SEC : A   CN LAB: 01   BATCH : A3

REG: 190905513   CSE 3113   NAME: MOHAMMAD DANISH EQBAL

SAMPLE PROGRAM:

1. Write a C program to demonstrate the working of UDP echo Client/Server.

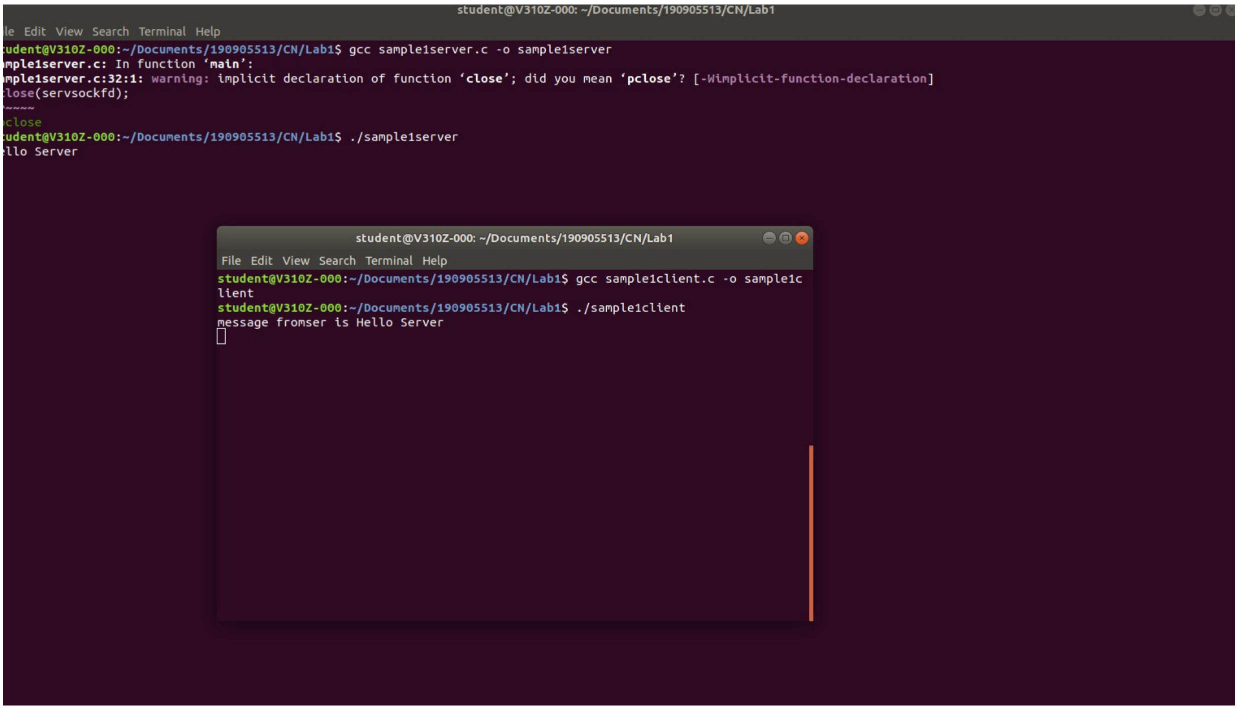
Program:

```
// Server program for udp connection
#include <stdio.h>
#include <strings.h>
#include <sys/types.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include<netinet/in.h>
#define PORT 5000
#define MAXLINE 1000

// Server code
int main()
{
	char buffer[100];
	int servsockfd, len,n;
	struct sockaddr_in servaddr, cliaddr;
	bzero(&servaddr, sizeof(servaddr));

	// Create a UDP Socket
	servsockfd = socket(AF_INET, SOCK_DGRAM, 0);
	servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
	servaddr.sin_port = htons(PORT);
	servaddr.sin_family = AF_INET;

	// bind server address to socket descriptor
	bind(servsockfd,         (struct         sockaddr*)&servaddr,
sizeof(servaddr));
```

```
	//receive the datagram
	len = sizeof(cliaddr);
	n = recvfrom(servsockfd, buffer, sizeof(buffer),0, (struct
sockaddr*)&cliaddr,&len);
	buffer[n] = '\0';
	puts(buffer);
//Echoing back to the client
                         sendto(servsockfd, buffer, n, 0, (struct
sockaddr*)&cliaddr, sizeof(cliaddr));
                  getchar();

	// close the descriptor
	close(sockfd);
}

// Udp client driver program
#include <stdio.h>
#include <strings.h>
#include <sys/types.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include<netinet/in.h>
#include<unistd.h>
#include<stdlib.h>

#define PORT 5000
#define MAXLINE 1000

// Driver code
int main()
{
	char buffer[100];
	char *message = "Hello Server";
	int sockfd, n,len;
	struct sockaddr_in servaddr, cliaddr;
```

```
    // clear servaddr
    bzero(&servaddr, sizeof(servaddr));
    servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
    servaddr.sin_port = htons(PORT);
    servaddr.sin_family = AF_INET;

    // create datagram socket
    sockfd = socket(AF_INET, SOCK_DGRAM, 0);
    sendto(sockfd, message, MAXLINE, 0, (struct
sockaddr*)&servaddr, sizeof(servaddr));
    len=sizeof(cliaddr);
        // waiting for response
    n=recvfrom(sockfd, buffer, sizeof(buffer), 0, (struct
sockaddr*)&cliaddr,&len );
       buffer[n]='\0';
    printf("message fromser is %s \n",buffer);
       getchar();

    // close the descriptor
    close(sockfd);
}
```

Output:



student@V310Z-000: ~/Documents/190905513/CN/Lab1
ile Edit View Search Terminal Help

```
udent@V310Z-000:~/Documents/190905513/CN/Lab1$ gcc sample1server.c -o sample1server
mple1server.c: In function ‘main’:
mple1server.c:32:1: warning: implicit declaration of function ‘close’; did you mean ‘pclose’? [-Wimplicit-function-declaration]
lose(servsockfd);
~~~~~
close
udent@V310Z-000:~/Documents/190905513/CN/Lab1$ ./sample1server
llo Server
```

student@V310Z-000: ~/Documents/190905513/CN/Lab1
File Edit View Search Terminal Help

```
student@V310Z-000:~/Documents/190905513/CN/Lab1$ gcc sample1client.c -o sample1c
lient
student@V310Z-000:~/Documents/190905513/CN/Lab1$ ./sample1client
message fromser is Hello Server
```

2. Write a C program to demonstrate the working of TCP client server as follows: After connection set up client send a message. Server will reply to this. If server decides to close the program then it will send a message exit to client then closes itself. Client will close after receiving this message.. (Note: In each program there is a function that handles the client and server function and main program is responsible for socket creation and connection setup.)

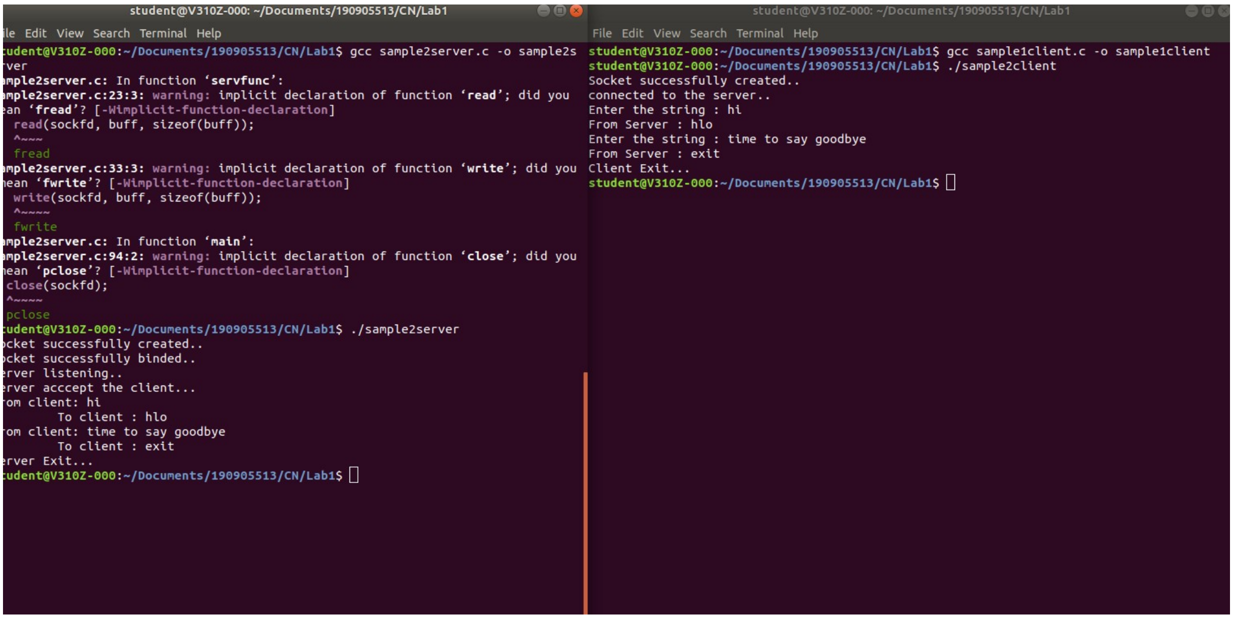
Program:

```
// TCP Server program
#include <stdio.h>
#include <netdb.h>
#include <netinet/in.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/types.h>
#define MAX 80
#define PORT 8080
#define SA struct sockaddr

// Function designed for chat between client and server.
void servfunc(int sockfd)
{
     char buff[MAX];
     int n;
     // infinite loop for chat
     for (;;) {
          bzero(buff, MAX);

          // read the message from client and copy it in buffer
          read(sockfd, buff, sizeof(buff));
          // print buffer which contains the client contents
          printf("From client: %s\t To client : ", buff);

          bzero(buff, sizeof(buff));
// Read server message from keyboard in the buffer
          n=0;
```

```
        while ((buff[n++] = getchar()) != '\n')
            ;
// and send that buffer to client
        write(sockfd, buff, sizeof(buff));

        // if msg contains "Exit" then server exit and session
ended.
        if (strncmp("exit", buff, 4) == 0) {
            printf("Server Exit...\n");
            break;
        }
    }
}

// Driver function
int main()
{
    int sockfd, connfd, len;
    struct sockaddr_in servaddr, cli;
    // socket create and verification
    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd == -1) {
        printf("socket creation failed...\n");
        exit(0);
    }
    else
        printf("Socket successfully created..\n");
    bzero(&servaddr, sizeof(servaddr));
    // assign IP, PORT
    servaddr.sin_family = AF_INET;
    servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
    servaddr.sin_port = htons(PORT);
    // Binding newly created socket to given IP and verification
    if ((bind(sockfd, (SA*)&servaddr, sizeof(servaddr))) != 0) {
        printf("socket bind failed...\n");
        exit(0);
    }
    else
        printf("Socket successfully binded..\n");
    // Now server is ready to listen and verification
    if ((listen(sockfd, 5)) != 0) {
        printf("Listen failed...\n");
        exit(0);
    }
    else
        printf("Server listening..\n");
    len = sizeof(cli);

    // Accept the data packet from client and verification
    connfd = accept(sockfd, (SA*)&cli, &len);
    if (connfd < 0) {
        printf("server acccept failed...\n");
        exit(0);
    }
    else
        printf("server acccept the client...\n");
    // Function for chatting between client and server
    servfunc(connfd);
    // After sending exit message close the socket
    close(sockfd);
}

//TCP Client program
#include <netdb.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#define MAX 80
#define PORT 8080
```

```
#define SA struct sockaddr
void clifunc(int sockfd)
{
    char buff[MAX];
    int n;
    for (;;) {
        bzero(buff, sizeof(buff));
        printf("Enter the string : ");
        n = 0;
        while ((buff[n++] = getchar()) != '\n')
            ;
        write(sockfd, buff, sizeof(buff));
        bzero(buff, sizeof(buff));
        read(sockfd, buff, sizeof(buff));
        printf("From Server : %s", buff);
        if ((strncmp(buff, "exit", 4)) == 0) {
            printf("Client Exit...\n");
            break;
        }
    }
}

int main()
{
    int sockfd, connfd;
    struct sockaddr_in servaddr, cli;

    // socket create and verification
    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd == -1) {
        printf("socket creation failed...\n");
        exit(0);
    }
    else
        printf("Socket successfully created..\n");
    bzero(&servaddr, sizeof(servaddr));

    // assign IP, PORT
    servaddr.sin_family = AF_INET;
    servaddr.sin_addr.s_addr = inet_addr("127.0.0.1");
    servaddr.sin_port = htons(PORT);

    // connect the client socket to server socket
    if (connect(sockfd, (SA*)&servaddr, sizeof(servaddr)) != 0) {
        printf("connection with the server failed...\n");
        exit(0);
    }
    else
        printf("connected to the server..\n");

    // function for client
    clifunc(sockfd);

    // close the socket
    close(sockfd);
}
```

Output:


student@V310Z-000: ~/Documents/190905513/CN/Lab1
ile Edit View Search Terminal Help
student@V310Z-000: ~/Documents/190905513/CN/Lab1
File Edit View Search Terminal Help

```
tudent@V310Z-000:~/Documents/190905513/CN/Lab1$ gcc sample2server.c -o sample2s
rver
mple2server.c: In function ‘servfunc’:
mple2server.c:23:3: warning: implicit declaration of function ‘read’; did you
ean ‘fread’? [-Wimplicit-function-declaration]
  read(sockfd, buff, sizeof(buff));
  ^~~~
  fread
mple2server.c:33:3: warning: implicit declaration of function ‘write’; did you
ean ‘fwrite’? [-Wimplicit-function-declaration]
  write(sockfd, buff, sizeof(buff));
  ^~~~~
  fwrite
mple2server.c: In function ‘main’:
mple2server.c:94:2: warning: implicit declaration of function ‘close’; did you
ean ‘pclose’? [-Wimplicit-function-declaration]
 close(sockfd);
 ^~~~~
 pclose
tudent@V310Z-000:~/Documents/190905513/CN/Lab1$ ./sample2server
ocket successfully created..
ocket successfully binded..
erver listening..
erver acccept the client...
rom client: hi
         To client : hlo
rom client: time to say goodbye
         To client : exit
erver Exit...
tudent@V310Z-000:~/Documents/190905513/CN/Lab1$
```

```
student@V310Z-000:~/Documents/190905513/CN/Lab1$ gcc sample1client.c -o sample1client
student@V310Z-000:~/Documents/190905513/CN/Lab1$ ./sample2client
Socket successfully created..
connected to the server..
Enter the string : hi
From Server : hlo
Enter the string : time to say goodbye
From Server : exit
Client Exit...
student@V310Z-000:~/Documents/190905513/CN/Lab1$
```

EXERCISE PROBLEMS:

1. Write a UDP client-server program where client sends rows of a matrix to the server combines them together as a two dimensional matrix and display the same.

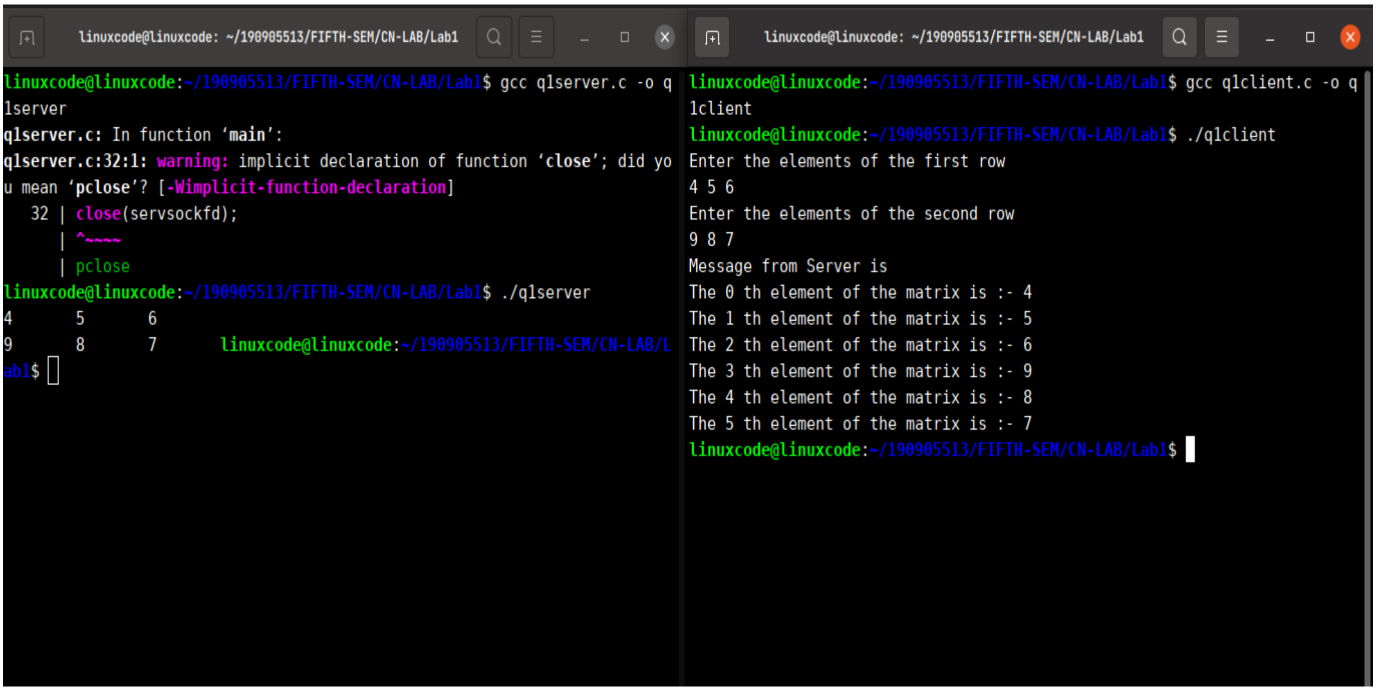
Program:

```
//Server side program
#include <stdio.h>
#include <strings.h>
#include <sys/types.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include <netinet/in.h>
#define PORT 5000
#define MAXLINE 1000
void main(){
int buffer[100];
int servsockfd,i,len,n;
struct sockaddr_in servaddr, cliaddr;
bzero(&servaddr, sizeof(servaddr));
// Create a UDP Socket
servsockfd = socket(AF_INET, SOCK_DGRAM, 0);
servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
servaddr.sin_port = htons(PORT);
servaddr.sin_family = AF_INET;
// bind server address to socket descriptor
bind(servsockfd, (struct sockaddr*)&servaddr, sizeof(servaddr));
//receive the datagram
len = sizeof(cliaddr);
n = recvfrom(servsockfd, buffer, sizeof(buffer),0,(struct
sockaddr*)&cliaddr,&len);
for(i=0;i<3;i++)printf("%d\t",buffer[i]);
printf("\n");
for(i=3;i<6;i++)
printf("%d\t",buffer[i]);
//Echoing back to the client
```

```
sendto(servsockfd, buffer, n, 0,(struct sockaddr*)&cliaddr,
sizeof(cliaddr));
// close the descriptor
close(servsockfd);
}

//Client side program
#include <stdio.h>
#include <strings.h>
#include <sys/types.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <unistd.h>
#include <stdlib.h>
#define PORT 5000
#define MAXLINE 1000
void main(){
int buffer[100];
int sockfd, n,len;
struct sockaddr_in servaddr, cliaddr;
//using a square matrix of 3*2
printf("Enter the elements of the first row\n");int a ,b, c;
scanf("%d %d %d",&a,&b, &c);
printf("Enter the elements of the second row \n");
int d ,e, f;
scanf("%d%d%d",&d ,&e, &f);
// clear servaddr
bzero(&servaddr, sizeof(servaddr));
servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
servaddr.sin_port = htons(PORT);
servaddr.sin_family = AF_INET;
int message[6];
message[0]=a;
message[1]=b;
```

```
message[2]=c;
message[3]=d;
message[4]=e;
message[5]=f;
// create datagram socket
sockfd = socket(AF_INET, SOCK_DGRAM, 0);
sendto(sockfd, message, MAXLINE, 0, (struct sockaddr*)&servaddr, sizeof(servaddr));
len=sizeof(cliaddr);
// waiting for response
n=recvfrom(sockfd, buffer, sizeof(buffer), 0, (struct sockaddr*)&cliaddr,&len );
buffer[n]='\0';
printf("Message from Server is \n");
//Just chck if it gives correct outpur or not , connection is already esablished
for(int i=0;i<6;i++){
//hardcoded till 3
printf("The %d th element of the matrix is :- %d \n",i,(buffer[i]));}
// close the descriptor
close(sockfd);
}
```

Output:



```
linuxcode@linuxcode:~/190905513/FIFTH-SEM/CN-LAB/Lab1$ gcc q1server.c -o q
1server
q1server.c: In function ‘main’:
q1server.c:32:1: warning: implicit declaration of function ‘close’; did yo
u mean ‘pclose’? [-Wimplicit-function-declaration]
   32 | close(servsockfd);
      | ^~~~~
      | pclose
linuxcode@linuxcode:~/190905513/FIFTH-SEM/CN-LAB/Lab1$ ./q1server
4       5       6
9       8       7       linuxcode@linuxcode:~/190905513/FIFTH-SEM/CN-LAB/L
ab1$ 
```

```
linuxcode@linuxcode:~/190905513/FIFTH-SEM/CN-LAB/Lab1$ gcc q1client.c -o q
1client
linuxcode@linuxcode:~/190905513/FIFTH-SEM/CN-LAB/Lab1$ ./q1client
Enter the elements of the first row
4 5 6
Enter the elements of the second row
9 8 7
Message from Server is
The 0 th element of the matrix is :- 4
The 1 th element of the matrix is :- 5
The 2 th element of the matrix is :- 6
The 3 th element of the matrix is :- 9
The 4 th element of the matrix is :- 8
The 5 th element of the matrix is :- 7
linuxcode@linuxcode:~/190905513/FIFTH-SEM/CN-LAB/Lab1$ 
```

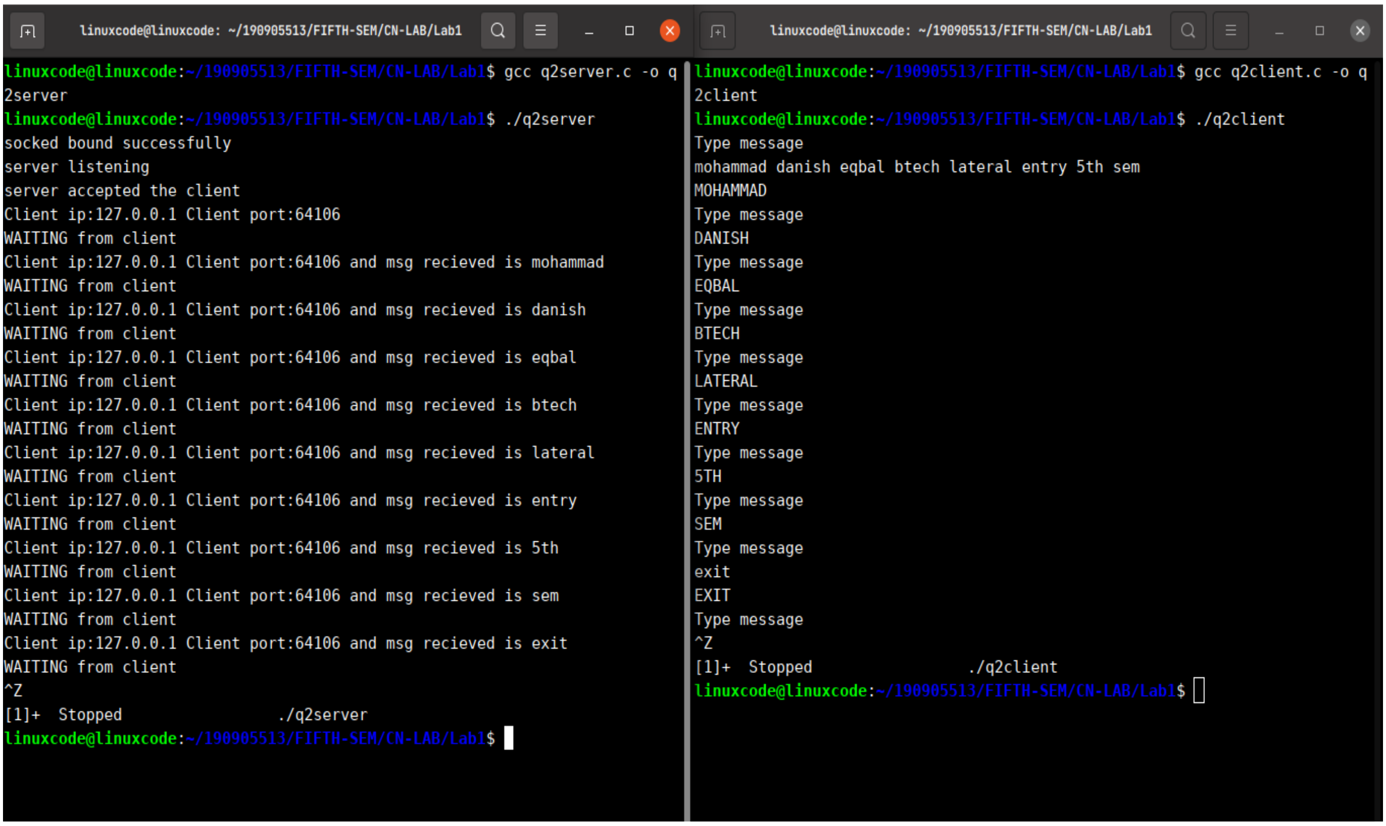
2. Write a TCP client which sends a string to a server program. Server displays the string along with client IP and ephemeral port number. Server then responds to the client by echoing back the string in uppercase. The process continues until one of them types “QUIT”.

Program:

```
//Server side program
#include <sys/types.h>
#include <sys/socket.h>
#include <stdio.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <stdlib.h>
#include <ctype.h>
#include <string.h>
#define MAX 50
void servfunc(int conn_fd,struct sockaddr_in client_address){
char buff[MAX];
int n=0;
char* ip_add=inet_ntoa(client_address.sin_addr);
int port=client_address.sin_port;
printf("Client ip:%s Client port:%d \n",ip_add,port);
while(1){
printf("WAITING from client\n");
memset(buff,0,sizeof(buff));
n = read(conn_fd,buff,sizeof(buff));
buff[n]='\n';
printf("Client ip:%s Client port:%d and msg recieved is %s \n",ip_add,port,buff);
if(strcmp("quit",buff)==0){
printf("server is closing..closed\n");
return;
}for(int i=0;i<n;i++){
buff[i]=toupper(buff[i]);
}
write(conn_fd,buff,sizeof(buff));
}}
int main(){
int server_sockfd, conn_sockfd;
int server_len,client_len;
struct sockaddr_in server_address;
struct sockaddr_in client_address;
//create a socket for the server
server_sockfd=socket(AF_INET,SOCK_STREAM,0);
//name the server socket
server_address.sin_family=AF_INET;
//inet_addr converts to unsigned long,
//else use htonl(INADDR_ANY)
server_address.sin_addr.s_addr=inet_addr("127.0.0.1");
server_address.sin_port=htons(7280);
server_len=sizeof(server_address);
if(bind(server_sockfd,(struct
sockaddr*)&server_address,server_len)!=0){
printf("socket binding failed\n");
exit(0);
}
else{
printf("socked bound successfully\n");}
//create a connection queue and wait for clients
if(listen(server_sockfd,2)!=0){
printf("listen failed\n");
exit(0);
} else{
printf("server listening\n");
} client_len=sizeof(client_address);
//when accepted a new client, a new socketfd is created
conn_sockfd=accept(server_sockfd,(struct
sockaddr*)&client_address,&client_len);
if(conn_sockfd<0){
printf("server accept failed\n");
exit(0);
}
```

```
else{
printf("server accepted the client\n");
}
servfunc(conn_sockfd,client_address);
close(server_sockfd);
return 0;
}

//Client side program
#include <string.h>
#include <ctype.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <stdio.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <stdlib.h>
#define MAX 50
void clifunc(int sockfd){
char buff[MAX];
int n=0,recv_len=0;
while(1){
memset(buff,0,sizeof(buff));
printf("Type message\n");
scanf("%s",buff);
write(sockfd,buff,sizeof(buff));
if(strcmp("quit",buff)==0){
printf("client closing\n");
return;
}
memset(buff,0,sizeof(buff));
n=read(sockfd,buff,sizeof(buff));
buff[n]='\n';
printf("%s\n",buff );
}}
void main(int argc, char const *argv[]){
int sockfd;
int len;
struct sockaddr_in server_address;int result;
char ch;
sockfd=socket(AF_INET,SOCK_STREAM,0);
server_address.sin_family=AF_INET;
server_address.sin_addr.s_addr=inet_addr("127.0.0.1");
server_address.sin_port=htons(7280);
len=sizeof(server_address);
result=connect(sockfd,(struct sockaddr*)&server_address,len);
if(result == -1){
printf("connection error\n");
exit(0);
}
clifunc(sockfd);
close(sockfd);
}
```

Output:



```
linuxcode@linuxcode:~/190905513/FIFTH-SEM/CN-LAB/Lab1$ gcc q2server.c -o q
2server
linuxcode@linuxcode:~/190905513/FIFTH-SEM/CN-LAB/Lab1$ ./q2server
socked bound successfully
server listening
server accepted the client
Client ip:127.0.0.1 Client port:64106
WAITING from client
Client ip:127.0.0.1 Client port:64106 and msg recieved is mohammad
WAITING from client
Client ip:127.0.0.1 Client port:64106 and msg recieved is danish
WAITING from client
Client ip:127.0.0.1 Client port:64106 and msg recieved is eqbal
WAITING from client
Client ip:127.0.0.1 Client port:64106 and msg recieved is btech
WAITING from client
Client ip:127.0.0.1 Client port:64106 and msg recieved is lateral
WAITING from client
Client ip:127.0.0.1 Client port:64106 and msg recieved is entry
WAITING from client
Client ip:127.0.0.1 Client port:64106 and msg recieved is 5th
WAITING from client
Client ip:127.0.0.1 Client port:64106 and msg recieved is sem
WAITING from client
Client ip:127.0.0.1 Client port:64106 and msg recieved is exit
WAITING from client
^Z
[1]+  Stopped                 ./q2server
linuxcode@linuxcode:~/190905513/FIFTH-SEM/CN-LAB/Lab1$
```

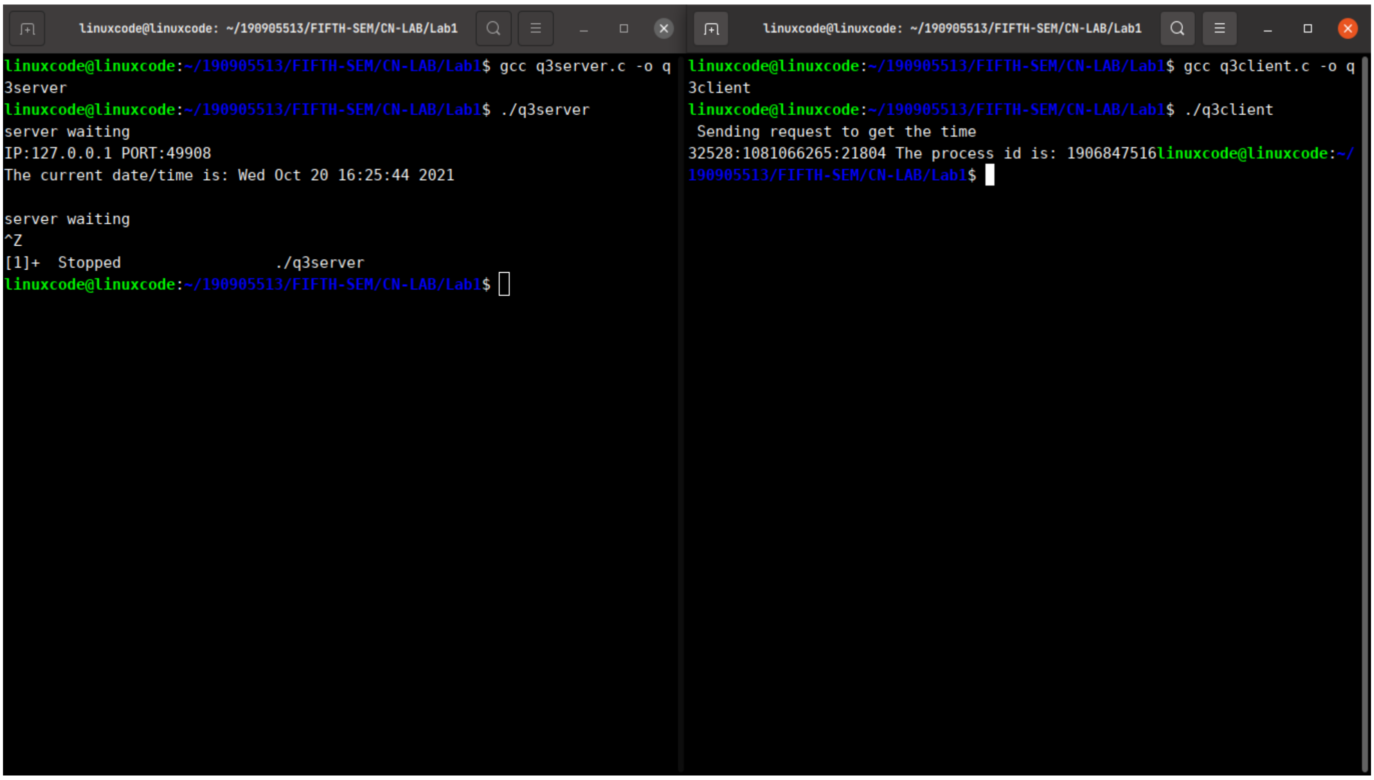
```
linuxcode@linuxcode:~/190905513/FIFTH-SEM/CN-LAB/Lab1$ gcc q2client.c -o q
2client
linuxcode@linuxcode:~/190905513/FIFTH-SEM/CN-LAB/Lab1$ ./q2client
Type message
mohammad danish eqbal btech lateral entry 5th sem
MOHAMMAD
Type message
DANISH
Type message
EQBAL
Type message
BTECH
Type message
LATERAL
Type message
ENTRY
Type message
5TH
Type message
SEM
Type message
exit
EXIT
Type message
^Z
[1]+  Stopped                 ./q2client
linuxcode@linuxcode:~/190905513/FIFTH-SEM/CN-LAB/Lab1$
```

3. DayTime Server: Where client sends request to time server to send current time. Server responds by sending the current time . [Hint: read man pages of asctime() and localtime()] . Display server process id at client side along with time.

Program:

```
//Server side program
#include <sys/types.h>
#include <sys/socket.h>
#include <stdio.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <stdlib.h>
#include <time.h>
int main(){
time_t rawtime;
struct tm * timeinfo;
char *reply;
int server_sockfd, client_sockfd;
int server_len, client_len;
struct sockaddr_in server_address;
struct sockaddr_in client_address;
int hour,mins,sec,pid;
//Create an unnamed socket for the server.
server_sockfd = socket(AF_INET, SOCK_STREAM, 0);
// Name the socket.
server_address.sin_family = AF_INET;server_address.sin_addr.s_addr
= inet_addr("127.0.0.1");
server_address.sin_port = 9734;
server_len = sizeof(server_address);
bind(server_sockfd, (struct sockaddr *)&server_address,
server_len);
// Create a connection queue and wait for clients.
listen(server_sockfd, 5);
while(1){
char ch;
printf("server waiting\n");
// Accept a connection.
client_len = sizeof(client_address);
client_sockfd = accept(server_sockfd, (struct sockaddr
*)&client_address, &client_len);
// We can now read/write to client on client_sockfd.
char * ip_add =inet_ntoa(client_address.sin_addr);
int port=client_address.sin_port;
printf("IP:%s PORT:%d\n", ip_add,port);
//get the time
time ( &rawtime );
timeinfo = localtime ( &rawtime );
reply = asctime(timeinfo);
printf ( "The current date/time is: %s\n", reply );
hour = timeinfo->tm_hour;
mins = timeinfo->tm_min;sec = timeinfo->tm_sec;
pid = getpid();
write(client_sockfd, &hour, 1);
write(client_sockfd, &mins, 1);
write(client_sockfd, &sec, 1);
write(client_sockfd, &pid, 1);
}
return 0;
}

//Client side program
#include <sys/types.h>
#include <sys/socket.h>
#include <stdio.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <stdlib.h>
#include <time.h>
int main(){
int sockfd;
int len;
struct sockaddr_in address;
struct tm * timeinfo;
int result;
char *reply;int hour,mins,sec,pid;
// Create a socket for the client.
sockfd = socket(AF_INET, SOCK_STREAM, 0);
// Name the socket, as agreed with the server.
address.sin_family = AF_INET;
address.sin_addr.s_addr = inet_addr("127.0.0.1");
address.sin_port = 9734;
len = sizeof(address);
// Now connect our socket to the server's socket.
result = connect(sockfd, (struct sockaddr *)&address, len);
if(result == -1)
{
perror("oops: client2");
exit(1);
}
// We can now read/write via sockfd.
printf(" Sending request to get the time\n");
read(sockfd, &hour , 1);
read(sockfd, &mins , 1);
read(sockfd, &sec , 1);
read(sockfd, &pid , 1);
printf("%d:%d:%d", hour, mins, sec);printf(" The process id is: %d",pid);
close(sockfd);
exit(0);
return 0;
}
```

Output:



linuxcode@linuxcode: ~/190905513/FIFTH-SEM/CN-LAB/Lab1

```
linuxcode@linuxcode:~/190905513/FIFTH-SEM/CN-LAB/Lab1$ gcc q3server.c -o q
3server
linuxcode@linuxcode:~/190905513/FIFTH-SEM/CN-LAB/Lab1$ ./q3server
server waiting
IP:127.0.0.1 PORT:49908
The current date/time is: Wed Oct 20 16:25:44 2021

server waiting
^Z
[1]+  Stopped                 ./q3server
linuxcode@linuxcode:~/190905513/FIFTH-SEM/CN-LAB/Lab1$
```

linuxcode@linuxcode: ~/190905513/FIFTH-SEM/CN-LAB/Lab1

```
linuxcode@linuxcode:~/190905513/FIFTH-SEM/CN-LAB/Lab1$ gcc q3client.c -o q
3client
linuxcode@linuxcode:~/190905513/FIFTH-SEM/CN-LAB/Lab1$ ./q3client
 Sending request to get the time
32528:1081066265:21804 The process id is: 1906847516linuxcode@linuxcode:~/
190905513/FIFTH-SEM/CN-LAB/Lab1$
```