

Ex.No:3	SIMPLE TCP/IP CLIENT SERVER COMMUNICATION
Date:	

Aim:

There are two hosts, Client and Server. The Client accepts the message from the user and sends it to the Server. The Server receives the message and prints it.

TECHNICAL OBJECTIVE:

To implement a simple TCP Client-Server application , where the Client on establishing a connection with the Server, sends a string to the Server. The Server reads the String and prints it.

METHODOLOGY:

Server:

- Include the necessary header files.
- Create a socket using socket function with family AF_INET, type as SOCK_STREAM.
- Initialize server address to 0 using the bzero function.
- Assign the sin_family to AF_INET, sin_addr to INADDR_ANY, sin_port to a dynamically assigned port number.
- Bind the local host address to socket using the bind function.
- Listen on the socket for connection request from the client.
- Accept connection request from the client using accept function.
- Within an infinite loop, using the recv function receive message from the client and print it on the console.

Client:

- Include the necessary header files.
- Create a socket using socket function with family AF_INET, type as SOCK_STREAM.
- Initialize server address to 0 using the bzero function.
- Assign the sin_family to AF_INET.
- Get the server IP address and port number from the console.
- Using gethostbyname function assign it to a hostent structure, and assign it to sin_addr of the server address structure.
- Request a connection from the server using the connect function.
- Within an infinite loop, read message from the console and send the message to the server using the send function.

CODING:

Server: tcpserver.c

```
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<netdb.h>
#include<arpa/inet.h>
#include<string.h>
#include<string.h>
#include<stdio.h>
int main(int asrgc,char*argv[])
{
```

```

        int bd,sd,ad;
        char buff[1024];
        struct sockaddr_in cliaddr,servaddr;
        socklen_t clilen;
        clilen=sizeof(cliaddr);
        bzero(&servaddr,sizeof(servaddr));

        /*Socket address structure*/
        servaddr.sin_family=AF_INET;
        servaddr.sin_addr.s_addr=htonl(INADDR_ANY);
        servaddr.sin_port=htons(1999);

        /*TCP socket is created, an Internet socket address structure is filled with wildcard address &
        server's well known port*/
        sd=socket(AF_INET,SOCK_STREAM,0);

        /*Bind function assigns a local protocol address to the socket*/
        bd=bind(sd,(struct sockaddr*)&servaddr,sizeof(servaddr));

        /*Listen function specifies the maximum number of connections that kernel should queue for
        this socket*/
        listen(sd,5);
        printf("Server is running...\n");

        /*The server to return the next completed connection from the front of the
        completed connection Queue calls it*/
        ad=accept(sd,(struct sockaddr*)&cliaddr,&clilen);
        while(1)
        {
            bzero(&buff,sizeof(buff));
            /*Receiving the request message from the client*/
            recv(ad,buff,sizeof(buff),0);
            printf("Message received is %s\n",buff);
        }
    }

```

Client: tcpclient.c

```

#include<stdio.h>
#include<string.h>
#include<sys/socket.h>
#include<sys/types.h>
#include<unistd.h>
#include<netinet/in.h>
#include<netdb.h>
#include<arpa/inet.h>
int main(int argc,char * argv[])
{
    int cd,sd,ad;
    char buff[1024];
    struct sockaddr_in cliaddr,servaddr;
    struct hostent *h;

    /*This function looks up a hostname and it returns a pointer to a hostent
    structure that contains all the IPV4 address*/

```

```

h=gethostbyname(argv[1]);
bzero(&servaddr,sizeof(servaddr));

/*Socket address structure*/
servaddr.sin_family=AF_INET;
memcpy((char *)&servaddr.sin_addr.s_addr,h->h_addr_list[0],h->h_length);
servaddr.sin_port = htons(1999);

/*Creating a socket, assigning IP address and port number for that socket*/
sd = socket(AF_INET,SOCK_STREAM,0);

/*Connect establishes connection with the server using server IP address*/
cd=connect(sd,(struct sockaddr*)&servaddr,sizeof(servaddr));
while(1)
{
    printf("Enter the message: \n");

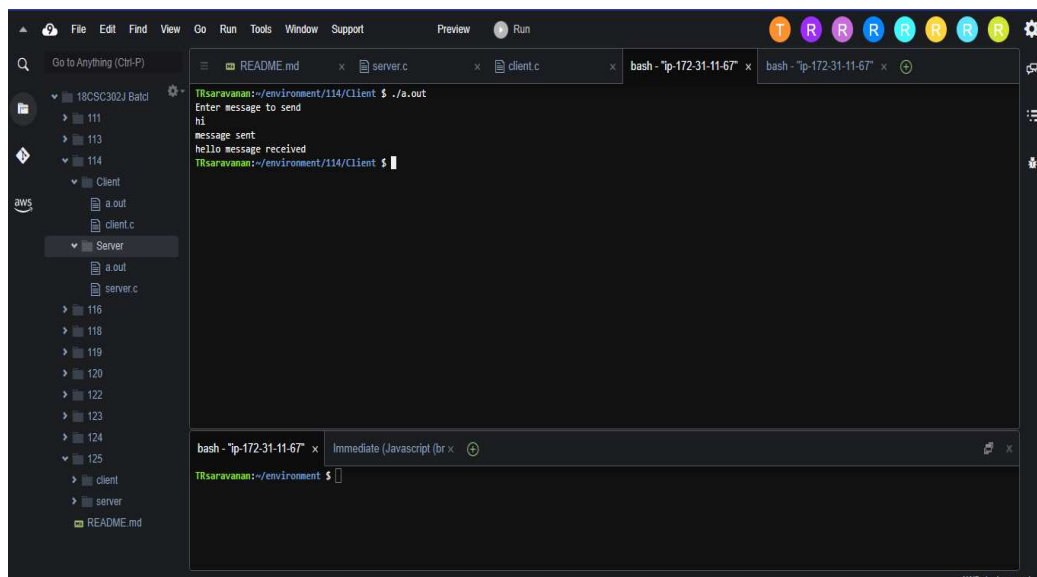
    /*Reads the message from standard input*/
    fgets(buff,100,stdin);

    /*Send function is used on client side to send data given by user on client
    side to the server*/
    send(sd,buff,sizeof(buff)+1,0);
    printf("\n Data Sent ");
    //recv(sd,buff,strlen(buff)+1,0);
    printf("%s",buff);
}
}

```

OUTPUT :

Client Output:

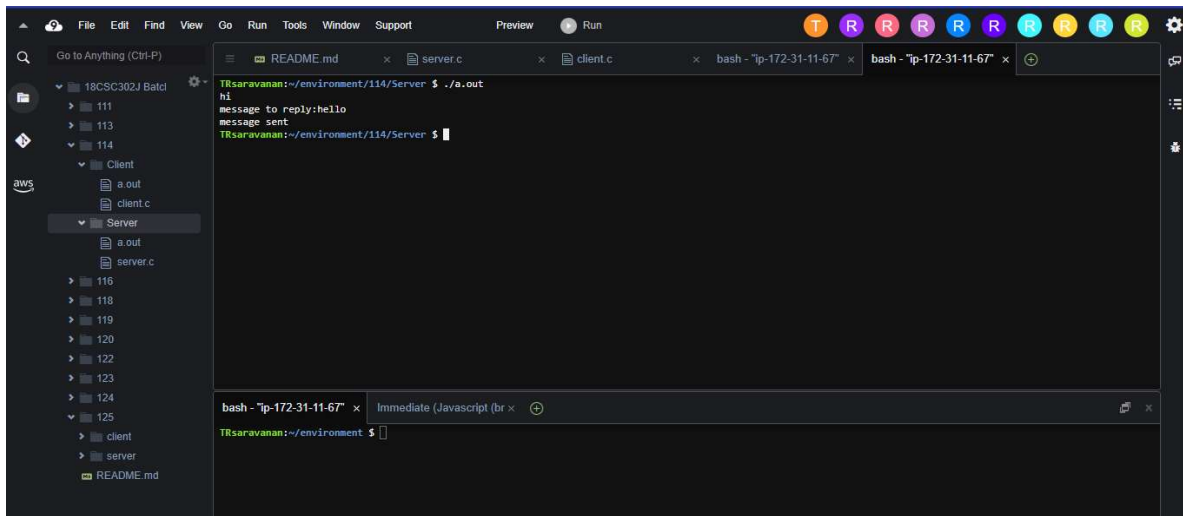


```

TRsaravanan:~/environment/114/Client $ ./a.out
Enter message to send
hi
message sent
hello message received
TRsaravanan:~/environment/114/Client $

```

Server Output:



The screenshot shows an IDE with a file explorer on the left and a terminal window in the center. The file explorer shows a project structure with folders 111, 113, 114, Client, and Server. The terminal window shows the output of a server program. The output is as follows:

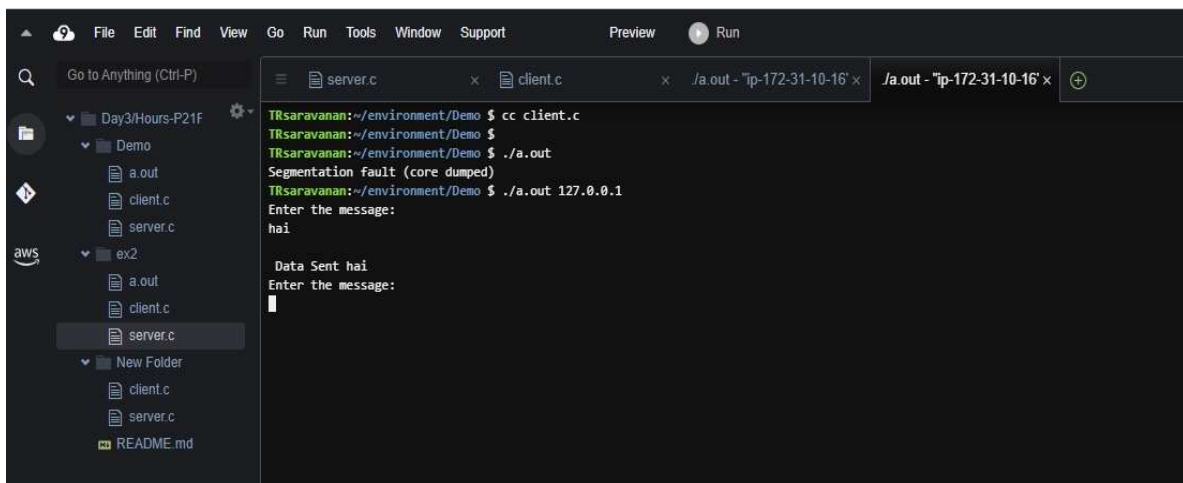
```
TRsaravanan:~/environment/114/Server $ ./a.out
hi
message to reply:hello
message sent
TRsaravanan:~/environment/114/Server $
```

Execution Procedure when we receive Segmentation Fault:



The screenshot shows a terminal window with the following output:

```
TRsaravanan:~/environment/Demo $ cc server.c
TRsaravanan:~/environment/Demo $ ./a.out
Server is running...
Message received is hai
Message received is
```



The screenshot shows an IDE with a file explorer on the left and a terminal window in the center. The file explorer shows a project structure with folders Day3/Hours-P21f, Demo, and ex2. The terminal window shows the output of a client program. The output is as follows:

```
TRsaravanan:~/environment/Demo $ cc client.c
TRsaravanan:~/environment/Demo $
TRsaravanan:~/environment/Demo $ ./a.out
Segmentation fault (core dumped)
TRsaravanan:~/environment/Demo $ ./a.out 127.0.0.1
Enter the message:
hai
Data Sent hai
Enter the message:
```

RESULT: Hence, the TPC/IP server client experiment is studied and performed.