

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

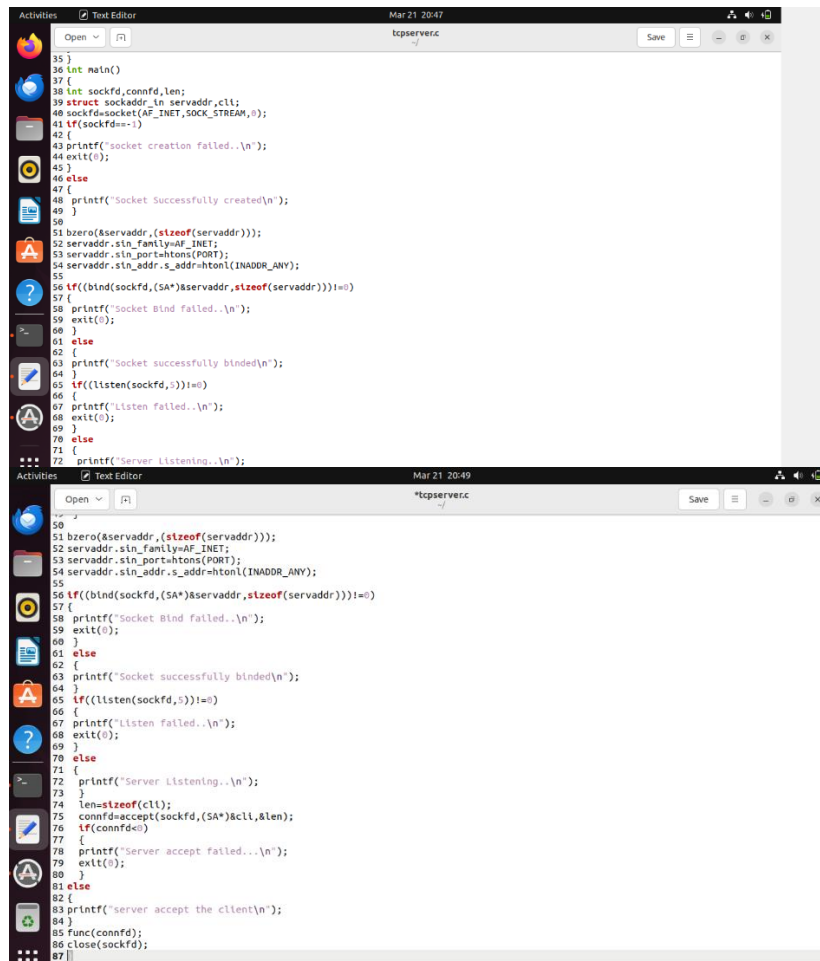
4.Write a server/client model socket program to exchange hello message between them.

Solution:

SERVER-SIDE PROGRAM

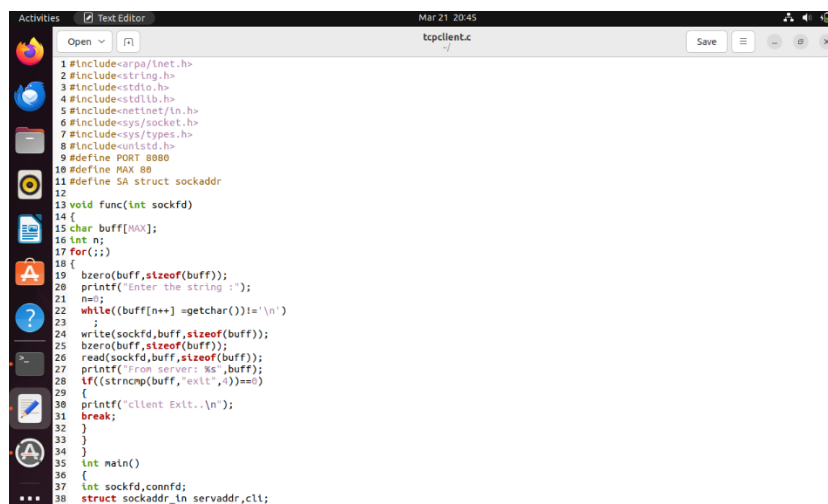


```
1 #include<stdio.h>
2 #include<netdb.h>
3 #include<netinet/in.h>
4 #include<stdlib.h>
5 #include<string.h>
6 #include<sys/socket.h>
7 #include<sys/types.h>
8 #include<unistd.h>
9 #define MAX 80
10 #define PORT 8080
11 #define SA struct sockaddr
12
13 void func(int sockfd)
14 {
15     char buff[MAX];
16     int n;
17     for(;;)
18     {
19         bzero(buff,MAX);
20         read(sockfd,buff,sizeof(buff));
21         printf("From Client: %s\t to Client : ",buff);
22         bzero(buff,MAX);
23         while((buff[n++]=getchar())!='\n')
24             ;
25         write(sockfd,buff,sizeof(buff));
26         if(strcmp(buff,"exit")==0)
27             break;
28         printf("Server Exit...\n");
29     }
30 }
31
32 int main()
33 {
34     int sockfd,connfd,len;
35     struct sockaddr_in serv_addr,cli_addr;
```




```
35 }
36 int main()
37 {
38     int sockfd, connfd, len;
39     struct sockaddr_in servaddr, cli;
40     sockfd=socket(AF_INET, SOCK_STREAM, 0);
41     if(sockfd==-1)
42     {
43         printf("socket creation failed..\n");
44         exit(0);
45     }
46     else
47     {
48         printf("Socket Successfully created\n");
49     }
50     bzero(servaddr, (sizeof(servaddr)));
51     servaddr.sin_family=AF_INET;
52     servaddr.sin_port=htons(PORT);
53     servaddr.sin_addr.s_addr=htonl(INADDR_ANY);
54     if(bind(sockfd, (SA*)&servaddr, sizeof(servaddr)))!=0)
55     {
56         printf("Socket Bind failed..\n");
57         exit(0);
58     }
59     else
60     {
61         printf("Socket successfully binded\n");
62     }
63     if(listen(sockfd, 5))!=0
64     {
65         printf("listen failed..\n");
66         exit(0);
67     }
68     else
69     {
70         printf("Server Listening..\n");
71     }
72 }
50
51 bzero(servaddr, (sizeof(servaddr)));
52 servaddr.sin_family=AF_INET;
53 servaddr.sin_port=htons(PORT);
54 servaddr.sin_addr.s_addr=htonl(INADDR_ANY);
55
56 if(bind(sockfd, (SA*)&servaddr, sizeof(servaddr)))!=0)
57 {
58     printf("Socket Bind failed..\n");
59     exit(0);
60 }
61 else
62 {
63     printf("Socket successfully binded\n");
64 }
65 if(listen(sockfd, 5))!=0
66 {
67     printf("listen failed..\n");
68     exit(0);
69 }
70 else
71 {
72     printf("Server Listening..\n");
73 }
74 len=sizeof(cli);
75 connfd=accept(sockfd, (SA*)&cli, &len);
76 if(connfd==0)
77 {
78     printf("Server accept failed...\n");
79     exit(0);
80 }
81 else
82 {
83     printf("server accept the client\n");
84 }
85 func(connfd);
86 close(sockfd);
87 }
```

CLIENT-SIDE PROGRAM



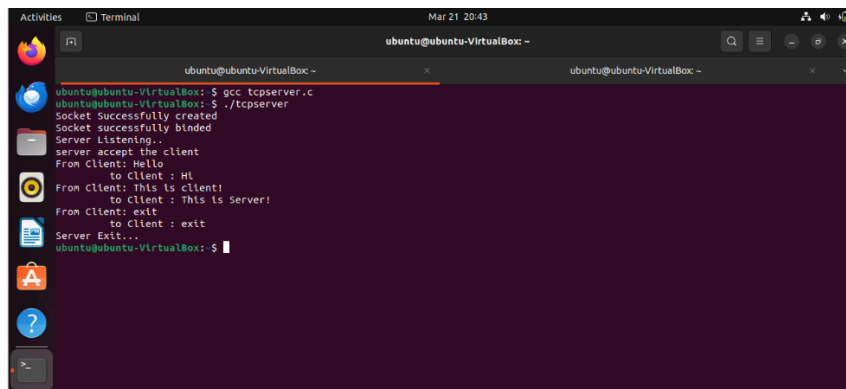
```
1 #include <arpa/inet.h>
2 #include <string.h>
3 #include <stdio.h>
4 #include <stdlib.h>
5 #include <netinet/in.h>
6 #include <sys/socket.h>
7 #include <sys/types.h>
8 #include <unistd.h>
9 #define PORT 8080
10 #define MAX 80
11 #define SA struct sockaddr
12
13 void func(int sockfd)
14 {
15     char buff[MAX];
16     int n;
17     for(;;)
18     {
19         bzero(buff, sizeof(buff));
20         printf("Enter the string : ");
21         n=0;
22         while((buff[n++] = getchar()) != '\n')
23             ;
24         write(sockfd, buff, sizeof(buff));
25         bzero(buff, sizeof(buff));
26         read(sockfd, buff, sizeof(buff));
27         printf("From server: %s", buff);
28         if(strcmp(buff, "exit") == 0)
29             break;
30         printf("client Exit..\n");
31     }
32 }
33
34 int main()
35 {
36     int sockfd, connfd;
37     struct sockaddr_in servaddr, cli;
```



```
28 if((strcmp(buff,"exit")==0))
29 {
30 printf("client Exit.\n");
31 break;
32 }
33 }
34 }
35 int main()
36 {
37 int sockfd,connfd;
38 struct sockaddr_in servaddr,cli;
39 sockfd=socket(AF_INET,SOCK_STREAM,0);
40 if(sockfd==-1)
41 {
42 printf("socket creation failed...\n");
43 exit(0);
44 }
45 else
46 {
47 printf("socket Successfully created.\n");
48 }
49 bzero(&servaddr,sizeof(servaddr));
50 servaddr.sin_family=AF_INET;
51 servaddr.sin_port=htonS(PORT);
52 servaddr.sin_addr.s_addr=inet_addr("127.0.0.1");
53 if(connect(sockfd,(SA*)&servaddr,sizeof(servaddr))!=0)
54 {
55 printf("connection with the server failed///\n");
56 exit(0);
57 }
58 else
59 {
60 printf("connected to the server...\n");
61 }
62 func(sockfd);
63 close(sockfd);
64 }
65 }
66 }
```

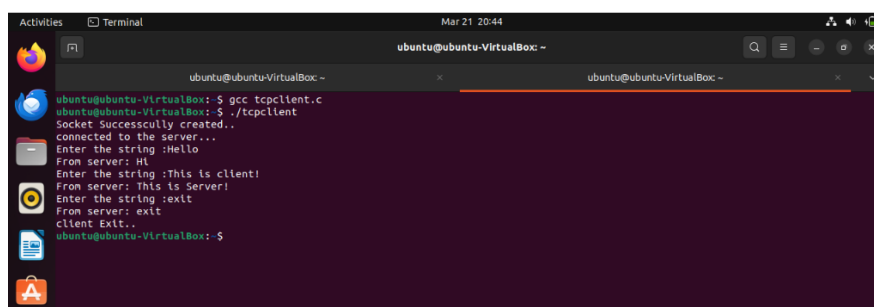
OUTPUT

SERVER-SIDE



```
ubuntu@ubuntu-VirtualBox: ~
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: ~
ubuntu@ubuntu-VirtualBox: ~$ gcc tcpclient.c
ubuntu@ubuntu-VirtualBox: ~$ ./tcpclient
Socket Successfully 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

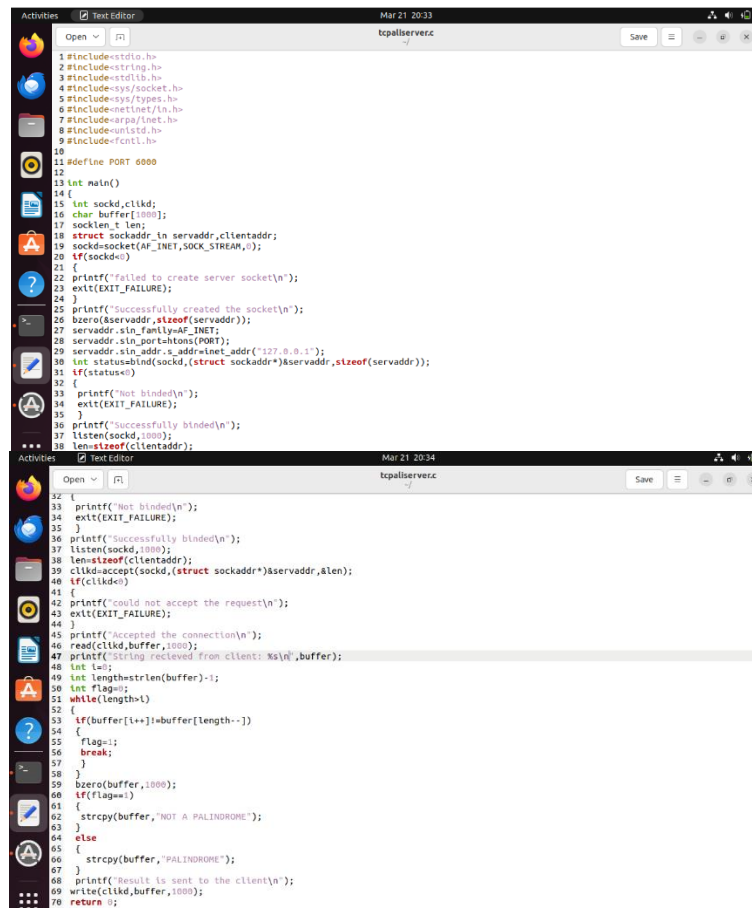
Output: Palindrome

Input: Assessment

Output: Not a Palindrome

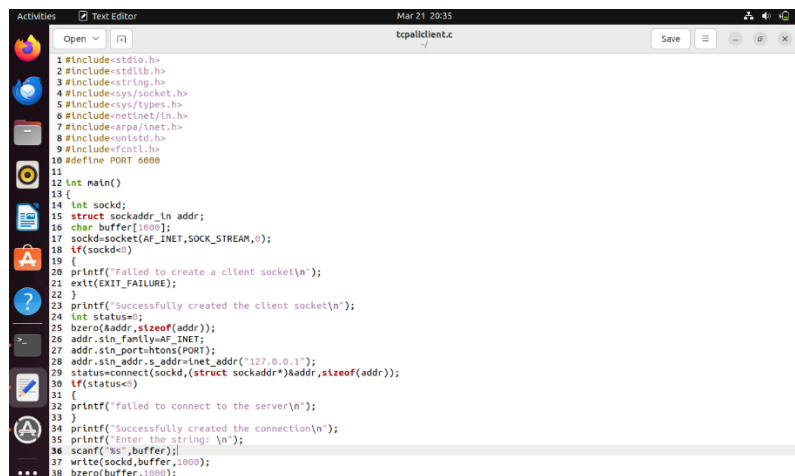
Solution:

SERVER-SIDE PROGRAM



```
1 #include<stdio.h>
2 #include<string.h>
3 #include<stdlib.h>
4 #include<sys/socket.h>
5 #include<sys/types.h>
6 #include<netinet/in.h>
7 #include<arpa/inet.h>
8 #include<unistd.h>
9 #include<fcntl.h>
10
11 #define PORT 6000
12
13 int main()
14 {
15     int sockd, cliId;
16     char buffer[1000];
17     socklen_t len;
18     struct sockaddr_in servaddr, clientaddr;
19     sockd=socket(AF_INET, SOCK_STREAM, 0);
20     if(sockd<0)
21     {
22         printf("Failed to create server socket\n");
23         exit(EXIT_FAILURE);
24     }
25     printf("Successfully created the socket\n");
26     bzero(servaddr, sizeof(servaddr));
27     servaddr.sin_family=AF_INET;
28     servaddr.sin_port=htons(PORT);
29     servaddr.sin_addr.s_addr=inet_addr("127.0.0.1");
30     int status=bind(sockd, (struct sockaddr*)&servaddr, sizeof(servaddr));
31     if(status<0)
32     {
33         printf("Not binded\n");
34         exit(EXIT_FAILURE);
35     }
36     printf("Successfully binded\n");
37     listen(sockd, 1000);
38     len=sizeof(clientaddr);
39     cliId=accept(sockd, (struct sockaddr*)&servaddr, &len);
40     if(cliId<0)
41     {
42         printf("could not accept the request\n");
43         exit(EXIT_FAILURE);
44     }
45     printf("Accepted the connection\n");
46     read(cliId, buffer, 1000);
47     printf("String recieved from client: %s\n", buffer);
48     int len;
49     int length=strlen(buffer)-1;
50     int flag=0;
51     while(length>0)
52     {
53         if(buffer[i++]==buffer[length--])
54         {
55             flag=1;
56             break;
57         }
58     }
59     bzero(buffer, 1000);
60     if(flag==1)
61     {
62         strcpy(buffer, "NOT A PALINDROME");
63     }
64     else
65     {
66         strcpy(buffer, "PALINDROME");
67     }
68     printf("Result is sent to the client\n");
69     write(cliId, buffer, 1000);
70     return 0;
}
```

CLIENT-SIDE PROGRAM



```
1 #include<stdio.h>
2 #include<stdlib.h>
3 #include<string.h>
4 #include<sys/socket.h>
5 #include<sys/types.h>
6 #include<netinet/in.h>
7 #include<arpa/inet.h>
8 #include<unistd.h>
9 #include<fcntl.h>
10 #define PORT 6000
11
12 int main()
13 {
14     int sockd;
15     struct sockaddr_in addr;
16     char buffer[1000];
17     sockd=socket(AF_INET, SOCK_STREAM, 0);
18     if(sockd<0)
19     {
20         printf("Failed to create a client socket\n");
21         exit(EXIT_FAILURE);
22     }
23     printf("Successfully created the client socket\n");
24     int status=0;
25     bzero(&addr, sizeof(addr));
26     addr.sin_family=AF_INET;
27     addr.sin_port=htons(PORT);
28     addr.sin_addr.s_addr=inet_addr("127.0.0.1");
29     status=connect(sockd, (struct sockaddr*)&addr, sizeof(addr));
30     if(status<0)
31     {
32         printf("Failed to connect to the server\n");
33     }
34     printf("Successfully created the connection\n");
35     printf("Enter the string: \n");
36     scanf("%s", buffer);
37     write(sockd, buffer, 1000);
38     bzero(buffer, 1000);
}
```

```

1 #include<sys/socket.h>
2 #include<sys/types.h>
3 #include<netinet/in.h>
4 #include<arpa/inet.h>
5 #include<unistd.h>
6 #include<fcntl.h>
7 #define PORT 6000
8
9 int main()
10 {
11     int sockd;
12     struct sockaddr_in addr;
13     char buffer[1000];
14     sockd=socket(AF_INET,SOCK_STREAM,0);
15     if(sockd<0)
16     {
17         printf("Failed to create a client socket\n");
18         exit(EXIT_FAILURE);
19     }
20     printf("Successfully created the client socket\n");
21     int status=0;
22     bzero(&addr,sizeof(addr));
23     addr.sin_family=AF_INET;
24     addr.sin_port=htons(PORT);
25     addr.sin_addr.s_addr=inet_addr("127.0.0.1");
26     status=connect(sockd,(struct sockaddr*)&addr,sizeof(addr));
27     if(status<0)
28     {
29         printf("Failed to connect to the server\n");
30     }
31     printf("Successfully created the connection\n");
32     printf("Enter the string: \n");
33     scanf("%s",buffer);
34     write(sockd,buffer,1000);
35     bzero(buffer,1000);
36     read(sockd,buffer,1000);
37     printf("%s",buffer);
38     return 0;
39 }

```

OUTPUT

SERVER-SIDE

```

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

```

CLIENT-SIDE

```

ubuntu@ubuntu-VirtualBox: ~$ gcc tcpalclient.c -o tcpalclient
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

```

1 #include<stdio.h>
2 #include<string.h>
3 #include<stdlib.h>
4 #include<sys/socket.h>
5 #include<sys/types.h>
6 #include<arpa/inet.h>
7 #include<netinet/in.h>
8 #define PORT 8080
9
10 int main()
11 {
12     int sockfd;
13     struct sockaddr_in servaddr,clientaddr;
14     char buffer[1000];
15     char hello[100]="Hello";
16
17     int n;
18     socklen_t len;
19
20     sockfd=socket(AF_INET,SOCK_DGRAM,0);
21     if(sockfd<0)
22     {
23         printf("Failed to create socket\n");
24         exit(1);
25     }
26     memset(&servaddr, '\0', sizeof(servaddr));
27     servaddr.sin_family=AF_INET;
28     servaddr.sin_port=htons(PORT);
29     servaddr.sin_addr.s_addr=inet_addr("127.0.0.1");
30     n=bind(sockfd,(struct sockaddr*)&servaddr,sizeof(servaddr));
31     len=sizeof(clientaddr);
32     recvfrom(sockfd,buffer,1000,0,(struct sockaddr*)&clientaddr,&len);
33     printf("Message from client: %s\n",buffer);
34     bzero(buffer,1000);
35     sendto(sockfd,hello,100,0,(struct sockaddr*)&clientaddr,len);
36     printf("Message to client: %s\n",hello);
37     return 0;
38 }

```

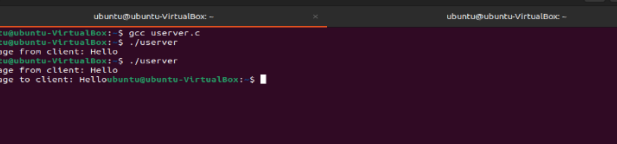
CLIENT-SIDE PROGRAM

The image shows a Linux desktop environment. At the top, there is a panel with the 'Activities' button, a 'Text Editor' window title, and a system clock showing 'Mar 21 18:50'. The desktop background is a solid light blue. On the left side, there is a vertical dock containing several application icons: a red and white Firefox icon, a blue and white Telegram icon, a grey Dash icon, a yellow and black Nautilus icon, a blue and white LibreOffice Writer icon, an orange and white App Store icon, a blue and white question mark icon, a blue and white terminal icon, a blue and white code editor icon, and a blue and white application icon. The main window is a 'Text Editor' titled '*uclient.c'. It has a menu bar with 'Open', a file icon, and 'Save'. The text area contains a C program for a simple TCP server. The code is as follows:

```
1 #include<stdio.h>
2 #include<string.h>
3 #include<stdlib.h>
4 #include<unistd.h>
5 #include<netinet/in.h>
6 #include<arpa/inet.h>
7 #include<sys/socket.h>
8 #include<sys/types.h>
9 #define PORT 8080
10
11 int main()
12 {
13     int sockfd;
14     struct sockaddr_in addr;
15     char buffer[100];
16     char hello[100]="Hello'S!";
17     socklen_t len;
18     sockfd=socket(AF_INET,SOCK_DGRAM,0);
19     memset(&addr,'\0',sizeof(addr));
20     addr.sin_family=AF_INET;
21     addr.sin_port=htons(PORT);
22     addr.sin_addr.s_addr=inet_addr("127.0.0.1");
23     bzero(buffer,1000);
24     printf("Message to server: %s\n",hello);
25     sendto(sockfd,hello,100,0,(struct sockaddr*)&addr,sizeof(addr));
26     len=sizeof(addr);
27     recvfrom(sockfd,buffer,1000,0,(struct sockaddr*)&addr,&len);
28     printf("Message from server:%s\n",buffer);
29     bzero(buffer,1000);
30     return 0;
31 }
```

OUTPUT

SERVER-SIDE

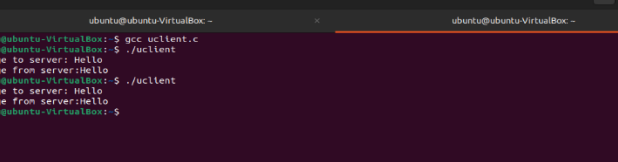


The screenshot shows a terminal window titled 'Activities' and 'Terminal'. The terminal is running on an Ubuntu virtual machine, as indicated by the title bar 'ubuntu@ubuntu-VirtualBox ~'. The user has executed the following commands and received the following output:

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

The terminal window is part of a desktop environment with a sidebar on the left containing various application icons. The terminal output is displayed in a dark background with light-colored text.

CLIENT-SIDE



The screenshot shows a Linux desktop environment. On the left is a vertical dock with icons for Activities, Firefox, LibreOffice Writer, and several other applications. The top panel displays the date and time as 'Mar 21 18:50'. A terminal window is open, titled 'ubuntu@ubuntu-VirtualBox: ~'. The terminal shows the following commands and output:

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