Ethan Nagelvoort
821234668

Project 3 (Socket Programming)

Client Code

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#define SIZE 1
int main(int count, char *ServMsg[])
{
int soccet;
soccet = socket(AF_INET,SOCK_STREAM,0);
if (soccet == -1) {
 printf("Socket has not been created yet \n");
 exit(0);
 }
if(count < 2){
exit(-1);
}
struct sockaddr_in serverAddr;
serverAddr.sin_family = AF_INET;
serverAddr.sin_port = htons(atoi(ServMsg[2]));
serverAddr.sin_addr.s_addr = inet_addr(ServMsg[1]);
int connecting_status = connect(soccet,(struct sockaddr *)
&serverAddr,sizeof(serverAddr));
 if(connecting_status == -1)
 {
 exit(-1);
 }
FILE *fp;
fp=fdopen(soccet,"w");
if (fp == NULL)
{
exit(-1);
}
char client_message[4096]="";
printf("Please enter a message - ");
```

```
int i = 0;
while ((client_message[i++] = getchar()) != '\n');
printf( "The entered message is: ");
puts( client_message );
fwrite(client_message, SIZE, sizeof(client_message), fp);
char serverMessage[4096]="";
fp=fdopen(soccet,"r+");
if (fp == NULL){
 exit(-1);
}
fread(serverMessage, SIZE, sizeof(serverMessage), fp);
printf("Message from server: %s\n",serverMessage);
strcpy(serverMessage, "");
strcpy(client_message, "");
close(soccet);
return 0;
}
```

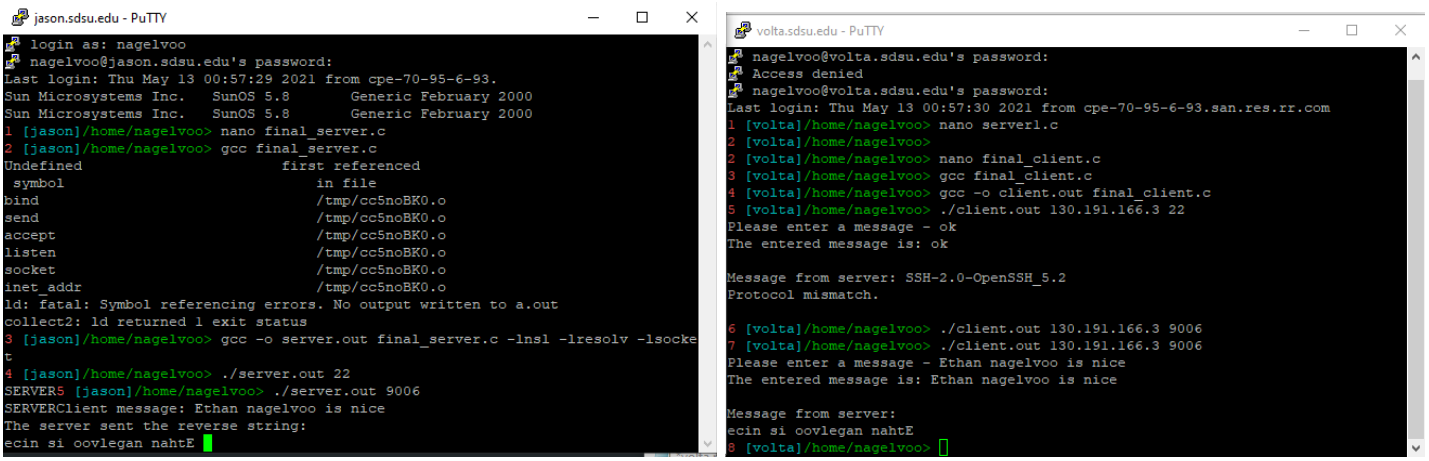
Server Code

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
#define SIZE 1
int main(int count, char *cmsg[])
{
printf("SERVER");
struct sockaddr_in serverAddr, clientAddr;
socklen_t client_length=sizeof(clientAddr);
int server_socket;
server_socket = socket(AF_INET, SOCK_STREAM, 0);
if(count < 2){
exit(-1);
}
serverAddr.sin_family = AF_INET;
serverAddr.sin_port = htons(atoi(cmsg[1]));
serverAddr.sin_addr.s_addr = inet_addr("130.191.166.3");
 if (bind(server_socket, (struct sockaddr*) &serverAddr, sizeof(serverAddr)) !=0)
 {
```

```
 exit(0);
 }
for(;;)
{
if ((listen(server_socket, 5)) != 0)
{
 exit(0);
 }
char clientMsg[4096] ="";
int clientSocket;
clientSocket = accept(server_socket,(struct sockaddr *) &clientAddr, &client_length);
if (clientSocket < 0)
{
exit(0);
}
FILE *fn;
fn=fdopen(clientSocket,"r");
if (fn == NULL){
 exit(-1);
}
fread(clientMsg, SIZE, sizeof(clientMsg), fn);
printf("Client message: %s",clientMsg);
if (strncmp("exit",clientMsg,4)==0)
{
break;
}
char reverse_string[4096]= "";
int x=0;
int i;
for (i = strlen(clientMsg)-1;i>=0;i--)
{
reverse_string[x]=clientMsg[i];
x ++;
}
fn=fdopen(clientSocket,"w");
if (fn == NULL){
exit(-1);
}
fwrite(reverse_string, SIZE, sizeof(reverse_string), fn);
send(clientSocket, reverse_string, sizeof(reverse_string), 0);
```

```
printf("The server sent the reverse string: %s ",reverse_string);
strcpy(reverse_string, "");
strcpy(clientMsg, "");
}
close(server_socket);
return 0;
}
```

Screenshot of Final Result



In the above screenshot, server is used in jason terminal and client is used in volta terminal.

Summary

_____In the client code, I first create a socket with the function socket. I use AF_INET in the function to designate the socket to communicate with IPv4 addresses. I also use SOCK_STREAM in the function to have a reliable, two way, connection-based service. I then check if it equals -1, because if it did then the socket would be a failure. I then check if count, one of the clients parameters, is less than 2. If it is, then a failure would occur since that would mean that there are less than 2 parameters when client needs 2. I then create a struct of type sockaddr_in. I have this struck's sin_family is equal to AF_INET. I have it's sin_port equal htons(atoi(ServMsg[2])). ServMsg is the second parameter. atoi() converts the string into an integer value and htons converts this integer value into a network byte order, which would be big endian. I then have it's s_addr parameter equal inet_addr(ServMsg[1]). inet_addr() is used to convert ServMsg[1] to an integer value to be used as an internet address. I then used the function connect() to have the client function make a connection with the server. I check if this is -1 because then the connection was not successful. I then create a FILE pointer and have it equal to the output of the socket being opened with the function fdopen(). I then check if the file is NULL, which means there was an issue with opening the socket. I then create a char array called client_message and prompt the user to enter a message. I then use a while loop and getchar() to

store that message into client_message. I then use fwrite() to place client_message into the file. I then create another char array called serverMessage and use fdopen() to update the socket. This update would put the file to go through the server process where the client_message would be reversed. fread() is then used to read the new string in fp into serverMessage. I then print this string. I then use the function, strcpy on both serverMessage and client_message to have them revert to equaling nothing. I then finally close the socket.

In the server code, I create two socket structs, one for server and one for client. I also check if there are two parameters by checking my count parameter. Then I have an integer equal socket(AF_INET, SOCK_STREAM, 0), like before in my client code. I then fill out the server sockets parameters and this time I set inet_addr to 130.191.166.3. This is the ip address for jason.sdsu.edu. I then use the bind() function to assign the server socket's address to the integer value previously made. Sockets made with socket() are unnamed and so the bind() gives them that name. If this function is not 0, then an error has occurred. I then go into a for loop that lasts forever and use the listen() to mark the server socket to wait for a client to make a connection with it. I set its queue length to five, meaning that this server cannot have more than five outstanding connections. I then create a char array for the client message and a client socket. I set this client socket to an incoming client connection. I do this through the accept(), this function is used to wait for a connection to be made with the client. Once it is made, then it will return the socket from that connection. Then if the socket is less than 0, an error has occurred with the accept() function. I then have a file pointer equal to that client socket by using fdopen to have the client socket ready for reading. I check if the file is null and then use fread() to have the char array for client messages equal to the message found in the file. The file contains the socket that was gained through the accept(), so it will have the message from the client. I then print out the message and check to see if the message was "exit". If it is "exit", then I end the program. I then declare a char array for a reversed message. I use a for loop to decrement through the clients message and set every character the loop goes through to the character in the reversed message. The for loop in incrementing through the reversed message char array and so it will contain a reversed version of the clients message. I then use fdopen to enable the file for writing and write this reversed message char array into the file. I then use the send() function to transmit the reversed message to the client. I then print the reversed string and use strcpy to set the reversed string char array and the client message char array to nothing.

Questions

What are sockets and on which layer do they operate?

A socket is a node within a network that acts as an endpoint for sending and receiving data. They operate on the transport layers.

Differentiate between TCP and UDP? This assignment is based on TCP or UDP? YouTube uses TCP or UDP?

UDP is a connectionless protocol, its data delivery is not guaranteed, has basic error checking, has data out of order, and is faster than TCP. TCP is a connection based protocol, reliable for data delivery, has thorough error checking, orders data, and is slower than UDP. This assignment is based on TCP. Youtube uses both.

What will happen if I use an out of range port number in my code? Will I encounter an error? If Yes then why and if No then why?

If the port number is out of range, then a connection will not be made. This will result in an error since data will be unable to be sent.

What is the maximum number of sockets that a client and a server can have?

The maximum number of sockets that a client and server can have is 65535.

Video

https://drive.google.com/file/d/1xR3gEYYpDOfN-l-2hHx0XwfpM9vG80Dk/view?usp=sharing