



VIT[®]

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CSE 1004

Network and Communication

LAB ASSESSMENT - 5

NAME: Vibhu Kumar Singh

REG. NO: 19BCE0215

TEACHER: Santhi H.

Q1) Write a program to perform exchange of any text message between server and client and vice versa using TCP Application.

Ans 1)

Aim: To perform exchange of any text message between server and client and vice versa using TCP Application.

Algorithm:

Server Side Algorithm:

- STEP 1: Start
- STEP 2: Declare the variables for the socket
- STEP 3: Specify the family, protocol, IP address and port number
- STEP 4: Create a socket using socket() function
- STEP 5: Bind the IP address and Port number
- STEP 6: Listen and accept the client's request for the connection
- STEP 7: Read the client's message
- STEP 8: Display the client's message
- STEP 9: Continue the chat
- STEP 10: Terminate the chat
- STEP 11: Close the socket
- STEP 12: Stop

Client Side Algorithm:

- STEP 1: Start
- STEP 2: Declare the variables for the socket
- STEP 3: Specify the family, protocol, IP address and port number
- STEP 4: Create a socket using socket() function
- STEP 5: Call the connect() function
- STEP 6: Read the input message
- STEP 7: Send the input message to the server
- STEP 8: Display the server's reply
- STEP 9: Continue the chat
- STEP 10: Terminate the chat
- STEP 11: Close the socket
- STEP 12: Stop

Functions used here are:

System calls that allow to access the network functionality of a Unix box are as given below. When you call one of these functions, the kernel takes over and does all the work for you automatically.

Server Side:

socket () bind() connect() listen() accept() send() recv() close()

Client Side:

socket () gethostbyname() connect() send() recv() close()

Source Code:

Q1Server.c

```
#include<sys/socket.h>
#include<stdio.h>
#include<string.h>
#include<netdb.h>
#include<stdlib.h>
int main()
{
    char buf[100];
    int k;
    socklen_t len;
    int sock_desc,temp_sock_desc;
    struct sockaddr_in server,client;
    memset(&server,0,sizeof(server));
    memset(&client,0,sizeof(client));
    sock_desc=socket(AF_INET,SOCK_STREAM,0);
    if(sock_desc== -1)
    {
        printf("Error in socket creation");
        exit(1);
    }
    server.sin_family=AF_INET;
    server.sin_addr.s_addr=inet_addr("127.0.0.1");
    server.sin_port=3002;
    k=bind(sock_desc,(struct sockaddr*)&server,sizeof(server));
    if(k== -1){
        printf("Error in binding");
        exit(1);
    }
    k=listen(sock_desc,20);
    if(k== -1)
    {
        printf("Error in listening");
        exit(1);
    }
    len=sizeof(client);
    temp_sock_desc=accept(sock_desc,(struct sockaddr*)&client,&len);
    if(temp_sock_desc== -1)
    {
        printf("Error in temporary socket creation");
        exit(1);
    }
    while(1)
    {
        k=recv(temp_sock_desc,buf,100,0);
        if(k== -1)
        {
```

```

        printf("Error in receiving");
        exit(1);
    }
    printf("Message got from client is : %s",buf);
    printf("\nEnter data to be send to client: ");
    fgets(buf,100,stdin);
    if(strncmp(buf,"end",3)==0)
        break;
    k=send(temp_sock_desc,buf,100,0);
    if(k==-1)
    {
        printf("Error in sending");
        exit(1);
    }
}
close(temp_sock_desc);
exit(0);
return 0;
}

```

Q1Client.c

```

#include<sys/socket.h>
#include<stdio.h>
#include<string.h>
#include<netdb.h>
#include<stdlib.h>
int main()
{
    char buf[100];
    int k;
    int sock_desc;
    struct sockaddr_in client;
    memset(&client,0,sizeof(client));
    sock_desc=socket(AF_INET,SOCK_STREAM,0);
    if(sock_desc==-1)
    {
        printf("Error in socket creation");
        exit(1);
    }
    client.sin_family=AF_INET;
    client.sin_addr.s_addr=INADDR_ANY;
    client.sin_port=3002;
    k=connect(sock_desc,(struct sockaddr*)&client,sizeof(client));
    if(k==-1)
    {
        printf("Error in connecting to server");
        exit(1);
    }
    while(1)
    {
        printf("\nEnter data to be send to server: ");
        fgets(buf,100,stdin);
        if(strncmp(buf,"end",3)==0)//Use "end" to end communication with server
            break;
        k=send(sock_desc,buf,100,0);
        if(k==-1)
        {
            printf("Error in sending");
            exit(1);
        }
    }
}

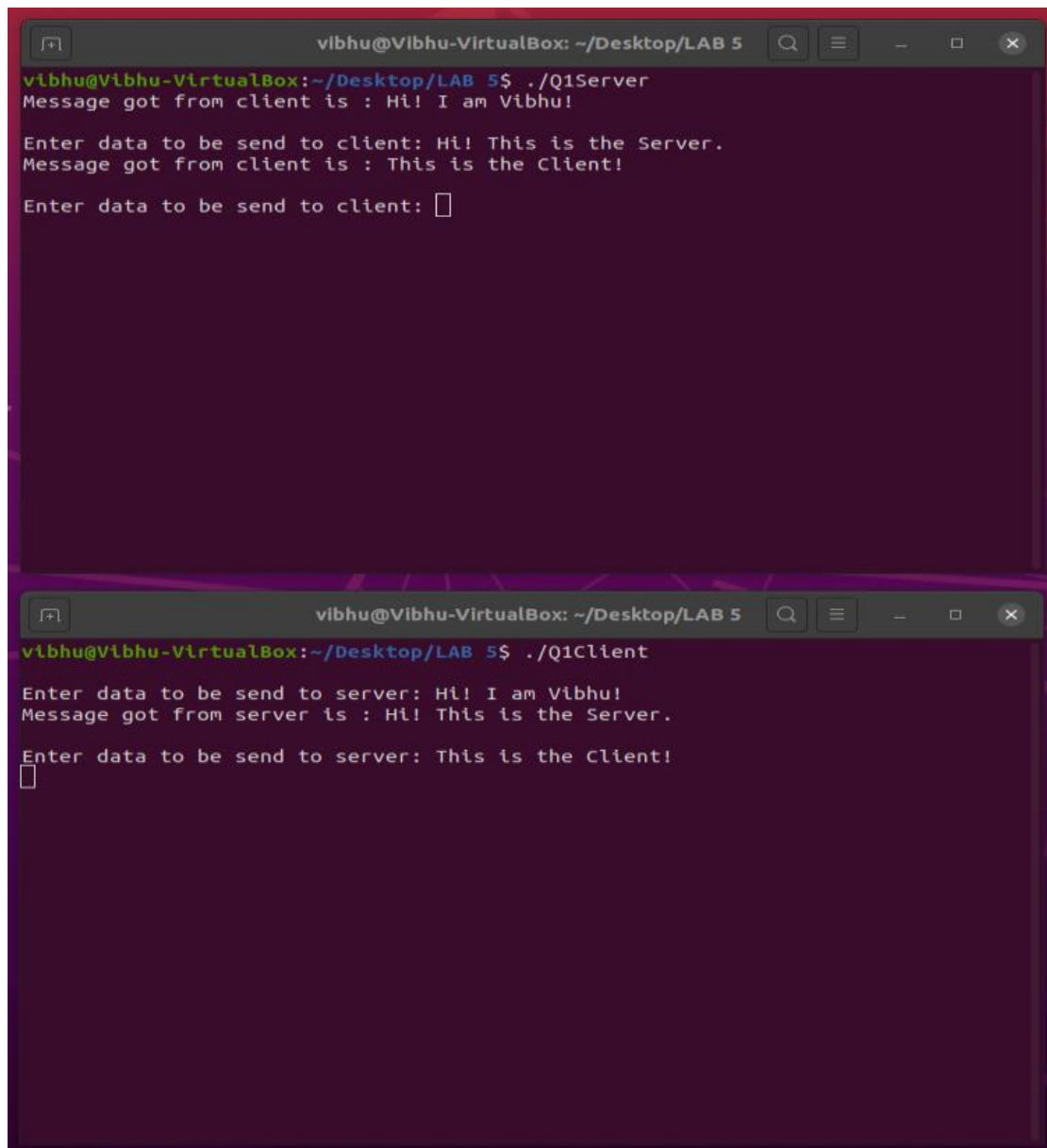
```

```

    }
    k=recv(sock_desc,buf,100,0);
    if(k==-1)
    {
        printf("Error in receiving");
        exit(1);
    }
    printf("Message got from server is : %s",buf);
}
close(sock_desc);
exit(0);
return 0;
}

```

OUTPUT SCREENSHOTS:



Q2) Write a program to display the server's date and time details at the client end.

Ans 2)

Aim: To display the server's date and time details at the client end.

Algorithm:

Server Side Algorithm

STEP 1: Start the program.

STEP 2: Declare the variables and structure for the socket..How to Find Day Time Server to Client Source code in C Programming

STEP 3: The socket is binded at the specified port.

STEP 4: Using the object the port and address are declared.

STEP 5: The listen and accept functions are executed.

STEP 6: If the binding is successful it waits for client request.

STEP 7: Execute the client program.

Client Side Algorithm

STEP 1: Start the program.

STEP 2: Declare the variables and structure.

STEP 3: Socket is created and connect function is executed.

STEP 4: If the connection is successful then server sends the message.

STEP 5: The date and time is printed at the client side.

STEP 6: Stop the program.

Source Code:

Q2Server.c

```
#include <stdio.h>
#include <unistd.h>
#include <time.h>
#include <errno.h>
#include <string.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#define MY_PORT_NUMBER 49999

int main(int argc, char **argv) {
    int listenfd;
    if ((listenfd = socket(AF_INET, SOCK_STREAM, 0)) < 0)
    {
        perror("socket");
        exit(1);
    }
```

```

    }
    printf("Server socket created\n");

    struct sockaddr_in servAddr;

    memset(&servAddr, 0, sizeof(servAddr));
    servAddr.sin_family = AF_INET;
    servAddr.sin_port = htons(MY_PORT_NUMBER);
    servAddr.sin_addr.s_addr = htonl(INADDR_ANY);

    if (bind(listenfd, (struct sockaddr *)&servAddr, sizeof(servAddr)) < 0)
    {
        perror("bind");
        exit(1);
    }
    printf("Socket bound to port\n");

    if ((listen(listenfd, 1)) < 0)
    {
        perror("listen");
        exit(1);
    }
    printf("Listening for connections...\n\n");

    int connectfd, status;
    int length = sizeof(struct sockaddr_in);
    struct sockaddr_in clientAddr;

    while (1) {
        if ((connectfd = accept(listenfd, (struct sockaddr *)&clientAddr, &length))
        < 0) {
            perror("accept");
            exit(1);
        }
        printf("Client connection found!\n");

        if (fork()) {
            close(connectfd);
            waitpid(-1, &status, 0);
        }
        else {
            struct hostent *hostEntry;
            char *hostName;
            if ((hostEntry = gethostbyaddr(&(clientAddr.sin_addr), sizeof(struct in_
            addr), AF_INET)) == NULL) {
                perror("gethostbyaddr");
                exit(1);
            }
            hostName = hostEntry->h_name;
            printf("Connected to: %s\n", hostName);

            char buffer[100];
            time_t ltime;
            time(&ltime);
            strcpy(buffer, ctime(&ltime));
            int len = strlen(buffer);
            buffer[len] = '\0';

            if ((write(connectfd, buffer, 100)) <= 0) {
                perror("write");
            }
        }
    }

```

```

        exit(1);
    }
    printf("Wrote time and date to client\n");
    printf("%s exiting...\n\n", hostName);
    close(connectfd);
    exit(0);
}
}
}

```

Q2Client.c

```

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#define MY_PORT_NUMBER 49999
int main(int argc, char **argv) {
    if(argc < 2) {
        printf("Invalid connection format, format is: <executable> <hostname/address>\n");
        exit(1);
    }

    int sockfd;

    if( (sockfd = socket( AF_INET, SOCK_STREAM, 0)) < 0) {
        perror("socket")
;
        exit(1);
    }
    struct sockaddr_in servAddr;
    struct hostent *hostEntry;
    struct in_addr **pptr;

    memset(&servAddr, 0, sizeof(servAddr));
    servAddr.sin_family = AF_INET;
    servAddr.sin_port = htons(MY_PORT_NUMBER);

    if( (hostEntry = gethostbyname(argv[1])) == NULL ) {
        perror("gethostbyname");
        exit(1);
    }

    /* Structure pointer magic */
    pptr = (struct in_addr**) hostEntry->h_addr_list;
    memcpy(&servAddr.sin_addr, *pptr, sizeof(struct in_addr));
    if( (connect(sockfd, (struct sockaddr *) &servAddr, sizeof(servAddr))) < 0 ) {
        perror("connect");
        exit(1);
    }

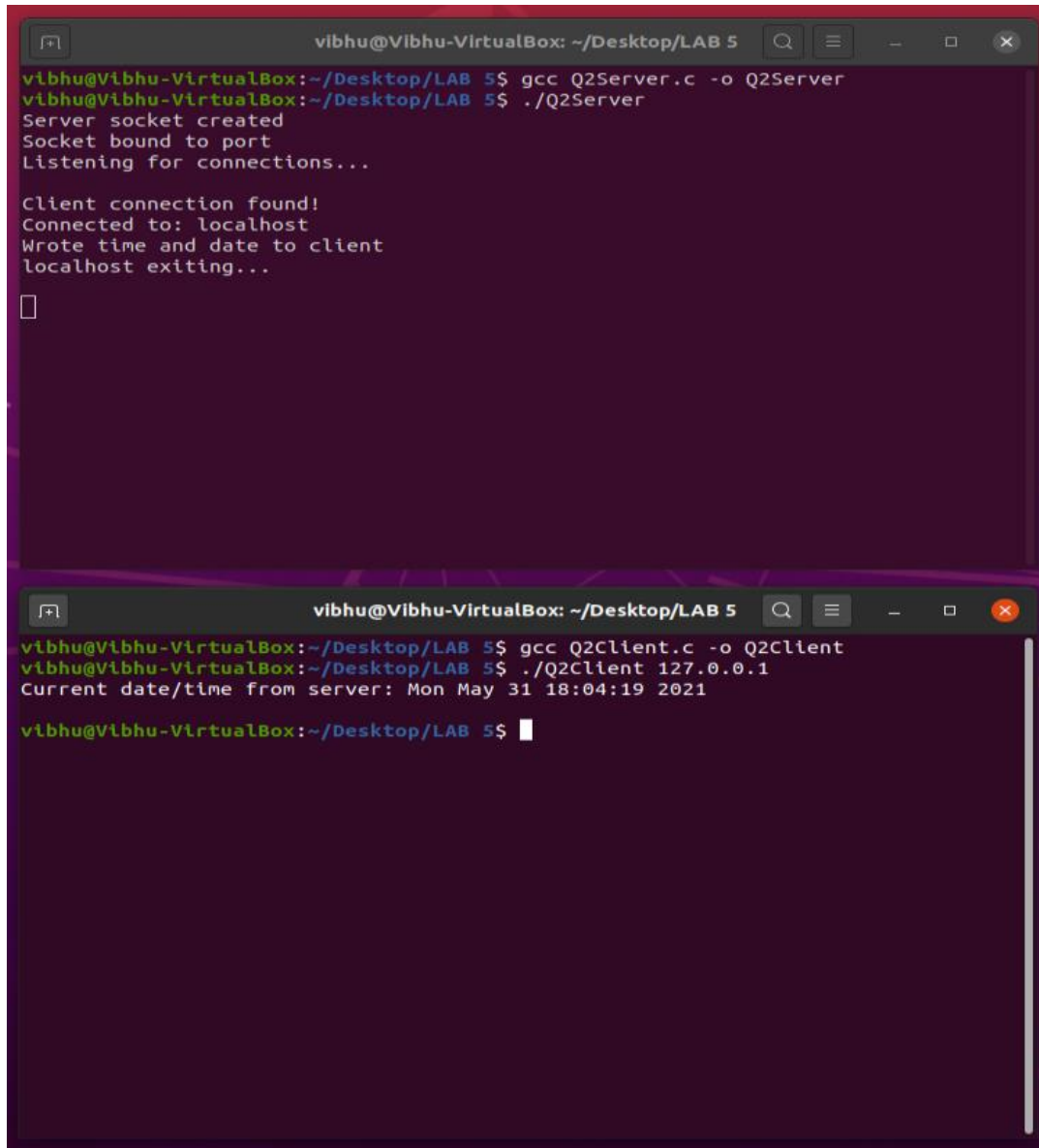
    char buffer[100];
    int bytesRead = read(sockfd, buffer, 100);
    if(bytesRead <= 0) {
        perror("read");
    }
}

```



```
        exit(1);
    }
    buffer[byteRead] = '\0';
    printf("Current date/time from server: %s\n", buffer);
    close(socketfd);
    exit(0);
}
```

OUTPUT SCREENSHOTS:



```
vibhu@Vibhu-VirtualBox: ~/Desktop/LAB 5
vibhu@Vibhu-VirtualBox:~/Desktop/LAB 5$ gcc Q2Server.c -o Q2Server
vibhu@Vibhu-VirtualBox:~/Desktop/LAB 5$ ./Q2Server
Server socket created
Socket bound to port
Listening for connections...

Client connection found!
Connected to: localhost
Wrote time and date to client
localhost exiting...

vibhu@Vibhu-VirtualBox:~/Desktop/LAB 5$ gcc Q2Client.c -o Q2Client
vibhu@Vibhu-VirtualBox:~/Desktop/LAB 5$ ./Q2Client 127.0.0.1
Current date/time from server: Mon May 31 18:04:19 2021

vibhu@Vibhu-VirtualBox:~/Desktop/LAB 5$
```

Q3) Implement a message transfer from client to server and vice versa process using UDP.

Ans 3)

Aim: To implement a message transfer from client to server and vice versa process using UDP.

Algorithm:

Server Side Algorithm:

- STEP 1: Start
- STEP 2: Declare the variables for the socket
- STEP 3: Specify the family, protocol, IP address and port number
- STEP 4: Create a socket using socket() function
- STEP 5: Bind the IP address and Port number
- STEP 6: Listen and accept the client's request for the connection
- STEP 7: Read the client's message
- STEP 8: Display the client's message
- STEP 9: Continue the chat
- STEP 10: Terminate the chat
- STEP 11: Close the socket
- STEP 12: Stop

Client Side Algorithm:

- STEP 1: Start
- STEP 2: Declare the variables for the socket
- STEP 3: Specify the family, protocol, IP address and port number
- STEP 4: Create a socket using socket() function
- STEP 5: Call the connect() function
- STEP 6: Read the input message
- STEP 7: Send the input message to the server
- STEP 8: Display the server's reply
- STEP 9: Continue the chat
- STEP 10: Terminate the chat
- STEP 11: Close the socket
- STEP 12: Stop

Source Code:

Q3Server.c

```
#include<stdio.h>
#include<netinet/in.h>
#include<sys/types.h>
#include<sys/socket.h>
```

```

#include<netdb.h>
#include<string.h>
#include<stdlib.h>
#define MAX 80
#define PORT 43454
#define SA struct sockaddr
void func(int sockfd)
{
    char buff[MAX];
    int n,clen;
    struct sockaddr_in cli;
    clen=sizeof(cli);
    for(;;)
    {
        bzero(buff,MAX);
        recvfrom(sockfd,buff,sizeof(buff),0,(SA *)&cli,&clen);
        printf("From client" ,%s " To client: ",buff);
        bzero(buff,MAX);
        n=0;
        while((buff[n++]=getchar())!='\n');
        sendto(sockfd,buff,sizeof(buff),0,(SA *)&cli,clen);
        if(strncmp("exit",buff,4)==0)
        {
            printf("Server Exit...\n");
            break;
        }
    }
}
int main()
{
    int sockfd;
    struct sockaddr_in servaddr;
    sockfd=socket(AF_INET,SOCK_DGRAM,0);
    if(sockfd==-1)
    {
        printf("socket creation failed...\n");
        exit(0);
    }
    else
        printf("Socket successfully created..\n");
    bzero(&servaddr,sizeof(servaddr));
    servaddr.sin_family=AF_INET;
    servaddr.sin_addr.s_addr=htonl(INADDR_ANY);
    servaddr.sin_port=htons(PORT);
    if((bind(sockfd,(SA *)&servaddr,sizeof(servaddr)))!=0)
    {
        printf("socket bind failed...\n");
        exit(0);
    }
    else
        printf("Socket successfully binded..\n");
    func(sockfd);
    close(sockfd);
}

```

Q3Client.c

```

#include<sys/socket.h>
#include<netdb.h>
#include<string.h>

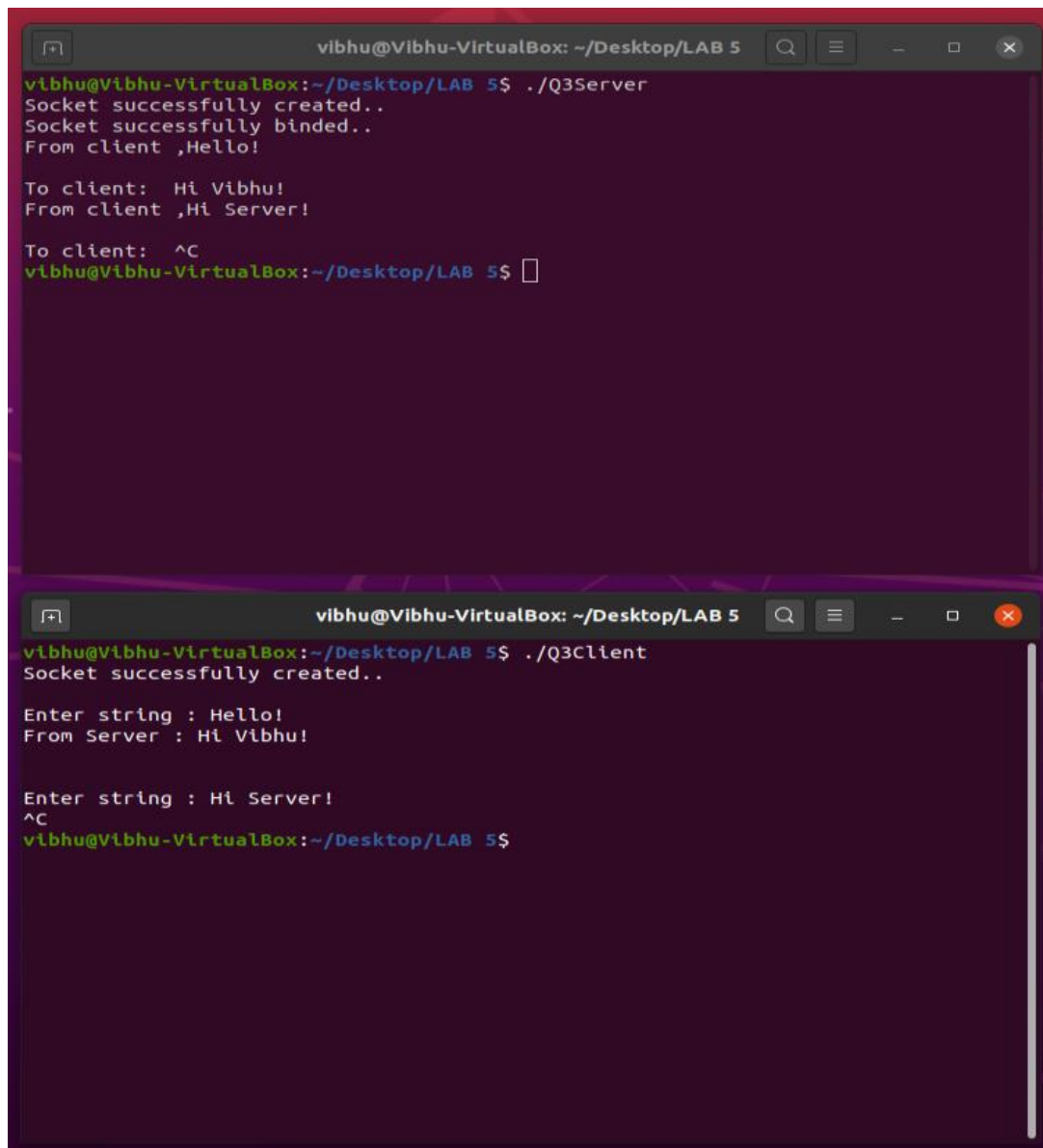
```

```

#include<stdlib.h>
#include<stdio.h>
#define MAX 80
#define PORT 43454
#define SA struct sockaddr
int main()
{
    char buff[MAX];
    int sockfd,len,n;
    struct sockaddr_in servaddr;
    sockfd=socket(AF_INET,SOCK_DGRAM,0);
    if(sockfd==-1)
    {
        printf("socket creation failed...\n");
        exit(0);
    }
    else
    printf("Socket successfully created..\n");
    bzero(&servaddr,sizeof(len));
    servaddr.sin_family=AF_INET;
    servaddr.sin_addr.s_addr=inet_addr("127.0.0.1");
    servaddr.sin_port=htons(PORT);
    len=sizeof(servaddr);
    for(;;)
    {
        printf("\nEnter string : ");
        n=0;
        while((buff[n++]=getchar())!='\n');
        sendto(sockfd,buff,sizeof(buff),0,(SA *)&servaddr,len);
        bzero(buff,sizeof(buff));
        recvfrom(sockfd,buff,sizeof(buff),0,(SA *)&servaddr,&len);
        printf("From Server : %s\n",buff);
        if(strncmp("exit",buff,4)==0)
        {
            printf("Client Exit...\n");
            break;
        }
    }
    close(sockfd);
}

```

OUTPUT SCREENSHOTS:



```
vibhu@Vibhu-VirtualBox: ~/Desktop/LAB 5
vibhu@Vibhu-VirtualBox:~/Desktop/LAB 5$ ./Q3Server
Socket successfully created..
Socket successfully binded..
From client ,Hello!

To client: Hi Vibhu!
From client ,Hi Server!

To client: ^C
vibhu@Vibhu-VirtualBox:~/Desktop/LAB 5$
```

```
vibhu@Vibhu-VirtualBox:~/Desktop/LAB 5$ ./Q3Client
Socket successfully created..

Enter string : Hello!
From Server : Hi Vibhu!

Enter string : Hi Server!
^C
vibhu@Vibhu-VirtualBox:~/Desktop/LAB 5$
```

Q4) Implement an Echo UDP server.

Ans 4)

Aim: To implement an Echo UDP server.

Algorithm:

Server Side Algorithm

STEP 1: Start

STEP 2: Declare the variables for the socket

STEP 3: Specify the family, protocol, IP address and port number

STEP 4: Create a socket using socket() function

STEP 5: Bind the IP address and Port number

STEP 6: Listen and accept the client's request for the connection

STEP 7: Read and Display the client's message

STEP 8: Stop

Client Side Algorithm

STEP 1: Start

STEP 2: Declare the variables for the socket

STEP 3: Specify the family, protocol, IP address and port number

STEP 4: Create a socket using socket() function

STEP 5: Call the connect() function

STEP 6: Read the input message

STEP 7: Send the input message to the server

STEP 8: Display the server's echo

STEP 9: Close the socket

STEP 10: Stop

Source Code:

Q4Server.c

```
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<unistd.h>
#include<netdb.h>
#include<stdio.h>
#include<string.h>
#include<arpa/inet.h>
#define MAXLINE 1024
int main(int argc,char **argv)
{
    int sockfd;
    int n;
    socklen_t len;
    char msg[1024];
    struct sockaddr_in servaddr,cliaddr;
    sockfd=socket(AF_INET,SOCK_DGRAM,0);
    bzero(&servaddr,sizeof(servaddr));
    servaddr.sin_family=AF_INET;
    servaddr.sin_addr.s_addr=INADDR_ANY;
    servaddr.sin_port=htons(5035);
    printf("\n\n Binded");
    bind(sockfd,(struct sockaddr*)&servaddr,sizeof(servaddr));
    printf("\n\n Listening...");
    for(;;)
    {
        printf("\n ");
        len=sizeof(cliaddr);
        n=recvfrom(sockfd,msg,MAXLINE,0,(struct sockaddr*)&cliaddr,&len);
        printf("\n Client's Message : %s\n",msg);
        if(n<6)
```

```

        perror("send error");
        sendto(sockfd,msg,n,0,(struct sockaddr*)&cliaddr,len);
    }
    return 0;
}

```

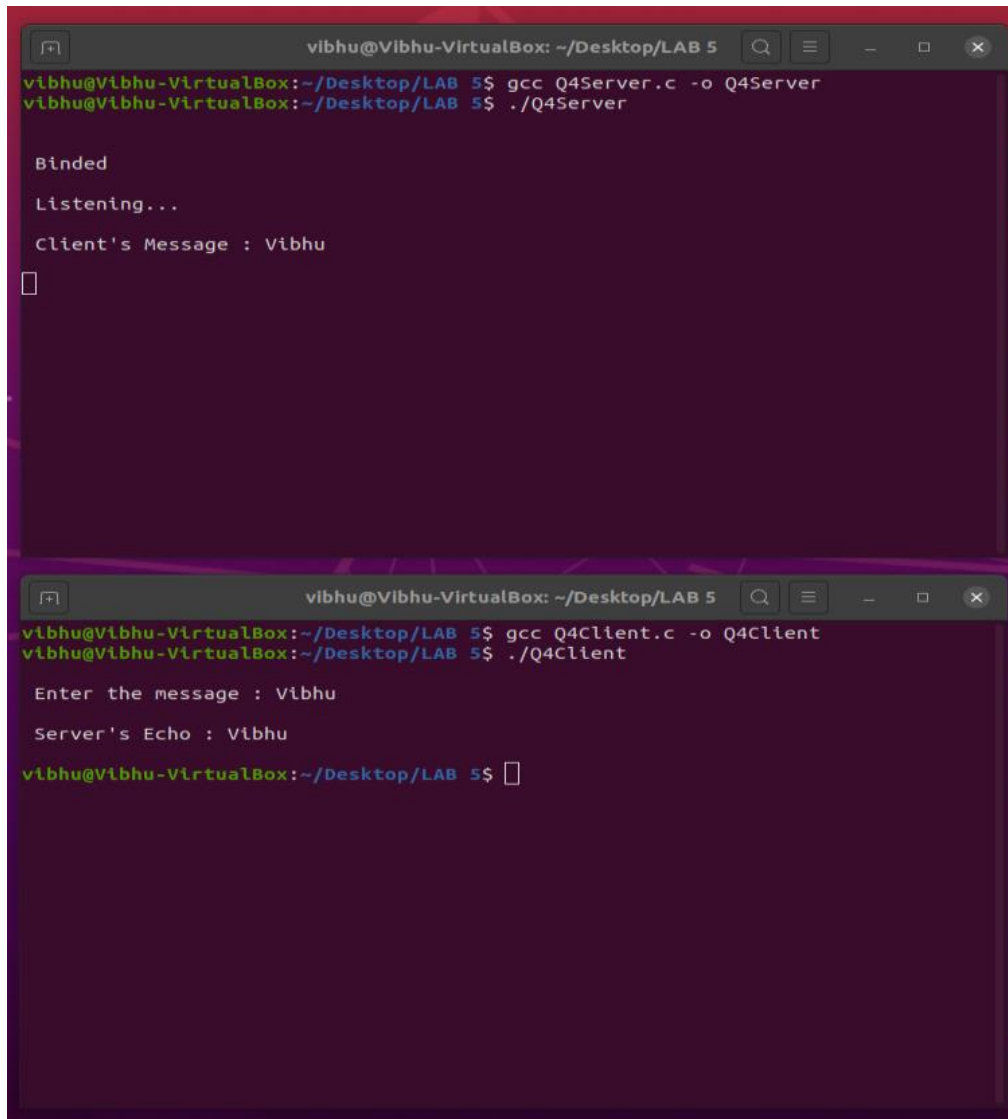
Q4Client.c

```

#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<string.h>
#include<arpa/inet.h>
#include<string.h>
#include<arpa/inet.h>
#include<stdio.h>
#define MAXLINE 1024
int main(int argc,char* argv[])
{
    int sockfd;
    int n;
    socklen_t len;
    char sendline[1024],recvline[1024];
    struct sockaddr_in servaddr;
    strcpy(sendline,"");
    printf("\n Enter the message : ");
    scanf("%s",sendline);
    sockfd=socket(AF_INET,SOCK_DGRAM,0);
    bzero(&servaddr,sizeof(servaddr));
    servaddr.sin_family=AF_INET;
    servaddr.sin_addr.s_addr=inet_addr("127.0.0.1");
    servaddr.sin_port=htons(5035);
    connect(sockfd,(struct sockaddr*)&servaddr,sizeof(servaddr));
    len=sizeof(servaddr);
    sendto(sockfd,sendline,MAXLINE,0,(struct sockaddr*)&servaddr,len);
    n=recvfrom(sockfd,recvline,MAXLINE,0,NULL,NULL);
    recvline[n]=0;
    printf("\n Server's Echo : %s\n\n",recvline);
    return 0;
}

```

OUTPUT SCREENSHOTS:



The image displays two terminal windows from a VirtualBox environment, showing the compilation and execution of a C program. The top window shows the compilation of Q4Server.c to Q4Server and its execution, which outputs 'Binded', 'Listening...', and 'Client's Message : Vibhu'. The bottom window shows the compilation of Q4Client.c to Q4Client and its execution, which prompts for a message, receives 'Vibhu', and outputs 'Server's Echo : Vibhu'.

```
vibhu@Vibhu-VirtualBox: ~/Desktop/LAB 5
vibhu@Vibhu-VirtualBox:~/Desktop/LAB 5$ gcc Q4Server.c -o Q4Server
vibhu@Vibhu-VirtualBox:~/Desktop/LAB 5$ ./Q4Server

Binded
Listening...
Client's Message : Vibhu
□
```

```
vibhu@Vibhu-VirtualBox: ~/Desktop/LAB 5
vibhu@Vibhu-VirtualBox:~/Desktop/LAB 5$ gcc Q4Client.c -o Q4Client
vibhu@Vibhu-VirtualBox:~/Desktop/LAB 5$ ./Q4Client

Enter the message : Vibhu
Server's Echo : Vibhu
vibhu@Vibhu-VirtualBox:~/Desktop/LAB 5$ □
```