Subject: WaitForMultipleObjects() analog?
Posted by Mindtraveller on Wed, 19 Mar 2008 23:53:26 GMT
View Forum Message <> Reply to Message

Is there one in U++? Or is there technique to wait for a number of *events*, awaiking on one of them?

Subject: Re: WaitForMultipleObjects() analog?
Posted by mirek on Sun, 23 Mar 2008 07:36:18 GMT
View Forum Message <> Reply to Message

No, AFAIK there is no unified way how to do that in Linux. (There is solution for sockets AFAIK, but not that nice general solution as in Win32).

Mirek

Subject: Re: WaitForMultipleObjects() analog?
Posted by Mindtraveller on Fri, 25 Apr 2008 01:35:25 GMT
View Forum Message <> Reply to Message

OK, who knows how to wait for multiple Semaphores in Linux? Sometimes it is very useful to awake on one of objects from list. All the complex applications usually wait for a number of objects in it`s threads (i.e. i/o event and quit event).
Are there any experienced Linux coders, how do you solve this problem? Maybe we will integrate it into U++ (if it`s not already).

For now I can`t write extensive portable applications under U++ mostly due to many MT unsupported features - like WaitForMultipleObjects. I believe this thing will be handy.

P.S. I found Linux` version of WaitForMultipleObjects function. May be it will be useful.

File Attachments
1) LINUX_WAIT_FOR_MULTIPLE_OBJS.ZIP, downloaded 677 times

Subject: Re: WaitForMultipleObjects() analog?
Posted by hojtsy on Wed, 28 May 2008 08:16:44 GMT
View Forum Message <> Reply to Message

Mindtraveller,

What are you trying to achieve with multiple Semaphores? I would expect that an alternative implementation could use Monitors. These are fancy mutexes which also have Wait and Pulse methods.
I am working on a U++ example code which uses Monitors to implement a Producer-Consumer

queue.

See the concept explained in http://www.albahari.com/threading/part4.html

Hojtsy

Subject: Re: WaitForMultipleObjects() analog?
Posted by Mindtraveller on Wed, 28 May 2008 20:19:43 GMT

View Forum Message <> Reply to Message

hojtsy wrote on Wed, 28 May 2008 12:16Mindtraveller,

What are you trying to achieve with multiple Semaphores? I would expect that an alternative implementation could use Monitors. These are fancy mutexes which also have Wait and Pulse methods.
I am working on a U++ example code which uses Monitors to implement a Producer-Consumer queue.

See the concept explained in http://www.albahari.com/threading/part4.html

Hojtsy
I just didn`t fully understand about Monitors usage as we don`t have ones in U++ for now.
Then about waiting a number of Semaphores. Let`s imagine the real world application. Your program must heavily work with a number of devices attached to serial port (or ethernet or something else). Of course it is solved by adding a class which creates new thread and have cycle which is sleeping most of time. But there is a number of awake signal types. And these signals could be activated from within: this thread, OS and main thread. These signals are:

 application close event. sleeping i/o thread should do finalization and close.
 OS i/o event. sleeping thread should handle the fact of newly sent or received data.
 queue event. sleeping thread just got a new task to do. this task was added by another thread.
 timer event (timeout). happens when system i/o functions didn`t  respond in time or final value of some deferred OS call didn`t arive in time.

These events are dramatically different from each other and when my thread is awakened I should know the type of this event to handle it. So I use one synchronization object (let`s say Semaphore for simplicity) for each event type. Doing this with one Semaphore would be a nightmare, wouldn`t it?

Subject: Re: WaitForMultipleObjects() analog?
Posted by hojtsy on Thu, 29 May 2008 13:12:30 GMT

View Forum Message <> Reply to Message

Mindtraveller,

AFAIK unix can only block on multiple conditions inside "select".
Anything else is emulation. The code you attached can indeed wait on multiple Semaphores, but not on file descriptors which you also need. That is the more interesting part I think.

Implementation alternative A:

You can block on a single Semaphore or Monitor, which protects a message queue. In this case any of the events you list should be an other thread putting a message to the queue, which the worker thread awakes to process. The direct IO communication (the select call) should be done on a different thread which puts incoming messages to the message queue and signals the Semaphore/Monitor.
In this case the problem arises how do you cancel the IO thread which is blocking inside select when you want to shutdown? You have three options: either call shutdown on the file description it is waiting for, or have it select on a pipe too which only gets written to when a shutdown occurs, or have the thread automatically wake up every 0,1 sec from the select and poll for the shutdown flag. In all cases the IO thread gets waken up from select and can exit. The worker thread could just wake up on a shutdown message arriving in it's message queue.

Implementation alternative B:

You can block inside a select which waits for the IO port and multiple pipes, each of which communicates different events. Timeout is automatically handled by select, shutdown and queue events could be other threads writing some bytes to the dedicated pipes, which also wakes up the thread doing the select.

I think solution A is better because it separates the low level pipe hacking from the main logic of the application, needs less pipes, and also makes the main part portable without preprocessor guards.

Subject: Re: WaitForMultipleObjects() analog?
Posted by Mindtraveller on Thu, 29 May 2008 18:32:40 GMT
View Forum Message <> Reply to Message

Agree. So we should have everything needed for such a tricky multithreading inside U++.
For now we have Mutex, Semaphore and no pipes.