CN LAB-1

Name: Nekkanti Keerthan
Reg_No: 180905480
Roll_No: 57
Sec: D

Examples

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

Server

```
// 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()
```

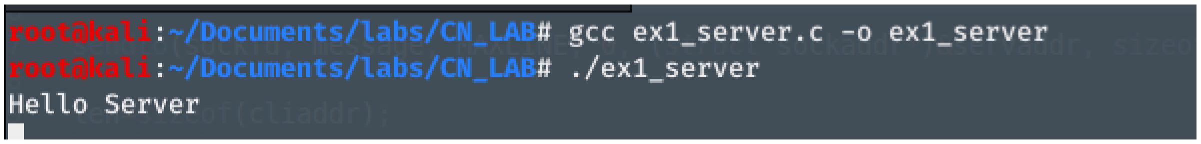
```
{

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();


}
```

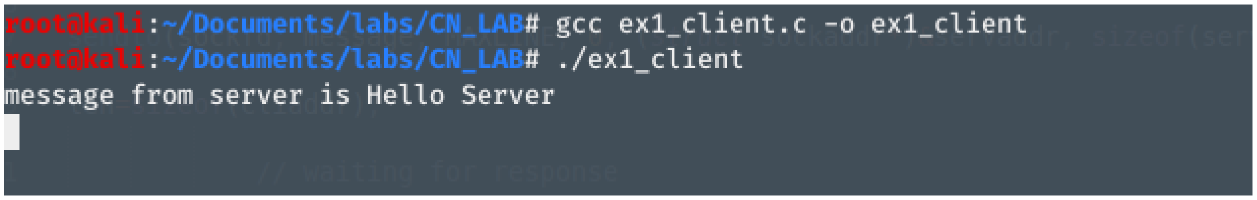
Client

```
#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 from server is %s \n",buffer);

     getchar();


// close the descriptor

close(sockfd);

}
```

Output



```
root@kali:~/Documents/labs/CN_LAB# gcc ex1_server.c -o ex1_server
root@kali:~/Documents/labs/CN_LAB# ./ex1_server
Hello Server
```



```
root@kali:~/Documents/labs/CN_LAB# gcc ex1_client.c -o ex1_client
root@kali:~/Documents/labs/CN_LAB# ./ex1_client
message from server 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.)

Server

```
#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); }
```

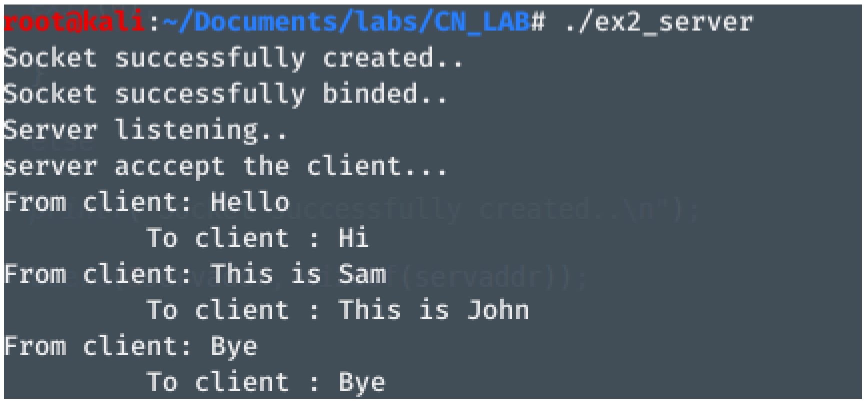
Client

```
#include <netdb.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>
```

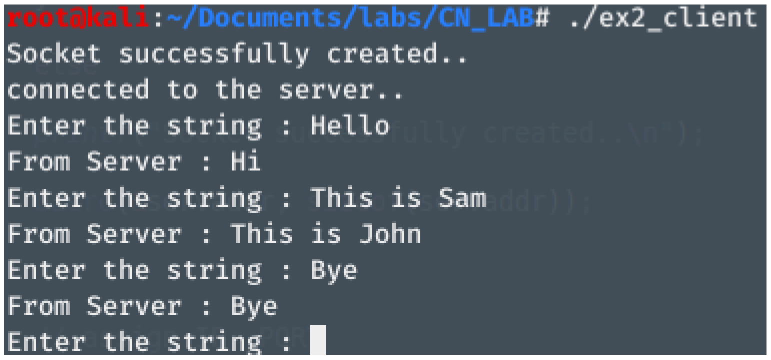
```
#include <sys/socket.h>

#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



```
root@kali:~/Documents/labs/CN_LAB# ./ex2_server
Socket successfully created..
Socket successfully binded..
Server listening..
server acccept the client...
From client: Hello
         To client : Hi
From client: This is Sam
         To client : This is John
From client: Bye
         To client : Bye
```



```
root@kali:~/Documents/labs/CN_LAB# ./ex2_client
Socket successfully created..
connected to the server..
Enter the string : Hello
From Server : Hi
Enter the string : This is Sam
From Server : This is John
Enter the string : Bye
From Server : Bye
Enter the string :
```

EXERCISE

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.

Server

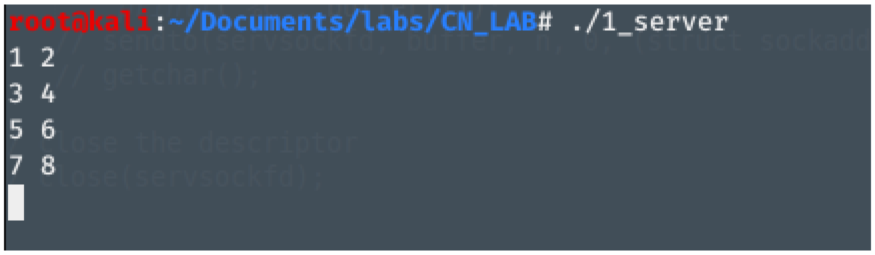
```
#include <stdio.h>
#include <strings.h>
#include <sys/types.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include <netinet/in.h>
#define PORT 5000
#define MAXLINE 1000
// Server code
int main()
{
   char buffer[100];
   int servsockfd, len, n;
   struct sockaddr_in servaddr, cliaddr;
   bzero(&servaddr, 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
    while(1)
    {
        bzero(buffer, sizeof(buffer));
        len = sizeof(cliaddr);
        n = recvfrom(servsockfd, buffer, sizeof(buffer), 0, (struct sockaddr *)&cliaddr, &len);
        // buffer[n] = '\0';
        //Echoing back to the client
        if ((strncmp(buffer, "exit", 4)) == 0)
        {
            printf("Client Exit\n");
            break;
        }
        for(int i = 0; i < n; i++)
            printf("%c", buffer[i]);
        // sendto(servsockfd, buffer, n, 0, (struct sockaddr *)&cliaddr, sizeof(cliaddr));
        // getchar();
    }
    // close the descriptor
    // close(servsockfd);
}
```

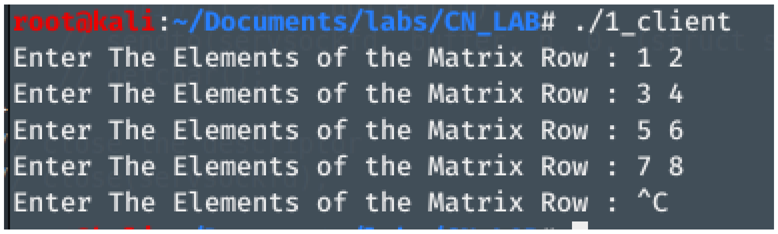
Client

```
#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 = "";
    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);
    char buff[100];
    for(;;)
    {
        bzero(buff, sizeof(buff));
        printf("Enter The Elements of the Matrix Row : ");
        n = 0;
        while ((buff[n++] = getchar()) != '\n');
        buff[n] = '\0';
        sendto(sockfd, &buff, MAXLINE, 0, (struct sockaddr *)&servaddr, sizeof(servaddr));
        if(strncmp(buff, "exit", 4) == 0)
        {
            printf("Closing client\n");
            break;
        }
        bzero(buff, sizeof(buff));
        len = sizeof(cliaddr);
    }
    // close the descriptor
    close(sockfd);
}
```

Output



```
root@kali:~/Documents/labs/CN_LAB# ./1_server
1 2
3 4
5 6
7 8
```



```
root@kali:~/Documents/labs/CN_LAB# ./1_client
Enter The Elements of the Matrix Row : 1 2
Enter The Elements of the Matrix Row : 3 4
Enter The Elements of the Matrix Row : 5 6
Enter The Elements of the Matrix Row : 7 8
Enter The Elements of the Matrix Row : ^C
```

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”.

<u>Server</u>

```
#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)
{
        //display client ip and port number
        //echo back received string in upper case
        //if quit is received from client/server, end program
        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");
                //while(n==0)
                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);
```

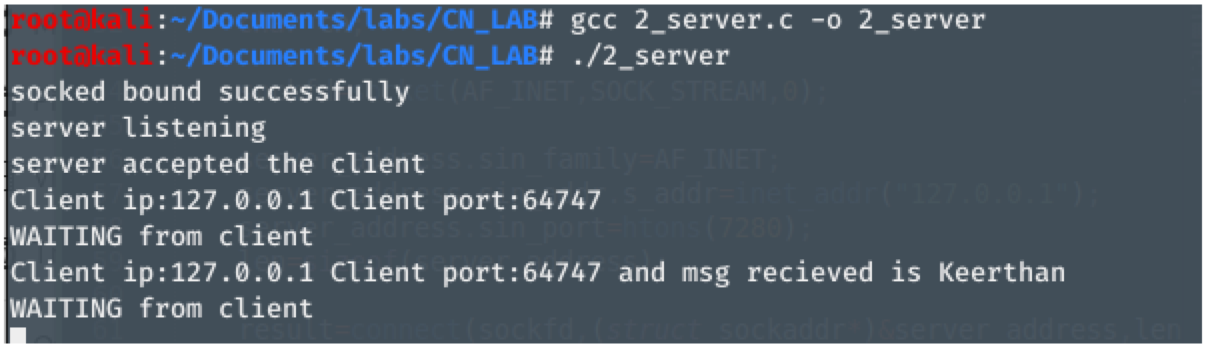
```
	if(bind(server_sockfd,(struct sockaddr*)&server_address,server_len)!=0)
	{
		printf("socket binding failed\n");
	exit(0);
	}

	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;
}
```

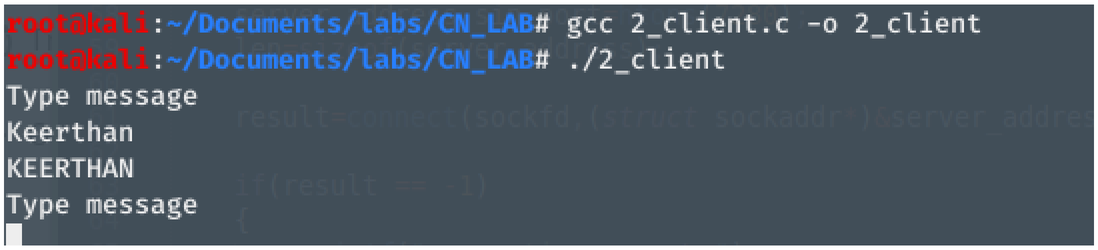
Client

```
#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 clifunc(int sockfd)
{
        //send string
        //print returned string
        //if returned string is quit, quit

        char buff[MAX];
        int n=0,recv_len=0;

        while(1)
        {
                memset(buff,0,sizeof(buff));
                printf("Type message\n");
                //while((buff[n++]=getchar())!='\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 );
        }
        }


int 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);

   return 0;

}
```

Output



```
root@kali:~/Documents/labs/CN_LAB# gcc 2_server.c -o 2_server
root@kali:~/Documents/labs/CN_LAB# ./2_server
socked bound successfully
server listening
server accepted the client
Client ip:127.0.0.1 Client port:64747
WAITING from client
Client ip:127.0.0.1 Client port:64747 and msg recieved is Keerthan
WAITING from client
```



```
root@kali:~/Documents/labs/CN_LAB# gcc 2_client.c -o 2_client
root@kali:~/Documents/labs/CN_LAB# ./2_client
Type message
Keerthan
KEERTHAN
Type message
```