CIS-427

With Dr. Zheng Song

Student: Demetrius Johnson

Program Assignment 1: Socket Programming

21 October 2022

Introduction and Special Notes

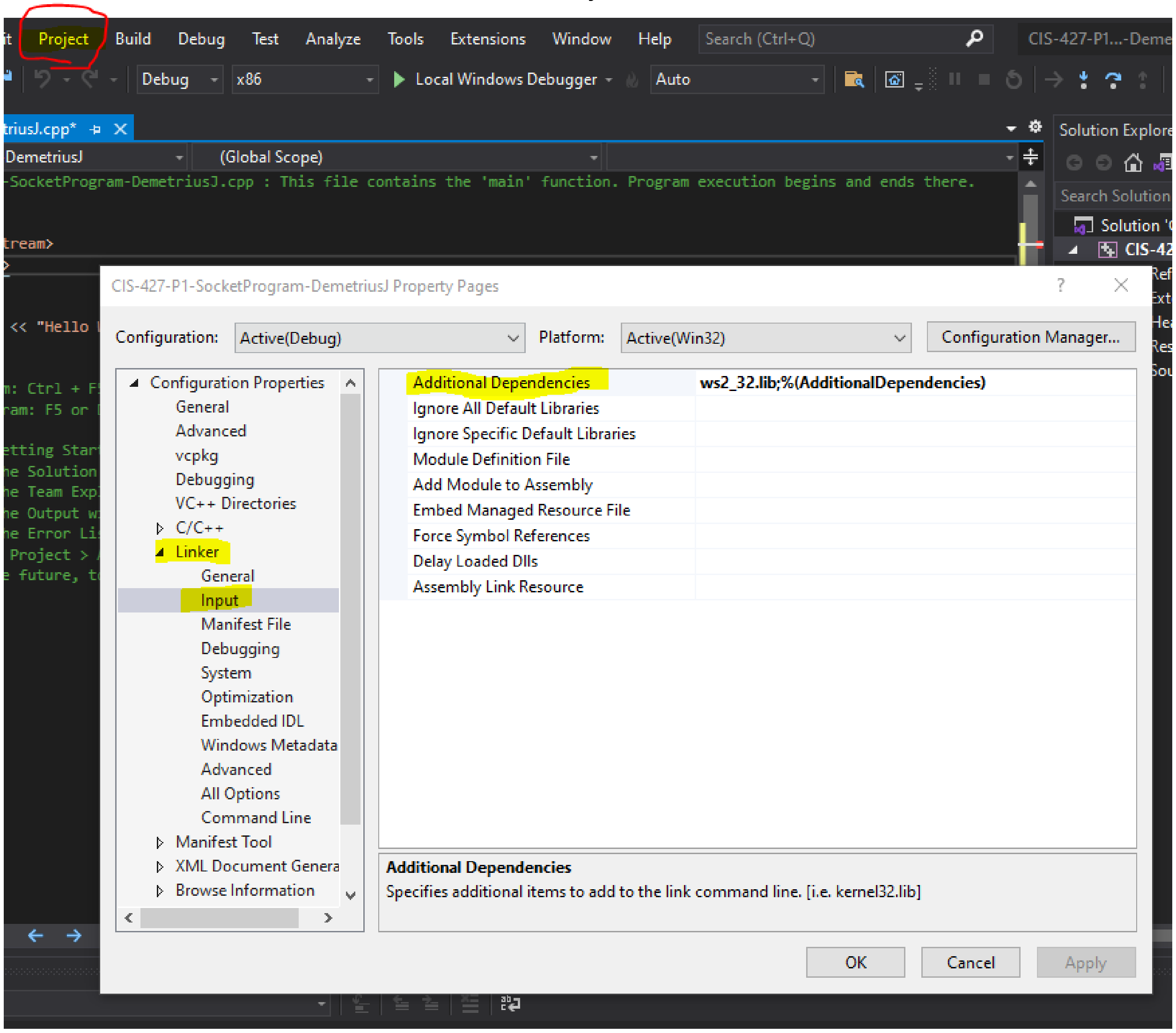
As in the next section, make sure to add the ws2_32.lib library. This must be linked before compiling otherwise the header files used will be missing the files with definitions (in ws2_32.lib) of their declarations. This is a Windows C++ server-client application that I wrote. I would like to recognize the udemy.com course: "*Socket Programming A to Z – Windows and Linux in C and C++*" by **Sonali Shrivastava**. I used the online course to help me to understand socket programming on both Linux and Windows machines, and she even does a demonstration and provides sample code for a basic server-client program. All of my program I wrote in the main function and the comments are very well done; you can easily follow along and see how I implemented and explain everything, and the code for both client and server is not that long at all (only about 200 lines each, including comments).

Lastly, I would like to mention that the README.TXT file in the *source code* folder provides information given here in this report and some additional information in the event that you run into any compilation issues. Please reach out to me if you have any issues as my program and its compilation on my machine works without any issues.

Setting up the IDE (integrated develop environment) with *Visual Studio 19 (Visual C++ 19)*

First, I needed to link the ws2_32.lib library to my project so that I can use the socket program functions defined in the library for Windows 32-bit machines and so that when I compile my program, I will not getting any errors due to missing code/libraries.

- I had to go to project→properties→Linker→Input→Additional Dependencies→click drop down and select “edit” so that I could add *ws2_32.lib* library:



```
-SocketProgram-DemetriusJ.cpp : This file contains the 'main' function. Program execution begins and ends there.
```

CIS-427-P1-SocketProgram-DemetriusJ Property Pages

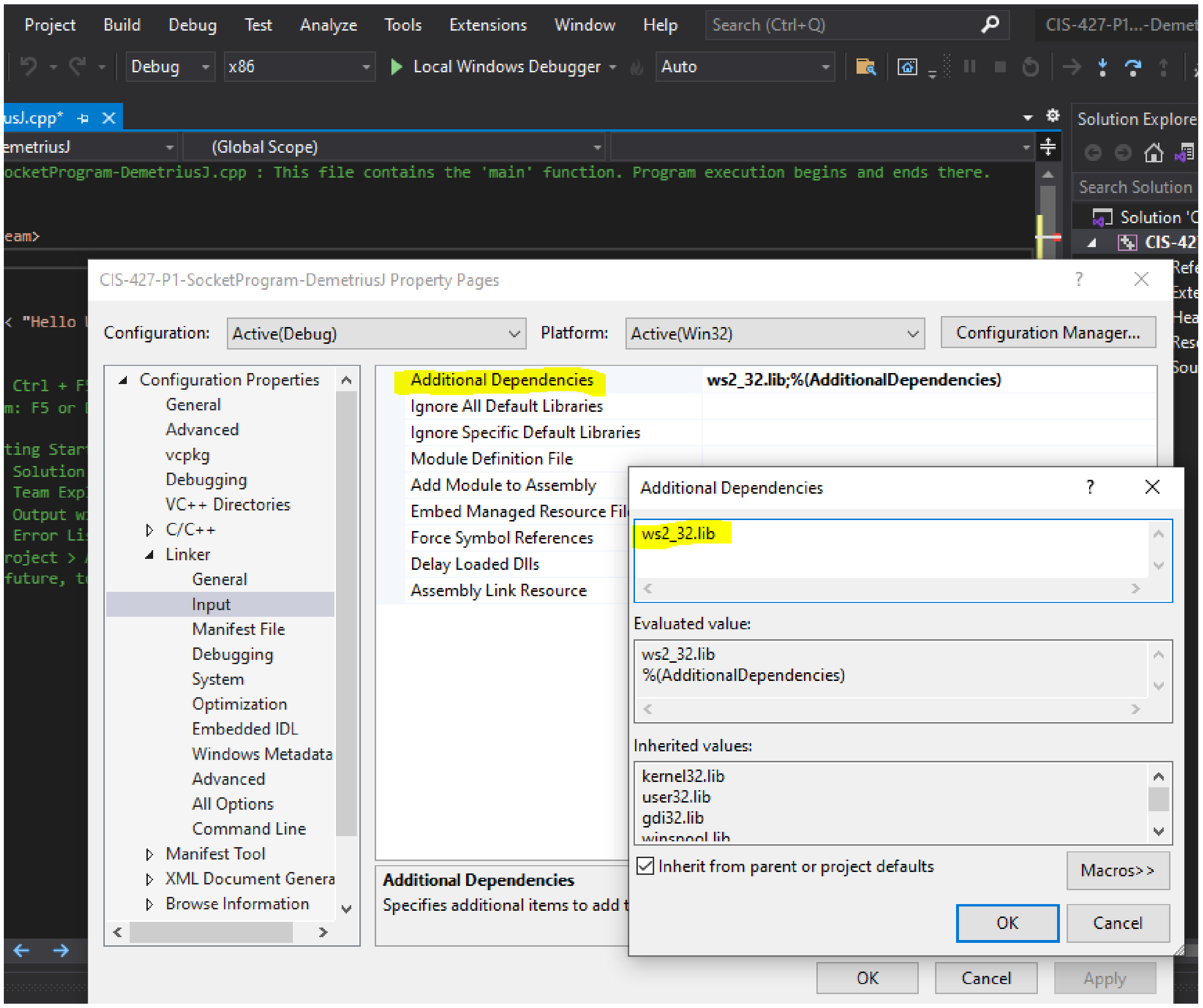
Configuration: Active(Debug) Platform: Active(Win32) Configuration Manager...

Configuration Properties
- General
- Advanced
- vcpkg
- Debugging
- VC++ Directories
- C/C++
- Linker
  - General
  - Input
  - Manifest File
  - Debugging
  - System
  - Optimization
  - Embedded IDL
  - Windows Metadata
  - Advanced
  - All Options
  - Command Line
- Manifest Tool
- XML Document Genera
- Browse Information

Additional Dependencies	ws2_32.lib;%(AdditionalDependencies)
Ignore All Default Libraries	
Ignore Specific Default Libraries	
Module Definition File	
Add Module to Assembly	
Embed Managed Resource File	
Force Symbol References	
Delay Loaded Dlls	
Assembly Link Resource	

Additional Dependencies
Specifies additional items to add to the link command line. [i.e. kernel32.lib]

-

- 
CIS-427-P1-SocketProgram-DemetriusJ Property Pages

Configuration: Active(Debug) Platform: Active(Win32)

Configuration Properties > Linker > Input

Additional Dependencies: ws2_32.lib;%(AdditionalDependencies)

Additional Dependencies

ws2_32.lib

Evaluated value:

ws2_32.lib
%(AdditionalDependencies)

Inherited values:

kernel32.lib
user32.lib
gdi32.lib
winspool.lib

☑ Inherit from parent or project defaults

Additional Dependencies
Specifies additional items to add t

    - Then click "OK" and "Apply" accordingly. Now, I have successfully linked the library (where the source code is) so that I can #include the appropriate header files and use the methods that are a part of that library.
    - Notice under "inherited values" which libraries are already added to the project by default, such as *kernel32.lib* or *user32.lib*.
- *Note to self:*

LIB vs DLL

- A DLL is a library that contains functions that can be called by applications at run-time while LIB is a static library whose code needs to be called during the compilation.
- Using LIB would result in a single file that is considerable bigger while you end up with multiple smaller files with DLL's.

Source code (client and server programs)

ServerSocket.cpp

```
//Author: Demetrius Johnson
//For: CIS-427 With Dr. Zheng Song at UM-Dearborn
//Date created: 10-21-22
//Date modified: 10-22-22
//purpose: Create the server for the client-server socket programming application
    //to send and update BANNER message of the day service ran on the server.

#include <iostream>
#include <Winsock2.h> //need this for socket programing in Windows
#include <Ws2tcpip.h> //need this for updated functions that replace depracated functions
(such as inet_addr() function)
#define APP_SOCKET_PORT 5555
using namespace std;

int main()
{
    WSADATA WSAData;    //WinSock data structure

    SOCKET serverSock, clientSock; //note: SOCKET is a descriptor to identify SOCKADDR_IN
structures: use serverSock to listen to incoming connection requests; use clientSock for
accepting incoming requests;
                                   //We do this so that the server can listen and accept
requests at the same time.

    SOCKADDR_IN serverAddr, clientAddr;  // The sockaddr_in structure specifies the
address family, IP address, and port of the server to be connected to.

    int check_WSAStartup = WSAStartup(MAKEWORD(2, 0), &WSAData); //initiates use of the
Winsock DLL by a process; we are using Winsock v 2.0 --> 2,0
    if (check_WSAStartup != NO_ERROR) {

        wprintf(L"WSAStartup function failed with error: %d\n", check_WSAStartup);
        return 1;
    }

    //initialize socket and get descriptor
    serverSock = socket(AF_INET, SOCK_STREAM, 0);

    //socket initialization error checking:
    if (serverSock == INVALID_SOCKET) {
        wprintf(L"socket function failed with error: %ld\n", WSAGetLastError());
        WSACleanup();
        return 1;
    }


    //inet_pton() function converts ASCII string to binary IP address to be stored in
addr object --> assign the IP to the serverAddr socket.
    //AF_INET specifies IPV4
    //addr.sin_port = htons(port #) assigns port# to the addr socket.

    serverAddr.sin_addr.s_addr = INADDR_ANY; //INADDR_ANY means listen to all ip
addresses of all interfaces on the local machine
    serverAddr.sin_family = AF_INET;
    serverAddr.sin_port = htons(APP_SOCKET_PORT);

    //serverSock descriptor to serverAddr structure
    bind(serverSock, (SOCKADDR *)&serverAddr, sizeof(serverAddr));

    //bind error checking:
    if (serverSock == SOCKET_ERROR) {
        wprintf(L"bind function failed with error %d\n", WSAGetLastError());
        serverSock = closesocket(serverSock);
        if (serverSock == SOCKET_ERROR)
            wprintf(L"closesocket function failed with error %d\n", WSAGetLastError());
        WSACleanup();
        return 1;
    }

    //listen on the serverSock and check for errors when we try to listen
    if (listen(serverSock, 0) == SOCKET_ERROR)
        wprintf(L"listen function failed with error: %d\n", WSAGetLastError());

    //WSAGetLastError allows you to get the most recent error code (if there were any)
stored in a winsock2.h variable
    if (WSAGetLastError() == 10060) //error code 10060 means connection timed out
(failed)
    {
        cout << "...connection timed out...\n...exiting program...\n";
        return 1;
    }

    cout << "Listening for incoming connections..." << endl;

    //create a buffer for sending and receiving data
    char buffer[1024];
    int clientAddrSize = sizeof(clientAddr); //get size of clientAddr struct size

    //now accept a client that arrives at the serverSock that we are listening on; set
clentSock descriptor
    //by setting clientAddr structure equal to the connection/structure (SOCKADDR_IN)
that serverSock is listening to.
    if((clientSock = accept(serverSock, (SOCKADDR *)&clientAddr, &clientAddrSize)) !=
INVALID_SOCKET)
    {
        cout << "Client connected!" << endl;

        //receive initial message from client:
        memset(buffer, 0, sizeof(buffer)); //reset buffer (zero it --> set all values in
the array to 0)
        recv(clientSock, buffer, sizeof(buffer), 0);
        cout << "Client says: " << buffer << endl;


        //send initial message to client:
        cout << "--SENDING a message to Client--" << endl;
        memset(buffer, 0, sizeof(buffer)); //reset buffer (zero it --> set all values in
the array to 0)
        strcpy_s(buffer, "\n\tWelcome to the Yet Another Message of the Day (YAMOTD)
Banner server.\n\t~We are connected over TCP/IP.~\n");
        send(clientSock, buffer, sizeof(buffer), 0);
        cout << "--message SENT to Client--" << endl;
    }
    else {//error checking output

        wprintf(L"accept failed with error: %ld\n", WSAGetLastError());
        cout << "\n...invalid socket...closing connection...\n";
        closesocket(serverSock);
        closesocket(clientSock);
        WSACleanup();
        return 1;
    }


////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////
    //CONNECTION IS NOW ESTABLISHED AND TESTED, WE CAN NOW SEND AND RECEIVE ON
ClientSocket - BEGIN "YET ANOTHER MESSEAGE OF THE DAY BANNER" (YAMOTD) PROGRAM:

////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////

    char YAMOTD[200] = "An apple a day keeps the doctor away.\n"; //default banner
message
    char MSGGET[20] = "MSGGET\n";
    char MSGSTORE[20] = "MSGSTORE\n";
    char QUIT[20] = "QUIT\n";
    char MSG_200_OK[20] = "200 OK\n\t";
    //NOTE: if strings are EQUAL, strcmp (string compare) returns 0

    //run this loop until client wishes to close the connection
    while (true) {

        //receive message from client:
        memset(buffer, 0, sizeof(buffer)); //reset buffer (zero it --> set all values in
the array to 0)
        recv(clientSock, buffer, sizeof(buffer), 0);
        cout << "Client says: " << buffer << endl;

     //MSGGET REQUEST:
        //if buffer == MSGGET
        if (strcmp(buffer, MSGGET) == 0) {

            //send 200 OK AND YAMOTD message to client:
            cout << "--SENDING YAMOTD to Client--" << endl;
            memset(buffer, 0, sizeof(buffer)); //reset buffer (zero it --> set all values
in the array to 0)
            //next 2 lines --> buffer =  MSG_200_OK + YAMOTD:
            strcpy_s(buffer, MSG_200_OK);
            strcat_s(buffer, YAMOTD);
            //send message to client:
            send(clientSock, buffer, sizeof(buffer), 0);
            cout << "--YAMOTD SENT to Client--" << endl;


        }
     //MSGSTORE REQUEST:
        //if buffer == MSFSTORE
        else if (strcmp(buffer, MSGSTORE) == 0) {
            //send "200 OK" message to client:
            cout << "--SENDING '200 OK' to Client--" << endl;
            memset(buffer, 0, sizeof(buffer)); //reset buffer (zero it --> set all values
in the array to 0)
            strcpy_s(buffer, "200 OK (input banner message terminated with newline, 200
char or less):\n");
            send(clientSock, buffer, sizeof(buffer), 0);
            cout << "--'200 OK' SENT to Client--" << endl;

            //receive message from client:
            memset(buffer, 0, sizeof(buffer)); //reset buffer (zero it --> set all values
in the array to 0)
            recv(clientSock, buffer, sizeof(buffer), 0);
            cout << "Client YAMOTD Banner reads: " << buffer << endl;

            //store message:
            cout << "~storing message~" << endl;
            memset(YAMOTD, 0, sizeof(YAMOTD)); //reset buffer (zero it --> set all values
in the array to 0)
            strcpy_s(YAMOTD, buffer);

            //send confirmation to client:
            cout << "--SENDING banner update confirmation to Client--" << endl;
            memset(buffer, 0, sizeof(buffer)); //reset buffer (zero it --> set all values
in the array to 0)
            strcpy_s(buffer, "200 OK ~banner updated by server succesfully~\n");
            send(clientSock, buffer, sizeof(buffer), 0);
            cout << "--banner update confirmation SENT to Client--" << endl;


        }
      //QUIT REQUEST:
        //if buffer == QUIT
        else if (strcmp(buffer, QUIT) == 0) {


            //send "200 OK" quit message to client:
            cout << "--SENDING '200 OK' quit confirmation to Client--" << endl;
            memset(buffer, 0, sizeof(buffer)); //reset buffer (zero it --> set all values
in the array to 0)
            strcpy_s(buffer, "200 OK...closing socket connection...\n");
            send(clientSock, buffer, sizeof(buffer), 0);
            cout << "--'200 OK' quit confirmation SENT to Client--" << endl;

            //close all sockets and exit server application
            closesocket(serverSock);
            closesocket(clientSock);
            WSACleanup();
            cout << "Client disconnected. Exiting server program..." << endl;
            system("pause");
            return 0;
        }
      //INVALID REQUEST:
        else
        {
            //send error message to client:
            cout << "--SENDING error message to Client--" << endl;
            memset(buffer, 0, sizeof(buffer)); //reset buffer (zero it --> set all values
in the array to 0)
            strcpy_s(buffer, "~error: invalid request. Please try again (MSGGET or
MSGSTORE or QUIT)\n");
            send(clientSock, buffer, sizeof(buffer), 0);
            cout << "--error message SENT to Client--" << endl;

        }
    }
}
```

ClientSocket.cpp

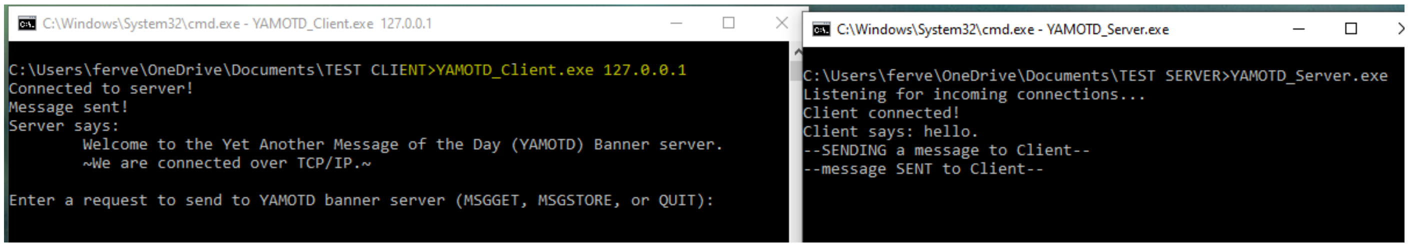
```
//Author: Demetrius Johnson
//For: CIS-427 With Dr. Zheng Song at UM-Dearborn
//Date created: 10-21-22
//Date modified: 10-22-22
//purpose: Create the client for the client-server socket programming application
    //to get and update BANNER message of the day service ran on the remote server.

#include <iostream>
#include <Winsock2.h> //need this for socket programing in Windows
#include <Ws2tcpip.h> //need this for updated functions that replace depracated functions
(such as inet_addr() function)
#define APP_SOCKET_PORT 5555
using namespace std;

//this program's main function takes 1 commmandline parameter: server ip address as per
the requirement for this assignment.

int main(int argc, char** argv ) //remember: element 0 in the vector is the file name;
first commandline argument begins at element 1
{
    WSADATA WSAData;
    SOCKET serverSock; //note: SOCKET is a descriptor to identify SOCKADDR_IN structures;
we will use serverSock/addr to connect to the server via IP address and TCP port#
    SOCKADDR_IN addr;  // The sockaddr_in structure specifies the address family, IP
address, and port of the server to be connected to.

    WSAStartup(MAKEWORD(2,0), &WSAData);            //initiates use of the Winsock DLL by
a process; we are using Winsock v 2.0 --> 2,0
    serverSock = socket(AF_INET, SOCK_STREAM, 0);   //AF_INET specifies IPv4, SOCK_STREAM
means use TCP, 0 means do not use a specific protocol (i.e. 1 = ICMP)
    char ip_address[16] = "127.0.0.1";              //default ip will be loopback address
of local machine

    //only update IP with command line parameter if there is one present and if no more
than 1 parameter has been passed in.
    if (argc == 2)
        strcpy_s(ip_address, argv[1]);

    //inet_pton() function converts ASCII string to binary IP address to be stored in
addr object --> assign the IP to the addr socket.
    //AF_INET specifies IPV4
    //addr.sin_port = htons(port #) assigns port# to the addr socket.
    //inet_pton() function will convert the IPv4 string into binary and store the address
in the the struct in the proper field (replaces the depracated inet_addr() function)
    inet_pton(AF_INET, ip_address, &addr.sin_addr.s_addr);

    addr.sin_family = AF_INET;
    addr.sin_port = htons(APP_SOCKET_PORT);

    connect(serverSock, (SOCKADDR*)&addr, sizeof(addr));   //try to connect to server
using server IP and Port number we would like to connect on

    //WSAGetLastError allows you to get the most recent error code (if there were any)
stored in a winsock2.h variable
    if (WSAGetLastError() == 10060) //error code 10060 means connection timed out
(failed)
    {
        cout << "...connection timed out...\n...exiting program...\n";
        closesocket(serverSock);
        WSACleanup();
        return 1;
    }
    if (WSAGetLastError() == 10061) //error code 10061 means connection refused by
host/target machine (server refused connection)
    {
        cout << "...connection refused by target machine...\n...exiting program...\n";
        closesocket(serverSock);
        WSACleanup();
        return 1;
    }

    cout << "Connected to server!" << endl;

    //test sending messages to server on serverSock:
    char buffer[1024]={'h', 'e', 'l', 'l', 'o', '.'};
    send(serverSock, buffer, sizeof(buffer), 0);
    cout << "Message sent!" << endl;


    //test receiving messages from server on serverSock:
    memset(buffer, 0, sizeof(buffer)); //reset buffer (zero it --> set all values in the
array to 0)
    recv(serverSock, buffer, sizeof(buffer), 0);
    cout << "Server says: " << buffer << endl;

    ////////////////////////////////////////////////////////////////////////
    //READY TO BEGIN REQUESTING YAMOTD Application services from server:
    ///////////////////////////////////////////////////////////////////////
    char REQUEST[20];

    //NOTE: if strings are EQUAL, strcmp (string compare) returns 0
   do {
       //get request input from USER:
       cout << "Enter a request to send to YAMOTD banner server (MSGGET, MSGSTORE, or
QUIT): ";
       cin >> REQUEST;          //get input from user
       strcat_s(REQUEST, "\n");  //concatenate newline character to input as required for
this client-server application
```

```
        //send request to server:
        memset(buffer, 0, sizeof(buffer)); //reset buffer (zero it --> set all values in
the array to 0)
        strcpy_s(buffer, REQUEST); //copy user request message into buffer
        send(serverSock, buffer, sizeof(buffer), 0); //send request to server
        cout << "Message sent!" << endl;

        //Receive request response from server:
        memset(buffer, 0, sizeof(buffer)); //reset buffer (zero it --> set all values in
the array to 0)
        recv(serverSock, buffer, sizeof(buffer), 0);
        cout << "Server says: " << buffer << endl;

        //case: requesting to update the banner
        if (strcmp(REQUEST, "MSGSTORE\n") == 0) {

            //send new banner to server:
            memset(buffer, 0, sizeof(buffer));  //reset buffer (zero it --> set all
values in the array to 0)
            cin.ignore(sizeof(buffer), '\n');   //need to flush cin buffer before the
next operation
            cin.getline(buffer, 200, '\n');     //banner is limited by 200 characters
            strcat_s(buffer, "\n");             //concatenate newline character to input
as required for this client-server application
            send(serverSock, buffer, sizeof(buffer), 0); //send banner to server
            cout << "Message sent!" << endl;

            //Receive update response from server:
            memset(buffer, 0, sizeof(buffer)); //reset buffer (zero it --> set all values
in the array to 0)
            recv(serverSock, buffer, sizeof(buffer), 0);
            cout << "Server says: " << buffer << endl;

        }

    } while (strcmp(REQUEST, "QUIT\n") != 0);


    closesocket(serverSock);
    WSACleanup();
    cout << "Socket closed." << endl << endl;

    system("pause");
    return 0;
}
```

Test cases:

Case 1: using default IP address (set to loopback: 127.0.0.1)

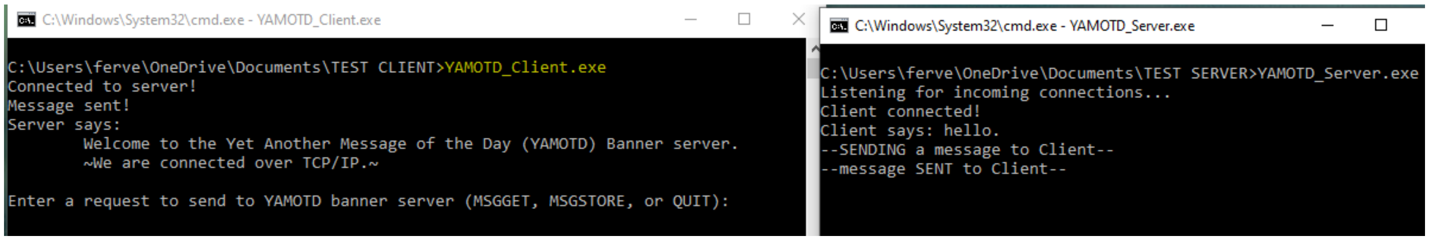
- I set the default IP address to 127.0.0.1 in the event that the IP address command line parameter receives no input, or the number of parameters passed in exceeds 2 (first parameter is default as file name, first command line parameter is the second parameter).
- I will show when I specify the default:



```
C:\Windows\System32\cmd.exe - YAMOTD_Client.exe 127.0.0.1

C:\Users\ferve\OneDrive\Documents\TEST CLIENT>YAMOTD_Client.exe 127.0.0.1
Connected to server!
Message sent!
Server says:
        Welcome to the Yet Another Message of the Day (YAMOTD) Banner server.
        ~We are connected over TCP/IP.~

Enter a request to send to YAMOTD banner server (MSGGET, MSGSTORE, or QUIT):
```

```
C:\Windows\System32\cmd.exe - YAMOTD_Server.exe

C:\Users\ferve\OneDrive\Documents\TEST SERVER>YAMOTD_Server.exe
Listening for incoming connections...
Client connected!
Client says: hello.
--SENDING a message to Client--
--message SENT to Client--
```

- Or when I simply just pass no command line parameters in:

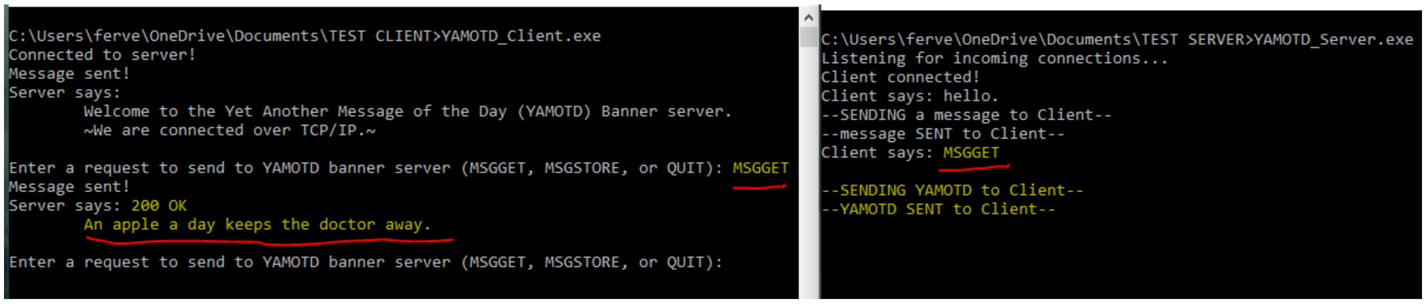


```
C:\Windows\System32\cmd.exe - YAMOTD_Client.exe

C:\Users\ferve\OneDrive\Documents\TEST CLIENT>YAMOTD_Client.exe
Connected to server!
Message sent!
Server says:
        Welcome to the Yet Another Message of the Day (YAMOTD) Banner server.
        ~We are connected over TCP/IP.~

Enter a request to send to YAMOTD banner server (MSGGET, MSGSTORE, or QUIT):
```

```
C:\Windows\System32\cmd.exe - YAMOTD_Server.exe

C:\Users\ferve\OneDrive\Documents\TEST SERVER>YAMOTD_Server.exe
Listening for incoming connections...
Client connected!
Client says: hello.
--SENDING a message to Client--
--message SENT to Client--
```

- Notice for both screenshots above that I wrote default communication checks between client and server as they establish a connection; the server listens on one socket while the client attempts to connect; upon connecting, the server accepts the connection on another socket, and it is this socket that they communicate on (named "ClientSock" in my main program of the Server application, and "serverSock" in the Client application main program).
- Once connection is established, they can begin communicating regularly (exchanging messages until connection is closed).

MSGGET

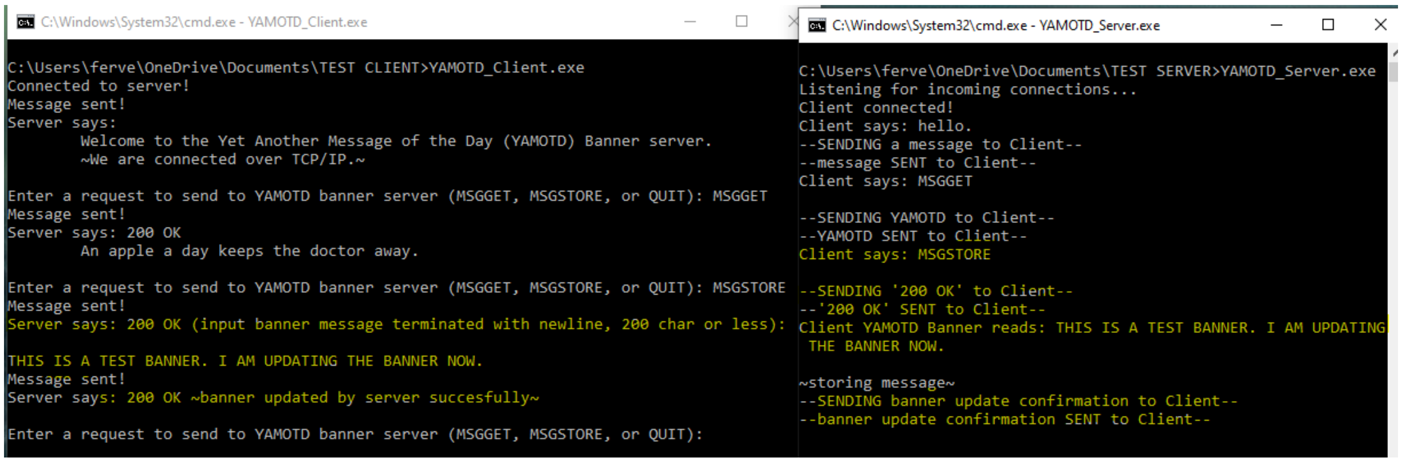


```
C:\Users\ferve\OneDrive\Documents\TEST CLIENT>YAMOTD_Client.exe
Connected to server!
Message sent!
Server says:
        Welcome to the Yet Another Message of the Day (YAMOTD) Banner server.
        ~We are connected over TCP/IP.~

Enter a request to send to YAMOTD banner server (MSGGET, MSGSTORE, or QUIT): MSGGET
Message sent!
Server says: 200 OK
        An apple a day keeps the doctor away.

Enter a request to send to YAMOTD banner server (MSGGET, MSGSTORE, or QUIT):
```

```
C:\Users\ferve\OneDrive\Documents\TEST SERVER>YAMOTD_Server.exe
Listening for incoming connections...
Client connected!
Client says: hello.
--SENDING a message to Client--
--message SENT to Client--
Client says: MSGGET

--SENDING YAMOTD to Client--
--YAMOTD SENT to Client--
```

- Notice above, that the default banner is displayed upon first request to server since the banner has not been updated.

MSGSTORE



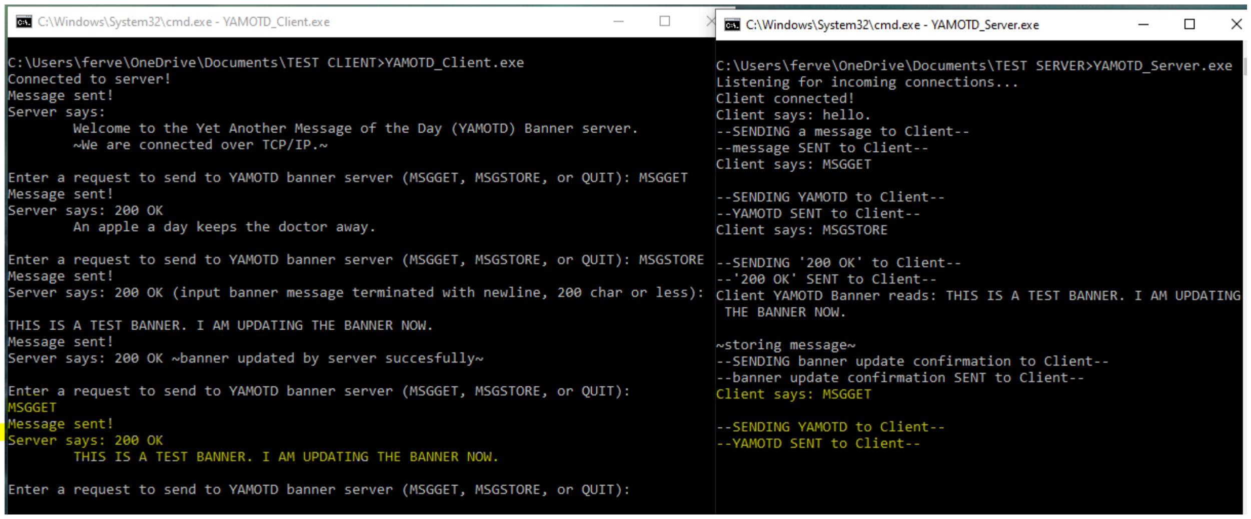
C:\Windows\System32\cmd.exe - YAMOTD_Client.exe

```
C:\Users\ferve\OneDrive\Documents\TEST CLIENT>YAMOTD_Client.exe
Connected to server!
Message sent!
Server says:
        Welcome to the Yet Another Message of the Day (YAMOTD) Banner server.
        ~We are connected over TCP/IP.~

Enter a request to send to YAMOTD banner server (MSGGET, MSGSTORE, or QUIT): MSGGET
Message sent!
Server says: 200 OK
        An apple a day keeps the doctor away.

Enter a request to send to YAMOTD banner server (MSGGET, MSGSTORE, or QUIT): MSGSTORE
Message sent!
Server says: 200 OK (input banner message terminated with newline, 200 char or less):

THIS IS A TEST BANNER. I AM UPDATING THE BANNER NOW.
Message sent!
Server says: 200 OK ~banner updated by server succesfully~

Enter a request to send to YAMOTD banner server (MSGGET, MSGSTORE, or QUIT):
```

C:\Windows\System32\cmd.exe - YAMOTD_Server.exe

```
C:\Users\ferve\OneDrive\Documents\TEST SERVER>YAMOTD_Server.exe
Listening for incoming connections...
Client connected!
Client says: hello.
--SENDING a message to Client--
--message SENT to Client--
Client says: MSGGET

--SENDING YAMOTD to Client--
--YAMOTD SENT to Client--
Client says: MSGSTORE

--SENDING '200 OK' to Client--
--'200 OK' SENT to Client--
Client YAMOTD Banner reads: THIS IS A TEST BANNER. I AM UPDATING
 THE BANNER NOW.

~storing message~
--SENDING banner update confirmation to Client--
--banner update confirmation SENT to Client--
```

- Above, I have updated the banner. Now I will call MSGGET again to prove banner has been updated:



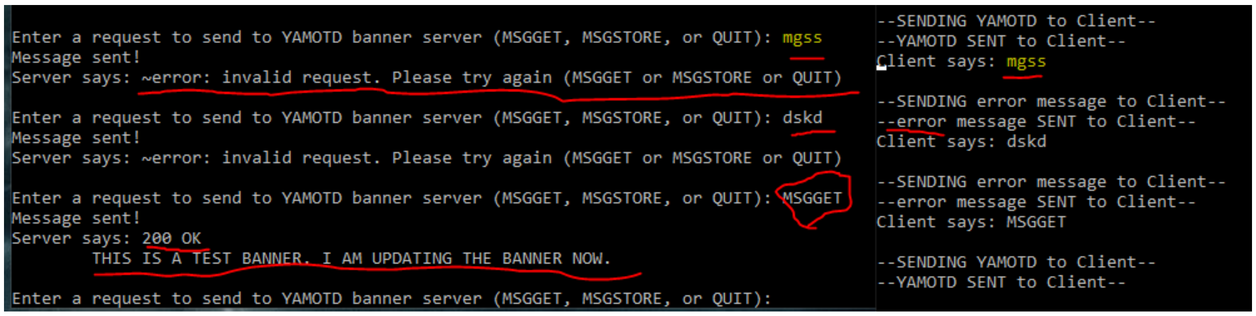
C:\Windows\System32\cmd.exe - YAMOTD_Client.exe

```
C:\Users\ferve\OneDrive\Documents\TEST CLIENT>YAMOTD_Client.exe
Connected to server!
Message sent!
Server says:
        Welcome to the Yet Another Message of the Day (YAMOTD) Banner server.
        ~We are connected over TCP/IP.~

Enter a request to send to YAMOTD banner server (MSGGET, MSGSTORE, or QUIT): MSGGET
Message sent!
Server says: 200 OK
        An apple a day keeps the doctor away.

Enter a request to send to YAMOTD banner server (MSGGET, MSGSTORE, or QUIT): MSGSTORE
Message sent!
Server says: 200 OK (input banner message terminated with newline, 200 char or less):

THIS IS A TEST BANNER. I AM UPDATING THE BANNER NOW.
Message sent!
Server says: 200 OK ~banner updated by server succesfully~

Enter a request to send to YAMOTD banner server (MSGGET, MSGSTORE, or QUIT):
MSGGET
Message sent!
Server says: 200 OK
        THIS IS A TEST BANNER. I AM UPDATING THE BANNER NOW.

Enter a request to send to YAMOTD banner server (MSGGET, MSGSTORE, or QUIT):
```

C:\Windows\System32\cmd.exe - YAMOTD_Server.exe

```
C:\Users\ferve\OneDrive\Documents\TEST SERVER>YAMOTD_Server.exe
Listening for incoming connections...
Client connected!
Client says: hello.
--SENDING a message to Client--
--message SENT to Client--
Client says: MSGGET

--SENDING YAMOTD to Client--
--YAMOTD SENT to Client--
Client says: MSGSTORE

--SENDING '200 OK' to Client--
--'200 OK' SENT to Client--
Client YAMOTD Banner reads: THIS IS A TEST BANNER. I AM UPDATING
 THE BANNER NOW.

~storing message~
--SENDING banner update confirmation to Client--
--banner update confirmation SENT to Client--
Client says: MSGGET

--SENDING YAMOTD to Client--
--YAMOTD SENT to Client--
```

Invalid Request input

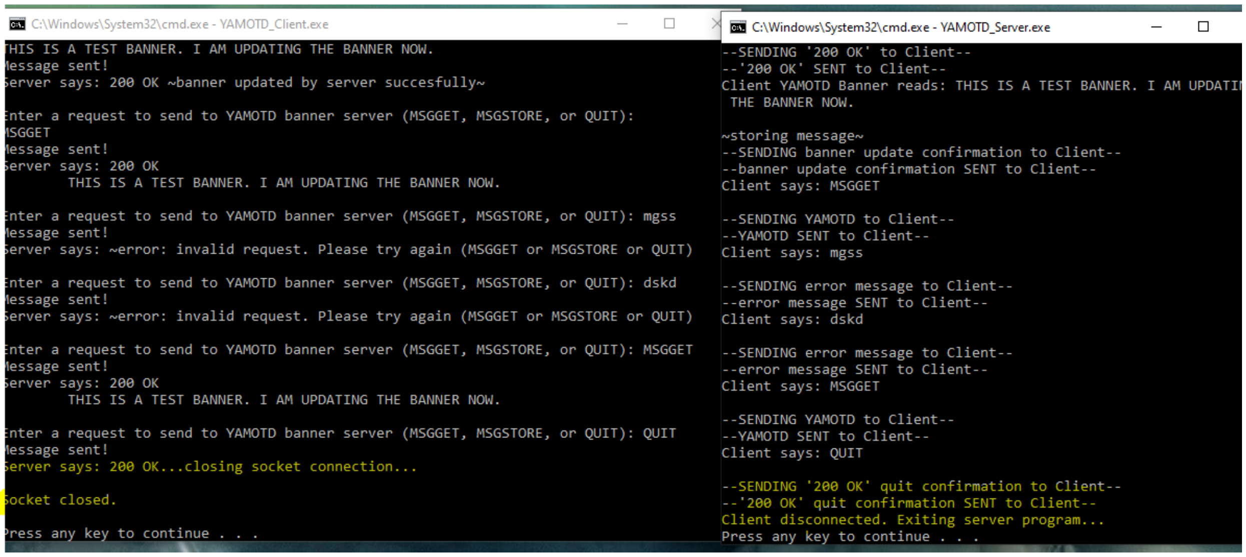


```
Enter a request to send to YAMOTD banner server (MSGGET, MSGSTORE, or QUIT): mgss
Message sent!
Server says: ~error: invalid request. Please try again (MSGGET or MSGSTORE or QUIT)

Enter a request to send to YAMOTD banner server (MSGGET, MSGSTORE, or QUIT): dskd
Message sent!
Server says: ~error: invalid request. Please try again (MSGGET or MSGSTORE or QUIT)

Enter a request to send to YAMOTD banner server (MSGGET, MSGSTORE, or QUIT): MSGGET
Message sent!
Server says: 200 OK
        THIS IS A TEST BANNER. I AM UPDATING THE BANNER NOW.

Enter a request to send to YAMOTD banner server (MSGGET, MSGSTORE, or QUIT):
```

```
--SENDING YAMOTD to Client--
--YAMOTD SENT to Client--
Client says: mgss

--SENDING error message to Client--
--error message SENT to Client--
Client says: dskd

--SENDING error message to Client--
--error message SENT to Client--
Client says: MSGGET

--SENDING YAMOTD to Client--
--YAMOTD SENT to Client--
```

- Notice above, I have sent several invalid request strings to the server and it responds appropriately and re-prompts the client to send another request. Then I show what happens when I send a valid request again (it works properly as server sends message of the day banner).

QUIT



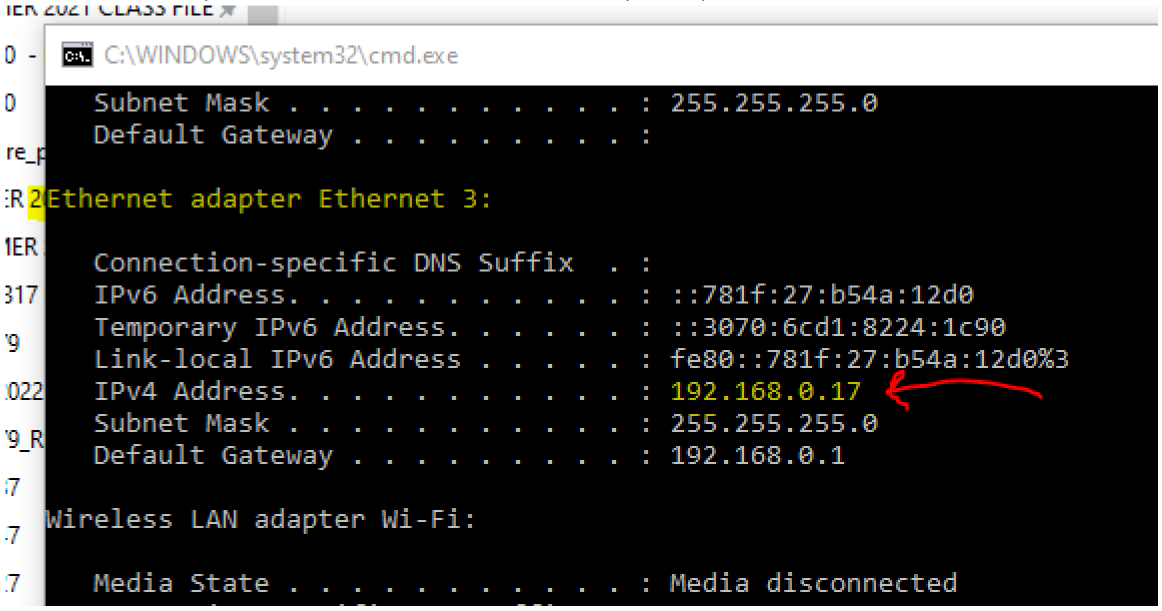
```
C:\Windows\System32\cmd.exe - YAMOTD_Client.exe
HIS IS A TEST BANNER. I AM UPDATING THE BANNER NOW.
lessage sent!
erver says: 200 OK ~banner updated by server succesfully~

nter a request to send to YAMOTD banner server (MSGGET, MSGSTORE, or QUIT):
ISGGET
lessage sent!
erver says: 200 OK
        THIS IS A TEST BANNER. I AM UPDATING THE BANNER NOW.

nter a request to send to YAMOTD banner server (MSGGET, MSGSTORE, or QUIT): mgss
lessage sent!
erver says: ~error: invalid request. Please try again (MSGGET or MSGSTORE or QUIT)

nter a request to send to YAMOTD banner server (MSGGET, MSGSTORE, or QUIT): dskd
lessage sent!
erver says: ~error: invalid request. Please try again (MSGGET or MSGSTORE or QUIT)

nter a request to send to YAMOTD banner server (MSGGET, MSGSTORE, or QUIT): MSGGET
lessage sent!
erver says: 200 OK
        THIS IS A TEST BANNER. I AM UPDATING THE BANNER NOW.

nter a request to send to YAMOTD banner server (MSGGET, MSGSTORE, or QUIT): QUIT
lessage sent!
erver says: 200 OK...closing socket connection...

ocket closed.

ress any key to continue . . .
```

```
C:\Windows\System32\cmd.exe - YAMOTD_Server.exe
--SENDING '200 OK' to Client--
--'200 OK' SENT to Client--
Client YAMOTD Banner reads: THIS IS A TEST BANNER. I AM UPDATI
 THE BANNER NOW.

~storing message~
--SENDING banner update confirmation to Client--
--banner update confirmation SENT to Client--
Client says: MSGGET

--SENDING YAMOTD to Client--
--YAMOTD SENT to Client--
Client says: mgss

--SENDING error message to Client--
--error message SENT to Client--
Client says: dskd

--SENDING error message to Client--
--error message SENT to Client--
Client says: MSGGET

--SENDING YAMOTD to Client--
--YAMOTD SENT to Client--
Client says: QUIT

--SENDING '200 OK' quit confirmation to Client--
--'200 OK' quit confirmation SENT to Client--
Client disconnected. Exiting server program...
Press any key to continue . . .
```

- Notice above, that the client requests to QUIT and receives confirmation from server before closing the socket connection. Server side also closes connections and exits program.

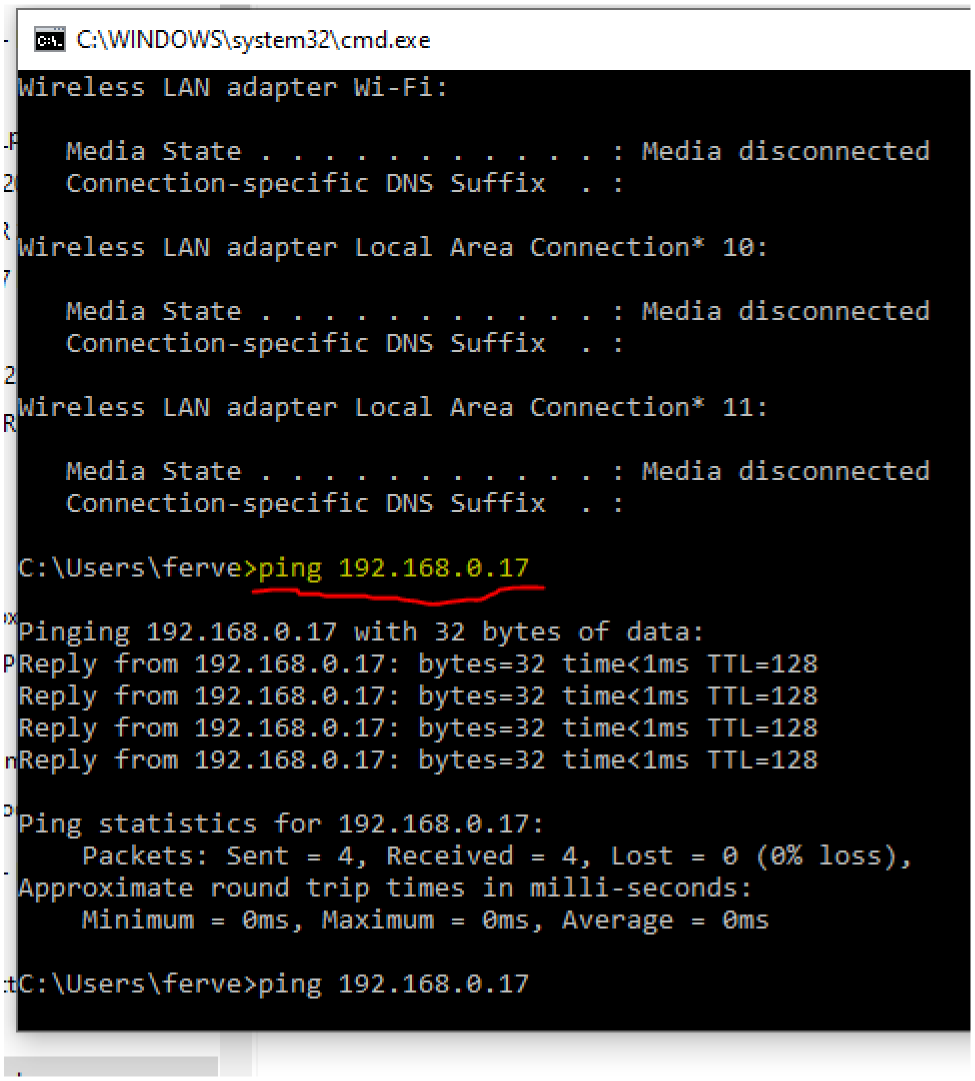
Case 2: using IP address of my ethernet adapter

- Now I will use my local host machine's ethernet adapter's private IP address:



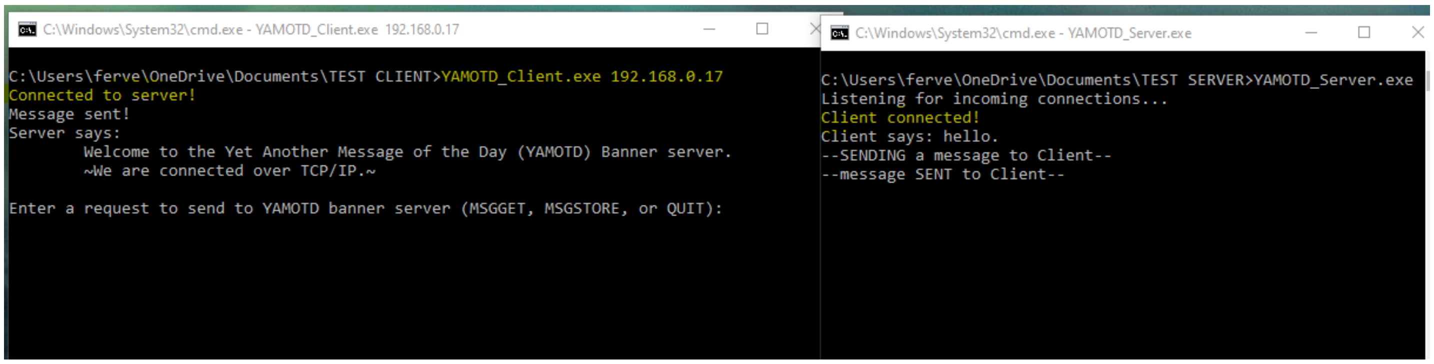
```
C:\WINDOWS\system32\cmd.exe
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . :

Ethernet adapter Ethernet 3:

   Connection-specific DNS Suffix  . :
   IPv6 Address. . . . . . . . . . . : ::781f:27:b54a:12d0
   Temporary IPv6 Address. . . . . . : ::3070:6cd1:8224:1c90
   Link-local IPv6 Address . . . . . : fe80::781f:27:b54a:12d0%3
   IPv4 Address. . . . . . . . . . . : 192.168.0.17
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.0.1

Wireless LAN adapter Wi-Fi:

   Media State . . . . . . . . . . . : Media disconnected
```

-
- Notice I have pinged it too to make sure it is working and visible to my machine (as it should be of course since it is assigned to the ethernet adapter on my local machine)L



```
C:\WINDOWS\system32\cmd.exe
Wireless LAN adapter Wi-Fi:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter Local Area Connection* 10:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter Local Area Connection* 11:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

C:\Users\ferve>ping 192.168.0.17

Pinging 192.168.0.17 with 32 bytes of data:
Reply from 192.168.0.17: bytes=32 time<1ms TTL=128
Reply from 192.168.0.17: bytes=32 time<1ms TTL=128
Reply from 192.168.0.17: bytes=32 time<1ms TTL=128
Reply from 192.168.0.17: bytes=32 time<1ms TTL=128

Ping statistics for 192.168.0.17:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\Users\ferve>ping 192.168.0.17
```

-
- Now I will actually call the client application using my local 192.168.0.17 adapter's IP; note, I specified IP_ADDR_ANY in my server application so that it will listen to all Network Adapters the machine it is running on. Note that for all test cases of this entire report, server and client are running on my same local host machine.

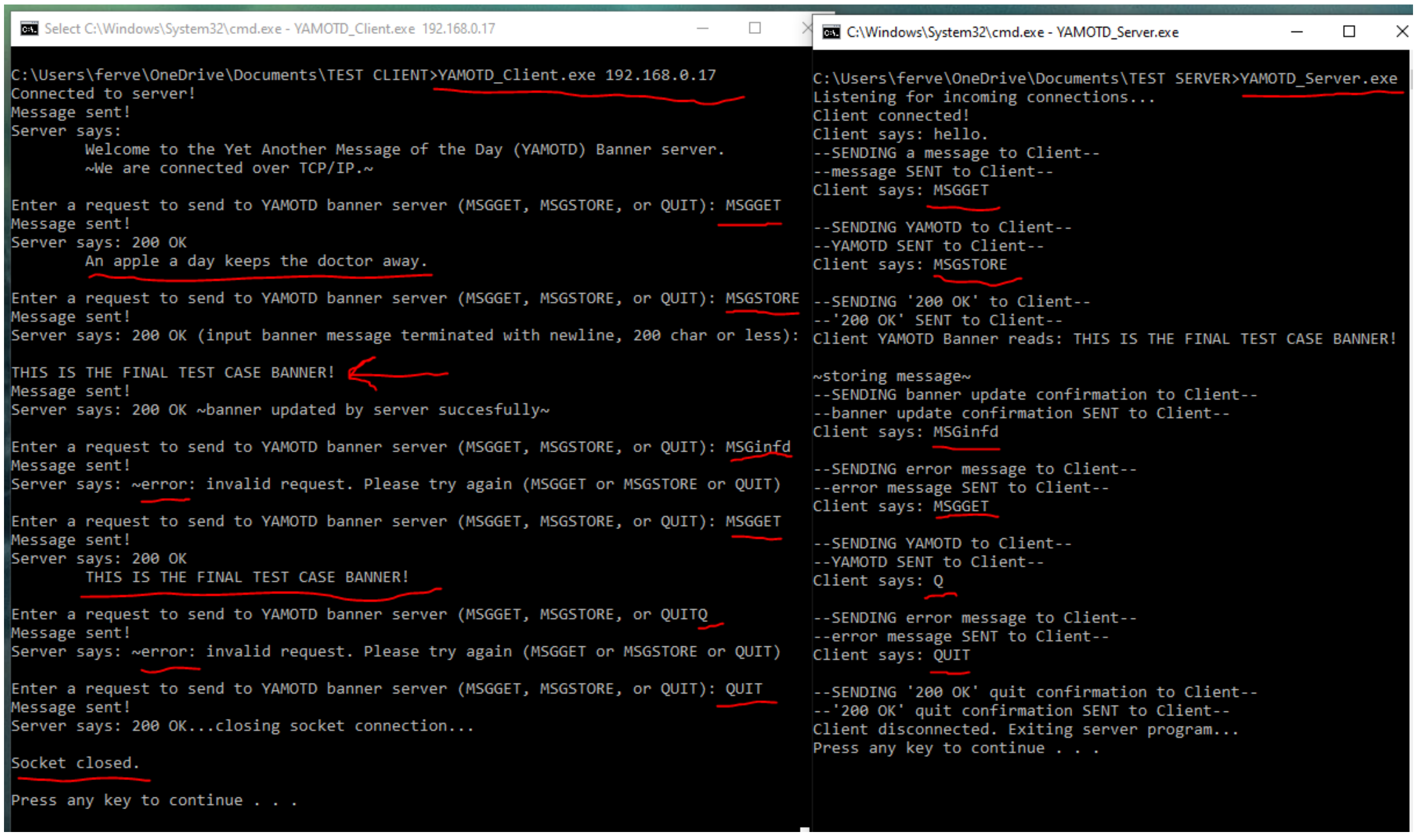


```
C:\Windows\System32\cmd.exe - YAMOTD_Client.exe 192.168.0.17

C:\Users\ferve\OneDrive\Documents\TEST CLIENT>YAMOTD_Client.exe 192.168.0.17
Connected to server!
Message sent!
Server says:
        Welcome to the Yet Another Message of the Day (YAMOTD) Banner server.
        ~We are connected over TCP/IP.~

Enter a request to send to YAMOTD banner server (MSGGET, MSGSTORE, or QUIT):
```

```
C:\Windows\System32\cmd.exe - YAMOTD_Server.exe

C:\Users\ferve\OneDrive\Documents\TEST SERVER>YAMOTD_Server.exe
Listening for incoming connections...
Client connected!
Client says: hello.
--SENDING a message to Client--
--message SENT to Client--
```

MSGGET, MSGSTORE, invalid input, MSGGET (shows updated banner), invalid input, QUIT

Now I will perform the same tests as I did previously but by connecting using the socket with the specified non-default IP address.

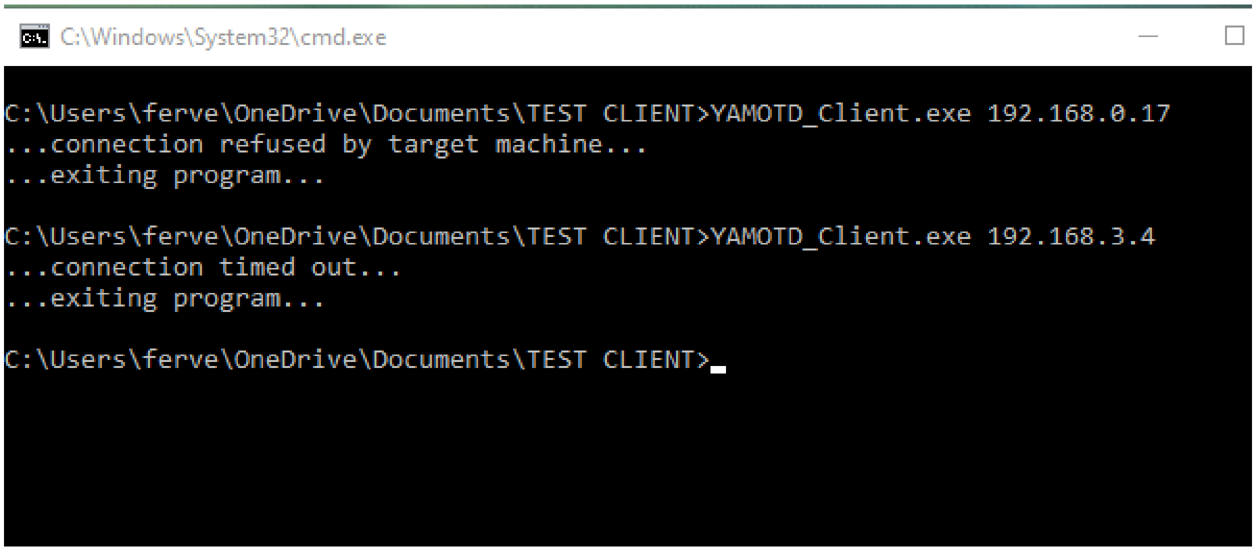

Select C:\Windows\System32\cmd.exe - YAMOTD_Client.exe 192.168.0.17
C:\Windows\System32\cmd.exe - YAMOTD_Server.exe

```
C:\Users\ferve\OneDrive\Documents\TEST CLIENT>YAMOTD_Client.exe 192.168.0.17
Connected to server!
Message sent!
Server says:
        Welcome to the Yet Another Message of the Day (YAMOTD) Banner server.
        ~We are connected over TCP/IP.~

Enter a request to send to YAMOTD banner server (MSGGET, MSGSTORE, or QUIT): MSGGET
Message sent!
Server says: 200 OK
        An apple a day keeps the doctor away.

Enter a request to send to YAMOTD banner server (MSGGET, MSGSTORE, or QUIT): MSGSTORE
Message sent!
Server says: 200 OK (input banner message terminated with newline, 200 char or less):

THIS IS THE FINAL TEST CASE BANNER!
Message sent!
Server says: 200 OK ~banner updated by server succesfully~

Enter a request to send to YAMOTD banner server (MSGGET, MSGSTORE, or QUIT): MSGinfd
Message sent!
Server says: ~error: invalid request. Please try again (MSGGET or MSGSTORE or QUIT)

Enter a request to send to YAMOTD banner server (MSGGET, MSGSTORE, or QUIT): MSGGET
Message sent!
Server says: 200 OK
        THIS IS THE FINAL TEST CASE BANNER!

Enter a request to send to YAMOTD banner server (MSGGET, MSGSTORE, or QUITQ
Message sent!
Server says: ~error: invalid request. Please try again (MSGGET or MSGSTORE or QUIT)

Enter a request to send to YAMOTD banner server (MSGGET, MSGSTORE, or QUIT): QUIT
Message sent!
Server says: 200 OK...closing socket connection...

Socket closed.

Press any key to continue . . .
```

```
C:\Users\ferve\OneDrive\Documents\TEST SERVER>YAMOTD_Server.exe
Listening for incoming connections...
Client connected!
Client says: hello.
--SENDING a message to Client--
--message SENT to Client--
Client says: MSGGET

--SENDING YAMOTD to Client--
--YAMOTD SENT to Client--
Client says: MSGSTORE

--SENDING '200 OK' to Client--
--'200 OK' SENT to Client--
Client YAMOTD Banner reads: THIS IS THE FINAL TEST CASE BANNER!

~storing message~
--SENDING banner update confirmation to Client--
--banner update confirmation SENT to Client--
Client says: MSGinfd

--SENDING error message to Client--
--error message SENT to Client--
Client says: MSGGET

--SENDING YAMOTD to Client--
--YAMOTD SENT to Client--
Client says: Q

--SENDING error message to Client--
--error message SENT to Client--
Client says: QUIT

--SENDING '200 OK' quit confirmation to Client--
--'200 OK' quit confirmation SENT to Client--
Client disconnected. Exiting server program...
Press any key to continue . . .
```

Case 3: connection timed out

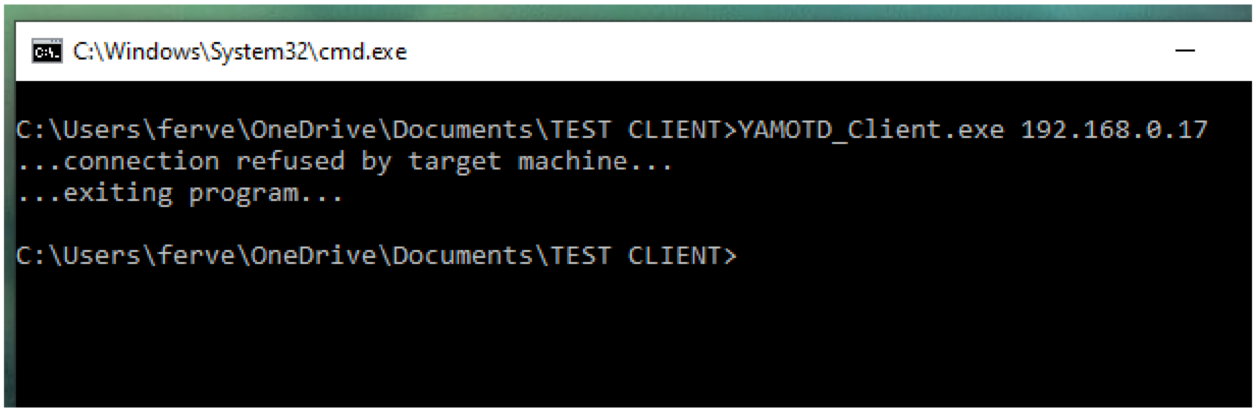
In this scenario, the IP address could not be reached (invalid host) so that the connection timed out code is thrown, which I wrote code to output that error message (and if the error message from connect() function throws a connection timed out error as well – see my code in the Client program).



C:\Windows\System32\cmd.exe

```
C:\Users\ferve\OneDrive\Documents\TEST CLIENT>YAMOTD_Client.exe 192.168.0.17
...connection refused by target machine...
...exiting program...

C:\Users\ferve\OneDrive\Documents\TEST CLIENT>YAMOTD_Client.exe 192.168.3.4
...connection timed out...
...exiting program...

C:\Users\ferve\OneDrive\Documents\TEST CLIENT>
```

Case 4: connection refused



C:\Windows\System32\cmd.exe

```
C:\Users\ferve\OneDrive\Documents\TEST CLIENT>YAMOTD_Client.exe 192.168.0.17
...connection refused by target machine...
...exiting program...

C:\Users\ferve\OneDrive\Documents\TEST CLIENT>
```

Notice I wrote an output method for when error code is thrown by winsocket telling the program that the connection was refused by target machine. The server application was not running, therefore, even though I could reach the IP address, no server process was running to allow me to make a TCP connection, so connection was refused by the machine that possessed the IP address (in this case my local machine's ethernet interface card).

Summary:

In summary, I used the Winsocket2 library from Windows to write a Windows client-server application using C++. The program works exactly as expected with no errors and even some limited error checking capabilities. It should be fairly easy to grade and compile and run as I have made all instructions, code comments, and screenshots with explanations very clear. Just about anyone with some literacy in computers and programming should be able to replicate my project and the results exactly.