# 

**Madhurya Mozumder**

**RA1911028010036**

**CSE-CC**

**I2**

# **Ex.No-7 FULL DUPLEX CHAT USING TCP**

**AIM:** To implement Full-Duplex chat using TCP client-server.

**Server.c**

**#include<sys/types.h>**

**#include<sys/socket.h>**

**#include<stdio.h>**

**#include<unistd.h>**

**#include<netdb.h>**

**#include<arpa/inet.h>**

**#include<netinet/in.h>**

**#include<string.h>**

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

**{**

**int clientSocketDescriptor,socketDescriptor;**

**struct sockaddr\_in serverAddress,clientAddress;**

**socklen\_t clientLength;**

**char recvBuffer[1000],sendBuffer[1000];**

**pid\_t cpid;**

**bzero(&serverAddress,sizeof(serverAddress));**

**/\*Socket address structure\*/**

**serverAddress.sin\_family=AF\_INET;**

**serverAddress.sin\_addr.s\_addr=htonl(INADDR\_ANY);**

**serverAddress.sin\_port=htons(6601);**

**/\*TCP socket is created, an Internet socket address structure is filled with**

**wildcard address & server’s well known port\*/**

**socketDescriptor=socket(AF\_INET,SOCK\_STREAM,0);**

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

**bind(socketDescriptor,(struct sockaddr\*)&serverAddress,sizeof(serverAddress));**

**/\*Listen function specifies the maximum number of connections that kernel should queue**

**for this socket\*/**

**listen(socketDescriptor,5);**

**printf("%s\n","Server is running ...");**

**/\*The server to return the next completed connection from the front of the**

**completed connection Queue calls it\*/**

**clientSocketDescriptor=accept(socketDescriptor,(struct sockaddr\*)&clientAddress,&clientLength);**

**/\*Fork system call is used to create a new process\*/**

**cpid=fork();**

**if(cpid==0)**

**{**

**while(1)**

**{**

**bzero(&recvBuffer,sizeof(recvBuffer));**

**/\*Receiving the request from client\*/**

**recv(clientSocketDescriptor,recvBuffer,sizeof(recvBuffer),0);**

**printf("\nCLIENT : %s\n",recvBuffer);**

**}**

**}**

**else**

**{**

**while(1)**

**{**

**bzero(&sendBuffer,sizeof(sendBuffer));**

**printf("\nType a message here ... ");**

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

**fgets(sendBuffer,10000,stdin);**

**/\*Sends the message to client\*/**

**send(clientSocketDescriptor,sendBuffer,strlen(sendBuffer)+1,0);**

**printf("\nMessage sent !\n");**

**}**

**}**

**return 0;**

**}**

**Client.c**

**#include "stdio.h"**

**#include "stdlib.h"**

**#include "string.h"**

**//headers for socket and related functions**

**#include <sys/types.h>**

**#include <sys/socket.h>**

**//for including structures which will store information needed**

**#include <netinet/in.h>**

**#include <unistd.h>**

**//for gethostbyname**

**#include "netdb.h"**

**#include "arpa/inet.h"**

**int main()**

**{**

**int socketDescriptor;**

**struct sockaddr\_in serverAddress;**

**char sendBuffer[1000],recvBuffer[1000];**

**pid\_t cpid;**

**bzero(&serverAddress,sizeof(serverAddress));**

**serverAddress.sin\_family=AF\_INET;**

**serverAddress.sin\_addr.s\_addr=inet\_addr("127.0.0.1");**

**serverAddress.sin\_port=htons(6601);**

**/\*Creating a socket, assigning IP address and port number for that socket\*/**

**socketDescriptor=socket(AF\_INET,SOCK\_STREAM,0);**

**/\*Connect establishes connection with the server using server IP address\*/**

**connect(socketDescriptor,(struct sockaddr\*)&serverAddress,sizeof(serverAddress));**

**/\*Fork is used to create a new process\*/**

**cpid=fork();**

**if(cpid==0)**

**{**

**while(1)**

**{**

**bzero(&sendBuffer,sizeof(sendBuffer));**

**printf("\nType a message here ... ");**

**/\*This function is used to read from server\*/**

**fgets(sendBuffer,10000,stdin);**

**/\*Send the message to server\*/**

**send(socketDescriptor,sendBuffer,strlen(sendBuffer)+1,0);**

**printf("\nMessage sent !\n");**

**}**

**}**

**else**

**{**

**while(1)**

**{**

**bzero(&recvBuffer,sizeof(recvBuffer));**

**/\*Receive the message from server\*/**

**recv(socketDescriptor,recvBuffer,sizeof(recvBuffer),0);**

**printf("\nSERVER : %s\n",recvBuffer);**

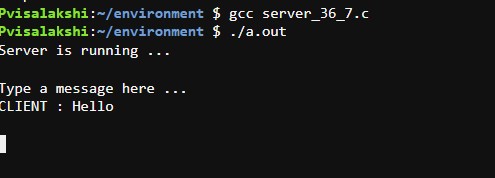
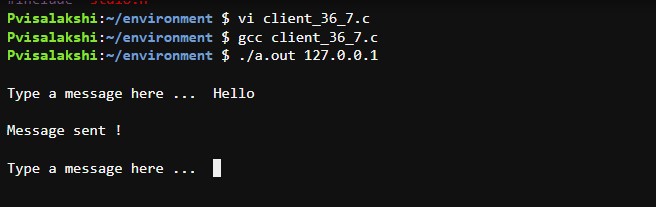
**}**

**}**

**return 0;**

**}**

**OUTPUT:**

****

**RESULT:** Full-Duplex chat using TCP client-server is successfully implemented.