Subject: Wait for other Threads
Posted by funky1221 on Wed, 01 Jun 2016 15:25:34 GMT
View Forum Message <> Reply to Message

My code as following. Compiled OK but when I run the program and push button to search file, No response and like crashed.

Someone help to solve this issue ???

```
void testU::PushRun()
{
 grid1.Clear();
 intfile=0;
 Thread().Run(THISBACK1(Searchfile,"C:"));
 Thread().Run(THISBACK1(Searchfile,"E:"));


 while(Thread().GetCount());

 String s = "Total " + AsString(intfile) + " Files";
 PromptOK(s);
}
void testU::Searchfile(String path)
{

 FindFile fs;
 String file = edit;
 file = path + "\\" + file;
 if(fs.Search(file))
 {

  do
  {
   Time t(fs.GetLastWriteTime());
      String lstime = Format("%",t);
   String fullname = fs.GetPath();
   INTERLOCKED{
     grid1.Add(fullname,lstime);
     intfile++;
   }

  }while(fs.Next());
 }

}
```

Subject: Re: Wait for other Threads
Posted by koldo on Thu, 02 Jun 2016 07:08:15 GMT
View Forum Message <> Reply to Message

Hello Funky

A detail from doc:

Design decisions and trade-offs
...
GUI is designed in a way that all user events are passed and processed by the main thread. The synchronization is done using single global mutex (operated by EnterGuiMutex, LeaveGuiMutex or GuiLock scoped helper). Other threads can directly work with GUI as long they do use this global mutex. There are some operations (like opening/closing windows, message loops) that cannot be performed by any other thread than main.
I propose you to try Reference/GuiLock example. It is simple and clear

Subject: Re: Wait for other Threads
Posted by funky1221 on Thu, 02 Jun 2016 14:41:55 GMT
View Forum Message <> Reply to Message

I try the GuiLock learned from the reference/GuiLock.
Change to the below but still no response and like be in the loop.

But if removed "while(Thread().GetCount());", it worked!

But intfile still 0, why ?

PromptOK show intfile prior to Grid that show file list. why Threads wait not work ?

```
void testU::PushRun()
{
 grid1.Clear();
 intfile=0;
 Thread().Run(THISBACK1(Searchfile,"C:"));
 Thread().Run(THISBACK1(Searchfile,"E:"));


 while(Thread().GetCount());

 String s = "Total " + AsString(intfile) + " Files";
 PromptOK(s);
}
void testU::Searchfile(String path)
{
```

```
 FindFile fs;
 String file = edit;
 file = path + "\\" + file;
 if(fs.Search(file))
 {

  do
  {
     Sleep(1);        //
     GuiLock __;      // New Add instead of INTERLOCKED

     Time t(fs.GetLastWriteTime());
     String lstime = Format("%",t);
     String fullname = fs.GetPath();

     grid1.Add(fullname,lstime);
     intfile++;


  }while(fs.Next());
 }

}
```

File Attachments

1) testU.7z, downloaded 434 times

Subject: Re: Wait for other Threads
Posted by koldo on Fri, 03 Jun 2016 06:50:18 GMT
View Forum Message <> Reply to Message

Hello funky

I am not an expert in multi threading and in my projects I use another focus. However I see two things in your code:

- I would replace while(Thread::GetCount()); with:

```
while(Thread::GetCount()) {
 ProcessEvents();
 Sleep(0);
}
```

I think that with while(Thread::GetCount()); you are locking main thread with an infinite loop, and Guilocks in threads will wait forever.
Calling ProcessEvents(); inside the loop you let the GUI to be refreshed by the threads you have launched.

- intfile

This is a variable handled globally by different threads without any control, so results will be undefined...
A safer way could be to declare intfile as Atomic and increase it with AtomicInc:
Atomic intfile;
...
AtomicInc(intfile);
intfile will behave as an int, but multiple threads will access it safely.

Subject: Re: Wait for other Threads
Posted by mr_ped on Fri, 03 Jun 2016 07:59:40 GMT
View Forum Message <> Reply to Message

I didn't study the original problem, this comment is more a reply to koldo's response.

Operations with int often are naturally atomic even without guards (either completely atomic like ++int, or with possible errors in result value, but undamaged content of int, like int++).

That said, you still can't just increase it in one thread, and display it in second, unless you tell the compiler, that the content can be changed outside of current thread. So you should either use "volatile int" (if you know, what you are doing, and how CPU works), or std::atomic<int> ..

like here is some (a bit too short?) tutorial:
http://baptiste-wicht.com/posts/2012/07/c11-concurrency-tuto rial-part-4-atomic-type.html

---

Subject: Re: Wait for other Threads
Posted by funky1221 on Sat, 04 Jun 2016 00:30:33 GMT
View Forum Message <> Reply to Message

I used ProcessEvents() but threads still inside the loop.
just refresh GUI.

If I use PromptOK inside the loop, threads seems to be finished, why ?

But I changed "void testU::Searchfile" to "void Searchfile" and used PostCallback referenced from GuiMT,
it worked.

koldo wrote on Fri, 03 June 2016 08:50Hello funky

I am not an expert in multi threading and in my projects I use another focus. However I see two things in your code:

- I would replace while(Thread::GetCount()); with:

```
while(Thread::GetCount()) {
 ProcessEvents();
 Sleep(0);
}
```
I think that with while(Thread::GetCount()); you are locking main thread with an infinite loop, and Guilocks in threads will wait forever.
Calling ProcessEvents(); inside the loop you let the GUI to be refreshed by the threads you have launched.

- intfile
This is a variable handled globally by different threads without any control, so results will be undefined...
A safer way could be to declare intfile as Atomic and increase it with AtomicInc:
```
Atomic intfile;
...
AtomicInc(intfile);
```
intfile will behave as an int, but multiple threads will access it safely.
