true;
 }
}
```

Well, thanks Mirek !
I'll post the correct code here for reference.

The problem was that IsOpen() just checks that the thread was STARTED right, not if the thread is still active.
To check for it, I added a new variable set inside the thread (inThread).

This snippet allows to have a long time backround calculation without stopping the gui and with results displayed as soon as available.
The thread routine can handle changes in data that happens during the calculation, re-entering itself when finished.

Ciao

Max

Subject: Re: MT again (SOLVED!)
Posted by mirek on Sat, 19 Dec 2009 10:55:09 GMT
View Forum Message <> Reply to Message

mdelfede wrote on Fri, 18 December 2009 06:51The problem was that IsOpen() just checks that the thread was STARTED right, not if the thread is still active.
To check for it, I added a new variable set inside the thread (inThread).

Well, I have spent some time thinking about the issue and came to the conclusion that IsOpen behaviour is in fact correct:

Even if thread is finished (returns from the thread routine), its OS representation still exists until the last reference to the thread is closed.

For example, in such situation, call to Wait returns immediately.

This behaviour is in fact required for correct thread synchronisation - it is always possible that thread finishes quick, but you still need to know that it has started successfully.

I was also thinking about adding IsRunning method, but for now I am against it - such status is very temporal thing, I am afraid that such method is invitation for race conditions to emerge.

Mirek

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

Subject: Re: MT again (SOLVED!)
Posted by koldo on Fri, 20 Aug 2010 10:22:08 GMT
View Forum Message <> Reply to Message

Uuups. Thank you Massimo!

I will have to fix some code

Subject: Re: MT again (SOLVED!)
Posted by mdelfede on Fri, 20 Aug 2010 14:42:16 GMT
View Forum Message <> Reply to Message

koldo wrote on Fri, 20 August 2010 12:22Uuups. Thank you Massimo!

I will have to fix some code

Ehehehehe... I had this damn'd bug in my app since loooong time, and couldn't understand why my calculation routine was being called in the middle.
That scrambled qtf text alot, and happened mostly ONLY on app startup, where controls triggered it very quickly upon loading from file.
It took me one day of debugging to find it.

Happy it's useful for you

Ciao

Max