Subject: MT app hangs
Posted by mdelfede on Tue, 15 Dec 2009 23:29:58 GMT
View Forum Message <> Reply to Message

This app :

```
#include <CtrlLib/CtrlLib.h>

using namespace Upp;

class TestThread : public TopWindow
{
 protected:
  bool stop;
  void buttonCb(void);
  Thread thr;
  virtual void thrCb(void);

  ProgressIndicator progress;
  Button button;
  StatusBar status;

 public:
  typedef TestThread CLASSNAME;

  TestThread();

};

void TestThread::thrCb(void)
{
 {
  GuiLock __;
  status.Set("Running, " + FormatInt(thr.GetCount()) + " threads");
 }
 for(;;)
 {
  {
   GuiLock __;
   if(progress < 100)
    progress++;
   else
    progress = 0;
   if(stop)
    break;
  }
  usleep(100000);
 }
 {
  GuiLock __;
  status.Set("Idle....");
 }
}

void TestThread::buttonCb(void)
{
 if(stop)
 {
  stop = false;
  button.SetLabel("STOP");
  thr.Run(THISBACK(thrCb));
 }
 else
 {
  stop = true;
  thr.Wait();
  button.SetLabel("START");
 }

}

TestThread::TestThread()
{
 SetRect(0, 0, 300, 150);
 Add(button);
 button.TopPos(90, 30).HCenterPos(100);
 button.SetLabel("START");
 Add(progress);
 progress.TopPos(30, 40).HCenterPos(250);
 progress.Set(0, 100);
 AddFrame(status);
 status.Set(" ");

 stop = true;
 button <<= THISBACK(buttonCb);

}

GUI_APP_MAIN
{
 TestThread().Run();
}
```

Hangs on "thr.Wait()" line.
Removing the Wait line the app seems work, but, as far as I can see, it leaves the thread callback

running (but blocked somehow...); the tread count shows erroneously the correct runnung thread count.
Looking deeper inside, the app seems to hang on callback return.

If I remove the callback code (but leave the empty callback function) the app behaves good.
In another non-gui testcase the app behaves also good.

Do I miss something or it's a bug ?

Ciao

Max

p.s.: ubuntu karmic, gcc 4.4.1

Subject: Re: MT app hangs
Posted by mirek on Thu, 17 Dec 2009 12:22:05 GMT

View Forum Message <> Reply to Message

```
void TestThread::thrCb(void)
{
 {
  GuiLock __;
  status.Set("Running, " + FormatInt(thr.GetCount()) + " threads");
 }
 for(;;)
 {
  {
   GuiLock __;
   if(progress < 100)
    progress++;
   else
    progress = 0;
  }
  Sleep(100);
  if(stop)
   break;
 }
}

void TestThread::buttonCb(void)
{
 if(stop)
 {
  stop = false;
  button.SetLabel("STOP");
  thr.Run(THISBACK(thrCb));
 }
 else
 {
  stop = true;
  thr.Wait();
  status.Set("Idle....");
  button.SetLabel("START");
 }

}
```

The problem was that GuiLock is locked in buttonCb (that is the point...).

Mirek

Subject: Re: MT app hangs
Posted by mirek on Thu, 17 Dec 2009 12:24:44 GMT
View Forum Message <> Reply to Message

P.S.: In some cases, you might want to serialize access to 'stop'. I believe that given supported memory architecture, it is not required in this case, but I might be wrong...

Subject: Re: MT app hangs
Posted by mdelfede on Thu, 17 Dec 2009 12:51:23 GMT
View Forum Message <> Reply to Message

Hi, Mirek, thank for your answer

Anyways, I don't see a simple solution to my problem, indeed.
As far as I can understand, I can't use Wait() from inside the button callback, or anyways can't wait for the thread to stop from inside a gui event handler, as the thread could try a GuiLock between the Wait() call and its termination.

Worse, the tread can (ad it's probably) waiting for a lock to become free when I'm inside a callback, so I can't communicate with it.

The example I posted is very simplified, but what I need is a thread that can react to a Gui or a timed event (stop = true, and that's ok...) AND to check that the thread received it (Wait(), and that's not ok).

Why does Gui need to be locked inside event handlers ?
Can I unlock it inside the handler just to wait my thread to finish ? Or there's a nicer solution ?

Ciao

Max

Subject: Re: MT app hangs
Posted by mirek on Thu, 17 Dec 2009 17:20:18 GMT
View Forum Message <> Reply to Message

mdelfede wrote on Thu, 17 December 2009 07:51Hi, Mirek, thank for your answer

Anyways, I don't see a simple solution to my problem, indeed.
As far as I can understand, I can't use Wait() from inside the button callback, or anyways can't wait for the thread to stop from inside a gui event handler, as the thread could try a GuiLock between the Wait() call and its termination.

Ha! That is true.

Well, that is the hard part of MT...

I guess the really bad issue here is the Wait call. I thing things would be much simplified if you avoided Wait and posted callback at the end of thread.

Quote:
Why does Gui need to be locked inside event handlers ?

Because it does a lot of things to GUI status. E.g. setting focuses, changing widget status etc...

In theory, the locking could be much fine-grained (like locking individual widgets before accessing them). But doubt it would be any easier....

Quote:
 Or there's a nicer solution ?

```
void TestThread::thrCb(void)
{
 {
  GuiLock __;
  status.Set("Running, " + FormatInt(thr.GetCount()) + " threads");
 }
 for(;;)
 {
  {
   GuiLock __;
   if(progress < 100)
    progress++;
   else
    progress = 0;
   if(stop)
    break;
  }
  Sleep(1000);
 }
 {
  GuiLock __;
  status.Set("Idle....");
 }
 PostCallback(THISBACK(Stop));
}

void TestThread::Stop()
{
 button.SetLabel("START");
}


void TestThread::buttonCb(void)
{
 if(stop)
 {
  stop = false;
  button.SetLabel("STOP");
  thr.Run(THISBACK(thrCb));
 }
 else
  stop = true;
}
```

Subject: Re: MT app hangs
Posted by mdelfede on Thu, 17 Dec 2009 20:15:05 GMT
View Forum Message <> Reply to Message

Well... now I'm running out of ideas !
Why this

```
// calculation thread function
void CalcPage::DoCalculation(void)
```

```
{
 // gets the status bar
 StatusBar &sb = ((Lamell *)(GetMainWindow()))->GetStatusBar();
 {
  GuiLock __;
  sb.Set("Calcolo in corso");
 }
 calcAborted = !Calculate();
 {
  GuiLock __;
  if(calcAborted)
   sb.Set("Errore");
  else
   sb.Set("Pronto");
 }
}

// modify handler -- triggers page calculation
// (triggered by button)
void CalcPage::ModifyCb(void)
{
 PostCallback(THISBACK(CalcAgainCb));
}

// callback in case modify handler is re-entered when
// calc thread still working
void CalcPage::CalcAgainCb(void)
{
 if(!calcThread.IsOpen())
  // restarts the calculation if thread not already working
  calcThread.Run(THISBACK(DoCalculation));
  // otherwise repost calculation for later
 else
  PostCallback(THISBACK(CalcAgainCb));
}
```

doesn't work ???
It just runs calc thread once, so I guess IsOpen() always returns true.... so, thread still blocked somewhere.

Ciao

Max