Subject: Basic MT queries
Posted by 281264 on Mon, 20 Sep 2010 15:16:55 GMT
View Forum Message <> Reply to Message

Hi,

I am attaching a simple snippet with a thread that performs a dummy loop. I have some very basic queries:

1.- how to stop/start the thread by using the buttons? I guess that ShutdownThreads function is the right one to raise a flag but although the loop stops, the application remains stuck.

2.- I have seen some global functions that are available, such is GuiLock_; is there any information about such a function and other related functions?. What is it for?

Many thanks.

Cheers,

Javier

File Attachments
1) prueba_multhreads2.7z, downloaded 431 times

Subject: Re: Basic MT queries
Posted by koldo on Tue, 21 Sep 2010 04:50:29 GMT
View Forum Message <> Reply to Message

Hello Javier

Try the samples Reference/Guilock and GuiMT.

They work very well and are simple.

Subject: Re: Basic MT queries
Posted by 281264 on Tue, 21 Sep 2010 10:26:08 GMT
View Forum Message <> Reply to Message

Ok. I have checked them and are fine.

Allow me to ask some questions:

1.- Atomic and associated functions, are they similar to INTERLOCKED?

2.- ShutDownThreads function: what is it for? How a thread can be paused and resumed?

3.- Apparently there are numerous global functions in U++ not documented. One of them is GuiLock__? What is this for?.

Cheers.

Thank you,

Javier

Subject: Re: Basic MT queries
Posted by dolik.rce on Tue, 21 Sep 2010 11:36:57 GMT
View Forum Message <> Reply to Message

Hi Javier
281264 wrote on Tue, 21 September 2010 12:261.- Atomic and associated functions, are they similar to INTERLOCKED?
If I'm not mistaken, Atomic*() functions use platform specific routines for atomic access to variables. INTERLOCKED macro gives you similar effect on entire block. The main difference is that INTERLOCK uses mutex to achieve this, so it is much slower.

281264 wrote on Tue, 21 September 2010 12:262.- ShutDownThreads function: what is it for? How a thread can be paused and resumed?
Calling Thread::ShutDownThreads() sets an internal flag that can be checked using Thread::IsShutDownThreads(). If I remember correctly, ShutDownThreads waits till all the threads end. To stop/resume thread you should use a flag for which you check in the thread. It is a good idea to to mark it volatile, e.g. "volatile bool running;".

281264 wrote on Tue, 21 September 2010 12:263.- Apparently there are numerous global functions in U++ not documented. One of them is GuiLock__? What is this for?.
GuiLock is one of the ways how to safely update GUI from threads. BTW: There is actually a space, the "__" is used just as a variable name The idea behind GUILock is that as long as the object __ (or gl) exists, the main thread is blocked from changing GUI, so it prevents dead-locks. The lock is actually acquired in constructor and released in destructor, so it can be easily limited using brackets.

Honza

PS: Looking at you sources from first post, I noticed that you use the thread1 variable to call the static functions (which can be called without an object). Actually the whole code in this case could be done without using the variable at all. To start thread in such case you can use Thread::Start(callback). The only downside is that you can't Wait() for such thread, but in many cases that is not necessary.

Subject: Re: Basic MT queries
Posted by 281264 on Tue, 21 Sep 2010 16:03:43 GMT
View Forum Message <> Reply to Message

Honza:

Many thanks. Fine explanation. What is Wait() function for?

Javier

Subject: Re: Basic MT queries
Posted by dolik.rce on Tue, 21 Sep 2010 16:46:52 GMT
View Forum Message <> Reply to Message

281264 wrote on Tue, 21 September 2010 18:03Honza:

Many thanks. Fine explanation. What is Wait() function for?

Javier
Wait() is for situations where you need to block the program until a given thread finishes (e.g. because you need it's result before your program can continue).

Example:
Thread t1,t2;
//start threads to do some work in parallel
t1.Run(fn1); // do first half of work
t2.Run(fn2); // do second half of work
// wait till both finish so you can process results
t1.Wait();
t2.Wait();
// now you can continue work that needs both threads to be finished ...

Notice that it doesn't matter in which order the threads finish. If you call Wait() on thread that has already finished, it should return immediately.

Subject: Re: Basic MT queries
Posted by 281264 on Tue, 21 Sep 2010 16:53:46 GMT
View Forum Message <> Reply to Message

Cristal clear.

Many thanks,

Javier