Subject: Socket functions calling order
Posted by captainc on Mon, 14 Jan 2008 17:04:12 GMT
View Forum Message <> Reply to Message

I'm writing a simple client/server socket app and need to confirm the order of operations for the functions and communication with the client. The app will run as a service/daemon and listen for connections from clients.

On the server side,
I first create the ServerSocket
if(ServerSocket(sock1, 1555, true, 5, false)){ ...
Then, in a continuous loop using Sleep() function, I check for data on the socket:

```
if( !sock1.IsOpen() ){
    ServerSocket(sock1, 1555, true, 5, false);
}
if( sock1.IsOpen() && !sock1.IsError() && sock1.Peek() && sock1.Accept(sock1) ){
    String sock_data(sock1.Read(2000));
    if(sock_data.GetLength() > 0){
        LOG(String().Cat() << "Data: " << sock_data);
        sock_data.Clear();
        sock1.Clear();
    }
}
```

The client code is simple:

```
if( ClientSocket(sock1, ip_addr, 1555, true, NULL, DEFAULT_CONNECT_TIMEOUT, false) ){
    sock1.Write(String("This is a test"));
    sock1.Close();
}
```

I found that the socket is closed on the server side after calling Read() and then Clear(). If I don't call Clear(), then the program hangs with any subsequent operation on the socket (and the client sees the socket as closed).
If I call Clear(), the socket ends up being closed (as IsOpen() the returns false). I then found that any other operations after Clear() such as Calling IsError() on a closed socket results in an "Assertion failed in ... Core/Other.h, line 17".

Can someone shed some light as to the correctness of the order of the code above?

How would I go about sending a response from the server to the client? Where/when would I call Write() on the socket?

Subject: Re: Socket functions calling order
Posted by zaurus on Thu, 17 Jan 2008 17:40:31 GMT
View Forum Message <> Reply to Message

Hi captainc!

I was also playing around with network connections some time ago. Here my loop which is reading the data from the socket. It works ok for me.

```
void NetServTest::TimerLoop()
{
 String sCommand;

 if(!m_Connection.IsOpen())
 {
  if(m_Socket.Accept(m_Connection, &m_ipaddr, true, 100))
   PromptOK("Accept.");
  Status = "OFF";
 }
 if(m_Connection.IsOpen())
 {
  Status = "ON";
  sCommand = m_Connection.Read(100);
  Output = sCommand;
  if(sCommand == "WHO")
   m_Connection.Write("It's me!");
  Sync();
 }
 SetTimeCallback(2000, THISBACK(TimerLoop));
}
```

I'm not doing anything to clear. I'm not a programming expert and cannot tell you what is really the correct way, but the above works for me.

I think in your code the sock1.Accept(sock1) is the problem. You should not accept the connection on the same socket on which you are listening.

Zaurus

Subject: Re: Socket functions calling order
Posted by dmcgeoch on Fri, 18 Jan 2008 21:01:04 GMT
View Forum Message <> Reply to Message

Hello,

I have also been experimenting with sockets. I've attached the code that I've been using. There is a Server and a Client project. The server is very loosely based on code I found on another thread.

Additionally, telnet can be used to connect to the server program to verify that the server is

behaving as expected.

Any comments would be greatly appreciated.

Dave

File Attachments
1) SocketServer.zip, downloaded 606 times
2) SocketClient.zip, downloaded 781 times

Subject: Re: Socket functions calling order
Posted by captainc on Tue, 22 Jan 2008 17:20:31 GMT
View Forum Message <> Reply to Message

zaurus, thanks for that example. I was definitely confused as to the reading and writing on the same socket as listening. I am new to socket programming, so I made the assumption that the socket that sees the initial connection had to be used to transfer information.

I still wonder, how does Accept() work together with ServerSocket()?

On my initial analysis through the source code,
Calling ServerSocket() on a socket doesn't seem to implicitly call Accept() for that socket, although both ServerSocket() and Accept() call Attach(data) on the socket.
Further, ServerSocket() calls OpenServer(), which calls lower level calls bind() and listen() on the socket. Accept() calls lower level accept() on the socket.

Would you mind showing the full example of the main loop you provided?

Subject: Re: Socket functions calling order
Posted by captainc on Tue, 22 Jan 2008 17:40:15 GMT
View Forum Message <> Reply to Message

dmcgeoch,

I really like your example. You implement a class that I have seen in the POCO C++ libraries (SocketServer). The SocketServer class is very intuitive and well done; I understand exactly what you are trying to do there. I compiled and tested the program(s) and found it to be a little buggy, but thats probably because it is a simple example and not a complete working program. It is a great idea nonetheless.