Subject: Thread based "conveyor" class for Win32
Posted by Mindtraveller on Wed, 10 Oct 2007 16:51:27 GMT
View Forum Message <> Reply to Message

In my work I have rather usual task of having some queue service, processing main thread requests in background. I just finished general purpose class template for simplified handling of such a task. It`s rather young (with interface unpolished) but look at what you can do with it:

```
template<class T> class ConveyorThread : protected Thread
{ typedef ConveyorThread CLASSNAME;
public:
 ConveyorThread(bool enabled = true);
 virtual ~ConveyorThread();
 void Enable(bool enable = true);

 void Request(Callback1<const T &>handler, const T &request);
 void ClearAllRequests();

 void SharedEnter();
 void SharedLeave();

 void RequestFinish();
 int  WaitFinished();
 bool IsFinished();
}
```

Just add request for asyncronous processing and define handler for it - and that`s all. Rather convenient, I suppose.
Class also has guarding functions for interacting with main thread`s objects: SharedEnter/SharedLeave (for updating interface, statistics, etc).

Any suggestions, critics and recommendations is appreciated.

File Attachments

1) ConveyorThread.h, downloaded 559 times

Subject: Re: Thread based "conveyor" class for Win32
Posted by mirek on Wed, 10 Oct 2007 17:35:36 GMT
View Forum Message <> Reply to Message

Mindtraveller wrote on Wed, 10 October 2007 12:51In my work I have rather usual task of having some queue service, processing main thread requests in background. I just finished general purpose class template for simplified handling of such a task. It`s rather young (with interface unpolished) but look at what you can do with it:

```
template<class T> class ConveyorThread : protected Thread
{ typedef ConveyorThread CLASSNAME;
public:
 ConveyorThread(bool enabled = true);
```

```
 virtual ~ConveyorThread();
 void Enable(bool enable = true);

 void Request(Callback1<const T &>handler, const T &request);
 void ClearAllRequests();

 void SharedEnter();
 void SharedLeave();

 void RequestFinish();
 int  WaitFinished();
 bool IsFinished();
}
```

Just add request for asyncronous processing and define handler for it - and that`s all. Rather convenient, I suppose.
Class also has guarding functions for interacting with main thread`s objects: SharedEnter/SharedLeave (for updating interface, statistics, etc).

Any suggestions, critics and recommendations is appreciated.

Interesting idea. Is there anything that prevents using Semaphore instead of events there? (The problem is that your code is win32 specific).

Subject: Re: Thread based "conveyor" class for Win32
Posted by Mindtraveller on Wed, 10 Oct 2007 17:44:38 GMT
View Forum Message <> Reply to Message

Subject: Re: Thread based "conveyor" class for Win32
Posted by Mindtraveller on Wed, 10 Oct 2007 17:50:58 GMT
View Forum Message <> Reply to Message

I just recollected the cause of using Windows objects instead of Semaphore class: the need of WaitForMultipleObjects call. Also, I was needed for testing of object state with WaitForSingleObject(finishEvent, 0)

Subject: Re: Thread based "conveyor" class for Win32
Posted by mirek on Wed, 10 Oct 2007 18:52:40 GMT
View Forum Message <> Reply to Message

Mindtraveller wrote on Wed, 10 October 2007 13:50I just recollected the cause of using Windows objects instead of Semaphore class: the need of WaitForMultipleObjects call. Also, I was needed

for testing of object state with WaitForSingleObject(finishEvent, 0)

Thanks.

Mirek

Subject: Re: Thread based "conveyor" class for Win32
Posted by Mindtraveller on Wed, 12 Dec 2007 22:16:41 GMT
View Forum Message <> Reply to Message

Recently I finished project. It included new upgraded version of ConveyorThread and brand new RS232Thread class.
RS232Thread is really advanced helper class useful i.e. for industrial automation, when it`s necessary to have some good tool for programming COM-port (RS232) i/o.
I`d like to contribute these classes into some package into U++. For now, they are targeted for Win32 only. It is due to Mutex and Semaphore limitations listed above.
I`m no professional in Linux, but if these base sync classes functionality extended, I`ll try to write complete multiplatform ConveyorThread and RS232Thread.
Is it possible to add necessary functionality to these classes?

Subject: Re: Thread based "conveyor" class for Win32
Posted by mirek on Sun, 16 Dec 2007 21:27:40 GMT
View Forum Message <> Reply to Message

Mindtraveller wrote on Wed, 12 December 2007 17:16Recently I finished project. It included new upgraded version of ConveyorThread and brand new RS232Thread class.
RS232Thread is really advanced helper class useful i.e. for industrial automation, when it`s necessary to have some good tool for programming COM-port (RS232) i/o.
I`d like to contribute these classes into some package into U++. For now, they are targeted for Win32 only. It is due to Mutex and Semaphore limitations listed above.
I`m no professional in Linux, but if these base sync classes functionality extended, I`ll try to write complete multiplatform ConveyorThread and RS232Thread.
Is it possible to add necessary functionality to these classes?

Maybe it will be necessarry to just #ifdef PLATFORM_LINUX and use Linux API here...

Mirek

Subject: Re: Thread based "conveyor" class for Win32
Posted by zsolt on Sun, 16 Dec 2007 23:03:21 GMT
View Forum Message <> Reply to Message

Mindtraveller wrote on Wed, 12 December 2007 23:16Recently I finished project. It included new upgraded version of ConveyorThread and brand new RS232Thread class.

RS232Thread is really advanced helper class useful i.e. for industrial automation, when it`s necessary to have some good tool for programming COM-port (RS232) i/o.
I`d like to contribute these classes into some package into U++. For now, they are targeted for Win32 only. It is due to Mutex and Semaphore limitations listed above.

I think, you can post it here and a good Linux programmer could finish it. These classes seems to be very interesting

---

Subject: Re: Thread based "conveyor" class for Win32
Posted by Mindtraveller on Mon, 17 Dec 2007 11:50:48 GMT
View Forum Message <> Reply to Message

Recently I`ve been thinking about implementing Mutex and Semaphore with additional functionality and how to do it the best way.

And that`s my suggestion: it is possible to make these classes Multithreading POSIX API based. It will work OK for Linux version. And for Win32 the same functions will be mapped by POSIX Threads for Win32 library calls. Here`s library description:

This will unify, ease debugging and reading of code.
