Advance C Programming

Weekly Assessment - Module 4 Assessment

VIVEK MUNNAA D

1. Explain the connection procedure followed in client server communication.

Ans: Socket Programming is one of the methods to connect two nodes over a network to establish a means of communication between those two nodes. Here client and server can be inferred as the nodes and sockets are created to establish communication between them. A socket is an endpoint used by one node to connect to another node.

The first step of connection procedure on the server side is creating a socket as endpoint initially. For this, the **socket()** function is used. Once a socket is created, the address information for the server including the port number and IP family is specified. Next, we must bind an address to the socket created using **bind()** function. Then, the **listen()** is used to make the server wait and listen for connections from the client node. To establish a connection, we use the **accept()** function and on a successful implication, the **read()** and **write()** functions to send and receive data from server to client or vice versa. After communication the established connections are closed using the **close()** function. However, the above steps describe the typical connection only in a TCP based connection whereas in a UDP based connection only two initial steps are necessary the **socket()** to create the socket and **bind()** to bind the address after this we can send and receive messages using the **sendto()** and **recvfrom().**

On the client side, for both TCP and UDP based communications only two functions are required. The first one being the **socket()** to create an endpoint for the client side. Then **connect()** is used to send connection request to the server. After that we can send and receive messages using the appropriate functions based on whether the established connection is a TCP based connection (read() and write()) or a UDP based connection (sendto() and recvfrom()).

2.What is the use of bind() function in socket programming ?

Ans: The bind() function in socket programming is used to assign an address to a socket created before using the socket() function. The bind() associates a socket with a specific IP address and port number. The function returns 0 on binding the address and port successfully and returns -1 on a failed binding. Binding to ‘INADDR_ANY’ (0.0.0.0) allows the server to accept connections on any available network interface.

The syntax is as follows:

int bind(int socket_descriptor, const struct sockaddr *address, socklen_t length_of_address);

Here,

- The socket_descriptor is the value of the file descriptor returned by the socket() function.
- The address is a struct type of sockaddr and a type sockaddr_in is used to represent this information as port and address information can only be stored in this structure. The sockaddr_in is cast to sockaddr data type when calling the bind() function.
- The length_of_address represents the size of the address passed as the second parameter.

3.What is Datagram Socket?

Ans: A datagram socket is a type of socket used in communication between networks where there is no need to establish a connection (i.e) connectionless networks. The main difference is stream sockets can provide reliable, orderly and bidirectional byte stream transfer whereas datagram sockets are unordered, unreliable and are used for transmitting discrete messages.

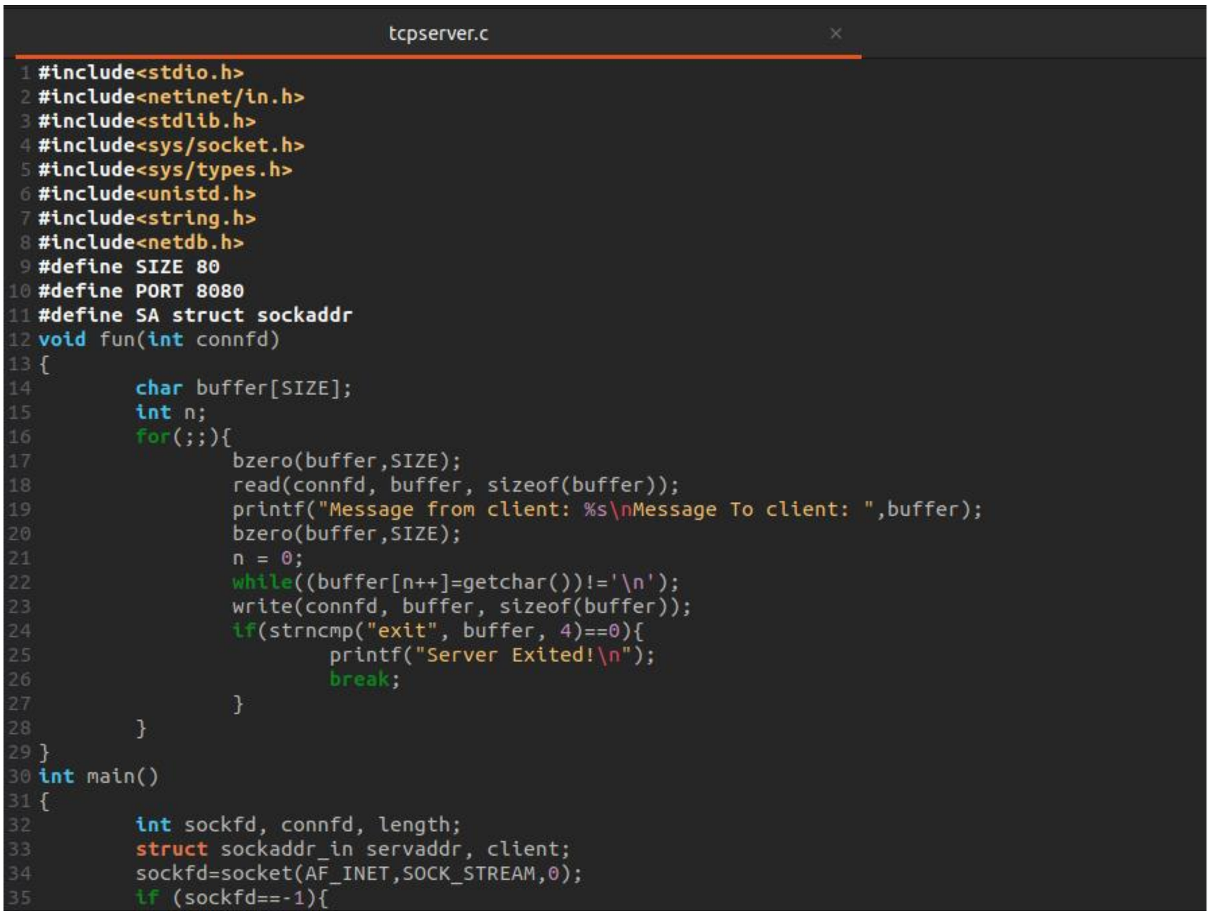
Datagram sockets do not establish a connection between the two nodes or sockets, instead every time they send a message it specifies the destination port or address. Datagram sockets are used with UDP (User Datagram Protocol) which is a connectionless protocol that provide best effort service but doesn’t guarantee whether datagrams will arrive at the destination or not.

Datagram sockets also doesn’t have any retransmission mechanisms for lost datagrams. Also, these sockets do not ensure whether the datagrams will arrive in the same order they were sent. Datagram sockets send individual messages unlike stream sockets which send a continuous stream of data. Here, each message has a maximum size and each message is sent along with the specified destination port.

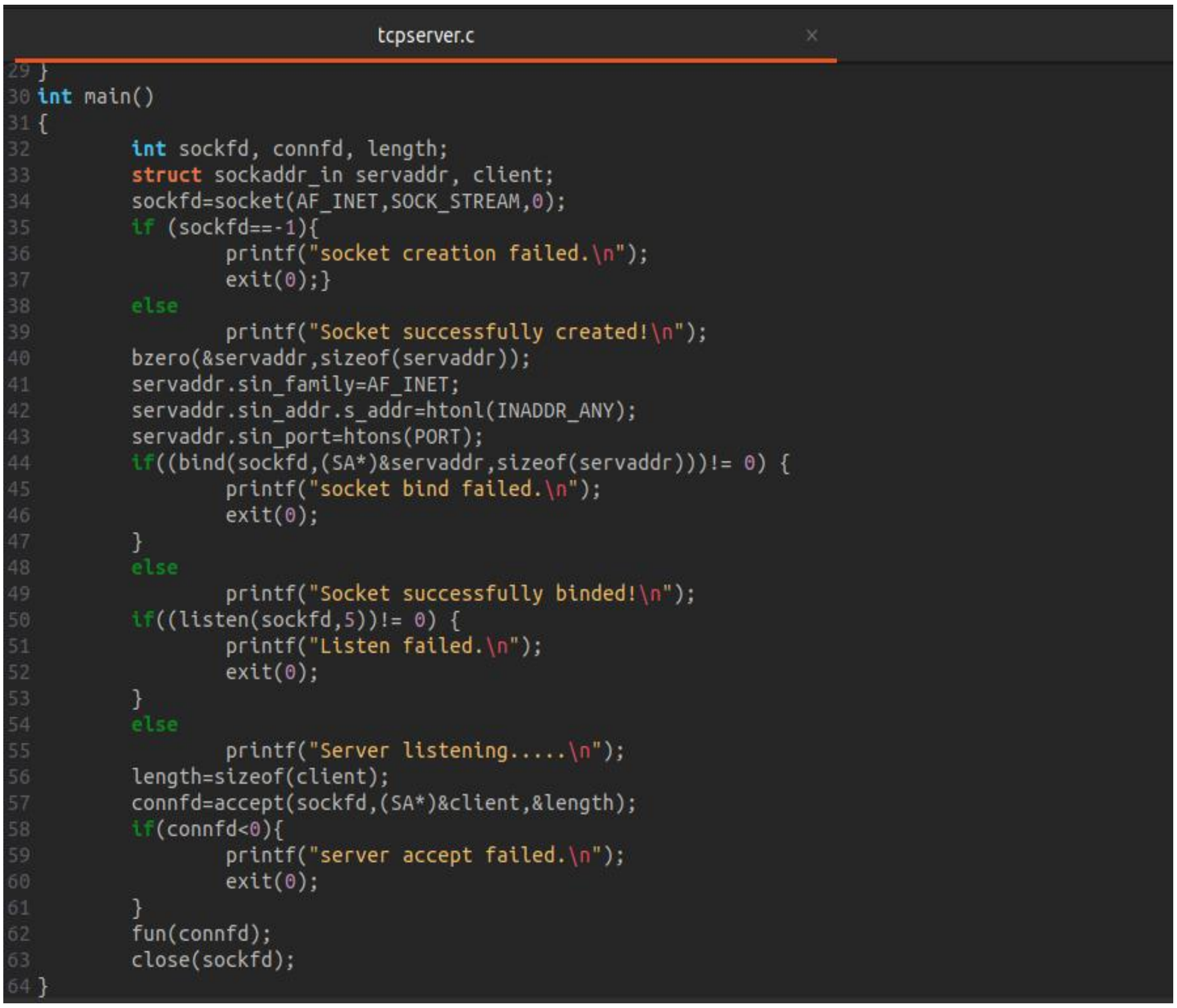
Datagram sockets are mainly used for real-time applications which require low latency, examples include Broadcasting, Streaming media, VOIP (Voice-Over-IP).

4.Write a server/client model socket program to exchange hello message between them.

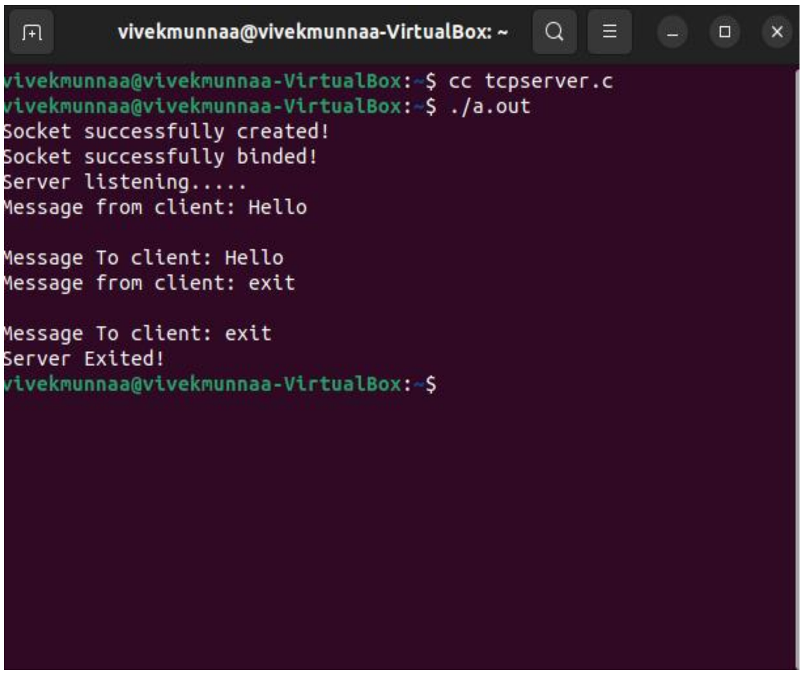
SERVER:



```
#include<stdio.h>
#include<netinet/in.h>
#include<stdlib.h>
#include<sys/socket.h>
#include<sys/types.h>
#include<unistd.h>
#include<string.h>
#include<netdb.h>
#define SIZE 80
#define PORT 8080
#define SA struct sockaddr
void fun(int connfd)
{
        char buffer[SIZE];
        int n;
        for(;;){
                bzero(buffer,SIZE);
                read(connfd, buffer, sizeof(buffer));
                printf("Message from client: %s\nMessage To client: ",buffer);
                bzero(buffer,SIZE);
                n = 0;
                while((buffer[n++]=getchar())!='\n');
                write(connfd, buffer, sizeof(buffer));
                if(strncmp("exit", buffer, 4)==0){
                        printf("Server Exited!\n");
                        break;
                }
        }
}
int main()
{
        int sockfd, connfd, length;
        struct sockaddr_in servaddr, client;
        sockfd=socket(AF_INET,SOCK_STREAM,0);
        if (sockfd==-1){
```

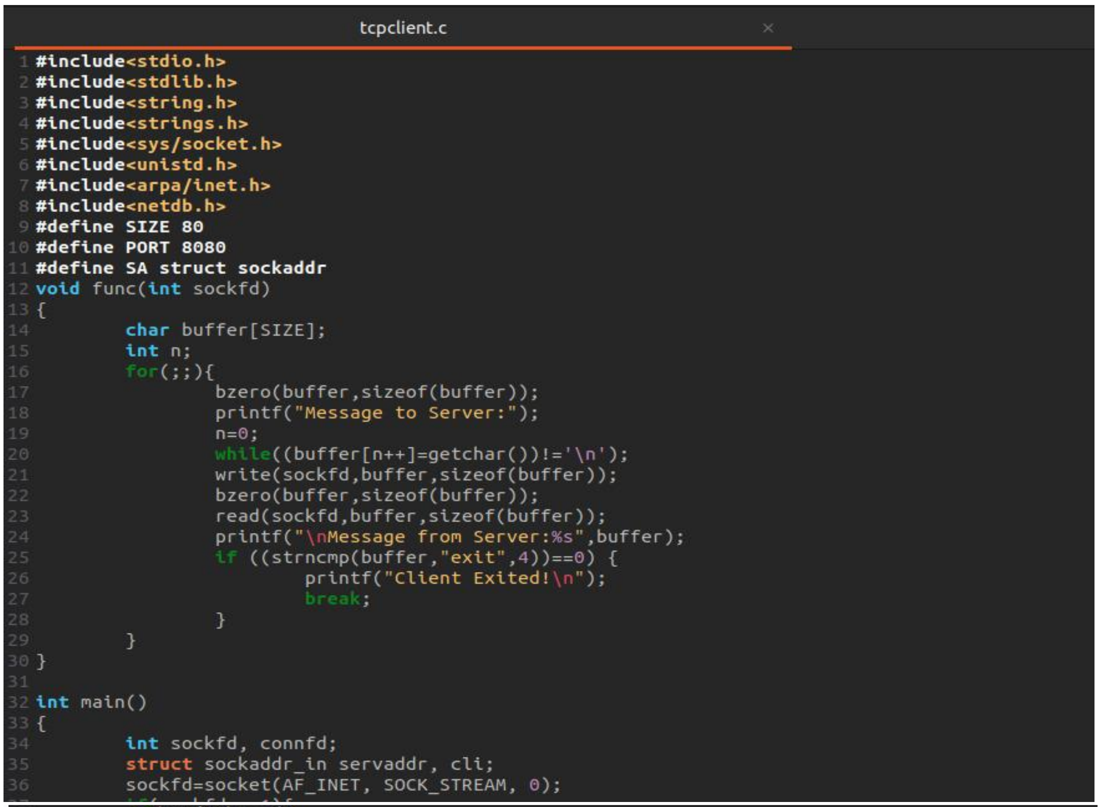


```c
}
int main()
{
        int sockfd, connfd, length;
        struct sockaddr_in servaddr, client;
        sockfd=socket(AF_INET,SOCK_STREAM,0);
        if (sockfd==-1){
                printf("socket creation failed.\n");
                exit(0);}
        else
                printf("Socket successfully created!\n");
        bzero(&servaddr,sizeof(servaddr));
        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");
        if((listen(sockfd,5))!= 0) {
                printf("Listen failed.\n");
                exit(0);
        }
        else
                printf("Server listening.....\n");
        length=sizeof(client);
        connfd=accept(sockfd,(SA*)&client,&length);
        if(connfd<0){
                printf("server accept failed.\n");
                exit(0);
        }
        fun(connfd);
        close(sockfd);
}
```

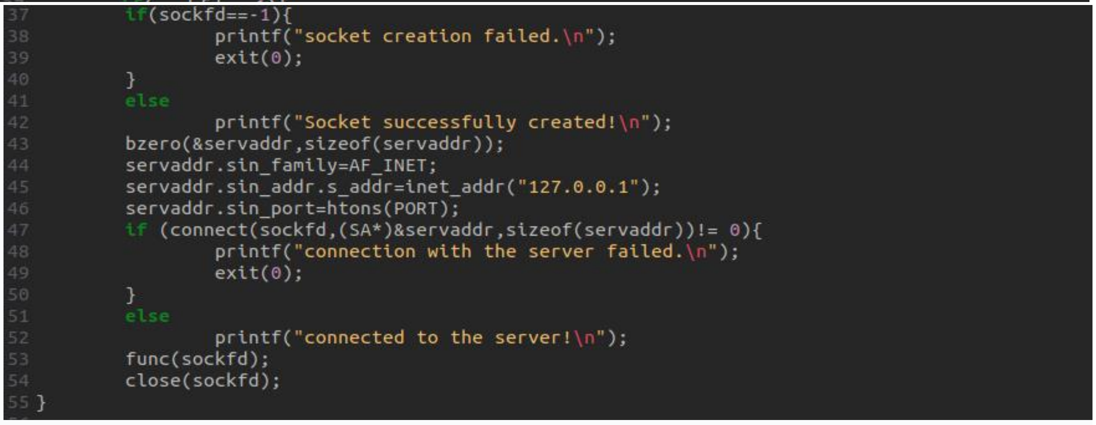


```
vivekmunnaa@vivekmunnaa-VirtualBox:~$ cc tcpserver.c
vivekmunnaa@vivekmunnaa-VirtualBox:~$ ./a.out
Socket successfully created!
Socket successfully binded!
Server listening.....
Message from client: Hello

Message To client: Hello
Message from client: exit

Message To client: exit
Server Exited!
vivekmunnaa@vivekmunnaa-VirtualBox:~$
```

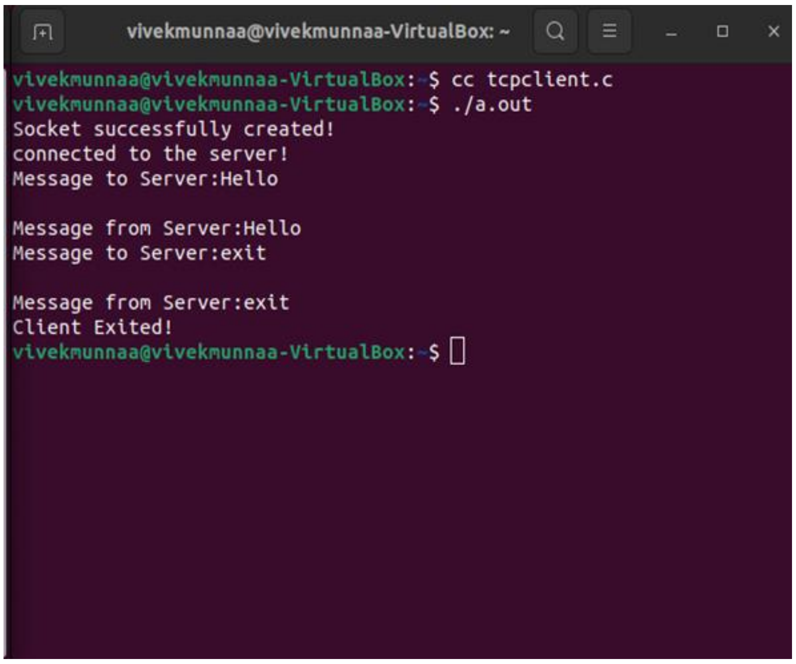
CLIENT:



```c
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<strings.h>
#include<sys/socket.h>
#include<unistd.h>
#include<arpa/inet.h>
#include<netdb.h>
#define SIZE 80
#define PORT 8080
#define SA struct sockaddr
void func(int sockfd)
{
        char buffer[SIZE];
        int n;
        for(;;){
                bzero(buffer,sizeof(buffer));
                printf("Message to Server:");
                n=0;
                while((buffer[n++]=getchar())!='\n');
                write(sockfd,buffer,sizeof(buffer));
                bzero(buffer,sizeof(buffer));
                read(sockfd,buffer,sizeof(buffer));
                printf("\nMessage from Server:%s",buffer);
                if ((strncmp(buffer,"exit",4))==0) {
                        printf("Client Exited!\n");
                        break;
                }
        }
}

int main()
{
        int sockfd, connfd;
        struct sockaddr_in servaddr, cli;
        sockfd=socket(AF_INET, SOCK_STREAM, 0);
```



```c
        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=inet_addr("127.0.0.1");
        servaddr.sin_port=htons(PORT);
        if (connect(sockfd,(SA*)&servaddr,sizeof(servaddr))!= 0){
                printf("connection with the server failed.\n");
                exit(0);
        }
        else
                printf("connected to the server!\n");
        func(sockfd);
        close(sockfd);
}
```



```
vivekmunnaa@vivekmunnaa-VirtualBox:~$ cc tcpclient.c
vivekmunnaa@vivekmunnaa-VirtualBox:~$ ./a.out
Socket successfully created!
connected to the server!
Message to Server:Hello

Message from Server:Hello
Message to Server:exit

Message from Server:exit
Client Exited!
vivekmunnaa@vivekmunnaa-VirtualBox:~$ 
```

(P.T.O)

5.Write a TCP server-client program to check if a given string is Palindrome

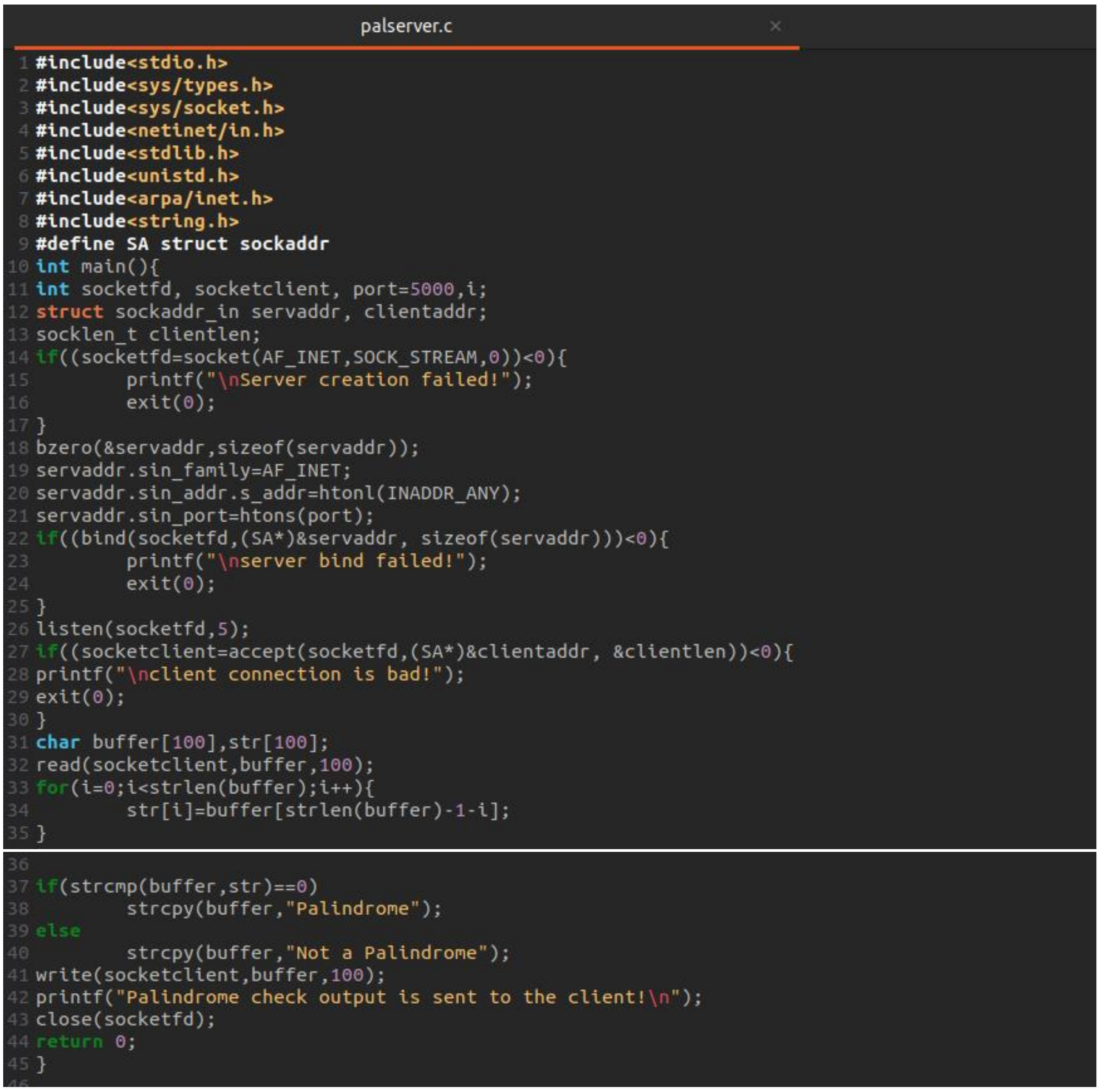
Input: level

Output: Palindrome

Input: Assessment

Output: Not a Palindrome

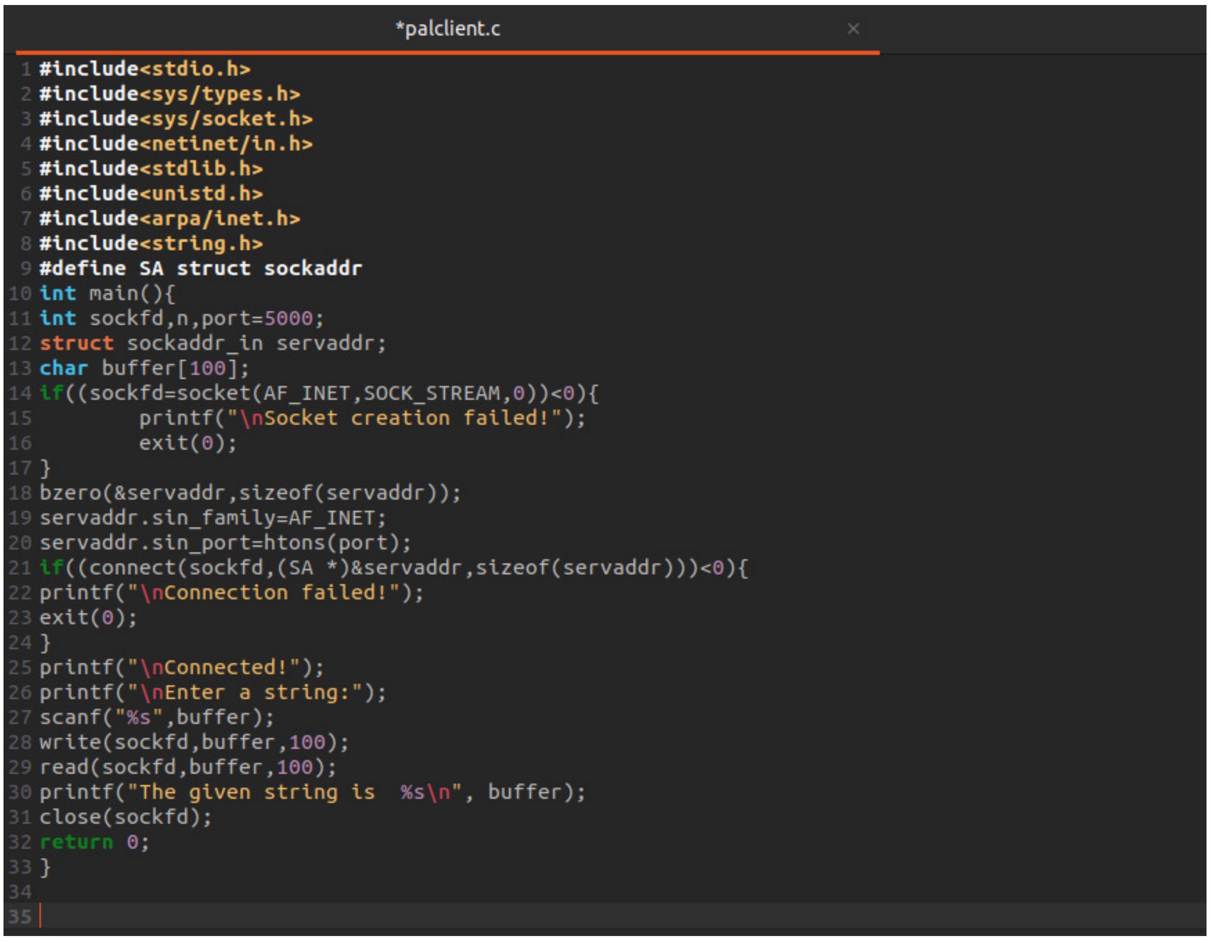
SERVER:



palserver.c

```c
#include<stdio.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<stdlib.h>
#include<unistd.h>
#include<arpa/inet.h>
#include<string.h>
#define SA struct sockaddr
int main(){
int socketfd, socketclient, port=5000,i;
struct sockaddr_in servaddr, clientaddr;
socklen_t clientlen;
if((socketfd=socket(AF_INET,SOCK_STREAM,0))<0){
        printf("\nServer creation failed!");
        exit(0);
}
bzero(&servaddr,sizeof(servaddr));
servaddr.sin_family=AF_INET;
servaddr.sin_addr.s_addr=htonl(INADDR_ANY);
servaddr.sin_port=htons(port);
if((bind(socketfd,(SA*)&servaddr, sizeof(servaddr)))<0){
        printf("\nserver bind failed!");
        exit(0);
}
listen(socketfd,5);
if((socketclient=accept(socketfd,(SA*)&clientaddr, &clientlen))<0){
printf("\nclient connection is bad!");
exit(0);
}
char buffer[100],str[100];
read(socketclient,buffer,100);
for(i=0;i<strlen(buffer);i++){
        str[i]=buffer[strlen(buffer)-1-i];
}

if(strcmp(buffer,str)==0)
        strcpy(buffer,"Palindrome");
else
        strcpy(buffer,"Not a Palindrome");
write(socketclient,buffer,100);
printf("Palindrome check output is sent to the client!\n");
close(socketfd);
return 0;
}
```

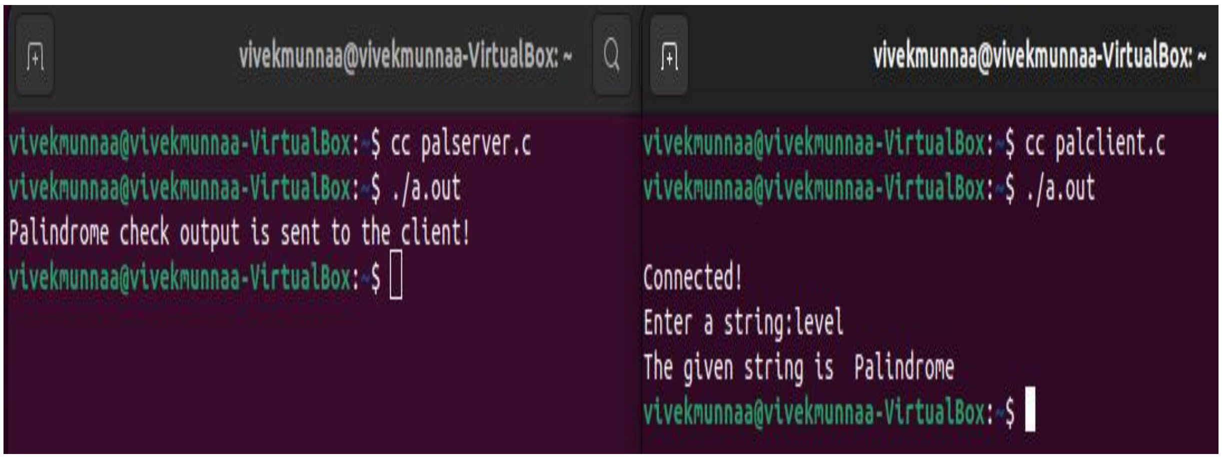
CLIENT:



*palclient.c

```c
#include<stdio.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<stdlib.h>
#include<unistd.h>
#include<arpa/inet.h>
#include<string.h>
#define SA struct sockaddr
int main(){
int sockfd,n,port=5000;
struct sockaddr_in servaddr;
char buffer[100];
if((sockfd=socket(AF_INET,SOCK_STREAM,0))<0){
        printf("\nSocket creation failed!");
        exit(0);
}
bzero(&servaddr,sizeof(servaddr));
servaddr.sin_family=AF_INET;
servaddr.sin_port=htons(port);
if((connect(sockfd,(SA *)&servaddr,sizeof(servaddr)))<0){
printf("\nConnection failed!");
exit(0);
}
printf("\nConnected!");
printf("\nEnter a string:");
scanf("%s",buffer);
write(sockfd,buffer,100);
read(sockfd,buffer,100);
printf("The given string is  %s\n", buffer);
close(sockfd);
return 0;
}

```

OUTPUT:

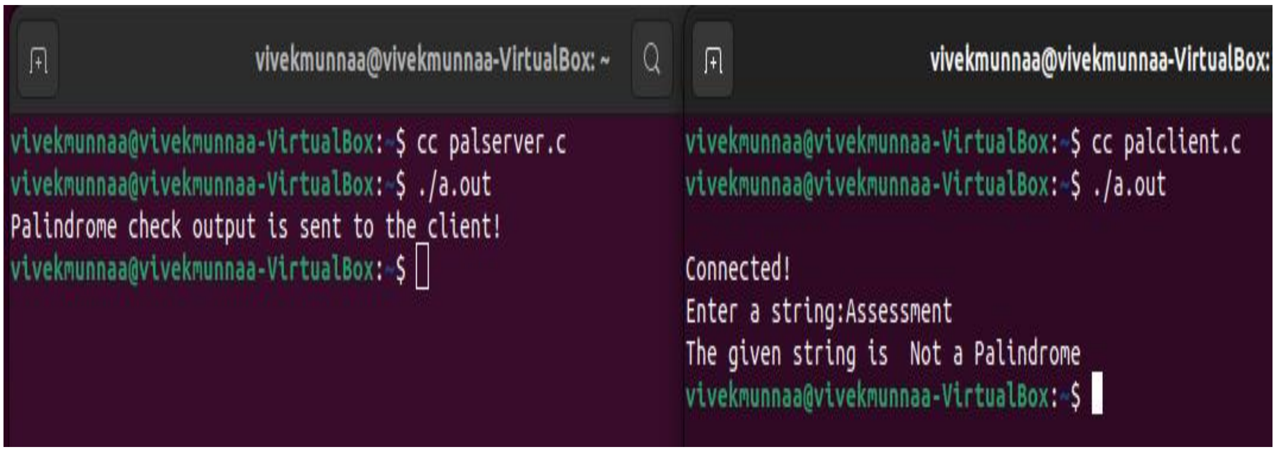


vivekmunnaa@vivekmunnaa-VirtualBox: ~

```
vivekmunnaa@vivekmunnaa-VirtualBox:~$ cc palserver.c
vivekmunnaa@vivekmunnaa-VirtualBox:~$ ./a.out
Palindrome check output is sent to the client!
vivekmunnaa@vivekmunnaa-VirtualBox:~$
```

vivekmunnaa@vivekmunnaa-VirtualBox: ~

```
vivekmunnaa@vivekmunnaa-VirtualBox:~$ cc palclient.c
vivekmunnaa@vivekmunnaa-VirtualBox:~$ ./a.out

Connected!
Enter a string:level
The given string is  Palindrome
vivekmunnaa@vivekmunnaa-VirtualBox:~$
```

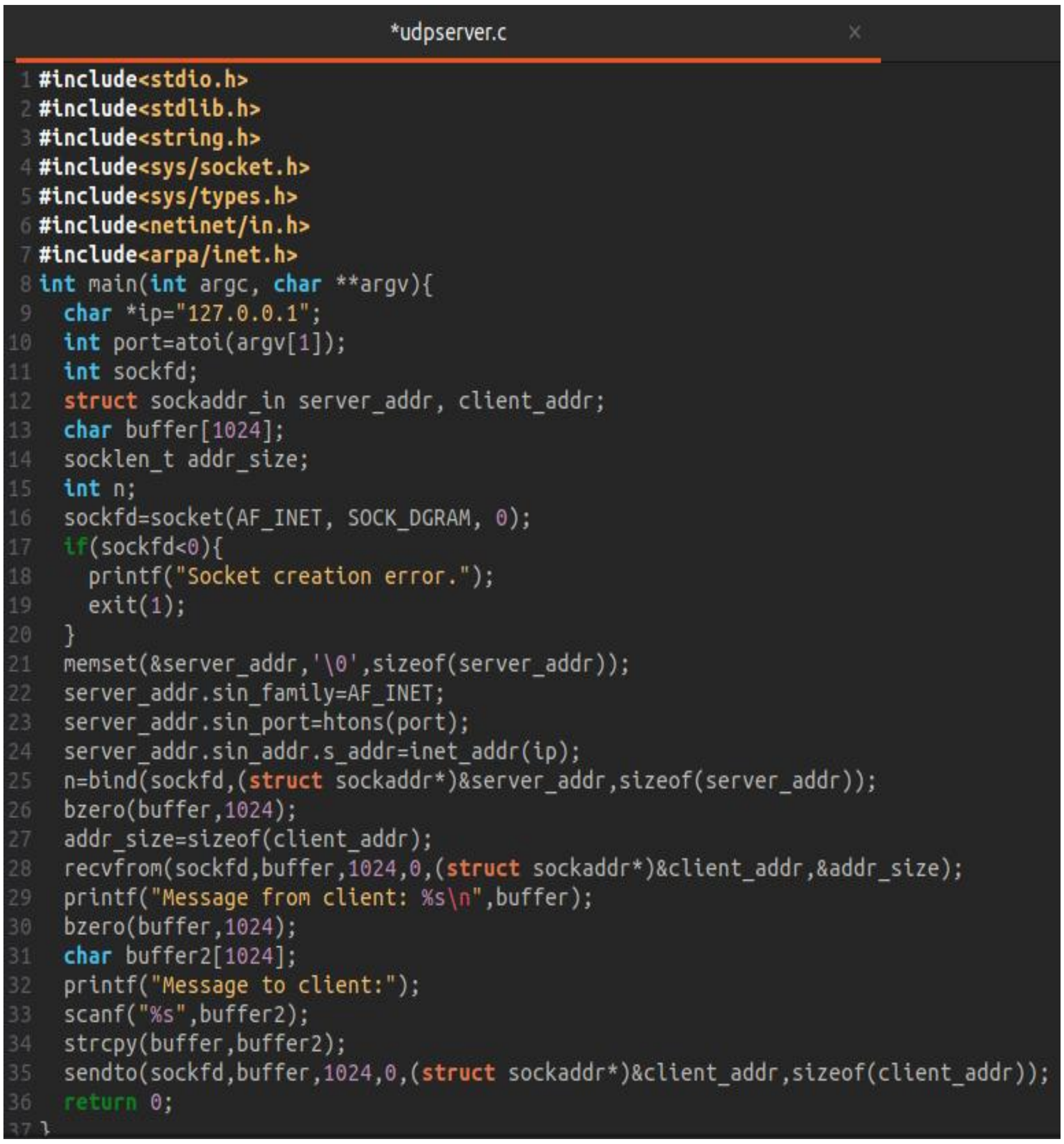


```
vivekmunnaa@vivekmunnaa-VirtualBox:~$ cc palserver.c
vivekmunnaa@vivekmunnaa-VirtualBox:~$ ./a.out
Palindrome check output is sent to the client!
vivekmunnaa@vivekmunnaa-VirtualBox:~$
```

```
vivekmunnaa@vivekmunnaa-VirtualBox:~$ cc palclient.c
vivekmunnaa@vivekmunnaa-VirtualBox:~$ ./a.out

Connected!
Enter a string:Assessment
The given string is  Not a Palindrome
vivekmunnaa@vivekmunnaa-VirtualBox:~$
```

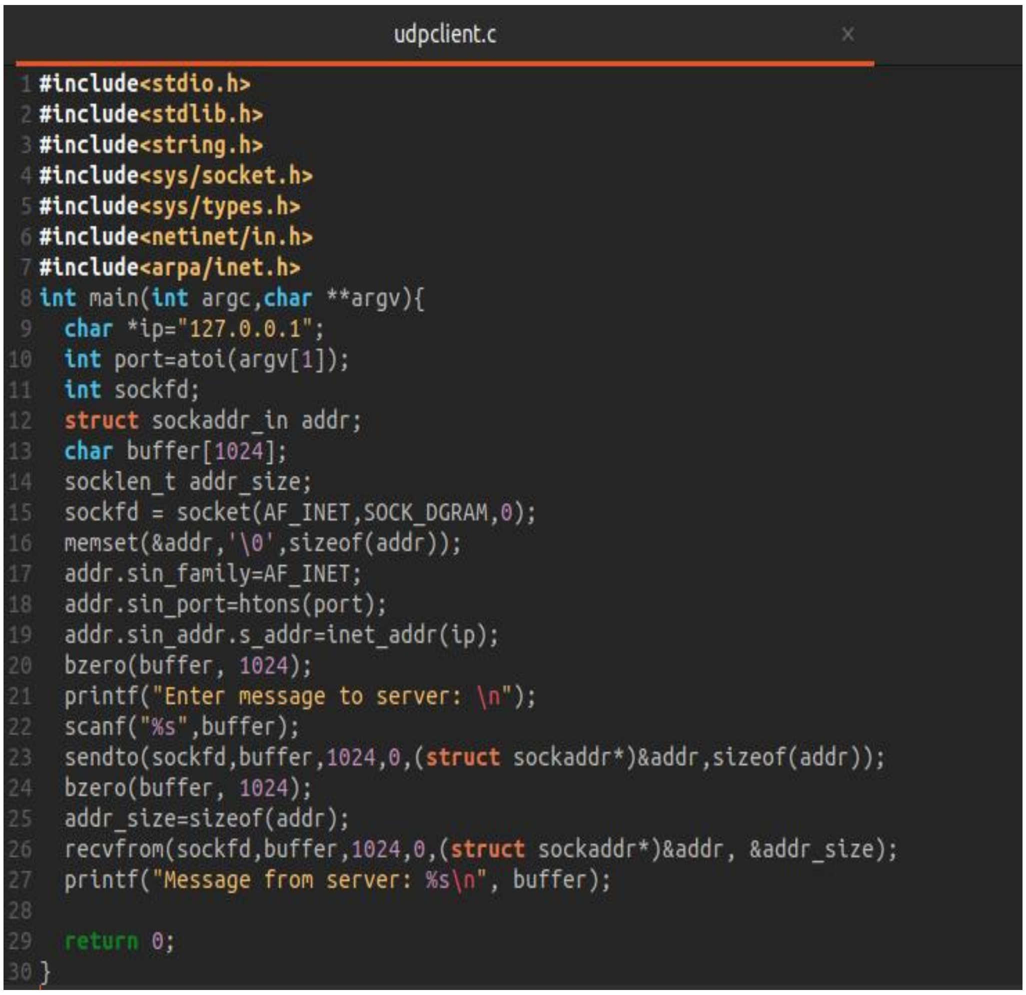
6.Write an example to demonstrate UDP server-client program

SERVER:



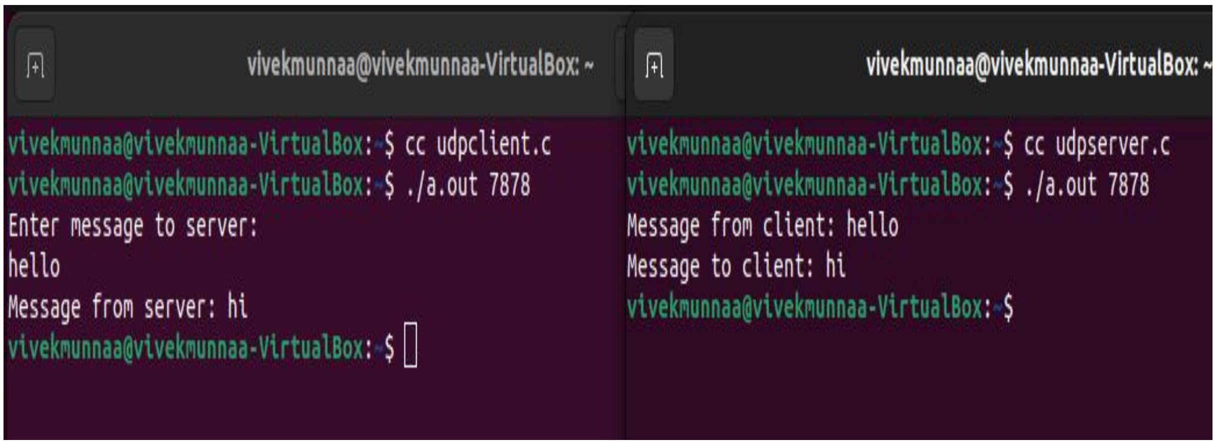
```c
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<sys/socket.h>
#include<sys/types.h>
#include<netinet/in.h>
#include<arpa/inet.h>
int main(int argc, char **argv){
  char *ip="127.0.0.1";
  int port=atoi(argv[1]);
  int sockfd;
  struct sockaddr_in server_addr, client_addr;
  char buffer[1024];
  socklen_t addr_size;
  int n;
  sockfd=socket(AF_INET, SOCK_DGRAM, 0);
  if(sockfd<0){
    printf("Socket creation error.");
    exit(1);
  }
  memset(&server_addr,'\0',sizeof(server_addr));
  server_addr.sin_family=AF_INET;
  server_addr.sin_port=htons(port);
  server_addr.sin_addr.s_addr=inet_addr(ip);
  n=bind(sockfd,(struct sockaddr*)&server_addr,sizeof(server_addr));
  bzero(buffer,1024);
  addr_size=sizeof(client_addr);
  recvfrom(sockfd,buffer,1024,0,(struct sockaddr*)&client_addr,&addr_size);
  printf("Message from client: %s\n",buffer);
  bzero(buffer,1024);
  char buffer2[1024];
  printf("Message to client:");
  scanf("%s",buffer2);
  strcpy(buffer,buffer2);
  sendto(sockfd,buffer,1024,0,(struct sockaddr*)&client_addr,sizeof(client_addr));
  return 0;
}
```

CLIENT:



```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<sys/socket.h>
#include<sys/types.h>
#include<netinet/in.h>
#include<arpa/inet.h>
int main(int argc,char **argv){
  char *ip="127.0.0.1";
  int port=atoi(argv[1]);
  int sockfd;
  struct sockaddr_in addr;
  char buffer[1024];
  socklen_t addr_size;
  sockfd = socket(AF_INET,SOCK_DGRAM,0);
  memset(&addr,'\0',sizeof(addr));
  addr.sin_family=AF_INET;
  addr.sin_port=htons(port);
  addr.sin_addr.s_addr=inet_addr(ip);
  bzero(buffer, 1024);
  printf("Enter message to server: \n");
  scanf("%s",buffer);
  sendto(sockfd,buffer,1024,0,(struct sockaddr*)&addr,sizeof(addr));
  bzero(buffer, 1024);
  addr_size=sizeof(addr);
  recvfrom(sockfd,buffer,1024,0,(struct sockaddr*)&addr, &addr_size);
  printf("Message from server: %s\n", buffer);

  return 0;
}
```

OUTPUT:



```
vivekmunnaa@vivekmunnaa-VirtualBox:~$ cc udpclient.c
vivekmunnaa@vivekmunnaa-VirtualBox:~$ ./a.out 7878
Enter message to server:
hello
Message from server: hi
vivekmunnaa@vivekmunnaa-VirtualBox:~$
```

```
vivekmunnaa@vivekmunnaa-VirtualBox:~$ cc udpserver.c
vivekmunnaa@vivekmunnaa-VirtualBox:~$ ./a.out 7878
Message from client: hello
Message to client: hi
vivekmunnaa@vivekmunnaa-VirtualBox:~$
```
