ASSIGNMENT-5

Name: Sayantan Roy

Roll Number: GCECTB-R18-3025

Subject: Computer Network Lab

Dept: CSE

Year:3rd

ASSIGNMENT -5

TCP ECHO CLIENT SERVER PROGRAM

IN C

TCP ECHO CLIENT SERVER PROGRAM

IN C

SYSTEM USED :

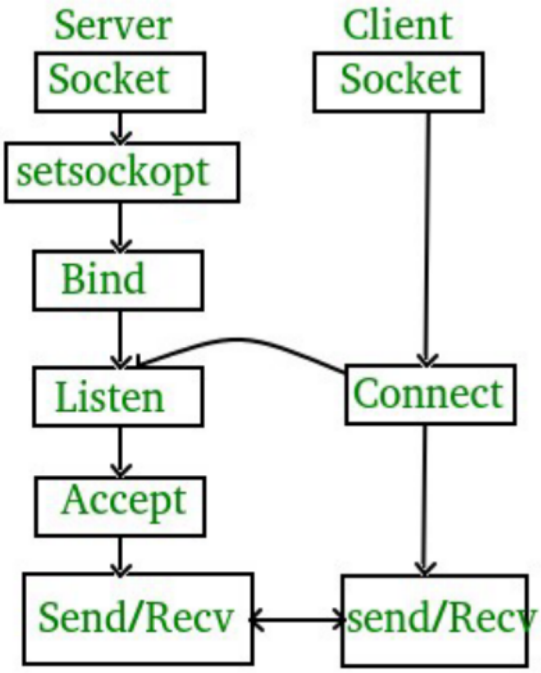
OPERATING SYSTEM : kali linux 2020.1
KERNEL: 4.19.153
GUI: DEFAULT XSESSION (XFCE4)
TEXT EDITOR :NANO

system owned and project done by :Sayantan Roy

TCP ECHO CLIENT SERVER PROGRAM IN C

If we are creating a connection between client and server using TCP then it has few functionality like, TCP is suited for applications that require high reliability, and transmission time is relatively less critical. It is used by other protocols like HTTP, HTTPs, FTP, SMTP, Telnet. TCP rearranges data packets in the order specified. There is absolute guarantee that the data transferred remains intact and arrives in the same order in which it was sent. TCP does Flow Control and requires three packets to set up a socket connection, before any user data can be sent. TCP handles reliability and congestion control. It also does error checking and error recovery. Erroneous packets are retransmitted from the source to the destination.

The entire process can be broken down into following steps:



The entire process can be broken down into following steps:

TCP Server –

1.using **create()**, Create TCP socket.
2.using **bind()**, Bind the socket to server address.
3.using **listen()**, put the server socket in a passive mode, where it waits for the client to approach the server to make a connection
4.using **accept()**, At this point, connection is established between client and server, and they are ready to transfer data.
5.Go back to Step 3.

TCP Client –

1.Create TCP socket.
2.connect newly created client socket to server.

Arguments :

domain –Specifies the communication
domain (AF_INET for IPv4/ AF_INET6 for IPv6)

type –Type of socket to be created
(SOCK_STREAM for TCP / SOCK_DGRAM for UDP)

protocol –Protocol to be used by socket.
0 means use default protocol for the address family.

bind : assigns address to the unbound socket

sockfd –File descriptor of socket.

addr –Structure in which address to be binded to is specified

Functions:

socket(int domain, int type, int protocol)

Creates an unbound socket in the specified domain.
Returns socket file descriptor.

AIM:

To write a program for TCP echo client server.

ALGORITHM:

SERVER:

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: Close the socket
STEP 10: Stop

CLIENT:

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:

FILENAME: tcpechoserver.c

SERVER:

```
#include<stdio.h>
#include<netinet/in.h>
#include<netdb.h>
#define SERV_TCP_PORT 5035
int main(int argc,char**argv)
{
	int sockfd,newsockfd,clength;
	struct sockaddr_in serv_addr,cli_addr;
	char buffer[4096];
	sockfd=socket(AF_INET,SOCK_STREAM,0);
	serv_addr.sin_family=AF_INET;
	serv_addr.sin_addr.s_addr=INADDR_ANY;
	serv_addr.sin_port=htons(SERV_TCP_PORT);
	printf("\nStart");
	bind(sockfd,(struct sockaddr*)&serv_addr,sizeof(serv_addr));
	printf("\nListening...");
	printf("\n");
	listen(sockfd,5);
	clength=sizeof(cli_addr);
	newsockfd=accept(sockfd,(struct sockaddr*)&cli_addr,&clength);
	printf("\nAccepted");
	printf("\n");
	read(newsockfd,buffer,4096);
	printf("\nClient message:%s",buffer);
	write(newsockfd,buffer,4096);
	printf("\n");
	close(sockfd);
	return 0;
}
```

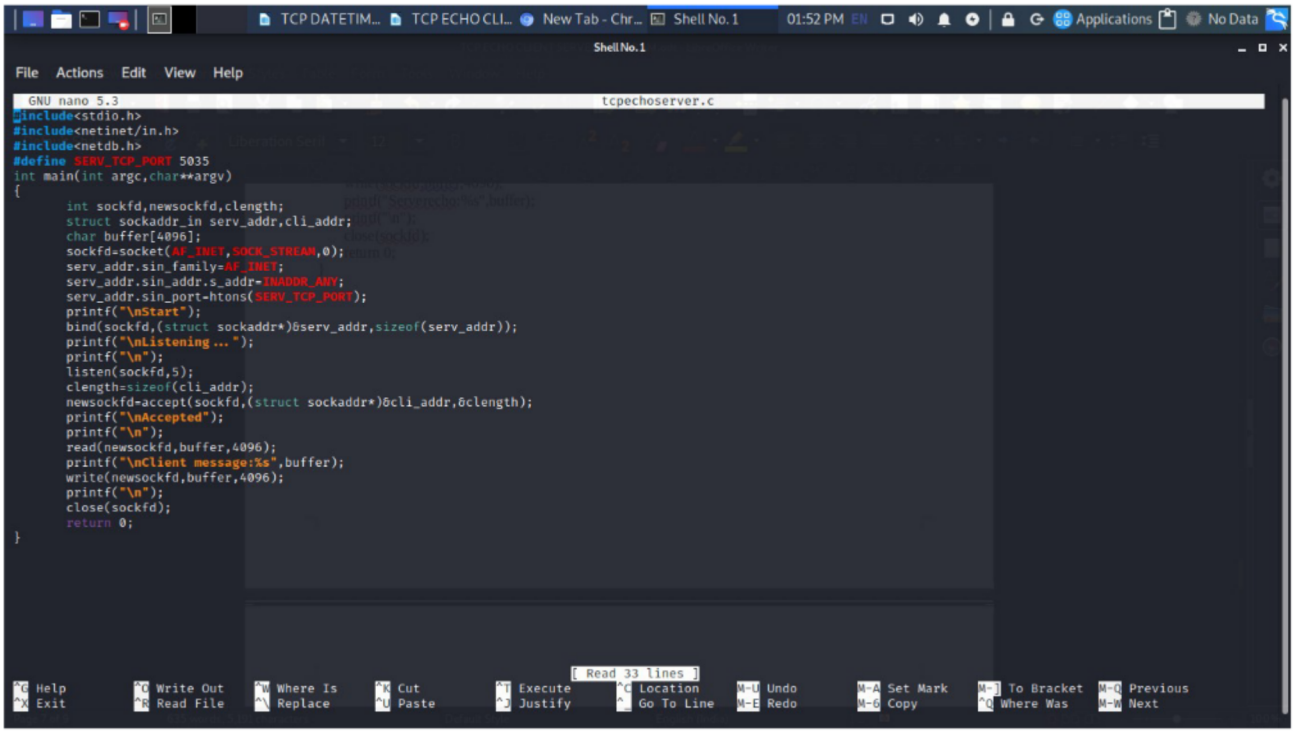
FILENAME:tcpechoclient.c

CLIENT:

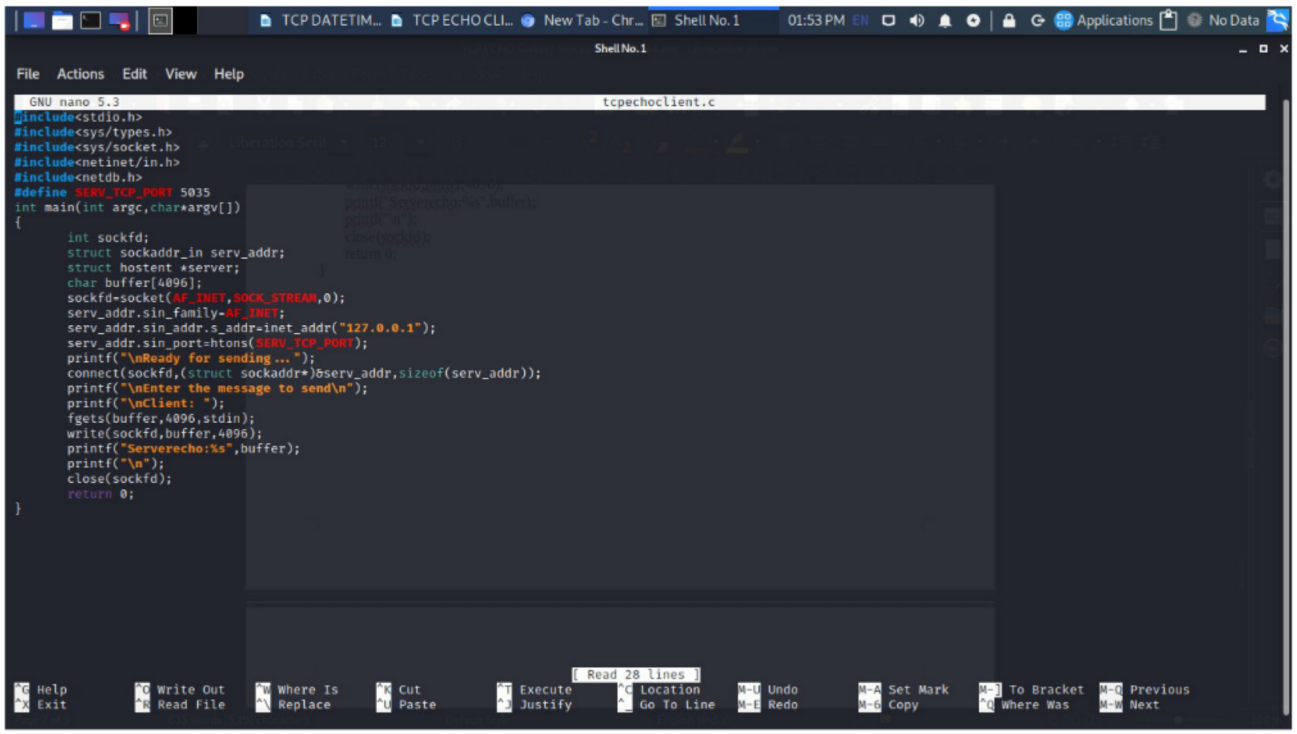
```
#include<stdio.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<netdb.h>
#define SERV_TCP_PORT 5035
int main(int argc,char*argv[])
{
	int sockfd;
	struct sockaddr_in serv_addr;
	struct hostent *server;
	char buffer[4096];
	sockfd=socket(AF_INET,SOCK_STREAM,0);
	serv_addr.sin_family=AF_INET;
	serv_addr.sin_addr.s_addr=inet_addr("127.0.0.1");
	serv_addr.sin_port=htons(SERV_TCP_PORT);
	printf("\nReady for sending...");
	connect(sockfd,(struct sockaddr*)&serv_addr,sizeof(serv_addr));
	printf("\nEnter the message to send\n");
	printf("\nClient: ");
	fgets(buffer,4096,stdin);
	write(sockfd,buffer,4096);
	printf("Serverecho:%s",buffer);
	printf("\n");
	close(sockfd);
	return 0;
}
```

output :

```
sayantan@localhost:~$ gcc tcpechoserver.c -o tcpechoserver

sayantan@localhost:~$ gcc tcpechoclient.c -o tcpechoclient

sayantan@localhost:~$ ./tcpechoserver

Start
Listening...

Accepted

Client message:hi ! how are you ?

sayantan@localhost:~$ ./tcpechoclient

Ready for sending...
Enter the message to send

Client: hi ! how are you ?
Serverecho:hi ! how are you ?

sayantan@localhost:~$
```



filename:tcpechoserver.c



filename:tcpechoclient.c

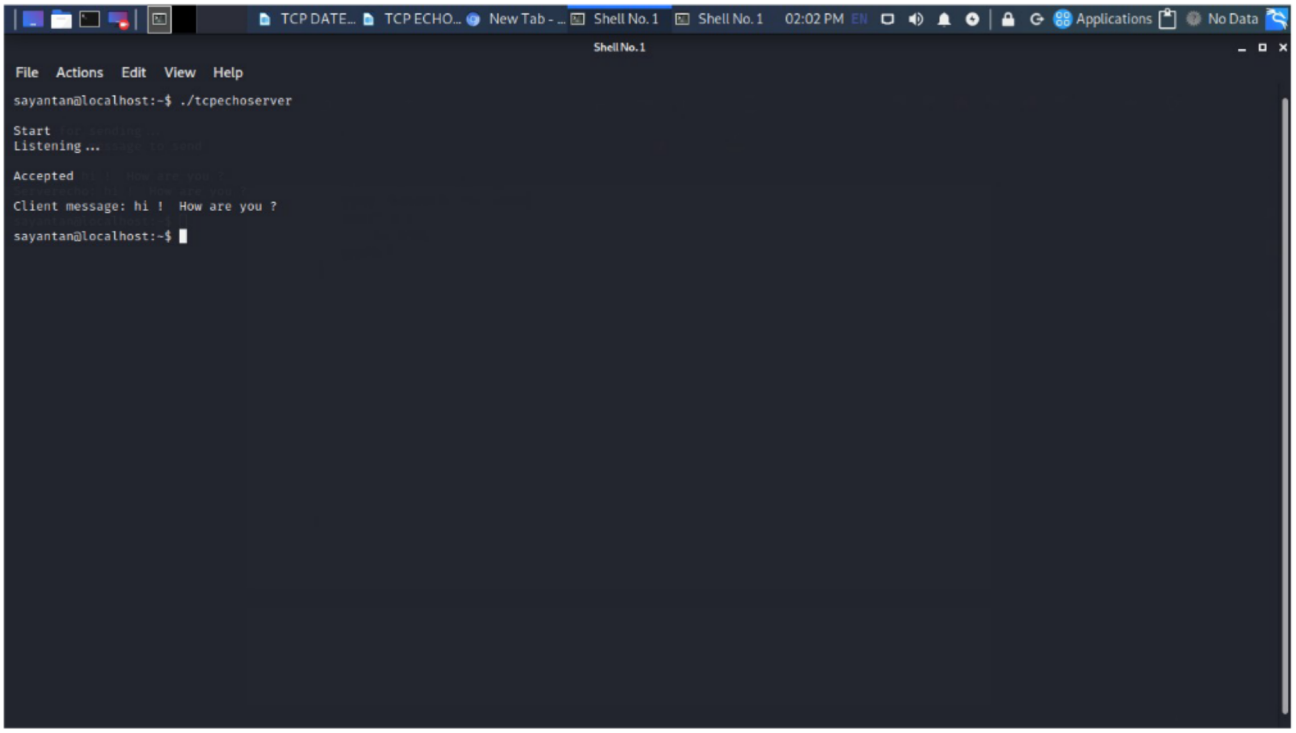


fig : TCP echo server is running

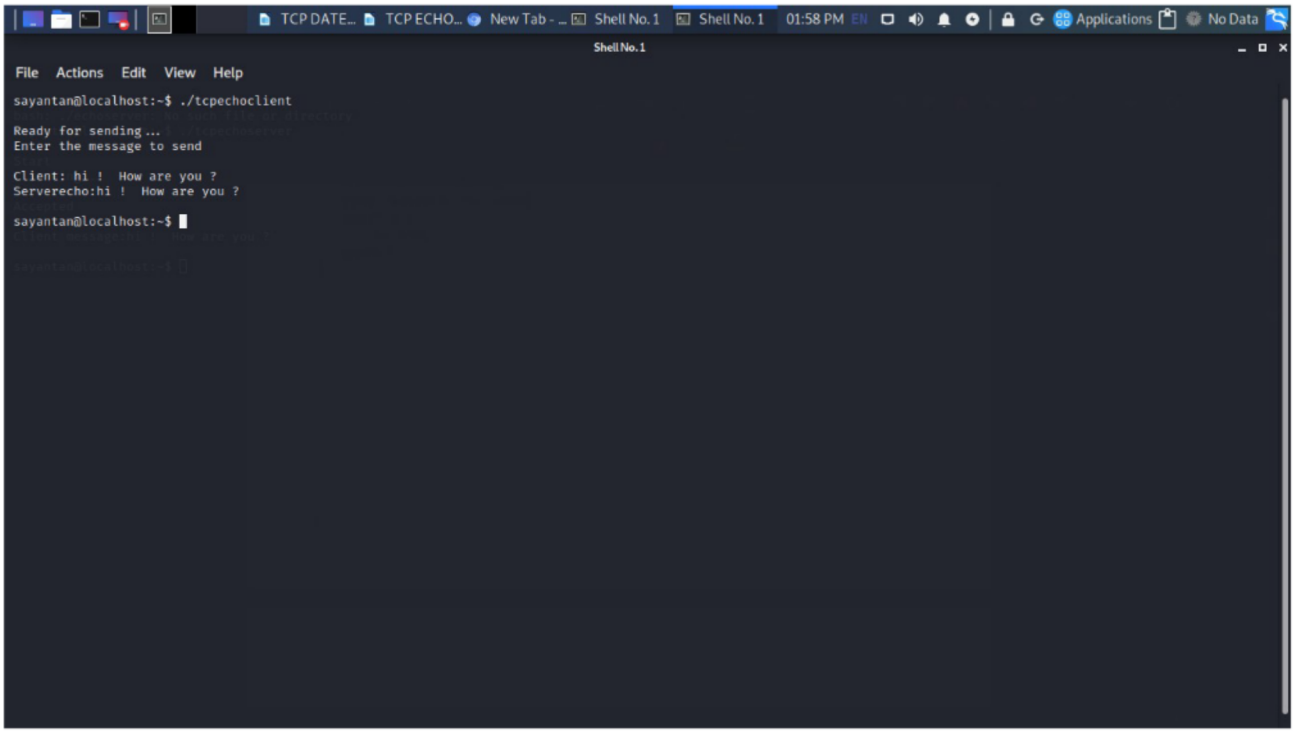


fig : TCP echo client is running , client's message got echo from server