

Submitted by – Paras Jain (2018KUCP1006)

Computer Networks Lab Assignment -3 Socket Programming: Socket Client Connection

Q. Write a program for TCP echo client server.

Server Side Code:

```
//Server Side Code
#include <stdio.h>
#include <string.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include <netinet/in.h>

int main(){
    int mySocket, clintConnt;
    char Send_buffer[1024], Rec_buffer[1024];
    struct sockaddr_in ipOfServer;
    //Creating the socket, arguments are: Internet domain, Stream socket, Default protocol (TCP)
    mySocket = socket(AF_INET, SOCK_STREAM, 0);
    //Configure settings of the server address struct
    ipOfServer.sin_family = AF_INET;
    // Set port number, using htons function to use proper byte order
    ipOfServer.sin_port = htons(2017);
    ipOfServer.sin_addr.s_addr = htonl(INADDR_ANY); //bind to any local address
    // Set all bits of the padding field to 0
    memset(ipOfServer.sin_zero, '\0', sizeof ipOfServer.sin_zero);
    // Bind the address struct to the socket
    bind(mySocket, (struct sockaddr *)&ipOfServer, sizeof(ipOfServer));
    //Listen on the socket, with 20 max connection requests queued
    listen(mySocket, 20);
    while (1){
        //Accept call creates a new socket for the incoming connection
        clintConnt = accept(mySocket, (struct sockaddr *)NULL, NULL);
        //Recieve message from the socket of the incoming connection
```

```

    recv(clintConnt, Rec_buffer, 1024, 0);
    //Printing recieved message
    printf("Msg recieved from client: %s\n", Rec_buffer);
    //Generating msg for sending
    strcpy(Send_buffer, "This is eco msg from server:\n");
    //Concatinating the recieved msg to the sending buffer
    strcat(Send_buffer, Rec_buffer);
    //Sending message to the socket of the incoming connection
    send(clintConnt, Send_buffer, 1024, 0);
}
return 0;
}

```

Client Side Code:

```

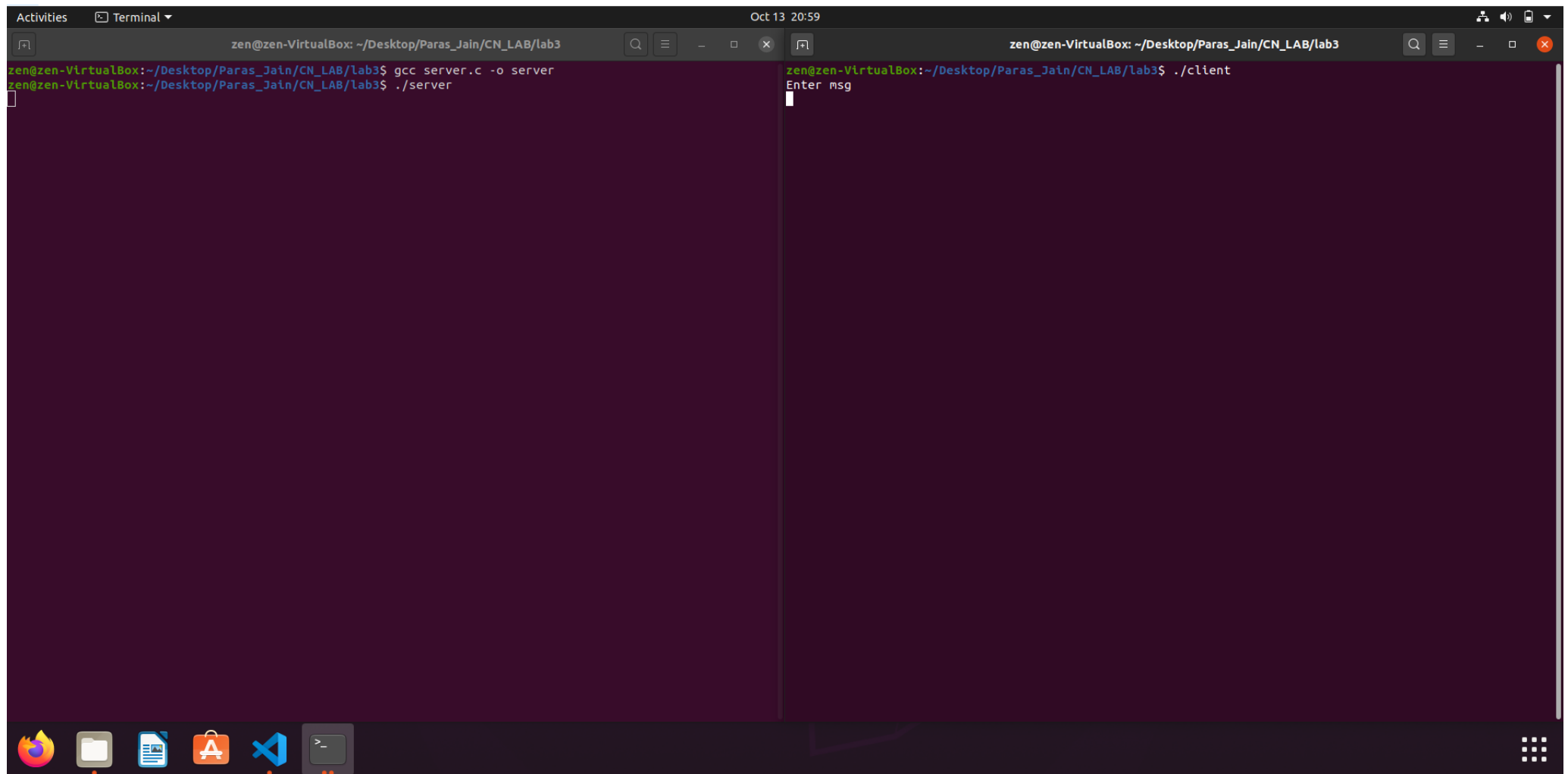
//Client Side Code
#include <stdio.h>
#include <string.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include <netinet/in.h>

int main(){
    int clientSocket;
    char Send_buffer[1024], Rec_buffer[1024];
    struct sockaddr_in ipOfServer;
    //Creating the socket, arguments are: Internet domain, Stream socket, Default protocol (TCP)
    clientSocket = socket(AF_INET, SOCK_STREAM, 0);
    //Configure settings of the server address struct

```

```
ipOfServer.sin_family = AF_INET;
//Set port number, using htons function to use proper byte order
ipOfServer.sin_port = htons(2017);
//Set IP address to localhost
ipOfServer.sin_addr.s_addr = inet_addr("127.0.0.1");
//Set all bits of the padding field to 0
memset(ipOfServer.sin_zero, '\0', sizeof ipOfServer.sin_zero);
//Connect the socket to the server using the address struct
if (connect(clientSocket, (struct sockaddr *)&ipOfServer, sizeof(ipOfServer)) < 0){
    //connecting to server, -ve value implies unsuccessful
    printf("Connection failed due to port and ip problems\n");
    return 1;
}
printf("Enter msg\n");
gets(Send_buffer);
// scanf("%s",Send_buffer);
send(clientSocket, Send_buffer, 1024, 0);
//Read the message from the server into the buffer
recv(clientSocket, Rec_buffer, 1024, 0);
//Print the received message
printf("Msg from server:\n");
puts(Rec_buffer);
return 0;
}
```

Output:



The image shows two terminal windows side-by-side in a virtual machine environment. The left window shows the compilation of a C program named 'server.c' into an executable named 'server'. The right window shows the execution of the 'server' program, which prompts the user to enter a message.

```
zen@zen-VirtualBox: ~/Desktop/Paras_Jain/CN_LAB/lab3
zen@zen-VirtualBox:~/Desktop/Paras_Jain/CN_LAB/lab3$ gcc server.c -o server
zen@zen-VirtualBox:~/Desktop/Paras_Jain/CN_LAB/lab3$ ./server

zen@zen-VirtualBox:~/Desktop/Paras_Jain/CN_LAB/lab3$ ./client
Enter msg
```

