**Socket – TCP Connection Between Volta(Client) and Jason(Server)**

**CompE560**

Introduction:

The Purpose of this Socket program was to get an introduction to socket connections. To do this we explored a Transmission Control Protocol (TCP) Connection between a client and a server.

I setup the connection so I would have the *Volta Server* as the Client and then the *Jason Server* as the server.

The code for both the server and the client was quite similar.

Both the codes starts with an error function that prints the error msg and then exits the code. This function is called whenever an error occurs throughout the code. Incorrect Port number, invalid host name, or bad connection will call this error function.

Both the programs then go into the struct creation. However rather than creating a struct in a regular compiler we had to use

#pragma(1)…struct…#pragma(0)

We did this so we can compress the struct and send it across the tcp connection as a whole.

In the client now we have one last function before the main. This function was created to convert the ‘int’ and ‘float’ from Big Indian to Little Indian. *Volta* is a Big Indian server and then *Jason* is a Windows system. This caused an issue when we transferred the numbers as they would appear to be different on the other side.

In the main the server and client are very similar. The only differences are at the beginning and towards the send of the code. They both create buffers and a portno. However the client is requesting and the server is accepting.

I start by creating a few integers sockfd and newsockfd for the file descriptors. Portno to store the portno that the two servers will communicate through.

The user is the prompted to enter a portno, if the user fails to do so or enters a portno that is not a minimum of 5 digits it will prompt it to the error function.

The program then creates the socket and is now listening for a client. The socket is created using three arguments: AF\_INET, SOCK\_STREAM, and 0.

A buffer is then created to store all incoming packets the buffer is first set to all zeros, so it is classified as an empty buffer.

At thus point the codes alter a little. The server is reading, or accepting a message from the client. And the Client is writing, or sending a message to the server. Once both have done such without errors the connection between both have been completed.

The main portion of the code begins here, now the socket has been created we need to send the struct. I created a struct variable where the user can input the credentials of his choice. Once this was done the struct was ready to send. I used the following codes to accept and read the struct on the server and client side respectively.

Client: n = write(sockfd,(char \*)&send1, sizeof(send1));

Server: n = read(newsockfd,&send1,sizeof(send1));

Once the struct was received I did the corresponding changes: Double the int, increase float by 1, and next letter. The final values were printed and the client printed “I got your message”

To run my code I first had to build and compile both the code for the Client and the Server.

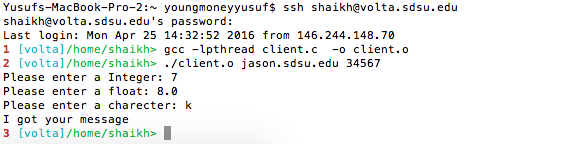
Client: gcc –lpthread client.c –o client.o

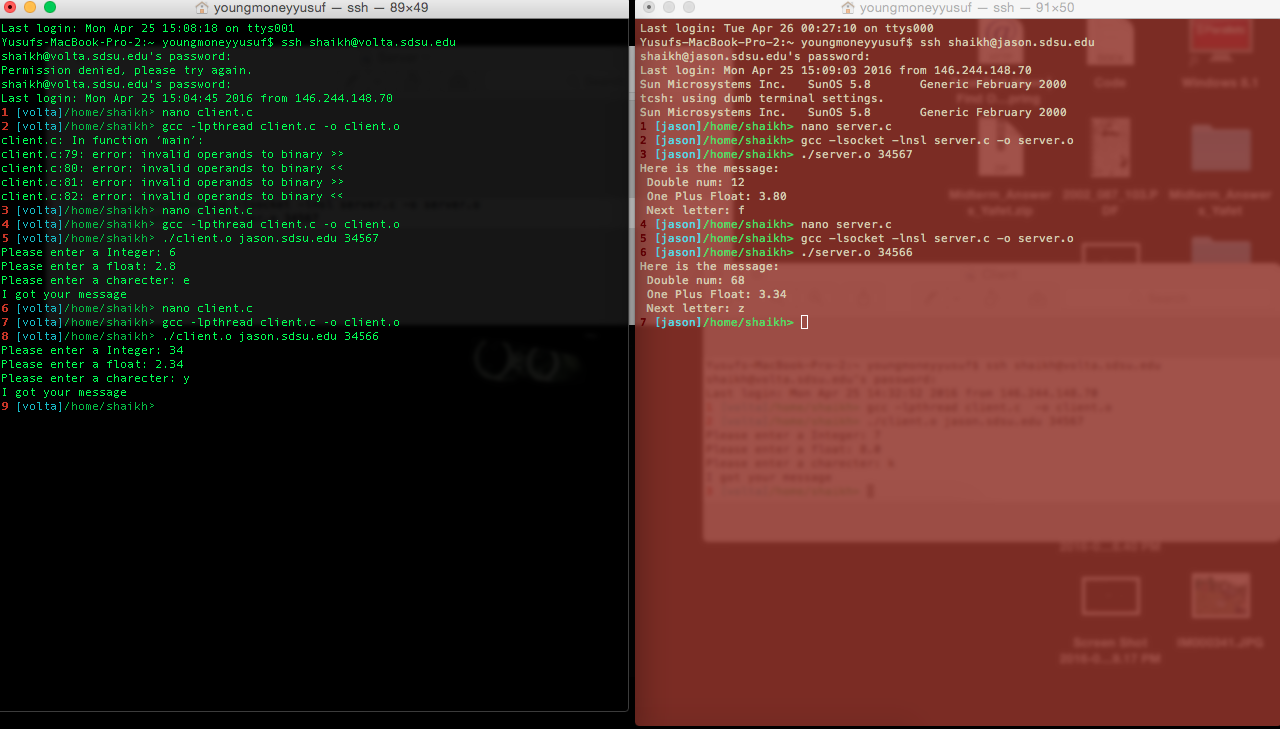
Server: gcc –lsocket –lnsl server.c –o server.o

Once both the Server and the Client codes were built and compiled I would start by giving the server a port number. I would then use this same port number when I ran the client; however, rather than simply entering the port number, I had to enter the address of the server followed the port number. After this was completed the client and server were connected through a socket connection and the program began.

Client: ./client.o jason.sdsu.edu 34567

Server: ./server.o 34567





Procedure:

1. Set up a client via *Volta Server*
2. Set up a server via *Jason Server*
3. Create the socket connection
   1. Have the server listen for a signal on a certain port #
   2. Have the client request a connection on the same port#
   3. Have the server accept the connection
   4. Have the client ack that the connection has been accepted
4. Have user input the struct credentials
5. Change the Int and Float to a little Indian value
6. Send the struct to the server
7. Have the server change the values in the struct
8. Print the Struct
9. Send back a message to the client confirming it got the message.