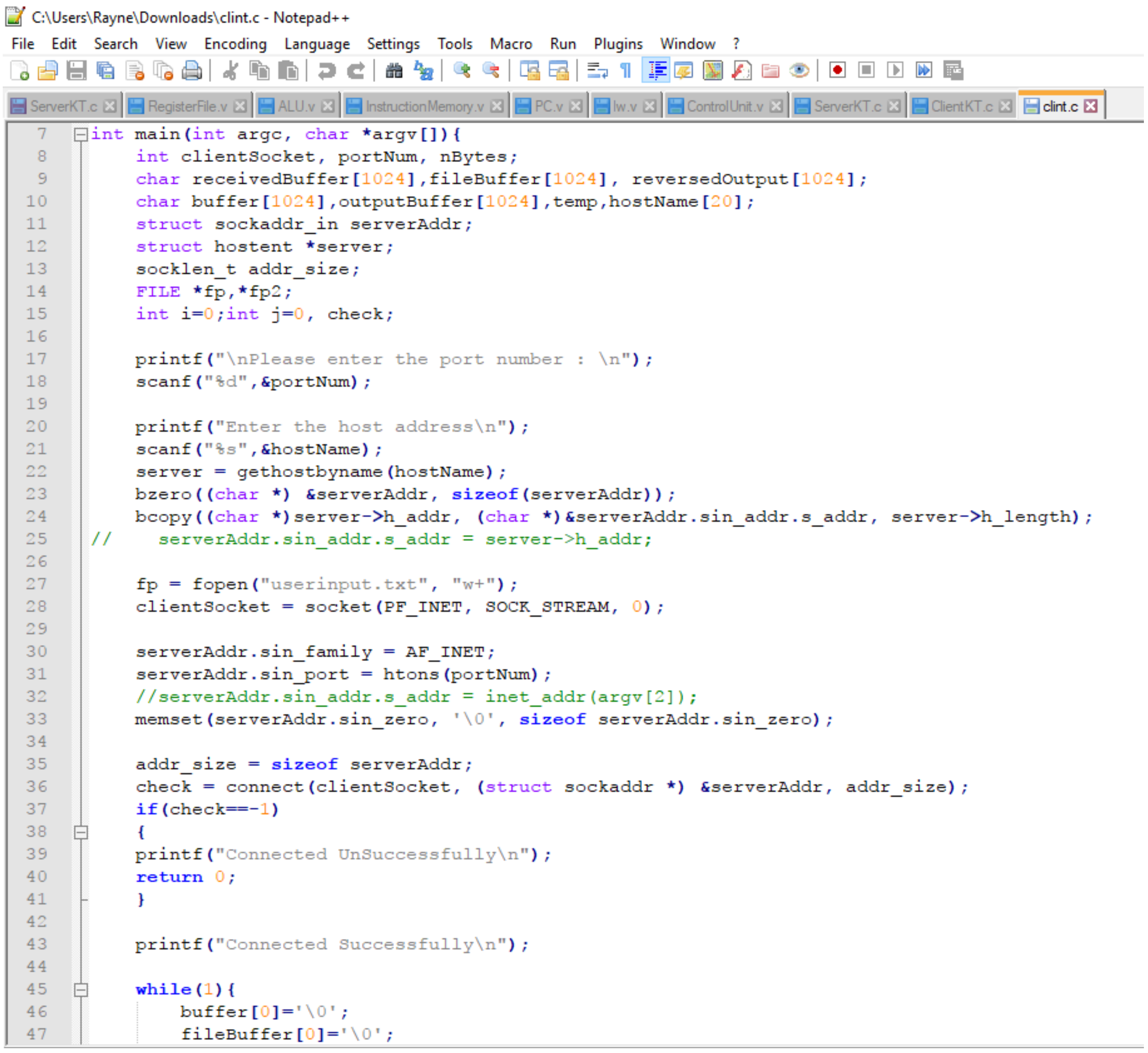
Socket Programming

11.30.2017

Rain Gopeng
818403684
Rgopeng@gmail.com

Tasneem Singh
822046102
tasneemsingh95@gmail.com

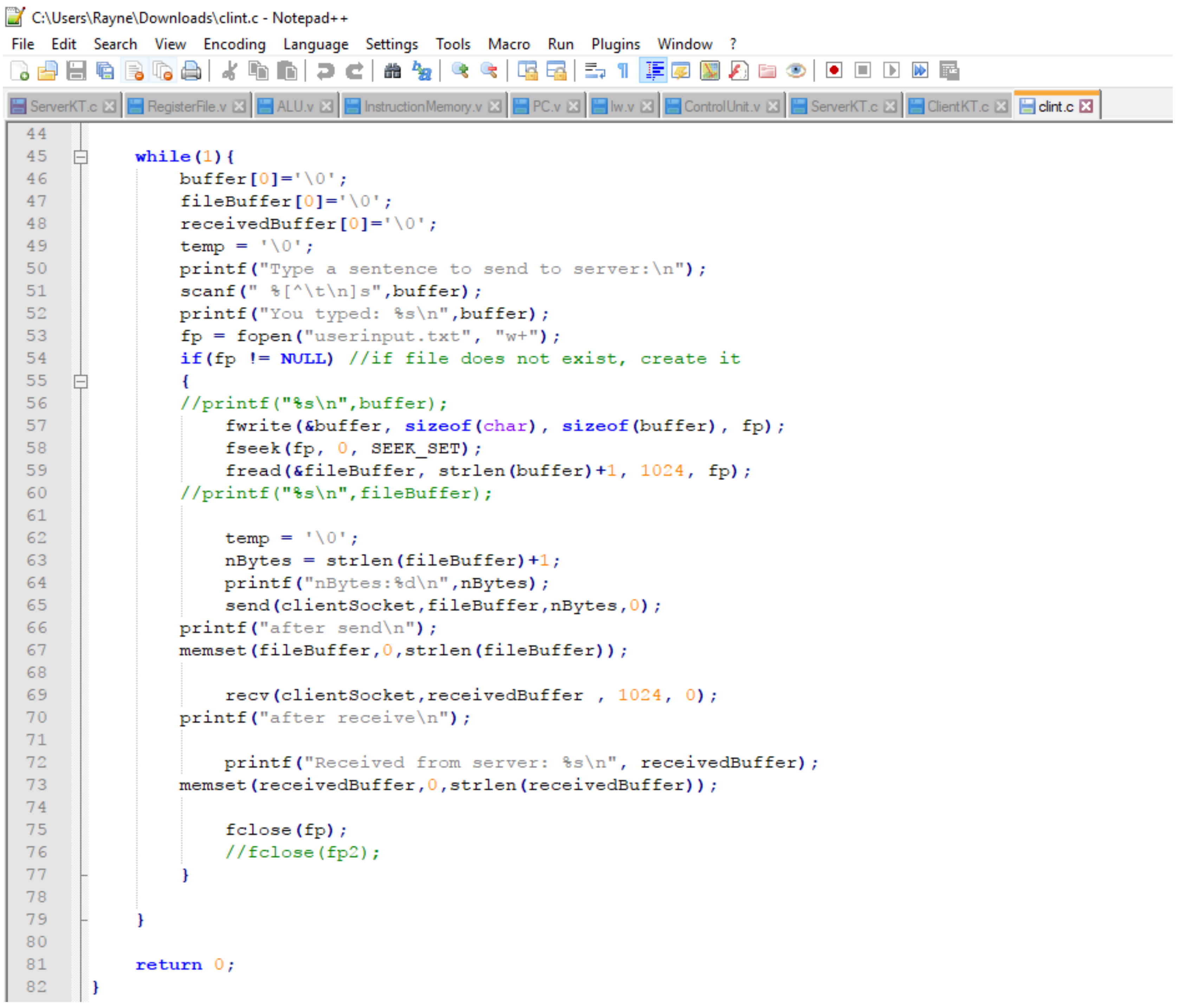
Kinjal Gala
822055813
kinjalgala215@gmail.com



C:\Users\Rayne\Downloads\clint.c - Notepad++

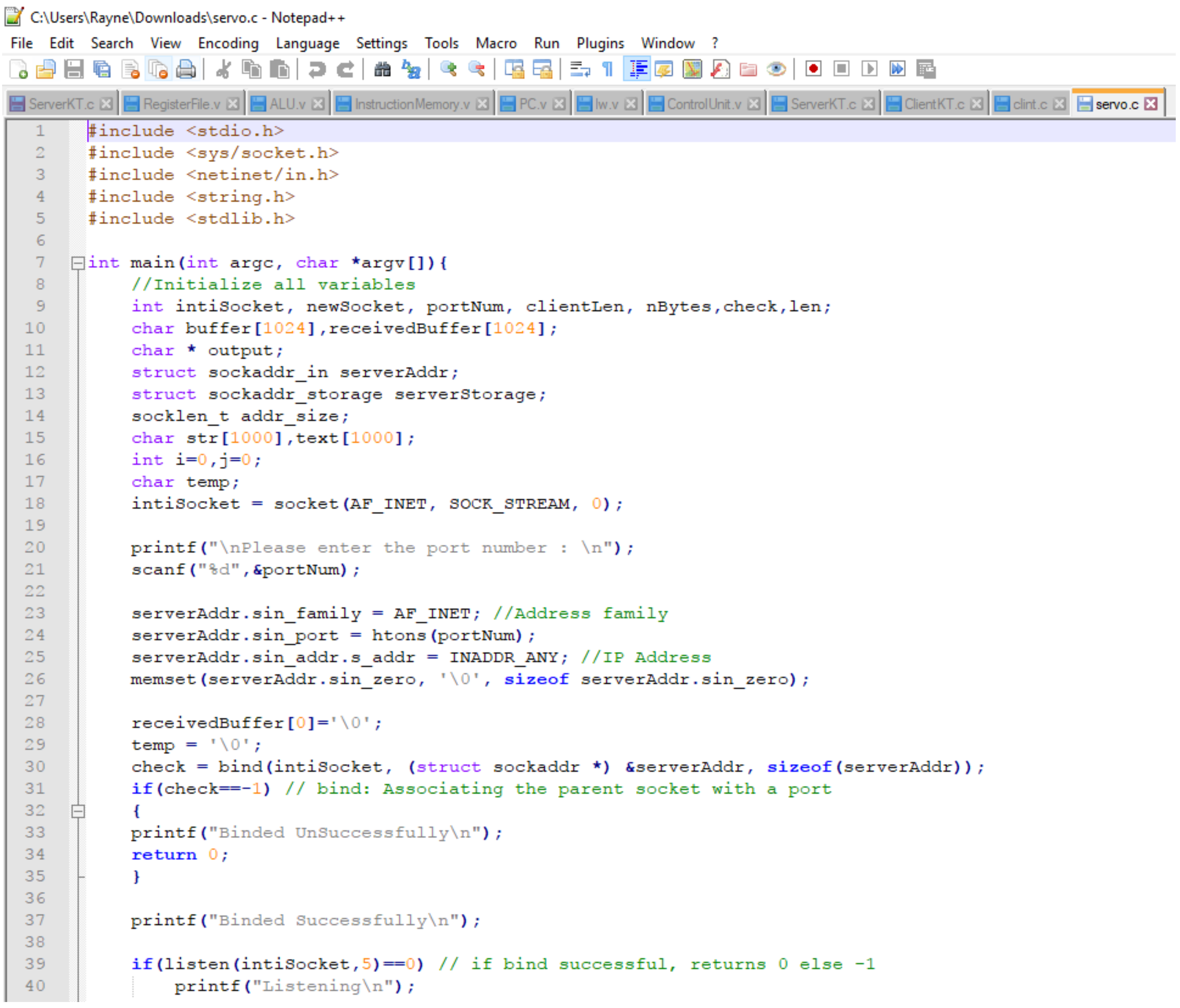
```c
int main(int argc, char *argv[]){
    int clientSocket, portNum, nBytes;
    char receivedBuffer[1024],fileBuffer[1024], reversedOutput[1024];
    char buffer[1024],outputBuffer[1024],temp,hostName[20];
    struct sockaddr_in serverAddr;
    struct hostent *server;
    socklen_t addr_size;
    FILE *fp,*fp2;
    int i=0;int j=0, check;

    printf("\nPlease enter the port number : \n");
    scanf("%d",&portNum);

    printf("Enter the host address\n");
    scanf("%s",&hostName);
    server = gethostbyname(hostName);
    bzero((char *) &serverAddr, sizeof(serverAddr));
    bcopy((char *)server->h_addr, (char *)&serverAddr.sin_addr.s_addr, server->h_length);
//    serverAddr.sin_addr.s_addr = server->h_addr;

    fp = fopen("userinput.txt", "w+");
    clientSocket = socket(PF_INET, SOCK_STREAM, 0);

    serverAddr.sin_family = AF_INET;
    serverAddr.sin_port = htons(portNum);
    //serverAddr.sin_addr.s_addr = inet_addr(argv[2]);
    memset(serverAddr.sin_zero, '\0', sizeof serverAddr.sin_zero);

    addr_size = sizeof serverAddr;
    check = connect(clientSocket, (struct sockaddr *) &serverAddr, addr_size);
    if(check==-1)
    {
    printf("Connected UnSuccessfully\n");
    return 0;
    }

    printf("Connected Successfully\n");

    while(1){
        buffer[0]='\0';
        fileBuffer[0]='\0';
```

Fig. 1 Client Code


C:\Users\Rayne\Downloads\clint.c - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
ServerKT.c
RegisterFile.v
ALU.v
InstructionMemory.v
PC.v
lw.v
ControlUnit.v
ServerKT.c
ClientKT.c
clint.c

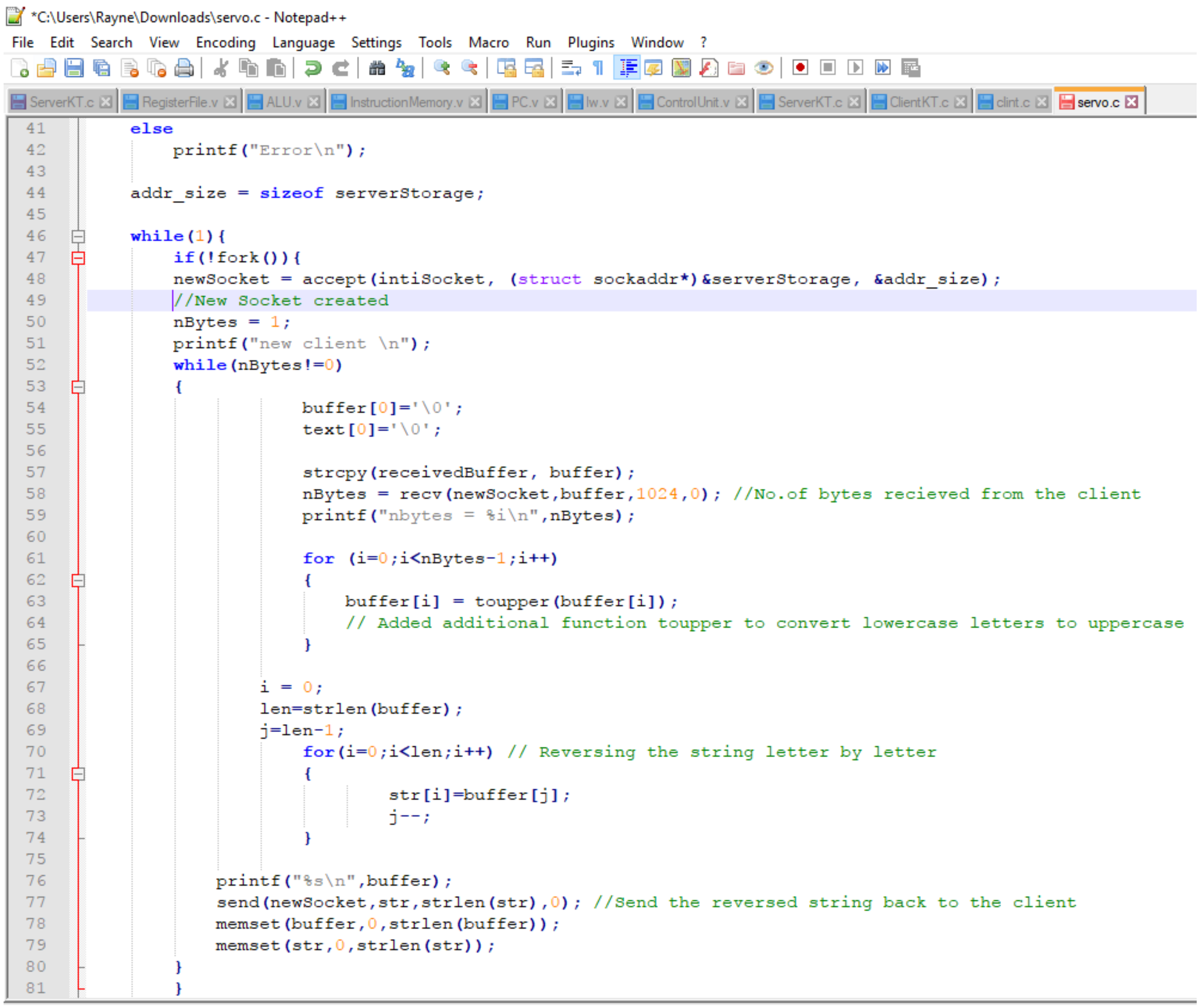
```c

    while(1){
        buffer[0]='\0';
        fileBuffer[0]='\0';
        receivedBuffer[0]='\0';
        temp = '\0';
        printf("Type a sentence to send to server:\n");
        scanf(" %[^\t\n]s",buffer);
        printf("You typed: %s\n",buffer);
        fp = fopen("userinput.txt", "w+");
        if(fp != NULL) //if file does not exist, create it
        {
        //printf("%s\n",buffer);
            fwrite(&buffer, sizeof(char), sizeof(buffer), fp);
            fseek(fp, 0, SEEK_SET);
            fread(&fileBuffer, strlen(buffer)+1, 1024, fp);
        //printf("%s\n",fileBuffer);

            temp = '\0';
            nBytes = strlen(fileBuffer)+1;
            printf("nBytes:%d\n",nBytes);
            send(clientSocket,fileBuffer,nBytes,0);
        printf("after send\n");
        memset(fileBuffer,0,strlen(fileBuffer));

            recv(clientSocket,receivedBuffer , 1024, 0);
        printf("after receive\n");

            printf("Received from server: %s\n", receivedBuffer);
        memset(receivedBuffer,0,strlen(receivedBuffer));

            fclose(fp);
            //fclose(fp2);
        }

    }

    return 0;
}
```

Fig. 2 Client Code -cont.


C:\Users\Rayne\Downloads\servo.c - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
ServerKT.c
RegisterFile.v
ALU.v
InstructionMemory.v
PC.v
lw.v
ControlUnit.v
ServerKT.c
ClientKT.c
clint.c
servo.c

```c
#include <stdio.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <string.h>
#include <stdlib.h>

int main(int argc, char *argv[]){
    //Initialize all variables
    int intiSocket, newSocket, portNum, clientLen, nBytes,check,len;
    char buffer[1024],receivedBuffer[1024];
    char * output;
    struct sockaddr_in serverAddr;
    struct sockaddr_storage serverStorage;
    socklen_t addr_size;
    char str[1000],text[1000];
    int i=0,j=0;
    char temp;
    intiSocket = socket(AF_INET, SOCK_STREAM, 0);

    printf("\nPlease enter the port number : \n");
    scanf("%d",&portNum);

    serverAddr.sin_family = AF_INET; //Address family
    serverAddr.sin_port = htons(portNum);
    serverAddr.sin_addr.s_addr = INADDR_ANY; //IP Address
    memset(serverAddr.sin_zero, '\0', sizeof serverAddr.sin_zero);

    receivedBuffer[0]='\0';
    temp = '\0';
    check = bind(intiSocket, (struct sockaddr *) &serverAddr, sizeof(serverAddr));
    if(check==-1) // bind: Associating the parent socket with a port
    {
    printf("Binded UnSuccessfully\n");
    return 0;
    }

    printf("Binded Successfully\n");

    if(listen(intiSocket,5)==0) // if bind successful, returns 0 else -1
        printf("Listening\n");
```

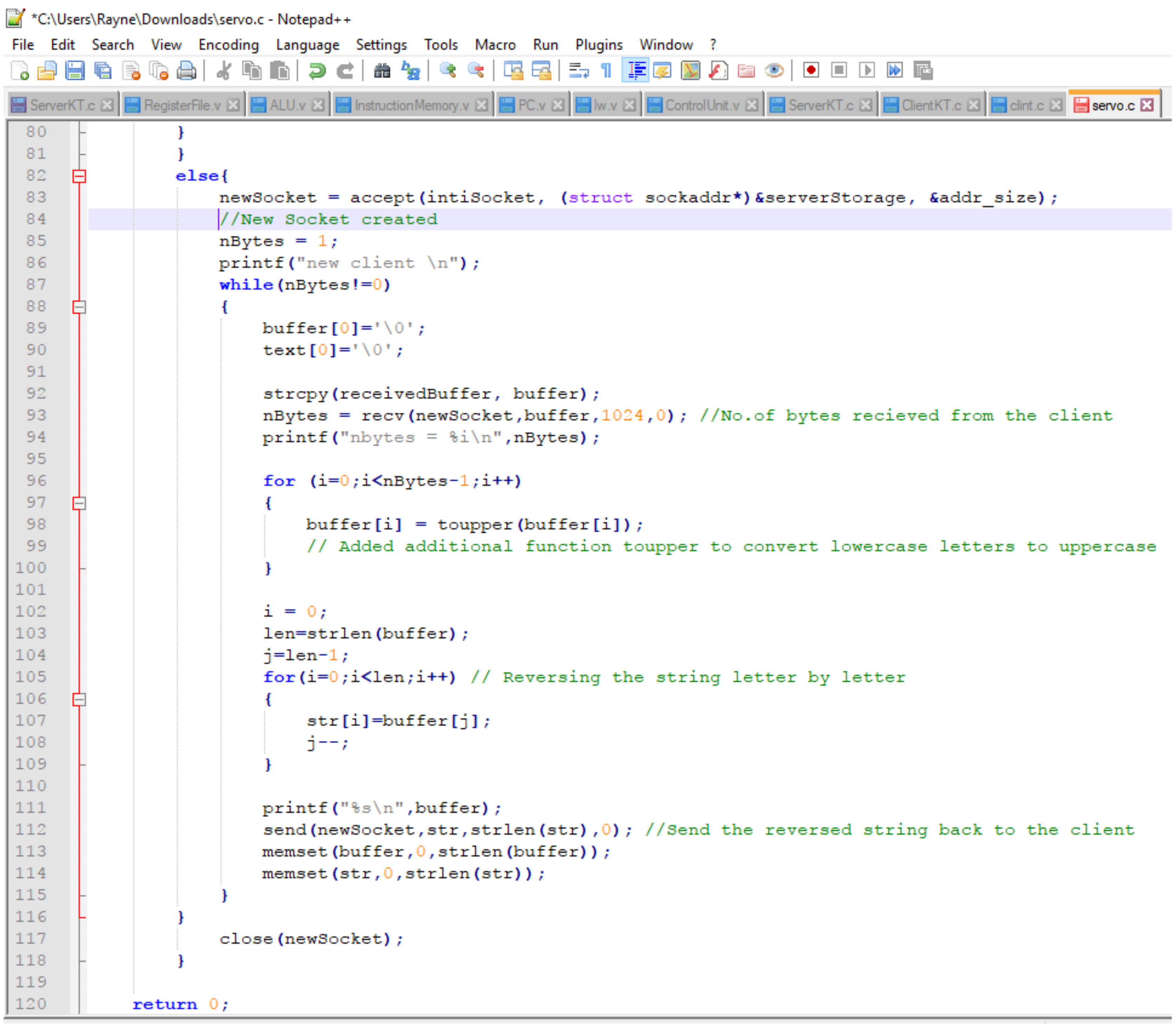
Fig. 3 Server Code



*C:\Users\Rayne\Downloads\servo.c - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
ServerKT.c RegisterFile.v ALU.v InstructionMemory.v PC.v lw.v ControlUnit.v ServerKT.c ClientKT.c clint.c servo.c

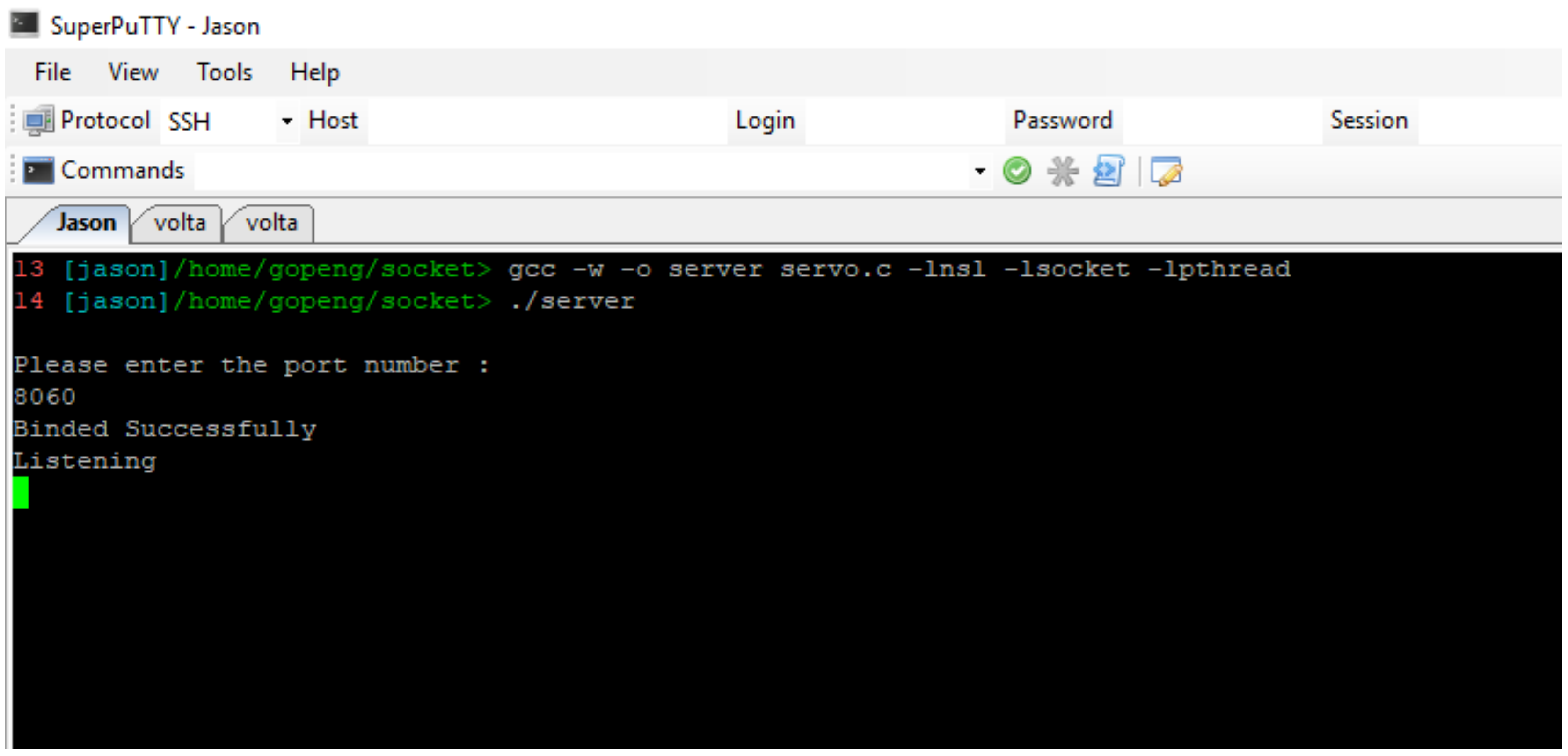
```c
    else
        printf("Error\n");

    addr_size = sizeof serverStorage;

    while(1){
        if(!fork()){
        newSocket = accept(intiSocket, (struct sockaddr*)&serverStorage, &addr_size);
        //New Socket created
        nBytes = 1;
        printf("new client \n");
        while(nBytes!=0)
        {
                    buffer[0]='\0';
                    text[0]='\0';

                    strcpy(receivedBuffer, buffer);
                    nBytes = recv(newSocket,buffer,1024,0); //No.of bytes recieved from the client
                    printf("nbytes = %i\n",nBytes);

                    for (i=0;i<nBytes-1;i++)
                    {
                        buffer[i] = toupper(buffer[i]);
                        // Added additional function toupper to convert lowercase letters to uppercase
                    }

                i = 0;
                len=strlen(buffer);
                j=len-1;
                    for(i=0;i<len;i++) // Reversing the string letter by letter
                    {
                            str[i]=buffer[j];
                            j--;
                    }

            printf("%s\n",buffer);
            send(newSocket,str,strlen(str),0); //Send the reversed string back to the client
            memset(buffer,0,strlen(buffer));
            memset(str,0,strlen(str));
        }
        }
```

Fig. 4 Server Code -cont.


*C:\Users\Rayne\Downloads\servo.c - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
ServerKT.c RegisterFile.v ALU.v InstructionMemory.v PC.v lw.v ControlUnit.v ServerKT.c ClientKT.c clint.c servo.c

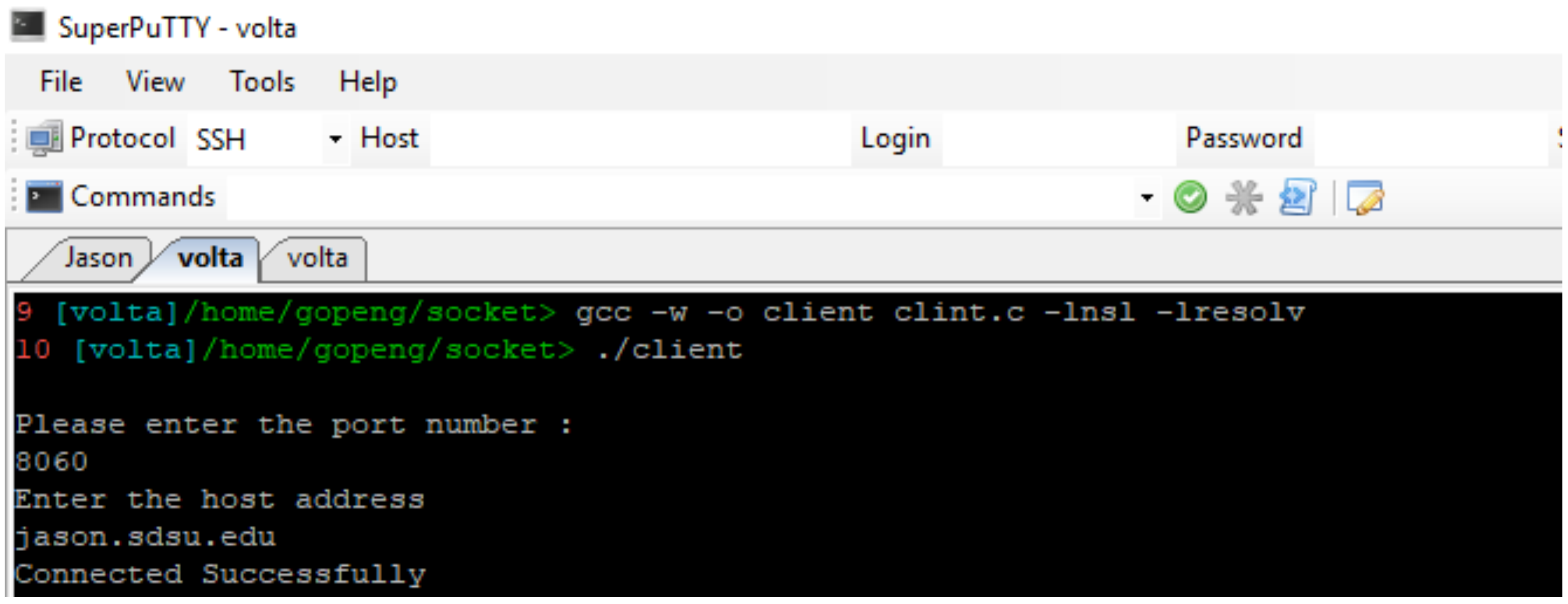
```c
        }
        }
        else{
            newSocket = accept(intiSocket, (struct sockaddr*)&serverStorage, &addr_size);
            //New Socket created
            nBytes = 1;
            printf("new client \n");
            while(nBytes!=0)
            {
                buffer[0]='\0';
                text[0]='\0';

                strcpy(receivedBuffer, buffer);
                nBytes = recv(newSocket,buffer,1024,0); //No.of bytes recieved from the client
                printf("nbytes = %i\n",nBytes);

                for (i=0;i<nBytes-1;i++)
                {
                    buffer[i] = toupper(buffer[i]);
                    // Added additional function toupper to convert lowercase letters to uppercase
                }

                i = 0;
                len=strlen(buffer);
                j=len-1;
                for(i=0;i<len;i++) // Reversing the string letter by letter
                {
                    str[i]=buffer[j];
                    j--;
                }

                printf("%s\n",buffer);
                send(newSocket,str,strlen(str),0); //Send the reversed string back to the client
                memset(buffer,0,strlen(buffer));
                memset(str,0,strlen(str));
            }
        }
            close(newSocket);
        }

    return 0;
```

Fig. 5 Server Code -cont.



```
13 [jason]/home/gopeng/socket> gcc -w -o server servo.c -lnsl -lsocket -lpthread
14 [jason]/home/gopeng/socket> ./server

Please enter the port number :
8060
Binded Successfully
Listening
```

Fig.6 Server setup code

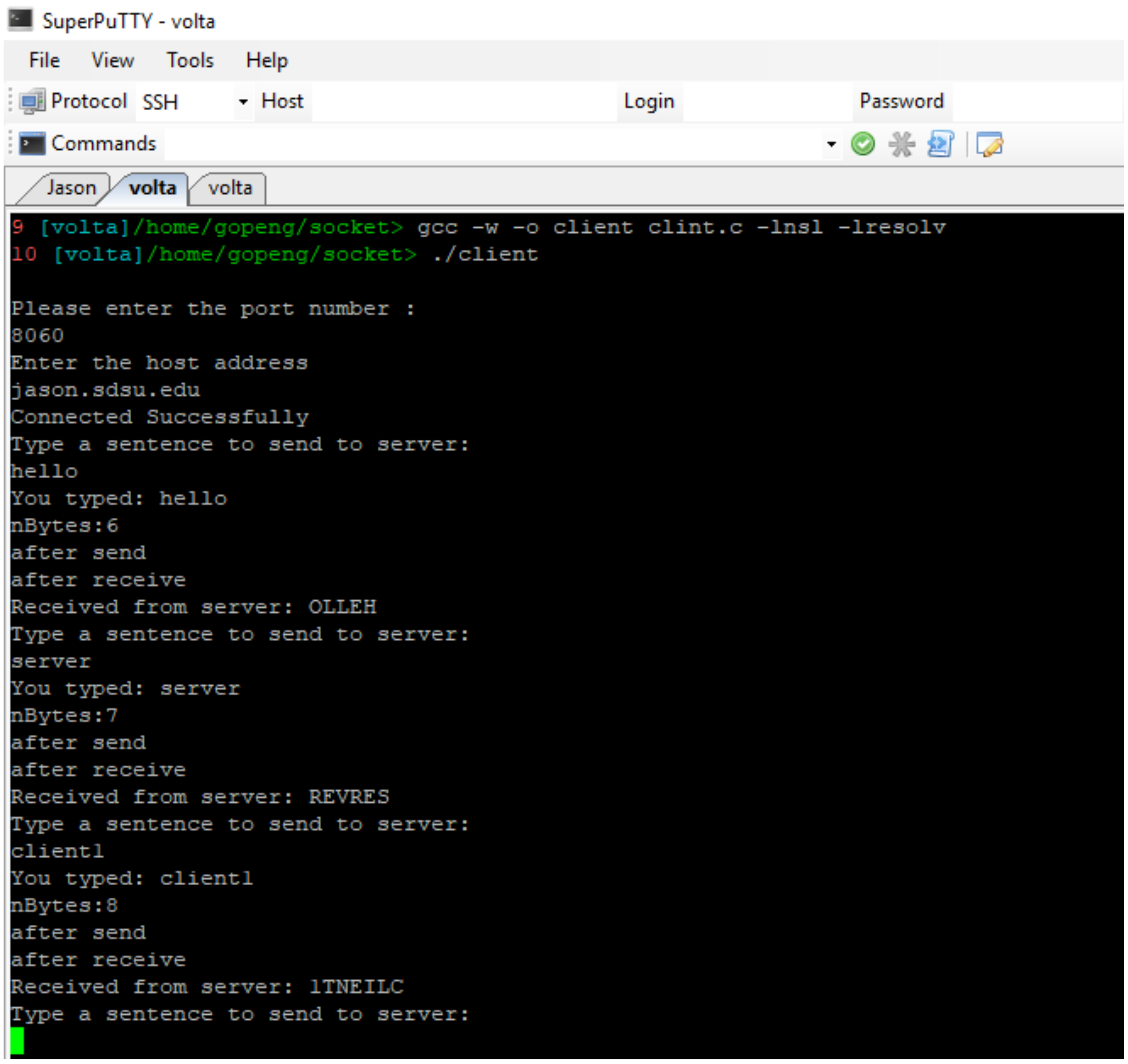


```
9 [volta]/home/gopeng/socket> gcc -w -o client clint.c -lnsl -lresolv
10 [volta]/home/gopeng/socket> ./client

Please enter the port number :
8060
Enter the host address
jason.sdsu.edu
Connected Successfully
```

Fig.7 Client setup code

The above commands are used by the Server and Client to setup a connection before transferring the data. The port number and the IP addresses are given along with the commands to establish a successful connection. Here, Jason is used as a Server and Volta is used to setup the client. 130.191.166.3 is the IP address of jason.sdsu.edu.

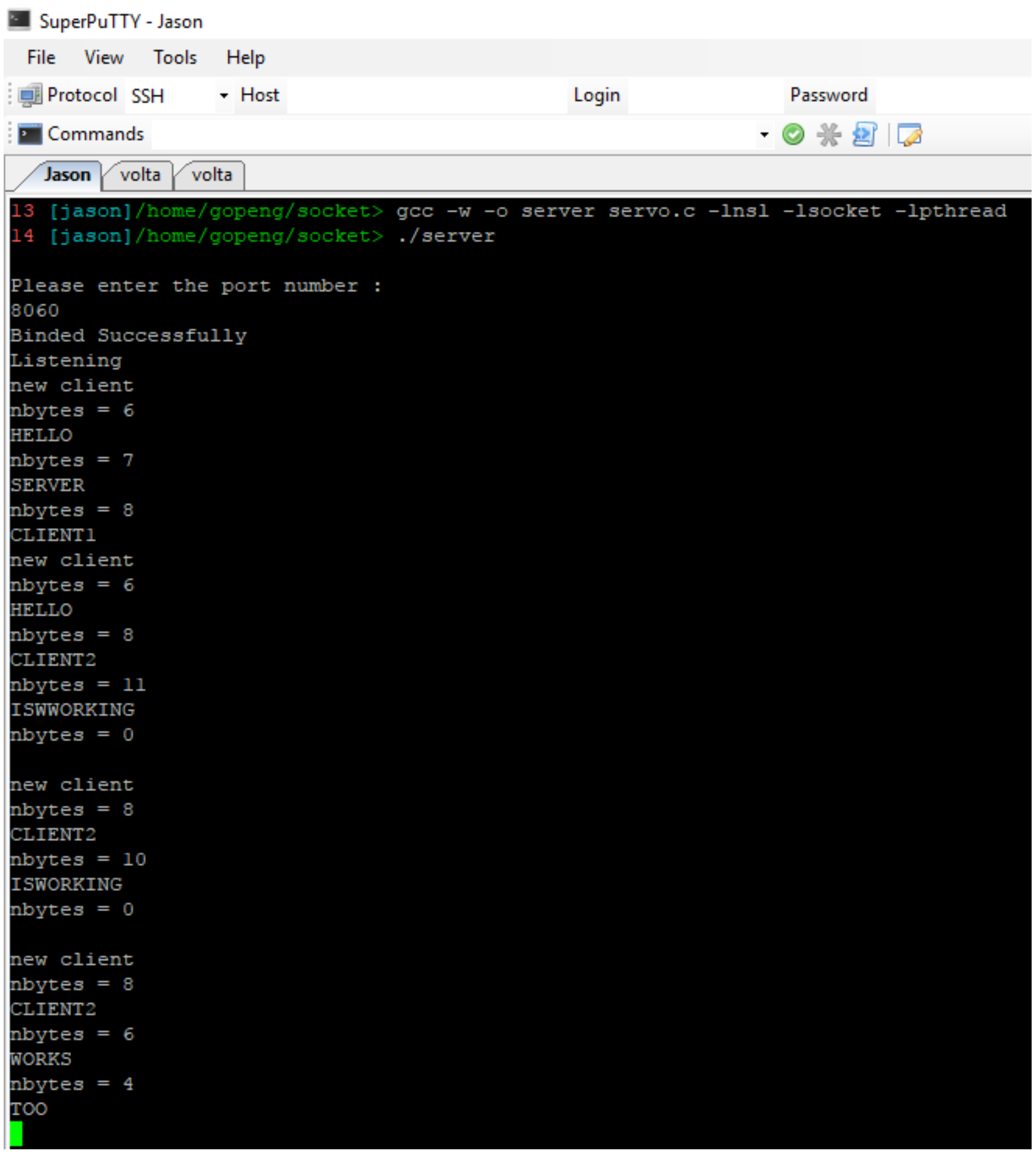
NOTE: We've made changes in the code as suggested in the demo, to take the port no. as an input from the user. It was hard-coded before.



```
9 [volta]/home/gopeng/socket> gcc -w -o client clint.c -lnsl -lresolv
10 [volta]/home/gopeng/socket> ./client

Please enter the port number :
8060
Enter the host address
jason.sdsu.edu
Connected Successfully
Type a sentence to send to server:
hello
You typed: hello
nBytes:6
after send
after receive
Received from server: OLLEH
Type a sentence to send to server:
server
You typed: server
nBytes:7
after send
after receive
Received from server: REVRES
Type a sentence to send to server:
client1
You typed: client1
nBytes:8
after send
after receive
Received from server: 1TNEILC
Type a sentence to send to server:
```

Fig.8 Client 1 after a successful connection

Above is the screenshot for the output we receive when we type in a message on the client side. nBytes here represent the number of bytes that are transferred and ‘Received from server:’ displays the reversed text as expected. Additionally, we have also added another functionality to our code; the toupper() function, which changes the lowercase letters to uppercase (as can been seen in the screenshot).

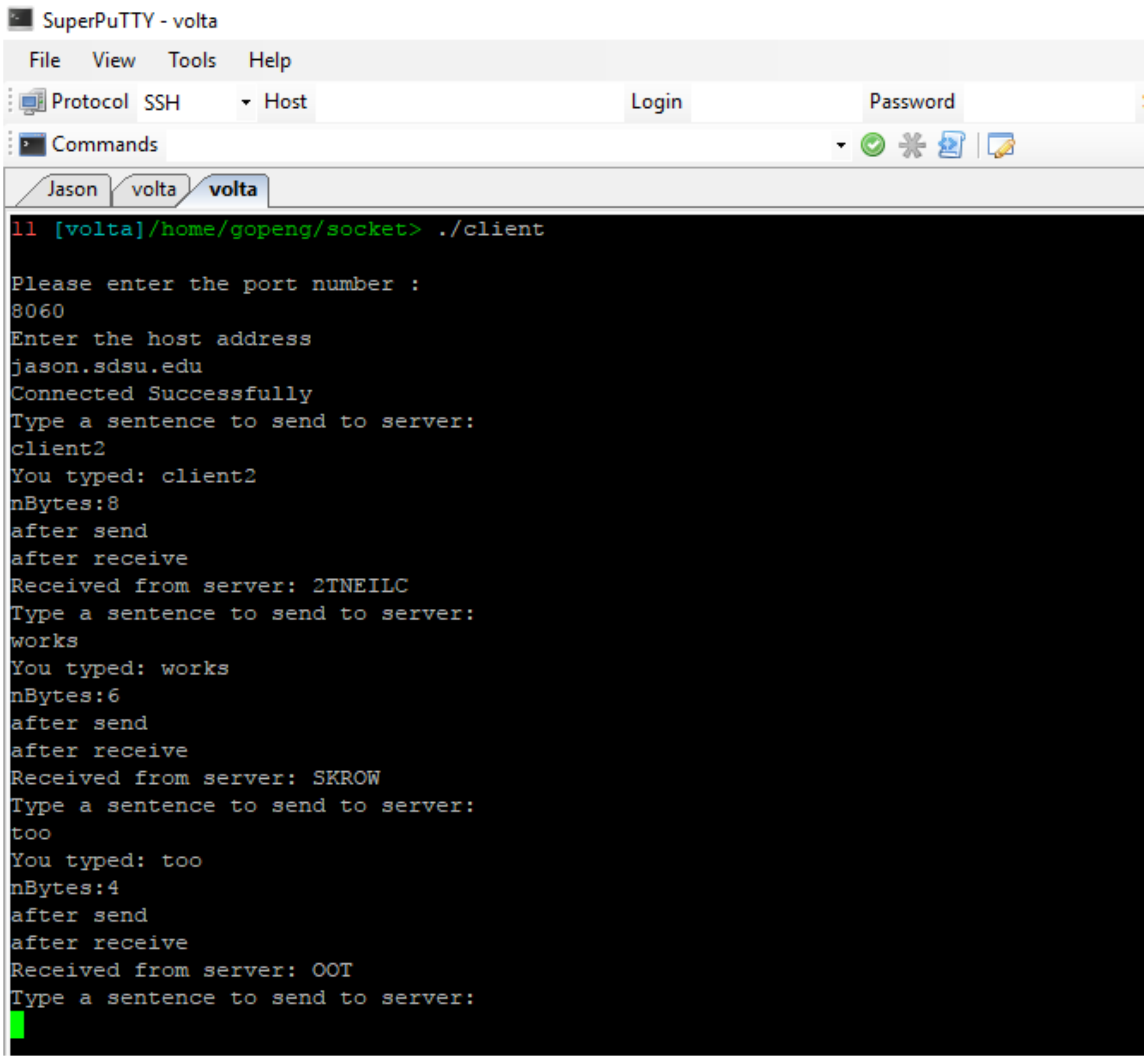


```
13 [jason]/home/gopeng/socket> gcc -w -o server servo.c -lnsl -lsocket -lpthread
14 [jason]/home/gopeng/socket> ./server

Please enter the port number :
8060
Binded Successfully
Listening
new client
nbytes = 6
HELLO
nbytes = 7
SERVER
nbytes = 8
CLIENT1
new client
nbytes = 6
HELLO
nbytes = 8
CLIENT2
nbytes = 11
ISWWORKING
nbytes = 0

new client
nbytes = 8
CLIENT2
nbytes = 10
ISWORKING
nbytes = 0

new client
nbytes = 8
CLIENT2
nbytes = 6
WORKS
nbytes = 4
TOO
```

Fig. 9 Server window after a successful connection

Fig.9 showcases the server window after client 1 has successfully connected to the server. The bind check 0 coveys that the connection was successful else the server would have displayed a value of -1 which indicates an unsuccessful connection. Similar to the Client window, we have nbytes to show the number of bytes transferred. The server receives the ‘capitalized’ inputs which its reverses word by word and sends back to the client.

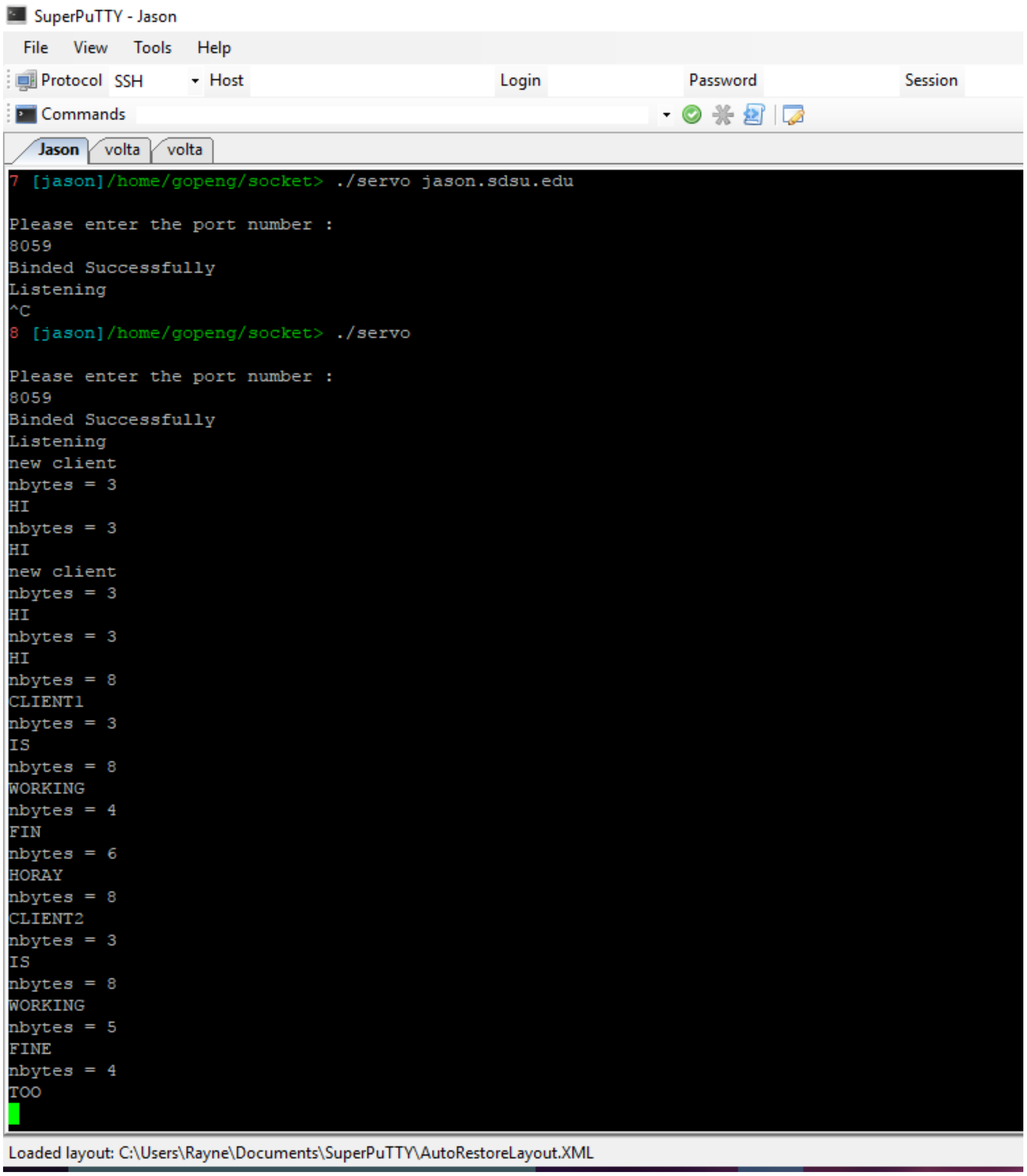
NOTE: The ‘new client’ here represents a new incoming connection. Everything before that is the communication that takes place between the first client connected.



```
11 [volta]/home/gopeng/socket> ./client

Please enter the port number :
8060
Enter the host address
jason.sdsu.edu
Connected Successfully
Type a sentence to send to server:
client2
You typed: client2
nBytes:8
after send
after receive
Received from server: 2TNEILC
Type a sentence to send to server:
works
You typed: works
nBytes:6
after send
after receive
Received from server: SKROW
Type a sentence to send to server:
too
You typed: too
nBytes:4
after send
after receive
Received from server: OOT
Type a sentence to send to server:
```

Fig. 10 Client 2 after a successful connection

Fig.10 demonstrates a new client connecting to the server. These clients work simultaneously ie. without the need to close one before connecting another.



```
7 [jason]/home/gopeng/socket> ./servo jason.sdsu.edu

Please enter the port number :
8059
Binded Successfully
Listening
^C
8 [jason]/home/gopeng/socket> ./servo

Please enter the port number :
8059
Binded Successfully
Listening
new client
nbytes = 3
HI
nbytes = 3
HI
new client
nbytes = 3
HI
nbytes = 3
HI
nbytes = 8
CLIENT1
nbytes = 3
IS
nbytes = 8
WORKING
nbytes = 4
FIN
nbytes = 6
HORAY
nbytes = 8
CLIENT2
nbytes = 3
IS
nbytes = 8
WORKING
nbytes = 5
FINE
nbytes = 4
TOO
```

Loaded layout: C:\Users\Rayne\Documents\SuperPuTTY\AutoRestoreLayout.XML

Fig. 11 Server and 2 clients running simultaneously.

Summary

Socket Programming involves developing a connection between the nodes to communicate with each other in a given network. Here, the socket which listens is a server and there are multiple nodes(Clients) which sends the data to the server.. The Server waits and listens for the data, reads it and then reverses it before sending it back to the respective clients. Also, here the user determines the port number to establish a connection by providing the same in the command prompt. In this project. we have implemented the concept of forking , which involves creation of a child process to a given parent process. This helps us to connect multiple clients to a single server for successful communication.

The snapshots in the report contain the images of the commands used during compilation, outputs obtained and the C codes for both client and server. It can be observed that when a client sends a statement or word to the server it receives the reversed string from the server side, where the string is reversed word by word and letter by letter. Adding multiple clients does not result in closing one connection and adding another. All the clients can work at the same time. Although, we can exit one client and connect other in between the process. We also made use of the fread and fwrite functions to read and write data to and from a socket and files. Also, we have added the gethostbyname() function to get the IP address using the host name. Thus, we were able to demonstrate a successfully running client(s) - server architecture.