# ADVANCED C PROGRAMMING ASSESSMENT MODULE 4

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

**Solution:**

**SERVER SIDE:**

**Step 1: Socket Creation**
Basic Component for sending and receiving of signals between two nodes (Communication).

**Step 2: Setsockopt**
Used to specify some options for the socket to control the behaviour of the socket.

**Step 3: Bind**
Bind function binds the socket to the address and port number specified in the addr(data Structure).

**Step 4: Listen**
Passive mode of the server, where it waits for the client to approach it.

**Step 5: Accept**
Server accepts the one of the connection requests from the queue of the listening socket.

**Step 6: Read**
Receive the message from the client.

**Step 7: Write**
Sends the message to the connected client.

**Step 8: Close**
Closes the file descriptor thus the server socket.

**CLIENT SIDE:**

**Step 1: Socket Creation**
Basic Component for sending and receiving of signals between two nodes (Communication).

**Step 2: Connect**
Connects the socket to by the file descriptor to the address  specified in addr(data structure).

**Step 3: Write**
Sends the message to the connected server.

**Step 4: Read**

Receive the message from the server.

**Step 5: Close File Descriptor (Socket)**

Closes the file descriptor thus the client socket.

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

**Solution:**

**Bind()** function binds the socket to the  address and port number specified in addr

int bind (int sockfd, const struct sockaddr *addr, socklen_t addrlen);
returns integer value, if 0 successful in binding else fails to bind.

## 3.What is Datagram Socket?

**Solution:**

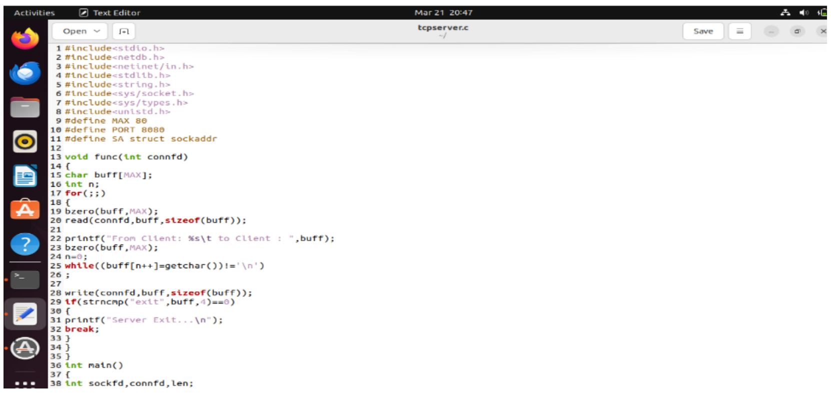
Datagram Socket is a connectionless socket, used for providing unreliable quick data packet transmission.

i. Best effort networking service.
ii. Packets may be lost or out of order.
iii. Applications: streaming, VOIP, SNMP, DNS

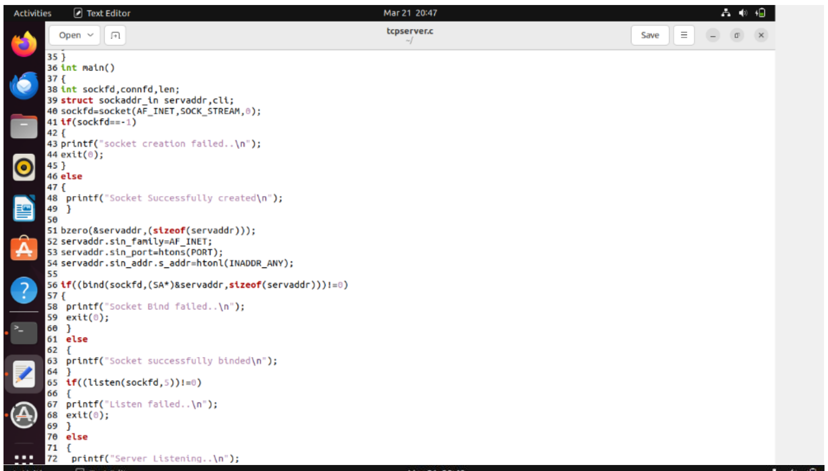
## 4.Write a server/client model socket program to exchange hello message between them.

**Solution:**

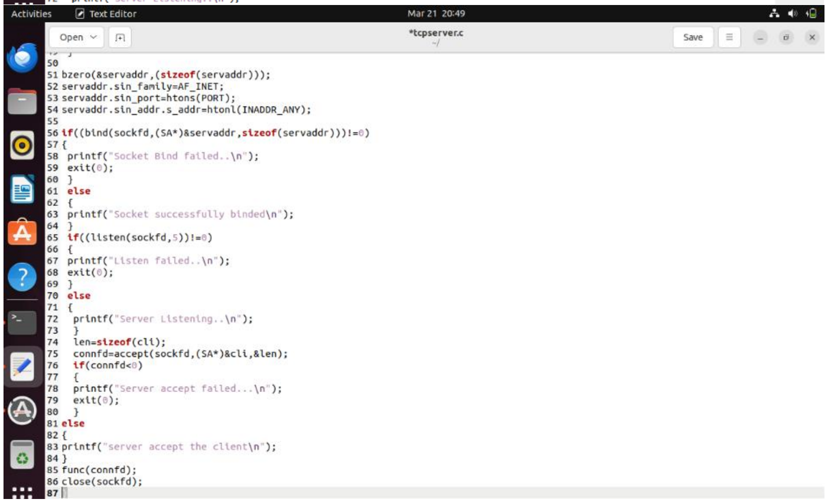
**SERVER-SIDE PROGRAM**



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

void func(int connfd)
{
char buff[MAX];
int n;
for(;;)
{
bzero(buff,MAX);
read(connfd,buff,sizeof(buff));

printf("From Client: %s\t to Client : ",buff);
bzero(buff,MAX);
n=0;
while((buff[n++]=getchar())!='\n')
;

write(connfd,buff,sizeof(buff));
if(strncmp("exit",buff,4)==0)
{
printf("Server Exit...\n");
break;
}
}
}
int main()
{
int sockfd,connfd,len;
```

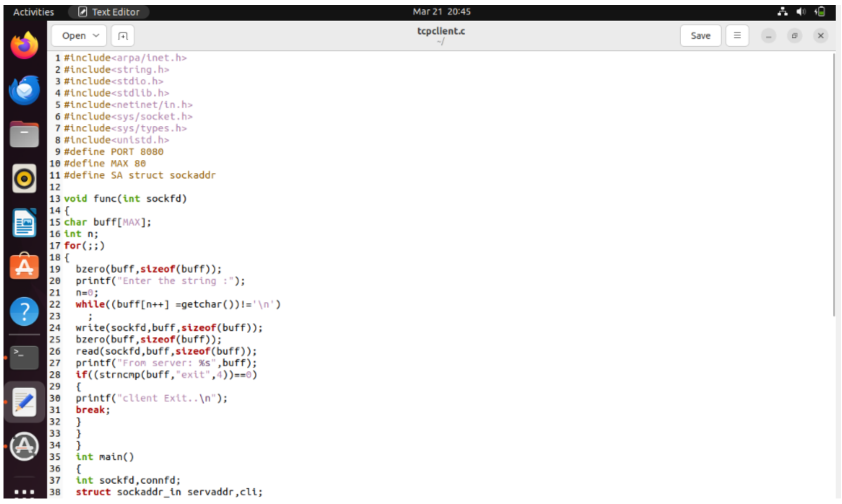


```c
}
int main()
{
int sockfd,connfd,len;
struct sockaddr_in servaddr,cli;
sockfd=socket(AF_INET,SOCK_STREAM,0);
if(sockfd==-1)
{
printf("socket creation failed..\n");
exit(0);
}
else
{
 printf("Socket Successfully created\n");
 }

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

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");
```

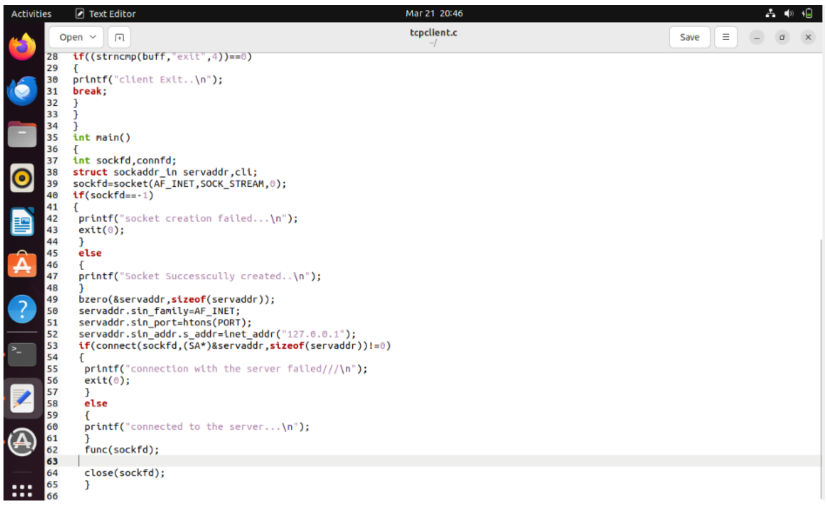


```c

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

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");
  }
  len=sizeof(cli);
  connfd=accept(sockfd,(SA*)&cli,&len);
  if(connfd<0)
  {
  printf("Server accept failed...\n");
  exit(0);
  }
else
{
printf("server accept the client\n");
}
func(connfd);
close(sockfd);
```

## CLIENT-SIDE PROGRAM



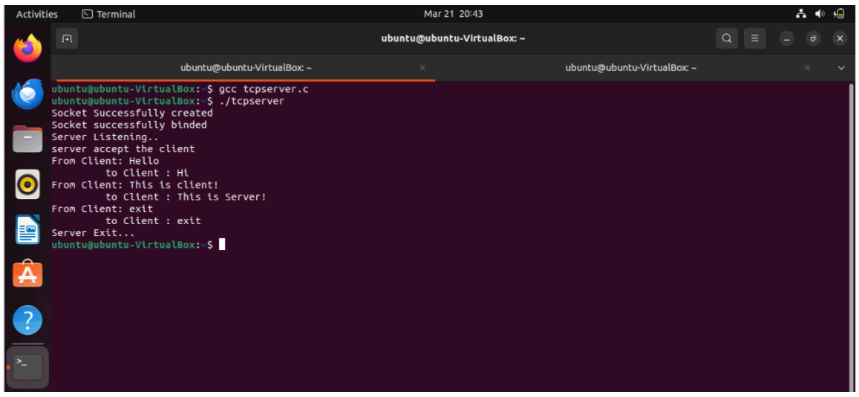
```c
#include<arpa/inet.h>
#include<string.h>
#include<stdio.h>
#include<stdlib.h>
#include<netinet/in.h>
#include<sys/socket.h>
#include<sys/types.h>
#include<unistd.h>
#define PORT 8080
#define MAX 80
#define SA struct sockaddr

void func(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;
```



```c
if((strncmp(buff,"exit",4))==0)
{
printf("client Exit..\n");
break;
}
}
}
int main()
{
int sockfd,connfd;
struct sockaddr_in servaddr,cli;
sockfd=socket(AF_INET,SOCK_STREAM,0);
if(sockfd==-1)
{
 printf("socket creation failed...\n");
 exit(0);
 }
 else
 {
 printf("Socket Successcully created..\n");
 }
 bzero(&servaddr,sizeof(servaddr));
 servaddr.sin_family=AF_INET;
 servaddr.sin_port=htons(PORT);
 servaddr.sin_addr.s_addr=inet_addr("127.0.0.1");
 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);
  }
```

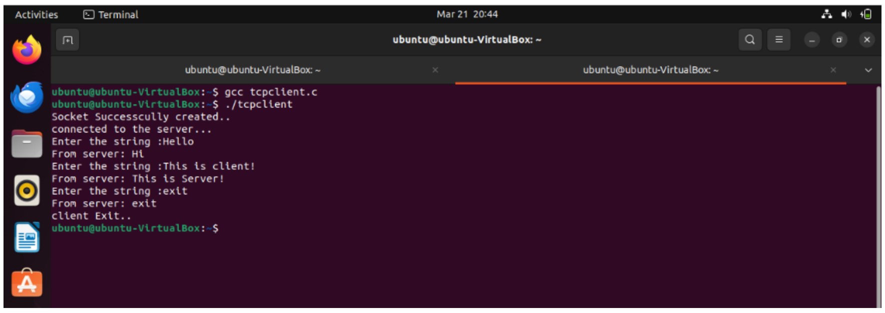
**OUTPUT**

**SERVER-SIDE**



```
ubuntu@ubuntu-VirtualBox:~$ gcc tcpserver.c
ubuntu@ubuntu-VirtualBox:~$ ./tcpserver
Socket Successfully created
Socket successfully binded
Server Listening..
server accept the client
From Client: Hello
         to Client : Hi
From Client: This is client!
         to Client : This is Server!
From Client: exit
         to Client : exit
Server Exit...
ubuntu@ubuntu-VirtualBox:~$ 
```

**CLIENT-SIDE**



```
ubuntu@ubuntu-VirtualBox:~$ gcc tcpclient.c
ubuntu@ubuntu-VirtualBox:~$ ./tcpclient
Socket Successcully created..
connected to the server...
Enter the string :Hello
From server: Hi
Enter the string :This is client!
From server: This is Server!
Enter the string :exit
From server: exit
client Exit..
ubuntu@ubuntu-VirtualBox:~$
```

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

Input: level

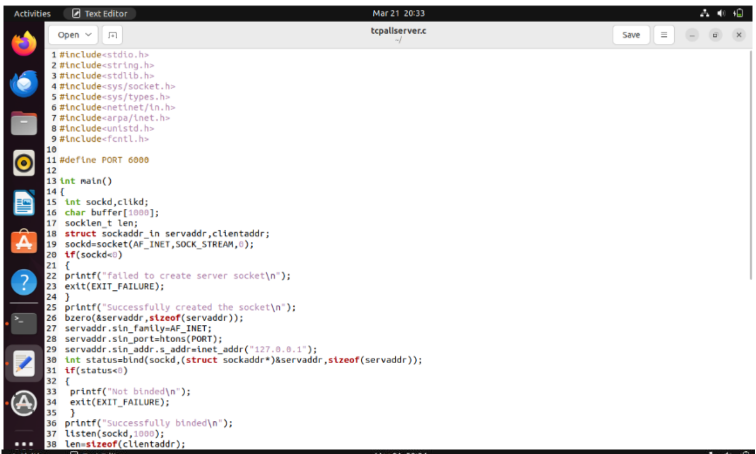
Output: Palindrome

Input: Assessment

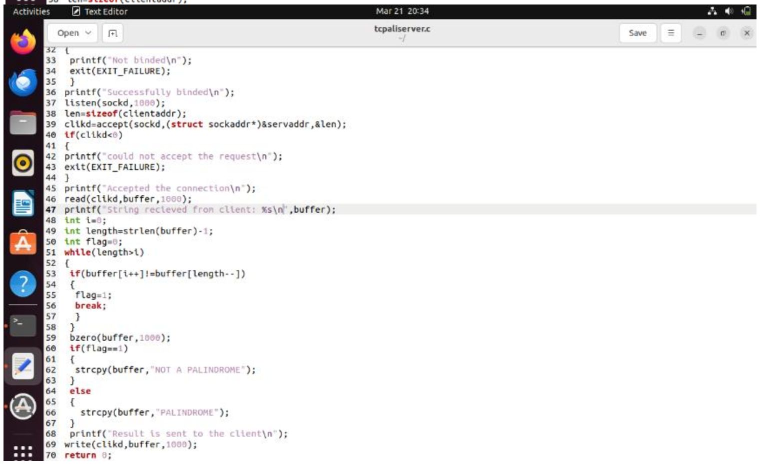
Output: Not a Palindrome

## Solution:

### SERVER-SIDE PROGRAM

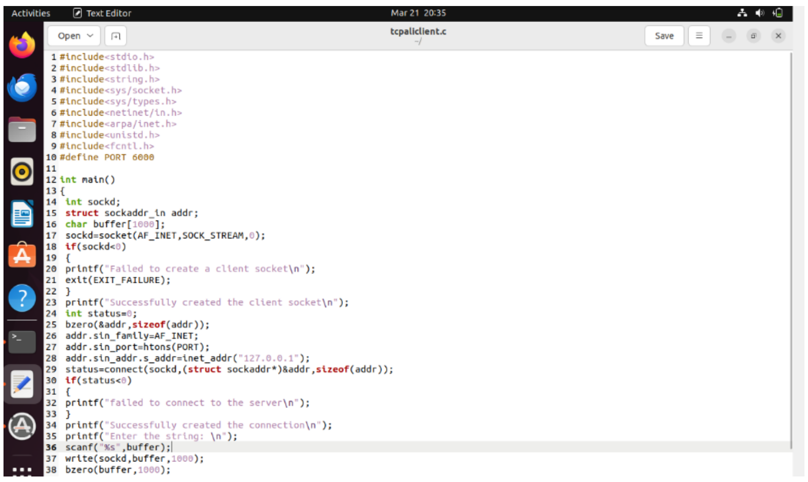


```c
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include<sys/socket.h>
#include<sys/types.h>
#include<netinet/in.h>
#include<arpa/inet.h>
#include<unistd.h>
#include<fcntl.h>

#define PORT 6000

int main()
{
 int sockd,clikd;
 char buffer[1000];
 socklen_t len;
 struct sockaddr_in servaddr,clientaddr;
 sockd=socket(AF_INET,SOCK_STREAM,0);
 if(sockd<0)
 {
 printf("failed to create server socket\n");
 exit(EXIT_FAILURE);
 }
 printf("Successfully created the socket\n");
 bzero(&servaddr,sizeof(servaddr));
 servaddr.sin_family=AF_INET;
 servaddr.sin_port=htons(PORT);
 servaddr.sin_addr.s_addr=inet_addr("127.0.0.1");
 int status=bind(sockd,(struct sockaddr*)&servaddr,sizeof(servaddr));
 if(status<0)
 {
  printf("Not binded\n");
  exit(EXIT_FAILURE);
  }
 printf("Successfully binded\n");
 listen(sockd,1000);
 len=sizeof(clientaddr);
```

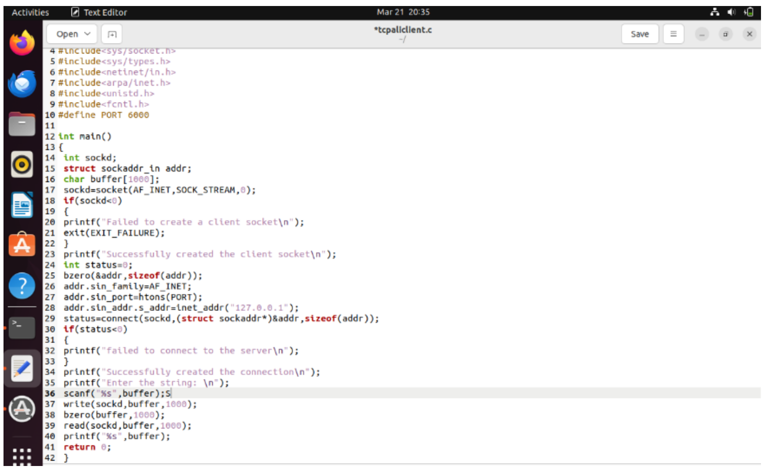


```c
 {
  printf("Not binded\n");
  exit(EXIT_FAILURE);
  }
 printf("Successfully binded\n");
 listen(sockd,1000);
 len=sizeof(clientaddr);
 clikd=accept(sockd,(struct sockaddr*)&servaddr,&len);
 if(clikd<0)
 {
 printf("could not accept the request\n");
 exit(EXIT_FAILURE);
 }
 printf("Accepted the connection\n");
 read(clikd,buffer,1000);
 printf("String recieved from client: %s\n",buffer);
 int i=0;
 int length=strlen(buffer)-1;
 int flag=0;
 while(length>i)
 {
  if(buffer[i++]!=buffer[length--])
  {
   flag=1;
   break;
   }
  }
  bzero(buffer,1000);
  if(flag==1)
  {
   strcpy(buffer,"NOT A PALINDROME");
  }
  else
  {
    strcpy(buffer,"PALINDROME");
  }
  printf("Result is sent to the client\n");
 write(clikd,buffer,1000);
 return 0;
```

### CLIENT-SIDE PROGRAM



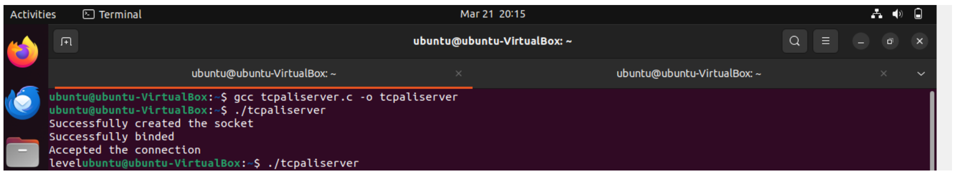
```c
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<sys/socket.h>
#include<sys/types.h>
#include<netinet/in.h>
#include<arpa/inet.h>
#include<unistd.h>
#include<fcntl.h>
#define PORT 6000

int main()
{
 int sockd;
 struct sockaddr_in addr;
 char buffer[1000];
 sockd=socket(AF_INET,SOCK_STREAM,0);
 if(sockd<0)
 {
 printf("Failed to create a client socket\n");
 exit(EXIT_FAILURE);
 }
 printf("Successfully created the client socket\n");
 int status=0;
 bzero(&addr,sizeof(addr));
 addr.sin_family=AF_INET;
 addr.sin_port=htons(PORT);
 addr.sin_addr.s_addr=inet_addr("127.0.0.1");
 status=connect(sockd,(struct sockaddr*)&addr,sizeof(addr));
 if(status<0)
 {
 printf("failed to connect to the server\n");
 }
 printf("Successfully created the connection\n");
 printf("Enter the string: \n");
 scanf("%s",buffer);
 write(sockd,buffer,1000);
 bzero(buffer,1000);
```



```c
#include<sys/socket.h>
#include<sys/types.h>
#include<netinet/in.h>
#include<arpa/inet.h>
#include<unistd.h>
#include<fcntl.h>
#define PORT 6000

int main()
{
 int sockd;
 struct sockaddr_in addr;
 char buffer[1000];
 sockd=socket(AF_INET,SOCK_STREAM,0);
 if(sockd<0)
 {
 printf("Failed to create a client socket\n");
 exit(EXIT_FAILURE);
 }
 printf("Successfully created the client socket\n");
 int status=0;
 bzero(&addr,sizeof(addr));
 addr.sin_family=AF_INET;
 addr.sin_port=htons(PORT);
 addr.sin_addr.s_addr=inet_addr("127.0.0.1");
 status=connect(sockd,(struct sockaddr*)&addr,sizeof(addr));
 if(status<0)
 {
 printf("failed to connect to the server\n");
 }
 printf("Successfully created the connection\n");
 printf("Enter the string: \n");
 scanf("%s",buffer);S
 write(sockd,buffer,1000);
 bzero(buffer,1000);
 read(sockd,buffer,1000);
 printf("%s",buffer);
 return 0;
 }
```

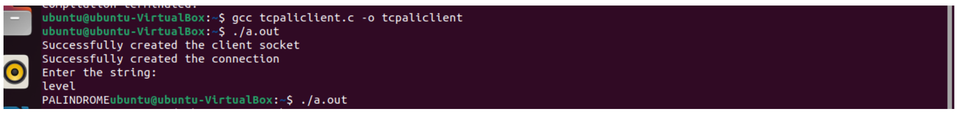
**OUTPUT**

**SERVER-SIDE**



```
ubuntu@ubuntu-VirtualBox:~$ gcc tcpaliserver.c -o tcpaliserver
ubuntu@ubuntu-VirtualBox:~$ ./tcpaliserver
Successfully created the socket
Successfully binded
Accepted the connection
levelubuntu@ubuntu-VirtualBox:~$ ./tcpaliserver
```

**CLIENT-SIDE**

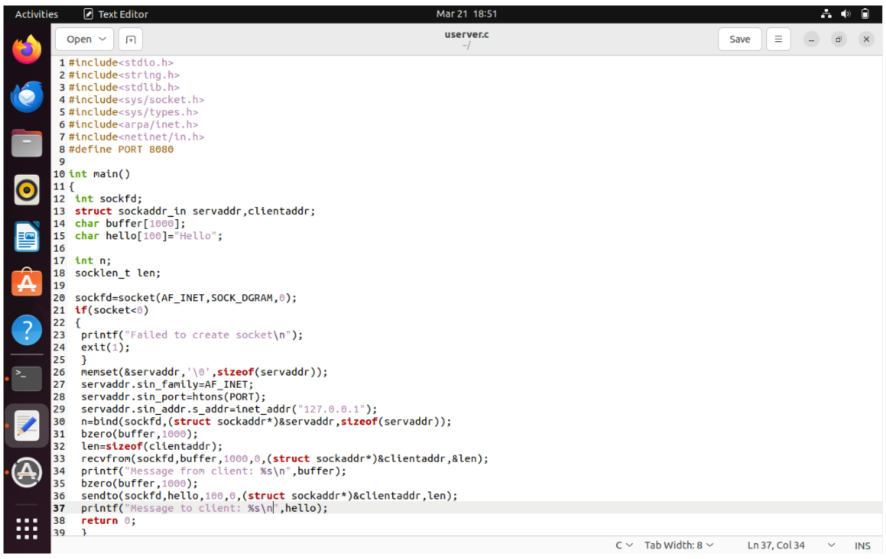


```
ubuntu@ubuntu-VirtualBox:~$ gcc tcpaliclient.c -o tcpaliclient
ubuntu@ubuntu-VirtualBox:~$ ./a.out
Successfully created the client socket
Successfully created the connection
Enter the string:
level
PALINDROMEubuntu@ubuntu-VirtualBox:~$ ./a.out
```

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

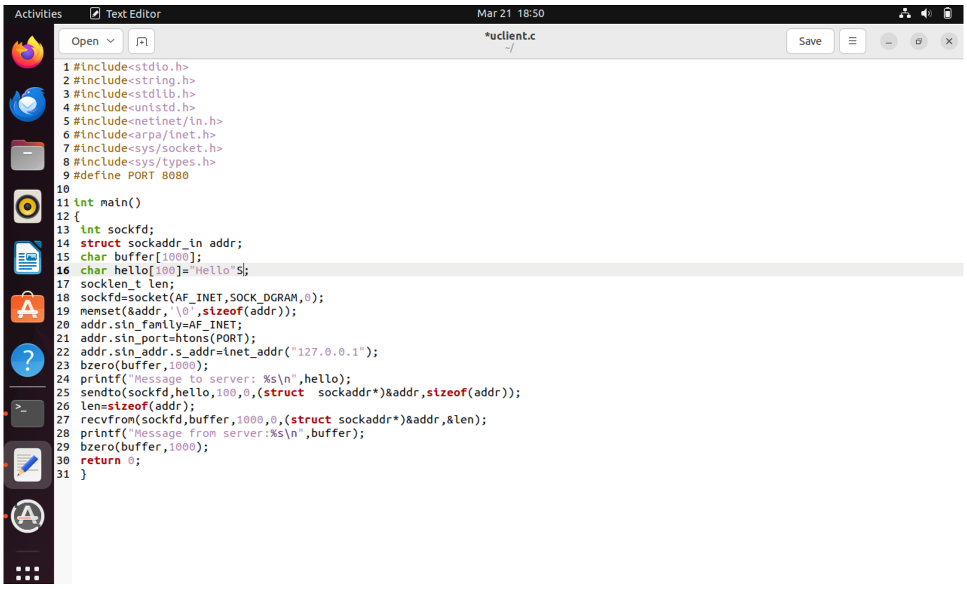
## Solution:

**SERVER-SIDE PROGRAM**



```c
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include<sys/socket.h>
#include<sys/types.h>
#include<arpa/inet.h>
#include<netinet/in.h>
#define PORT 8080

int main()
{
 int sockfd;
 struct sockaddr_in servaddr,clientaddr;
 char buffer[1000];
 char hello[100]="Hello";

 int n;
 socklen_t len;

 sockfd=socket(AF_INET,SOCK_DGRAM,0);
 if(socket<0)
 {
  printf("Failed to create socket\n");
  exit(1);
  }
  memset(&servaddr,'\0',sizeof(servaddr));
  servaddr.sin_family=AF_INET;
  servaddr.sin_port=htons(PORT);
  servaddr.sin_addr.s_addr=inet_addr("127.0.0.1");
  n=bind(sockfd,(struct sockaddr*)&servaddr,sizeof(servaddr));
  bzero(buffer,1000);
  len=sizeof(clientaddr);
  recvfrom(sockfd,buffer,1000,0,(struct sockaddr*)&clientaddr,&len);
  printf("Message from client: %s\n",buffer);
  bzero(buffer,1000);
  sendto(sockfd,hello,100,0,(struct sockaddr*)&clientaddr,len);
  printf("Message to client: %s\n",hello);
  return 0;
  }
```

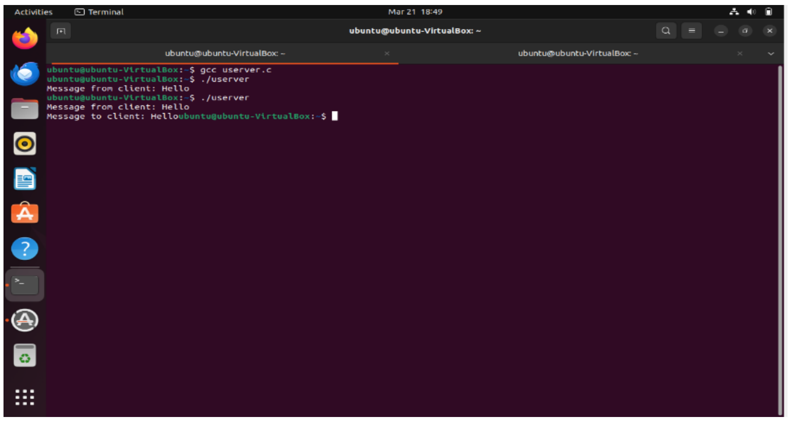
## CLIENT-SIDE PROGRAM



```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include<unistd.h>
#include<netinet/in.h>
#include<arpa/inet.h>
#include<sys/socket.h>
#include<sys/types.h>
#define PORT 8080

int main()
{
 int sockfd;
 struct sockaddr_in addr;
 char buffer[1000];
 char hello[100]="Hello"S;
 socklen_t len;
 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("127.0.0.1");
 bzero(buffer,1000);
 printf("Message to server: %s\n",hello);
 sendto(sockfd,hello,100,0,(struct  sockaddr*)&addr,sizeof(addr));
 len=sizeof(addr);
 recvfrom(sockfd,buffer,1000,0,(struct sockaddr*)&addr,&len);
 printf("Message from server:%s\n",buffer);
 bzero(buffer,1000);
 return 0;
 }
```

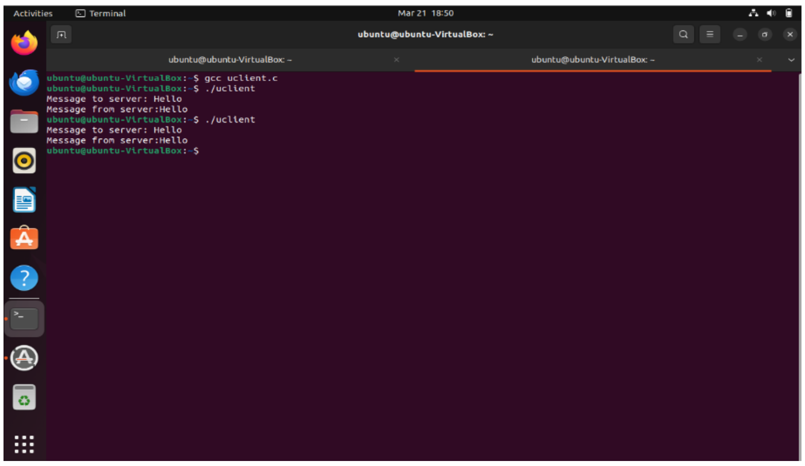
## OUTPUT

## SERVER-SIDE



```
ubuntu@ubuntu-VirtualBox:~$ gcc userver.c
ubuntu@ubuntu-VirtualBox:~$ ./userver
Message from client: Hello
ubuntu@ubuntu-VirtualBox:~$ ./userver
Message from client: Hello
Message to client: Helloubuntu@ubuntu-VirtualBox:~$
```

## CLIENT-SIDE



```
ubuntu@ubuntu-VirtualBox:~$ gcc uclient.c
ubuntu@ubuntu-VirtualBox:~$ ./uclient
Message to server: Hello
Message from server:Hello
ubuntu@ubuntu-VirtualBox:~$ ./uclient
Message to server: Hello
Message from server:Hello
ubuntu@ubuntu-VirtualBox:~$
```