Subject: how to stop a thread that is waiting, e.g., listen()
Posted by bonami on Wed, 26 Sep 2007 09:40:20 GMT
View Forum Message <> Reply to Message

i know how to do this neither in GNU C++ nor in U++.
one thread create the other. the second thread binds a socket and listen, waiting for incoming connections, and obviously it is blocked. how does the first thread stop the second?
thank you.

Subject: Re: how to stop a thread that is waiting, e.g., listen()
Posted by tvanriper on Wed, 26 Sep 2007 11:05:28 GMT
View Forum Message <> Reply to Message

You could probably do this with a signal. Normally, threads allow you to have multiple signals at one time. You are using one of those signals right now to wait on your networking socket (or whatever you're doing)... so you could create another signal to indicate that you have some kind of message for the thread.

The message could be stored in a variable that's protected by a mutex.

When the thread gets the signal, it looks in the variable, sees that you want it to close, and then stops itself (cleaning up any variables it needs to clean up).

It's not recommended to try and forcibly close a thread, as you seem to suggest wanting to do, because this will not deallocate variables you have (implicitly or explicitly) allocated for that thread.

Subject: Re: how to stop a thread that is waiting, e.g., listen()
Posted by bonami on Thu, 27 Sep 2007 08:57:11 GMT
View Forum Message <> Reply to Message

i used signals only b/t processes and only saw a tkill() sth. signaling a thread, though not yet knowing how to use it. and U++ probably cannot support this with Thread class.

furthermore, do not pthread_kill() and _cancel() send signals, too? so these functions work the same way as tkill()?

Subject: Re: how to stop a thread that is waiting, e.g., listen()
Posted by tvanriper on Thu, 27 Sep 2007 11:20:04 GMT
View Forum Message <> Reply to Message

Sorry... I have mostly worked with Windows programming, where the term 'signal' takes on a slightly different meaning, I think. At the very least, I meant the word in a far more generic way than I think you took it.

I can't really speak too clearly on POSIX-oriented programming, and I still need to peek a little more closely at how Ultimate++ handles threading in general, but while there might be commands to kill a thread, that normally means you're ending the thread before it has had a chance to clean up. Especially for long-running applications, this is not a good idea.

In general, regardless of platform, you want to somehow tell the thread, while it's running, that it should stop running, so it can clean itself up. That's all I'm trying to say.

In the Windows world, you do that through an event/signal.

I get the impression that, in the POSIX world, a signal interrupts the flow of the thread... that isn't what I'm aiming for. At least, I don't think it's what I'm aiming for... but again, I am not very experienced in POSIX. I could be completely wrong.

Somehow, while your thread waits for information to arrive on the socket, something tells the thread that it can stop waiting, and handle the network information. That same mechanism should also be able to provide a way of telling your thread that it can stop waiting, and start handling some other kind of information (in this case, checking a variable to see if the thread should quit). While I know how to do such a thing in Windows, I do not know how to do such a thing with POSIX, and I'm not entirely sure (yet) how Ultimate++ handles it... but I expect you could probably figure it out by examining the Ultimate++ code.

Subject: Re: how to stop a thread that is waiting, e.g., listen()
Posted by tvanriper on Thu, 27 Sep 2007 11:43:12 GMT
View Forum Message <> Reply to Message

Out of my own curiosity, I took a peek at Thread.h/Thread.cpp in Core, and socket.h/socket.cpp in Web.

Thread.h/.cpp detail the Thread class, a class you can use for handling multi-threaded needs in an application. It exposes a Wait function, which returns an int. The integer it returns is either the exit code for the thread, or a numeric identifier for an event that was sent to the thread (well... in Windows... in POSIX, it seems to work differently, if I am reading this correctly... pthread_join [which will return 0] or the return value of the terminated thread).

For Windows, the Socket class seems to support the use of the Event object, but that object is not exposed to POSIX, if I'm reading everything correctly.

At a guess, you could probably create a Socket, set it up for networking however you need to, and use the static 'Wait' command with a timeout to wait on network traffic. If there's traffic, do what you need to do with the traffic. Otherwise, check a variable shared between your threads to see if you need to shut down your thread.

Perhaps someone else might have a better way to do this, but this seems to be the safest way to handle Sockets, multi-threading, and keeping your application working in both Windows and POSIX environments.

Subject: Re: how to stop a thread that is waiting, e.g., listen()
Posted by bonami on Fri, 28 Sep 2007 03:13:01 GMT
View Forum Message <> Reply to Message

MS way IS what i want.
here's my plan to implement and hope to get some advice from ya,

in the main window thread, create a thread.
in the 2nd thread, open a network socket.
in the 1st thread, when it needs to tell sth. to or shut down the other thread, SetEvent.
in the 2nd thread, WaitForMultipleObjects on both the network socket and the event from the 1st thread.

pretty much like write() and select() on POSIX. how's this? thank you so much.

Subject: Re: how to stop a thread that is waiting, e.g., listen()
Posted by tvanriper on Sat, 29 Sep 2007 12:46:08 GMT
View Forum Message <> Reply to Message

Well, at least as of 2007.1 (I don't know about the dev builds, as I haven't been using them), it looks like you may need to create your own objects to get the behavior you're describing, at least for POSIX.

I have the impression that this area of Ultimate++ is still a little new.

Subject: Re: how to stop a thread that is waiting, e.g., listen()
Posted by bonami on Tue, 16 Oct 2007 09:59:43 GMT
View Forum Message <> Reply to Message

then if i choose fd_set and select() way, how could i create a socket/fd between threads? MinGW does not seem to support machine's local connection.

Subject: Re: how to stop a thread that is waiting, e.g., listen()
Posted by mirek on Wed, 17 Oct 2007 17:16:14 GMT
View Forum Message <> Reply to Message

tvanriper wrote on Sat, 29 September 2007 08:46Well, at least as of 2007.1 (I don't know about the dev builds, as I haven't been using them), it looks like you may need to create your own objects to get the behavior you're describing, at least for POSIX.

I have the impression that this area of Ultimate++ is still a little new.

I think that sort of problem is the fact that these thing are quite platform specific, especially if you want to have optimal solution - it is hard to support them in multi-platform library..

Mirek

Subject: Re: how to stop a thread that is waiting, e.g., listen()
Posted by bonami on Fri, 19 Oct 2007 09:52:48 GMT
View Forum Message <> Reply to Message

ok. i understand this is rather platform specific.
does that mean i have to write separate code for win & unix and control them using macros?

Subject: Re: how to stop a thread that is waiting, e.g., listen()
Posted by tvanriper on Fri, 19 Oct 2007 13:35:18 GMT
View Forum Message <> Reply to Message

I can definitely appreciate the challenge of creating something uniform that works across both platforms, where multi-threading is concerned.

This requires some thought.

Here's what boost has been thinking, on this topic, on the odd chance it helps:

http://www.boost.org/doc/html/thread.html

Subject: Re: how to stop a thread that is waiting, e.g., listen()
Posted by bonami on Mon, 22 Oct 2007 09:47:38 GMT
View Forum Message <> Reply to Message

wow. boost is a good thing. wonder whether it can be used here (in U++)

Subject: Re: how to stop a thread that is waiting, e.g., listen()
Posted by tvanriper on Mon, 22 Oct 2007 15:47:07 GMT
View Forum Message <> Reply to Message

Well, it's maintained separately, by the boost folks, but since it's just C++, I'm sure you could use it in addition to the Ultimate stuff if accomplishes your needs.

Goodness knows, I mix Ultimate++ with the standard C++ library (io streams, maps, etc) whenever I feel the need, without any adverse side effects. I haven't tried mixing with boost yet, but I can't imagine there'd be any problems.

I would heed the advice boost gave regarding event driven multitasking. They suggest that it's fraught with peril. Er, that it has a tendency to lead to some multi-tasking problems of one form or

another. Basically, it has a tendency to lead to unsafe code, while some of the other approaches they recommend lead to safer practices.

Subject: Re: how to stop a thread that is waiting, e.g., listen()
Posted by bonami on Wed, 24 Oct 2007 02:20:02 GMT
View Forum Message <> Reply to Message

yeah i heard of boost, just never used it, or even got to know it more.
speaking of my needs, i still had the tendancy to KILL the thread. so boost probably cannot fit in my project.

more considerations left later since it seems much time needed.

Subject: Re: how to stop a thread that is waiting, e.g., listen()
Posted by mirek on Sun, 11 Nov 2007 17:49:23 GMT
View Forum Message <> Reply to Message

bonami wrote on Tue, 23 October 2007 22:20yeah i heard of boost, just never used it, or even got to know it more.
speaking of my needs, i still had the tendancy to KILL the thread. so boost probably cannot fit in my project.

more considerations left later since it seems much time needed.

BTW, one note: Current U++ memory allocator needs a specific call at the end of each thread to release per-thread allocation cache.

This is done automatically with U++ Thread, but has to be called explicitely if you do threads using anything else.

Mirek

Subject: Re: how to stop a thread that is waiting, e.g., listen()
Posted by bonami on Mon, 12 Nov 2007 01:19:37 GMT
View Forum Message <> Reply to Message

IC. thanks.
i'll use U++ thread.
