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.