## Develop client server-based TCP applications using UNIX socket programming functions.

### AIM:

To run the program of top\_echoserver and top\_echoclient.

### THEORY:

The echo server **receives data from its client and echoes it back**. The EchoClient example creates a socket, thereby getting a connection to the echo server.

### PROGRAM:

Top\_echoserver.c:

/\*Required Headers\*/

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

#include arpa/inet.h

Unistd.h

int main()

{

char str[100];

int listen\_fd, comm\_fd;

struct sockaddr\_in servaddr;

listen\_fd = socket(AF\_INET, SOCK\_STREAM, 0); bzero( &servaddr, sizeof(servaddr));

servaddr.sin\_family = AF\_INET; servaddr.sin\_addr.s\_addr = htons(INADDR\_ANY); servaddr.sin\_port = htons(22000);

bind(listen\_fd, (struct sockaddr \*) &servaddr, sizeof(servaddr)); listen(listen\_fd, 10);

comm\_fd = accept(listen\_fd, (struct sockaddr\*) NULL, NULL);

while(1)

{

bzero( str, 100); read(comm\_fd,str,100); printf("Echoing back - %s",str); write(comm\_fd, str, strlen(str)+1);

}

}

Top\_echoclient.c: #include <sys/types.h> #include <sys/socket.h> #include <netdb.h> #include <stdio.h> #include<string.h>

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

{

int sockfd,n;

char sendline[100]; char recvline[100];

struct sockaddr\_in servaddr;

sockfd=socket(AF\_INET,SOCK\_STREAM,0); bzero(&servaddr,sizeof servaddr);

servaddr.sin\_family=AF\_INET; servaddr.sin\_port=htons(22000);

inet\_pton(AF\_INET,"127.0.0.1",&(servaddr.sin\_addr));

connect(sockfd,(struct sockaddr \*)&servaddr,sizeof(servaddr));

while(1)

{

bzero( sendline, 100);

bzero( recvline, 100);

fgets(sendline,100,stdin); /\*stdin = 0 , for standard input \*/

write(sockfd,sendline,strlen(sendline)+1); read(sockfd,recvline,100);

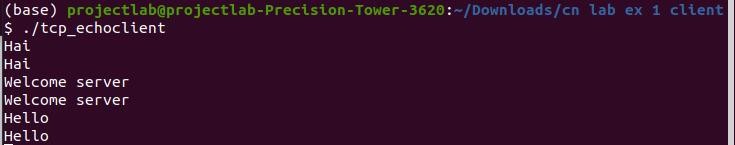
printf("%s",recvline);

}

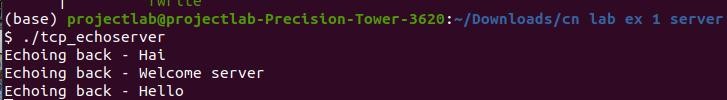
}

OUTPUT:

top\_echoclient.c



Top\_echoserver.c:



## 2.Develop client server based UDP applications using UNIX socket programming functions.

### AIM:

To run the program of udp\_client and udp\_server.

### THEORY:

In UDP, the client does not form a connection with the server like in TCP and instead just sends a datagram. Similarly, the server need not accept a connection and just waits for datagrams to arrive. Datagrams upon arrival contain the address of the sender which the server uses to send data to the correct client.

### PROGRAM:

udp\_server.c: #include<sys/types.h> #include<sys/socket.h> #include<stdio.h> #include<netinet/in.h>

#define MAX 100

#define SERPORT 1090 #define SA struct sockaddr

void str\_echo(FILE\*,int,SA\*,socklen\_t);

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

{

int sockfd;

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=htonl(0); servaddr.sin\_port=htons(SERPORT); bind(sockfd,(SA\*)&servaddr,sizeof(servaddr)); str\_echo(stdin,sockfd,(SA\*)&cliaddr,sizeof(cliaddr)); exit(0);

}

void str\_echo(FILE\* fp,int sockfd,SA\* cliaddr,socklen\_t clilen)

{

char msg[MAX],send[MAX]; int n;

while(1)

{

if((n=recvfrom(sockfd,msg,MAX,0,cliaddr,&clilen))>0)

{

msg[n]='\0'; printf("Client msg : "); fputs(msg,stdout); printf("SERVER msg : "); fgets(msg,MAX,fp);

sendto(sockfd,msg,strlen(msg),0,cliaddr,clilen);

}

}

}

Udp\_client.c: #include<sys/types.h> #include<sys/socket.h> #include<stdio.h> #include<netinet/in.h>

#define MAX 100

#define SERPORT 1090 #define SA struct sockaddr

void str\_cli(FILE\*,int,SA\*,socklen\_t);

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

{

int sockfd;

struct sockaddr\_in servaddr; 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(SERPORT); printf("Client msg : ");

str\_cli(stdin,sockfd,(SA\*)&servaddr,sizeof(servaddr)); exit(0);

}

void str\_cli(FILE\* fp,int sockfd,SA\* seraddr,socklen\_t len)

{

char msg[MAX],rcv[MAX]; int n;

while(fgets(msg,MAX,fp)!=NULL)

{

sendto(sockfd,msg,MAX,0,seraddr,len); if((n=recvfrom(sockfd,rcv,MAX,0,seraddr,&len))>0)

{

rcv[n]='\0'; fputs(rcv,stdout);

}

printf("Client msg : ");

}

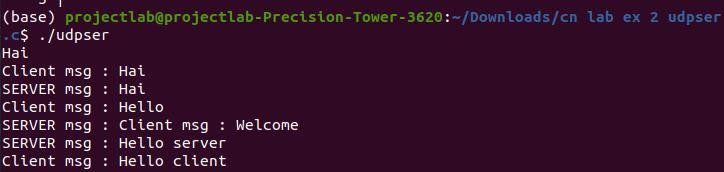
}

### OUTPUT:

Udpclient.c:



Udp\_server.c:



### Demonstrate the working of network tools such as Ping, TCP Dump, Traceroute, Netstat, Ipconfig.

**Ping:**

The ping command is a general utility which is used for checking whether any network is present and if a host is attainable. We can test if the server is up and executing using this command. Also, it helps several connectivity issues with troubleshooting.

Graphical user interface, text

Description automatically generated

**Tcpdump:**

Tcpdump is a packet analyzer that is launched from the command line. It can be used to analyze network traffic by intercepting and displaying packets that are being created or received by the computer it's running on. It runs on Linux and most UNIX-type operating systems.

Text

Description automatically generated

Text

Description automatically generated

**Traceroute:**

The traceroute command attempts to trace the route an IP packet follows to an Internet host by launching UDP probe packets with a small maximum time-to-live (Max\_ttl variable), then listening for an ICMP TIME\_EXCEEDED response from gateways along the way.

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with medium confidence

**Netstat:**

The network statistics (netstat ) command is a networking tool used for troubleshooting and configuration, that can also serve as a monitoring tool for connections over the network. Both incoming and outgoing connections, routing tables, port listening, and usage statistics are common uses for this command.

Text

Description automatically generated

A screen shot of a computer

Description automatically generated with low confidence

**Ipconfig:**

Ipconfig displays useful information such as IP address, subnet mask and the default gateway for all of the different network connections in the computer.

Text

Description automatically generated

Text

Description automatically generated

1. **Analyze the network traffic using Wireshark tool/Packet tracer tool.**

### TUTORIAL FOR WIRESHARK:

Graphical user interface, text, application

Description automatically generated

FILTERING OF HHTP:

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

COLORING RULES:

Graphical user interface, text, application

Description automatically generated

TIME SEQUENCE (STEVENS)

Chart, line chart

Description automatically generated

TIME SEQUENCE(TCP TRACES)

Chart, line chart

Description automatically generated

Wireshark conversations

Graphical user interface, application, table

Description automatically generated