Subject: Problem with non-blocking mode
Posted by Weras on Fri, 23 Oct 2009 09:02:55 GMT
View Forum Message <> Reply to Message

Hi!

I have a problem with sockets, namely, a non-blocking mode. I must say that until that moment I had never worked with sockets and my
question may seem silly.
In my project, I repeated the tutorial from the Help Topics "Connection-Oriented Socket Tutorial"

Function accept_socket.Accept (data_socket, & ip_addr)
suspends the entire program, so I point out that the socket nonblocking:

```
        Socket::Init();

 if( !ServerSocket(accept_socket, portInfo.portNumber) )
  throw Exc("Couldn't bind socket on the local port.");

 if( accept_socket.IsOpen() )
 {
  dword ip_addr;
  accept_socket.Block(false);
                accept_socket.Peek();

  if( !accept_socket.IsError() && accept_socket.Accept(data_socket, &ip_addr) )
  {
   Cout() << "Connection from " << FormatIP(ip_addr) << "\n";

   while(data_socket.IsOpen() && !data_socket.IsEof() && !data_socket.IsError())
    Cout() << data_socket.Read();
  }
 }
```

But the program still awaits all perform the function Accept. What I missed or am doing wrong?

Also I read that non-blocking socket's have callback when connection started(or something like that). Could you give examples code using this opportunity socket?

P.S.
Sorry my english)

Subject: Re: Problem with non-blocking mode
Posted by mirek on Sat, 24 Oct 2009 16:49:36 GMT
View Forum Message <> Reply to Message

Weras wrote on Fri, 23 October 2009 05:02Hi!

I have a problem with sockets, namely, a non-blocking mode. I must say that until that moment I had never worked with sockets and my
question may seem silly.
In my project, I repeated the tutorial from the Help Topics "Connection-Oriented Socket Tutorial"

Function accept_socket.Accept (data_socket, & ip_addr)
suspends the entire program, so I point out that the socket nonblocking:

```
        Socket::Init();

 if( !ServerSocket(accept_socket, portInfo.portNumber) )
  throw Exc("Couldn't bind socket on the local port.");

 if( accept_socket.IsOpen() )
 {
  dword ip_addr;
  accept_socket.Block(false);
                accept_socket.Peek();

  if( !accept_socket.IsError() && accept_socket.Accept(data_socket, &ip_addr) )
  {
   Cout() << "Connection from " << FormatIP(ip_addr) << "\n";

   while(data_socket.IsOpen() && !data_socket.IsEof() && !data_socket.IsError())
    Cout() << data_socket.Read();
  }
 }
```

But the program still awaits all perform the function Accept. What I missed or am doing wrong?

Also I read that non-blocking socket's have callback when connection started(or something like that). Could you give examples code using this opportunity socket?

P.S.
 Sorry my english)

I belive: Call Peek and only do Accept if it returns true.

Or you can sepcify timeout in Accept (e.g. to zero).

(I belive you in fact do not need non-blocking sockets in that case, but I might be wrong).

Mirek

Subject: Re: Problem with non-blocking mode
Posted by Weras on Sat, 24 Oct 2009 19:14:01 GMT
View Forum Message <> Reply to Message

Well, i see.

Yes, in that case i exactly need non-blocked socket. As i know, there are some callback function in non-block mode. Something about a special thread for a non-block sockets. Is it true? If yes, can you show me example of using this socket's feature please?

Dmitry.

Subject: Re: Problem with non-blocking mode
Posted by rylek on Sun, 25 Oct 2009 23:22:21 GMT
View Forum Message <> Reply to Message

Hello there!

As far as I know, in principle there are two basically different ways to make a socket communication run in background. One is to make the sockets non-blocking, the other to run the socket management in a different thread. Moreover, under Windows, the WSA socket management functions let you bind certain socket operation notifications to the message loop and so to avoid polling and emulate the worker thread behaviour in a single-threaded application.

The current socket wrapper in U++ doesn't support this option, mainly because it requires low-level interception of the message queue which would either have to be built into the U++ core (which would be ugly) or would require lots of low-level hackery; moreover it's available in Windows only, Linux has no similar functionality, which makes usability of such feature very limited in a cross-platform environment like U++.

Normally when you open a nonblocking server socket (using the 5-th "is_blocking" argument of the ServerSocket function), calls to Accept fail during the initial client-server handshake and you can use Peek on the server socket to check for pending client connections.

Regards

Tomas