



# Socket Programming

11.30.2017

Rain Gopeng

818403684

[Rgopeng@gmail.com](mailto:Rgopeng@gmail.com)

Tasneem Singh

822046102

[tasneemsingh95@gmail.com](mailto:tasneemsingh95@gmail.com)

Kinjal Gala

822055813

[kinjalgala215@gmail.com](mailto:kinjalgala215@gmail.com)

```
C:\Users\Rayne\Downloads\clint.c - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

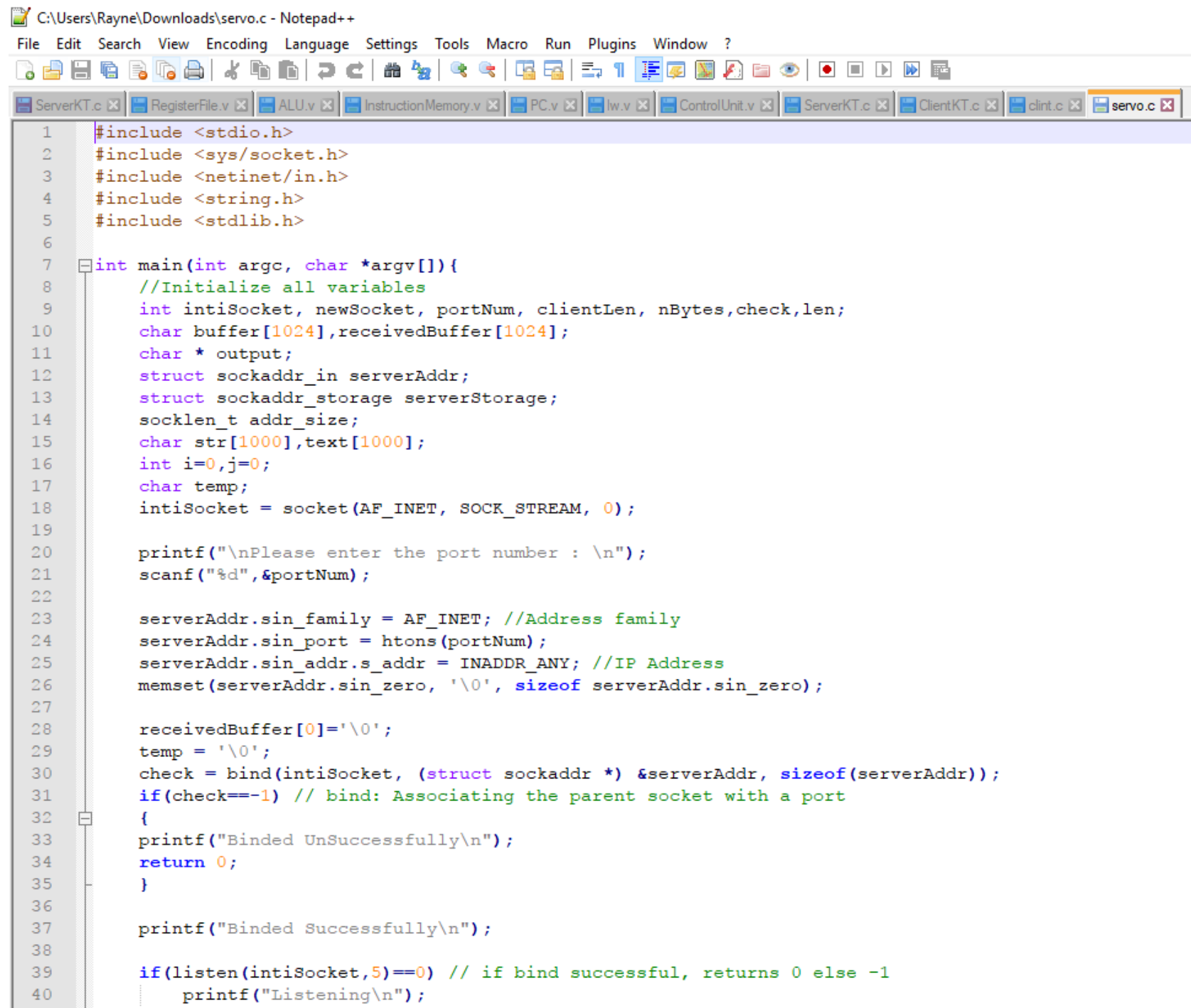
ServerKT.c x RegisterFile.v x ALU.v x InstructionMemory.v x PC.v x lw.v x ControlUnit.v x ServerKT.c x ClientKT.c x clint.c x

7 int main(int argc, char *argv[]){
8     int clientSocket, portNum, nBytes;
9     char receivedBuffer[1024],fileBuffer[1024], reversedOutput[1024];
10    char buffer[1024],outputBuffer[1024],temp,hostName[20];
11    struct sockaddr_in serverAddr;
12    struct hostent *server;
13    socklen_t addr_size;
14    FILE *fp,*fp2;
15    int i=0;int j=0, check;
16
17    printf("\nPlease enter the port number : \n");
18    scanf("%d",&portNum);
19
20    printf("Enter the host address\n");
21    scanf("%s",&hostName);
22    server = gethostbyname(hostName);
23    bzero((char *) &serverAddr, sizeof(serverAddr));
24    bcopy((char *)server->h_addr, (char *)&serverAddr.sin_addr.s_addr, server->h_length);
25    // serverAddr.sin_addr.s_addr = server->h_addr;
26
27    fp = fopen("userinput.txt", "w+");
28    clientSocket = socket(PF_INET, SOCK_STREAM, 0);
29
30    serverAddr.sin_family = AF_INET;
31    serverAddr.sin_port = htons(portNum);
32    //serverAddr.sin_addr.s_addr = inet_addr(argv[2]);
33    memset(serverAddr.sin_zero, '\0', sizeof serverAddr.sin_zero);
34
35    addr_size = sizeof serverAddr;
36    check = connect(clientSocket, (struct sockaddr *) &serverAddr, addr_size);
37    if(check==-1)
38    {
39        printf("Connected UnSuccessfully\n");
40        return 0;
41    }
42
43    printf("Connected Successfully\n");
44
45    while(1){
46        buffer[0]='\0';
47        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
44
45 while(1){
46     buffer[0]='\0';
47     fileBuffer[0]='\0';
48     receivedBuffer[0]='\0';
49     temp = '\0';
50     printf("Type a sentence to send to server:\n");
51     scanf(" %[^\\t\\n]s",buffer);
52     printf("You typed: %s\\n",buffer);
53     fp = fopen("userinput.txt", "w+");
54     if(fp != NULL) //if file does not exist, create it
55     {
56         //printf("%s\\n",buffer);
57         fwrite(&buffer, sizeof(char), sizeof(buffer), fp);
58         fseek(fp, 0, SEEK_SET);
59         fread(&fileBuffer, strlen(buffer)+1, 1024, fp);
60         //printf("%s\\n",fileBuffer);
61
62         temp = '\0';
63         nBytes = strlen(fileBuffer)+1;
64         printf("nBytes:%d\\n",nBytes);
65         send(clientSocket,fileBuffer,nBytes,0);
66         printf("after send\\n");
67         memset(fileBuffer,0,strlen(fileBuffer));
68
69         recv(clientSocket,receivedBuffer , 1024, 0);
70         printf("after receive\\n");
71
72         printf("Received from server: %s\\n", receivedBuffer);
73         memset(receivedBuffer,0,strlen(receivedBuffer));
74
75         fclose(fp);
76         //fclose(fp2);
77     }
78
79 }
80
81 return 0;
82 }
```

Fig. 2 Client Code -cont.



```
1 #include <stdio.h>
2 #include <sys/socket.h>
3 #include <netinet/in.h>
4 #include <string.h>
5 #include <stdlib.h>
6
7 int main(int argc, char *argv[]){
8     //Initialize all variables
9     int intiSocket, newSocket, portNum, clientLen, nBytes,check,len;
10    char buffer[1024],receivedBuffer[1024];
11    char * output;
12    struct sockaddr_in serverAddr;
13    struct sockaddr_storage serverStorage;
14    socklen_t addr_size;
15    char str[1000],text[1000];
16    int i=0,j=0;
17    char temp;
18    intiSocket = socket(AF_INET, SOCK_STREAM, 0);
19
20    printf("\nPlease enter the port number : \n");
21    scanf("%d",&portNum);
22
23    serverAddr.sin_family = AF_INET; //Address family
24    serverAddr.sin_port = htons(portNum);
25    serverAddr.sin_addr.s_addr = INADDR_ANY; //IP Address
26    memset(serverAddr.sin_zero, '\0', sizeof serverAddr.sin_zero);
27
28    receivedBuffer[0]='\0';
29    temp = '\0';
30    check = bind(intiSocket, (struct sockaddr *) &serverAddr, sizeof(serverAddr));
31    if(check==-1) // bind: Associating the parent socket with a port
32    {
33        printf("Binded UnSuccessfully\n");
34        return 0;
35    }
36
37    printf("Binded Successfully\n");
38
39    if(listen(intiSocket,5)==0) // if bind successful, returns 0 else -1
40        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 x RegisterFile.v x ALU.v x InstructionMemory.v x PC.v x lw.v x ControlUnit.v x ServerKT.c x ClientKT.c x clint.c x servo.c x

41     else
42         printf("Error\n");
43
44     addr_size = sizeof serverStorage;
45
46     while(1){
47         if(!fork()){
48             newSocket = accept(intiSocket, (struct sockaddr*)&serverStorage, &addr_size);
49             //New Socket created
50             nBytes = 1;
51             printf("new client \n");
52             while(nBytes!=0)
53             {
54                 buffer[0]='\0';
55                 text[0]='\0';
56
57                 strcpy(receivedBuffer, buffer);
58                 nBytes = recv(newSocket,buffer,1024,0); //No.of bytes recieved from the client
59                 printf("nbytes = %i\n",nBytes);
60
61                 for (i=0;i<nBytes-1;i++)
62                 {
63                     buffer[i] = toupper(buffer[i]);
64                     // Added additional function toupper to convert lowercase letters to uppercase
65                 }
66
67                 i = 0;
68                 len=strlen(buffer);
69                 j=len-1;
70                 for(i=0;i<len;i++) // Reversing the string letter by letter
71                 {
72                     str[i]=buffer[j];
73                     j--;
74                 }
75
76                 printf("%s\n",buffer);
77                 send(newSocket,str,strlen(str),0); //Send the reversed string back to the client
78                 memset(buffer,0,strlen(buffer));
79                 memset(str,0,strlen(str));
80             }
81         }
}
```

Fig. 4 Server Code -cont.

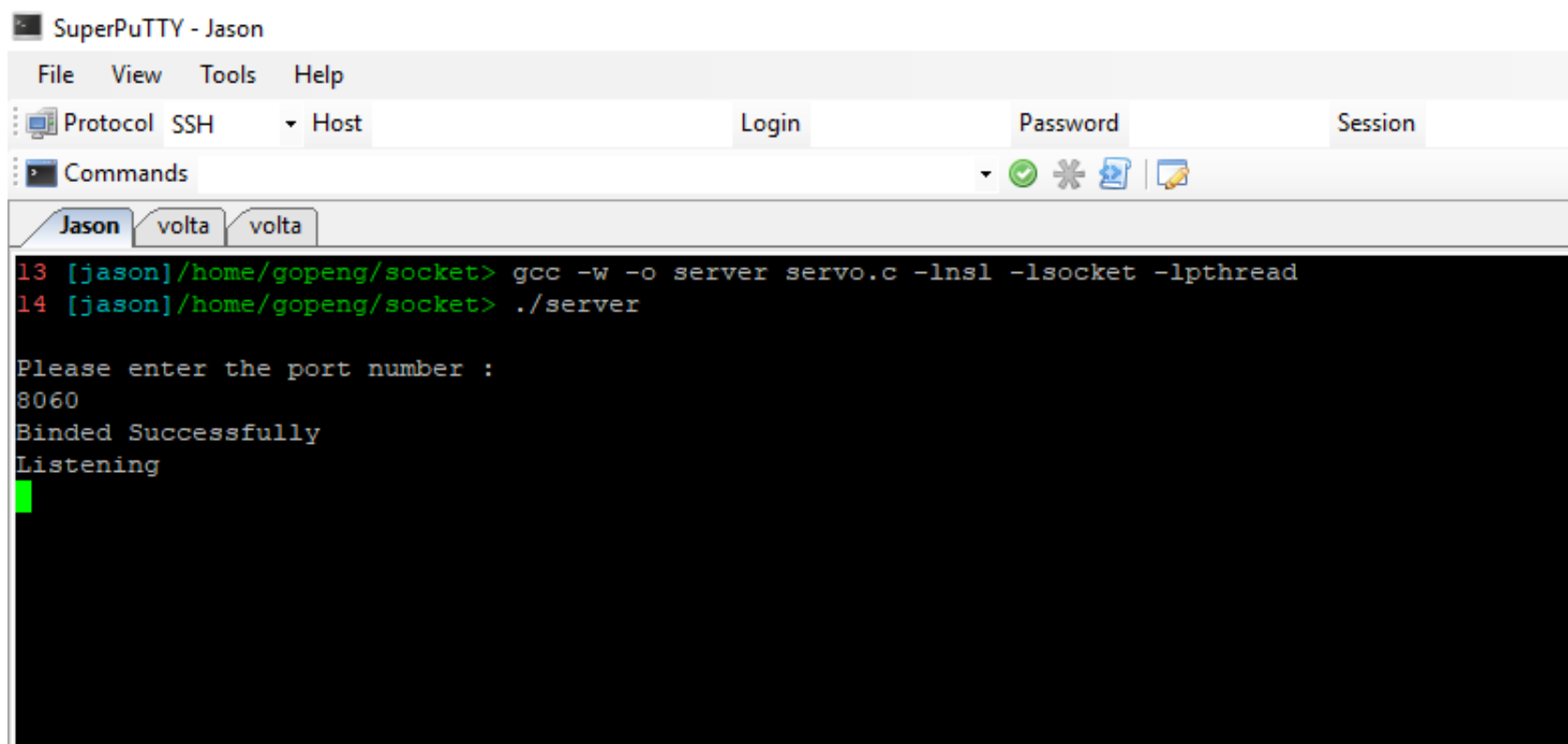
```
*C:\Users\Rayne\Downloads\servo.c - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

ServerKT.c x RegisterFile.v x ALU.v x InstructionMemory.v x PC.v x lw.v x ControlUnit.v x ServerKT.c x ClientKT.c x clint.c x servo.c x

80      }
81      }
82      else{
83          newSocket = accept(intiSocket, (struct sockaddr*)&serverStorage, &addr_size);
84          //New Socket created
85          nBytes = 1;
86          printf("new client \n");
87          while(nBytes!=0)
88          {
89              buffer[0]='\0';
90              text[0]='\0';
91
92              strcpy(receivedBuffer, buffer);
93              nBytes = recv(newSocket,buffer,1024,0); //No.of bytes recieved from the client
94              printf("nbytes = %i\n",nBytes);
95
96              for (i=0;i<nBytes-1;i++)
97              {
98                  buffer[i] = toupper(buffer[i]);
99                  // Added additional function toupper to convert lowercase letters to uppercase
100             }
101
102             i = 0;
103             len=strlen(buffer);
104             j=len-1;
105             for(i=0;i<len;i++) // Reversing the string letter by letter
106             {
107                 str[i]=buffer[j];
108                 j--;
109             }
110
111             printf("%s\n",buffer);
112             send(newSocket,str,strlen(str),0); //Send the reversed string back to the client
113             memset(buffer,0,strlen(buffer));
114             memset(str,0,strlen(str));
115         }
116     }
117     close(newSocket);
118 }
119
120 return 0;
```

Fig. 5 Server Code -cont.





The screenshot shows a SuperPuTTY terminal window titled "SuperPuTTY - Jason". The interface includes a menu bar (File, View, Tools, Help), a toolbar with icons for Protocol (SSH), Host, Login, Password, and Session, and a Commands section. The terminal session is for a user named "jason" in the directory "/home/gopeng/socket". The commands entered are:

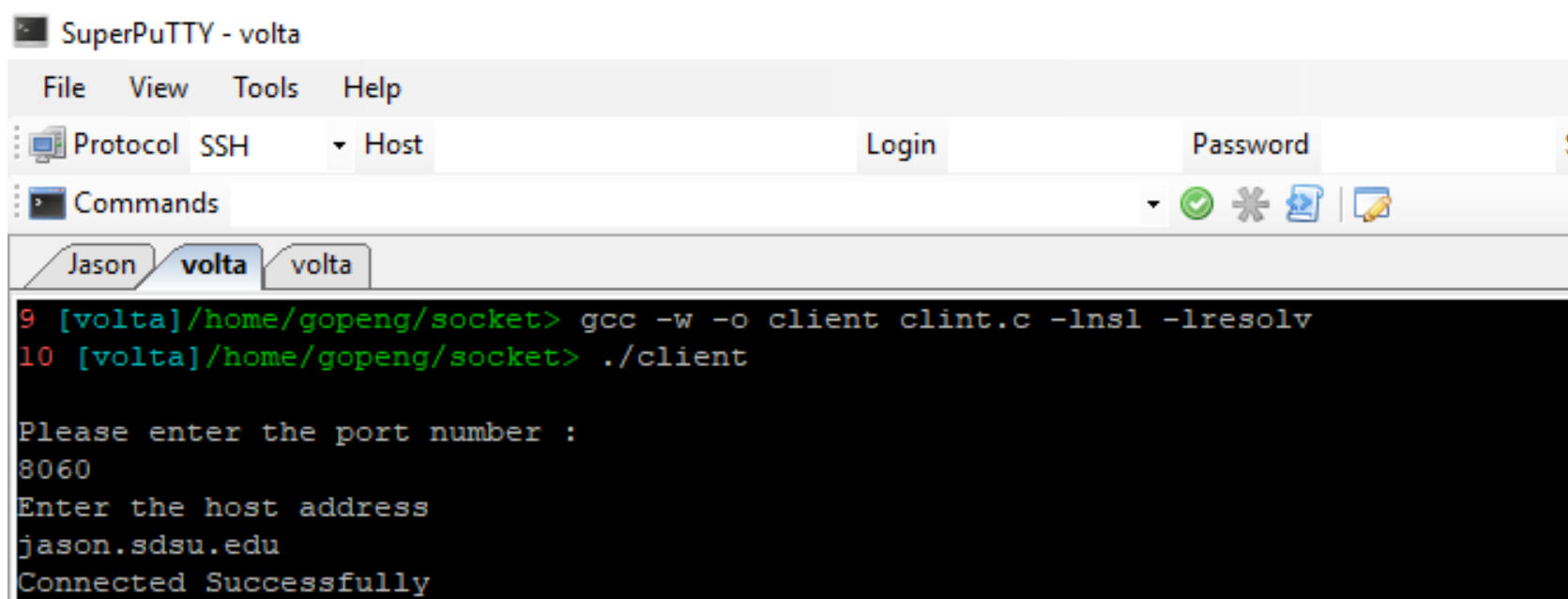
```
13 [jason]/home/gopeng/socket> gcc -w -o server servo.c -lnsl -lsocket -lpthread
14 [jason]/home/gopeng/socket> ./server
```

The output of the program is:

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

A green cursor is visible on the line following "Listening".

Fig.6 Server setup code



The screenshot shows a SuperPuTTY terminal window titled "SuperPuTTY - volta". The interface is similar to the previous one, with a menu bar, toolbar, and Commands section. The terminal session is for a user named "volta" in the directory "/home/gopeng/socket". The commands entered are:

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

The output of the program is:

```
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.

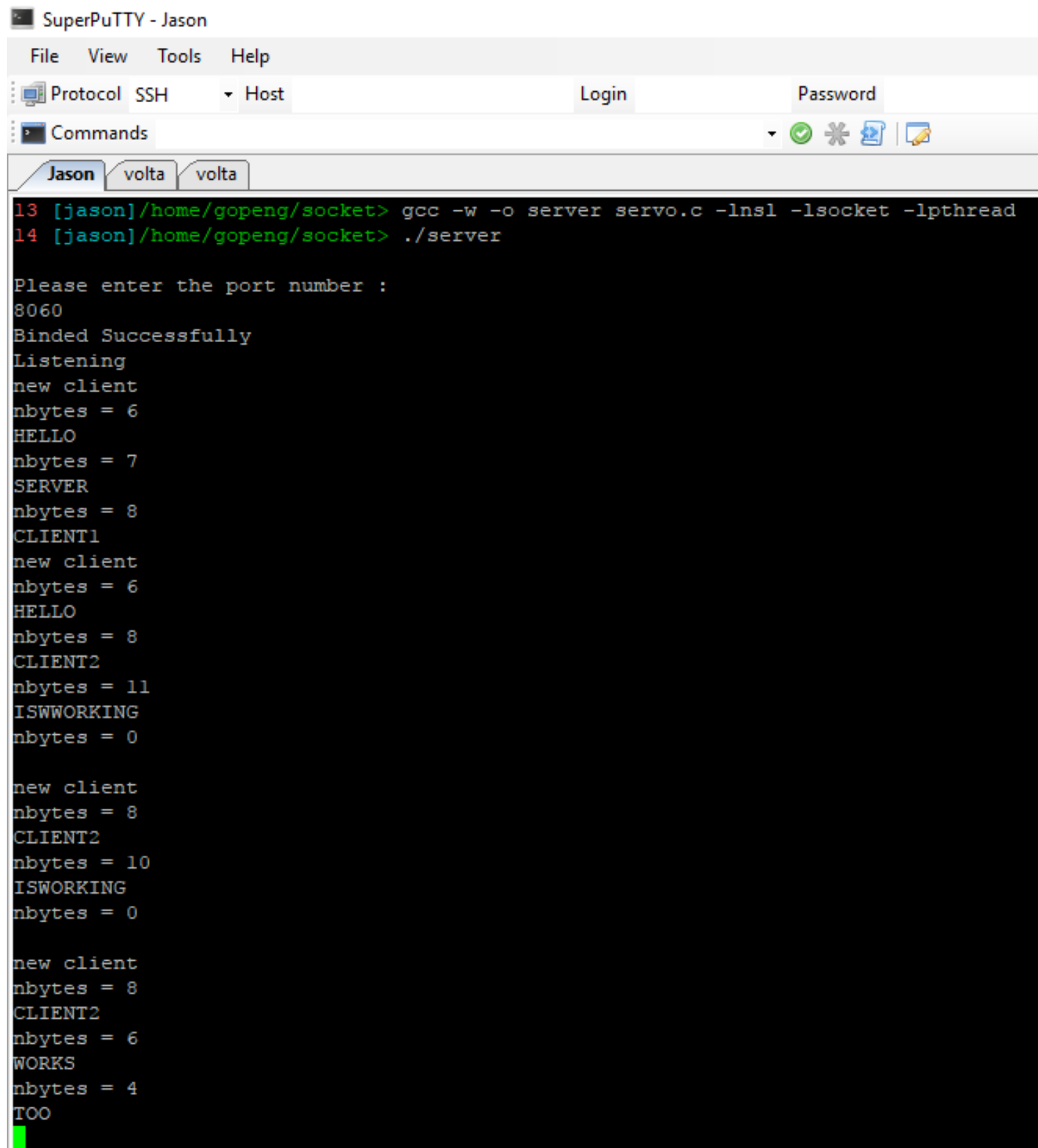
```
SuperPuTTY - volta
File View Tools Help
Protocol SSH Host Login Password
Commands
Jason volta volta
9 [volta]/home/gopeng/socket> gcc -w -o client clint.c -lnsl -lresolv
10 [volta]/home/gopeng/socket> ./client

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:
clientl
You typed: clientl
nBytes:8
after send
after receive
Received from server: lTNEILC
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 be seen in the screenshot).





```
SuperPuTTY - Jason
File View Tools Help
Protocol SSH Host Login Password
Commands
Jason volta volta
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
ISWORKING
nbytes = 0

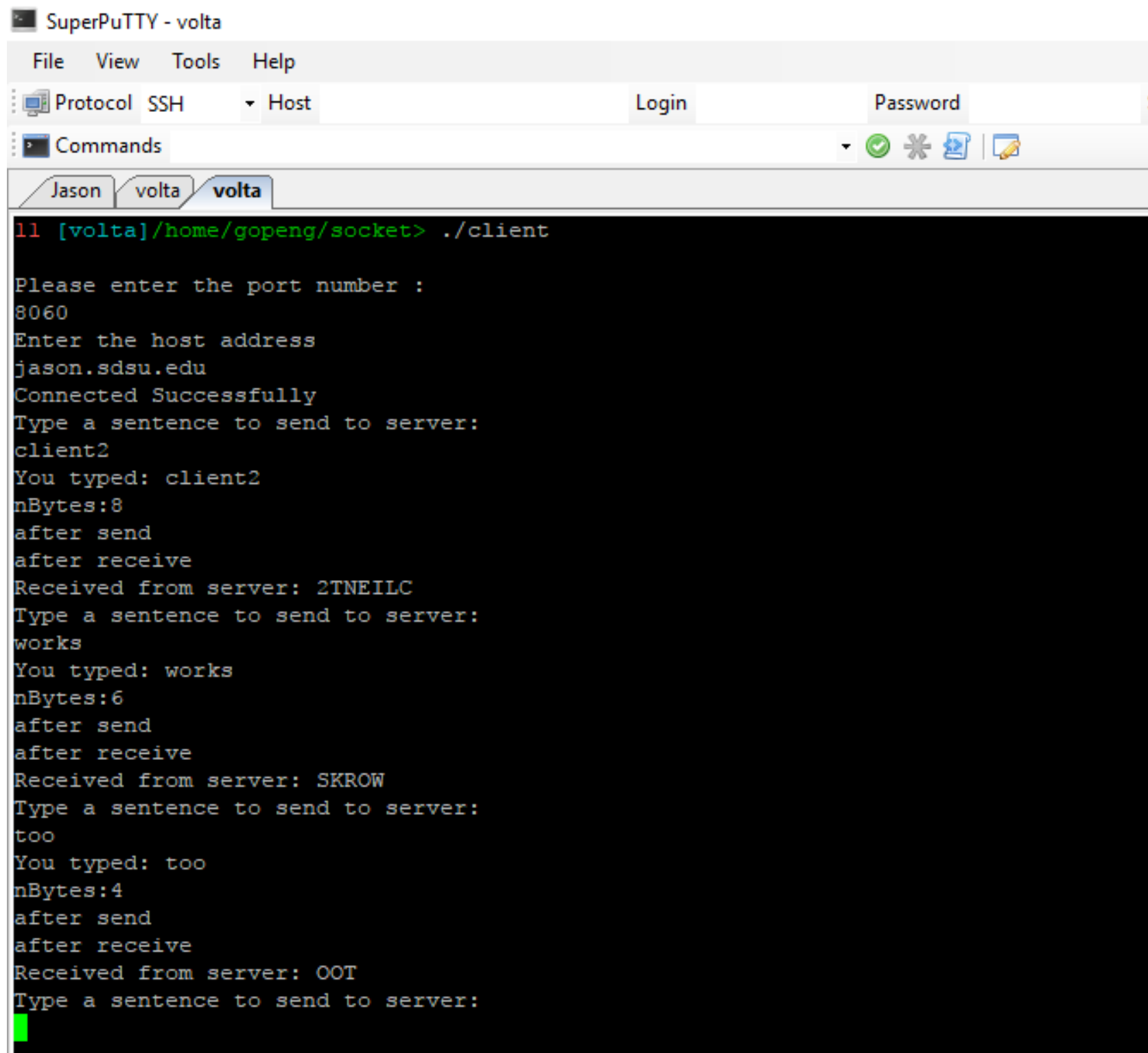
new client
nbytes = 8
CLIENT2
nbytes = 10
ISWORKING
nbytes = 0

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 conveys 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 it 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.



The screenshot shows a SuperPuTTY terminal window titled "SuperPuTTY - volta". The window has a menu bar with "File", "View", "Tools", and "Help". Below the menu bar, there are tabs for "Protocol" (SSH), "Host", "Login", and "Password". A "Commands" section is visible on the right. The terminal window has three tabs: "Jason", "volta", and "volta" (selected). The terminal output shows a client connecting to a server. The client prompts for a port number (8060) and host address (jason.sdsu.edu). The connection is successful. The client then sends "client2" and receives "2TNEILC". It then sends "works" and receives "SKROW". Finally, it sends "too" and receives "OOT". The terminal output is as follows:

```
ll [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.

```
SuperPuTTY - Jason
File View Tools Help
Protocol SSH Host Login Password Session
Commands
Jason volta volta
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.