Ex No: 4 Pranit Tandon

Date: 09/08/2021 RA1911028010014 (CSE – H2) UDP ECHO CLIENT SERVER COMMUNICATION

# GIVEN REQUIREMENTS:

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, prints it and echoes the message back to the Client.

# TECHNICAL OBJECTIVE:

To implement an UDP Echo Client-Server application , where the Client on establishing a connection with the Server, sends a string to the Server. The Server reads the String, prints it and echoes it back to the Client.

# METHODOLOGY:

**Server:**

# Client:

* Include the necessary header files.
* Create a socket using socket function with family AF\_INET, type as SOCK\_DGRAM.
* Initialize server address to 0 using the bzero function.
* Assign the sin\_family to AF\_INET, sin\_addr to INADDR\_ANY, sin\_port to SERVER\_PORT, a macro defined port number.
* Bind the local host address to socket using the bind function.
* Within an infinite loop, receive message from the client using recvfrom function, print it on the console and send (echo) the message back to the client using sendto function.
* Include the necessary header files.
* Create a socket using socket function with family AF\_INET, type as SOCK\_DGRAM.
* Initialize server address to 0 using the bzero function.
* Assign the sin\_family to AF\_INET.
* Get the server IP address from the console.
* Using gethostbyname function assign it to a hostent structure, and assign it to sin\_addr of the server address structure.
* Within an infinite loop, read message from the console and send the message to the server using the sendto function.
* Receive the echo message using the recvfrom function and print it on the console.

# CODING:

**Server: udpserver.c**

#include<sys/socket.h> #include<stdio.h> #include<unistd.h> #include<string.h> #include<netinet/in.h>

#include<netdb.h> #include<arpa/inet.h> #include<sys/types.h>

int main(int argc,char \*argv[])

{

int sd;

char buff[1024];

struct sockaddr\_in cliaddr,servaddr; socklen\_t clilen; clilen=sizeof(cliaddr);

# /\*UDP socket is created, an Internet socket address structure is filled with wildcard address & server’s well known port\*/

sd=socket(AF\_INET,SOCK\_DGRAM,0); if (sd<0)

{

perror ("Cannot open Socket"); exit(1);

}

bzero(&servaddr,sizeof(servaddr));

**/\*Socket address structure\*/** servaddr.sin\_family=AF\_INET; servaddr.sin\_addr.s\_addr=htonl(INADDR\_ANY); servaddr.sin\_port=htons(5669);

# /\*Bind function assigns a local protocol address to the socket\*/

if(bind(sd,(struct sockaddr\*)&servaddr,sizeof(servaddr))<0)

{

perror("error in binding the port"); exit(1);

}

printf("%s","Server is Running…\n"); while(1)

{

bzero(&buff,sizeof(buff));

# /\*Read the message from the client\*/

if(recvfrom(sd,buff,sizeof(buff),0,(struct sockaddr\*)&cliaddr,&clilen)<0)

{

perror("Cannot rec data"); exit(1);

}

printf("Message is received \n",buff);

# /\*Sendto function is used to echo the message from server to client side\*/

if(sendto(sd,buff,sizeof(buff),0,(struct sockadddr\*)&cliaddr,clilen)<0)

{

perror("Cannot send data to client"); exit(1);

}

printf("Send data to UDP Client: %s",buff);

}

cloSe(sd); return 0;

}

# Client: udpclient.c

#include<sys/types.h> #include<sys/socket.h> #include<stdio.h> #include<unistd.h> #include<string.h> #include<netinet/in.h> #include<netdb.h>

int main(int argc,char\*argv[])

{

int sd;

char buff[1024];

struct sockaddr\_in servaddr; socklen\_t len; len=sizeof(servaddr);

# /\*UDP socket is created, an Internet socket address structure is filled with wildcard address & server’s well known port\*/

sd = socket(AF\_INET,SOCK\_DGRAM,0); if(sd<0)

{

perror("Cannot open socket"); exit(1);

}

bzero(&servaddr,len);

**/\*Socket address structure\*/** servaddr.sin\_family=AF\_INET; servaddr.sin\_addr.s\_addr=htonl(INADDR\_ANY); servaddr.sin\_port=htons(5669);

while(1)

{

printf("Enter Input data : \n"); bzero(buff,sizeof(buff));

# /\*Reads the message from standard input\*/

fgets(buff,sizeof (buff),stdin);

# /\*sendto is used to transmit the request message to the server\*/

if(sendto (sd,buff,sizeof (buff),0,(struct sockaddr\*)&servaddr,len)<0)

{

perror("Cannot send data"); exit(1);

}

printf("Data sent to UDP Server:%s",buff); bzero(buff,sizeof(buff));

# /\*Receiving the echoed message from server\*/

if(recvfrom (sd,buff,sizeof(buff),0,(struct sockaddr\*)&servaddr,&len)<0)

{

perror("Cannot receive data"); exit(1);

}

printf("Received Data from server: %s",buff);

}

close(sd); return 0;

}

# SAMPLE OUTPUT:

**Server**:

# (Host Name:Root1)

[root@localhost 4ita33]# vi udpserver.c [root@localhost 4ita33]# cc udpserver.c [root@localhost 4ita33]# ./a.out

Server is Running…

Message is received

Send data to UDP Client: hi

Message is received

Send data to UDP Client: how are u

# Client:

**(Host Name:Root2)**

[root@localhost 4ita33]# vi udpclient.c [root@localhost 4ita33]# cc udpclient.c [root@localhost 4ita33]# ./a.out 127.0.0.1 Enter input data :

hi

Data sent to UDP Server:hi Received Data from server: hi

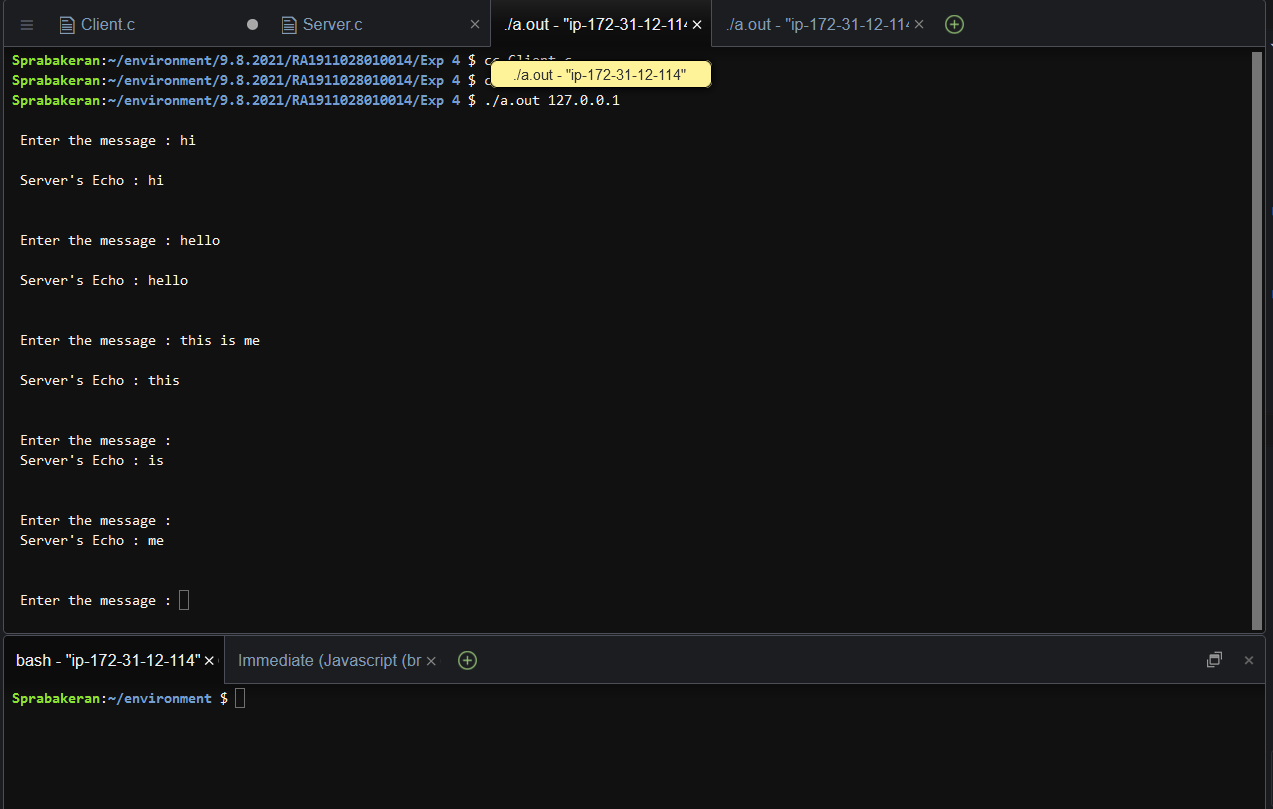
Enter input data :

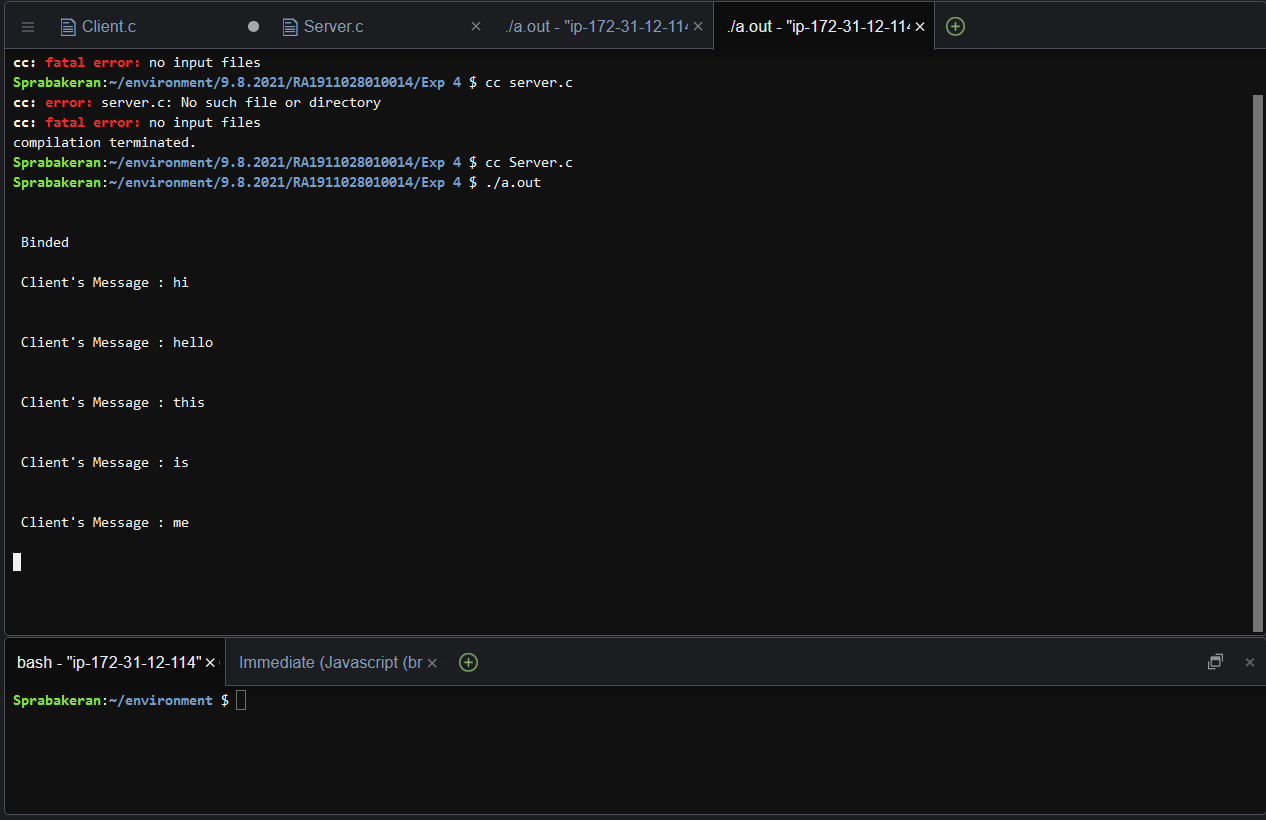
how are u

Data sent to UDP Server:how are u Received Data from server: how are u

Enter input data :

# OUTPUT:

****

****

**INFERENCE:**

Thus, the UDP ECHO client server communication is established by sending the message from the client to the server and server prints it and echoes the message back to the client.