

William Munnich

Programming Assignment

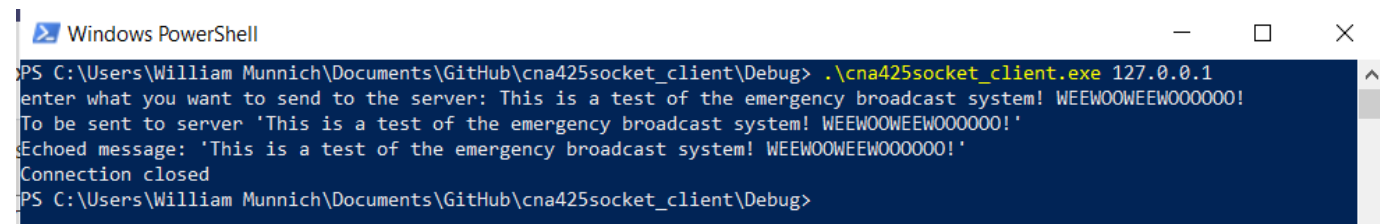
Dr. Mark Petzold

CNA 425

Write an echo server using the Winsock code provided. On the client side, the program should accept text input from the user and transmit that to the server. The server should then display the text received and echo the message back to the client. The client should display the received message and indicate that it is the received message.

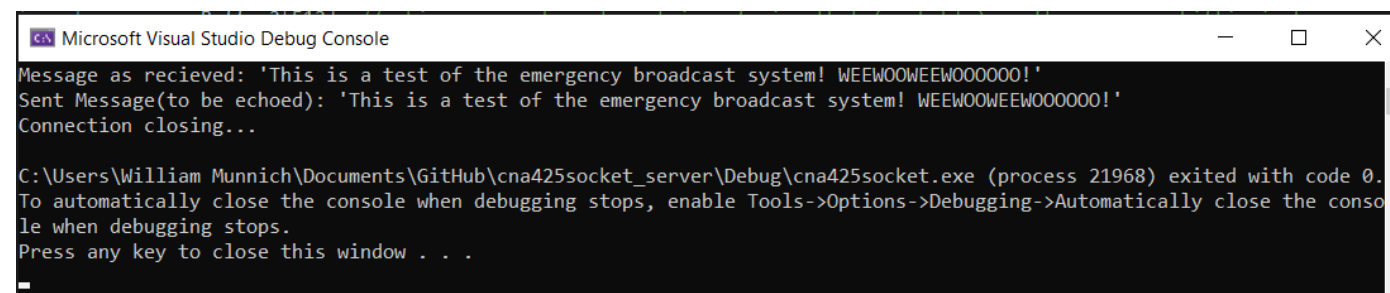
Please turn in your code along with evidence that you have run the program and that it works.

Proof Client:



```
Windows PowerShell
PS C:\Users\William Munnich\Documents\GitHub\cna425socket_client\Debug> .\cna425socket_client.exe 127.0.0.1
Enter what you want to send to the server: This is a test of the emergency broadcast system! WEEW00WEEW000000!
To be sent to server 'This is a test of the emergency broadcast system! WEEW00WEEW000000!'
Echoed message: 'This is a test of the emergency broadcast system! WEEW00WEEW000000!'
Connection closed
PS C:\Users\William Munnich\Documents\GitHub\cna425socket_client\Debug>
```

Proof Server:



```
Microsoft Visual Studio Debug Console
Message as recieved: 'This is a test of the emergency broadcast system! WEEW00WEEW000000!'
Sent Message(to be echoed): 'This is a test of the emergency broadcast system! WEEW00WEEW000000!'
Connection closing...

C:\Users\William Munnich\Documents\GitHub\cna425socket_server\Debug\cna425socket.exe (process 21968) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

What I did to set up:

I made two separate coding assignments in Visual studio, one for the client, one for the server. I put all the c++ files in the “source files” folder.

I compiled both server and client in Visual Studio.

I first ran the server in Visual Studio, then I ran the client executable (in the visual studio project debug folder)

I ran the executable with the address parameter as follows: “.\client.exe 127.0.0.1”

(127.0.0.1 is the internal loopback address)

What I did in the Server:

```

if (iResult > 0) {
    char messageBuffer1[512]; //making a new character string of size that (probably) won't go over, shifting in here so can put null character at end so not a bunch of goballdygook at the end
    for (int i = 0; i < iResult; i++) //within the array...
    {
        messageBuffer1[i] = recvbuf[i]; //move value over one at a time
    }
    messageBuffer1[iResult] = '\0'; //append null value at end so knows it is the end of the message
    printf("Message as recieved: "); //print out
    printf(messageBuffer1); //print out, prints out an integer instead of full message if all on same print statement for some reason
    printf("\n"); //print out
}

```

```

char messageBuffer2[512]; //making a new character string of size that (probably) won't go over, shifting in here so can put null character at end so not a bunch of goballdygook at the end
for (int i = 0; i < iSendResult; i++) //within the array...
{
    messageBuffer2[i] = recvbuf[i]; //move value over one at a time
}
messageBuffer2[iSendResult] = '\0'; //append null value at end so knows it is the end of the message
printf("Sent Message(to be echoed): "); //print out
printf(messageBuffer2);
printf("\n");

```

For The server, I moved the data in the initial character array into another one that's easier to edit then added a null character at the end (the length of the message) so that it doesn't print out a bunch of bad stuff at the end of the message. I repeated this for the incoming and the sending of the message.

What I did on the Client:

```

163
164
165     cout << "To be sent to server " << message << "\n";
166

```

```

186
187
188     iResult = recv(ConnectSocket, recvbuf, recvbuflen, 0);
189     char messageBuffer1[512];
190     if (iResult > 0) {
191         //making a new character string of size that (probably) won't go over, shifting in here so can put null character at end so not a bunch of goballdygook at the end
192         for (int i = 0; i < iResult; i++) //within the array...
193         {
194             messageBuffer1[i] = recvbuf[i]; //move value over one at a time
195         }
196         messageBuffer1[iResult] = '\0'; //append null value at end so knows it is the end of the message
197         printf("Echoed message: "); //print out
198         printf(messageBuffer1); //print out, prints out an integer instead of full message if all on same print statement for some reason
199         printf("\n"); //print out

```

For the client, the sending part, it's easy to print out, just used "cout".

For the Echo back I had to use the same method as with the server. Move to another char array, append a null character, then print that array.