PART II

SOCKET PROGRAMMING IN UNIX

Experiment No:1	UNIDIRECTIONAL CHAT USING TCP
Date:06/12/2018	

AIM

To write a program to implement unidirectional chat using tcp.

ALGORITHM

Step1: Start

Step2: Create client tcp socket on client and server tcp socket on server.

Step3: Bind server socket.

Step4: Make server socket on listen mode.

Step5: Connect client socket with server socket in client side

Step6: On server, accept the client socket connection request.

Step7: On client, read the message from user.

Step8: Send the message to the socket on client side.

Step10: On serverside, receive the message from socket and display it.

Step11: End.

PROGRAM

CLIENT SIDE

```
#include<stdio.h>
#include<sys/types.h>
#include<netinet/in.h>
#include<sys/socket.h>
#include<string.h>
void main()
{
intcsd,len;
charsendmsg[30],recvmsg[20];
structsockaddr_incliaddr,servaddr;
csd=socket(AF_INET,SOCK_STREAM,0);
servaddr.sin_family=AF_INET;
servaddr.sin_addr.s_addr=htonl(INADDR_ANY);
servaddr.sin_port=htons(33345);
connect(csd,(structsockaddr*)&servaddr,sizeof(servaddr));
printf("\nEnter the message:\n");
fgets(sendmsg,20,stdin);
len=strlen(sendmsg);
sendmsg[len-1]='\0';
send(csd,sendmsg,20,0);
}
```

SERVER SIDE

```
#include<stdio.h>
#include<sys/types.h>
#include<netinet/in.h>
#include<sys/socket.h>
#include<string.h>
void main()
{
intsd,len,nsd,clilen;
charsendmsg[30],recvmsg[20];
structsockaddr_incliaddr,servaddr;
sd=socket(AF_INET,SOCK_STREAM,0);
servaddr.sin_family=AF_INET;
servaddr.sin_addr.s_addr=htonl(INADDR_ANY);
servaddr.sin_port=htons(33345);
bind(sd,(structsockaddr*)&servaddr,sizeof(servaddr));
listen(sd,5);
clilen=sizeof(cliaddr);
nsd=accept(sd,(structsockaddr*)&cliaddr,&clilen);
printf("\nReceived string:");
recv(nsd,recvmsg,20,0);
printf("%s\n",recvmsg);
}
```

OUTPUT

CLIENT SIDE

```
user2@user-HP-15-Notebook-PC: ~/Desktop
user2@user-HP-15-Notebook-PC:~$ cd Desktop
user2@user-HP-15-Notebook-PC:~/Desktop$  gcc tcpcr.c -o n
user2@user-HP-15-Notebook-PC:~/Desktop$ ./n

Enter the message:
HAI ! MY SOULMATE
user2@user-HP-15-Notebook-PC:~/Desktop$
```

SERVER SIDE

```
user2@user-HP-15-Notebook-PC: ~/Desktop
user2@user-HP-15-Notebook-PC:~$ cd Desktop
user2@user-HP-15-Notebook-PC:~/Desktop$ gcc tcpsr.c -o d
user2@user-HP-15-Notebook-PC:~/Desktop$ ./d

Received string:HAI ! MY SOULMATE
user2@user-HP-15-Notebook-PC:~/Desktop$
```

RESULT

The program is compiled and output is obtained successfully.

Experiment No:2	BIDIRECTIONAL CHAT USING TCP
Date:06/12/2018	

AIM

To write a program to implement Bidirectional chat using TCP.

ALGORITHM

Step1: Start

Step2: Create client tcp socket on client and server tcp socket on server.

Step3: Bind server socket.

Step4: Make server socket on listen mode.

Step5: Connect client socket with server socket in client side

Step6: On server, accept the client socket connection request.

Step7: Repeat step8 to step14.

Step8: Read input from user on client.

Step9: Send message to server.

Step10: Receive message on server side.

Step11: Display message on server side.

Step12: Read the input from the user on server side.

Step13: Send this message to client.

Step14: Receive the message on client side and display the message.

Step15: End.

PROGRAM

CLIENT SIDE

```
#include<stdio.h>
#include<sys/types.h>
#include<netinet/in.h>
#include<sys/socket.h>
#include<string.h>
void main()
{
intcsd,len;
charsendmsg[30],recvmsg[20];
structsockaddr_incliaddr,servaddr;
csd=socket(AF_INET,SOCK_STREAM,0);
servaddr.sin_family=AF_INET;
servaddr.sin_addr.s_addr=htonl(INADDR_ANY);
servaddr.sin_port=htons(33345);
connect(csd,(structsockaddr*)&servaddr,sizeof(servaddr));
printf("\n messages:\n");
do
{
fgets(sendmsg,20,stdin);
len=strlen(sendmsg);
sendmsg[len-1]='\0';
send(csd,sendmsg,20,0);
```

```
wait(20);
recv(csd,recvmsg,20,0);
printf("%s\n",recvmsg);
}
while(strcmp(recvmsg,”bye”)!=0);
}
```

SERVER SIDE

```
#include<stdio.h>
#include<sys/types.h>
#include<netinet/in.h>
#include<sys/socket.h>
#include<string.h>
void main()
{
intsd,len,nsd,clilen;
charsendmsg[30],recvmsg[20];
structsockaddr_incliaddr,servaddr;
sd=socket(AF_INET,SOCK_STREAM,0);
servaddr.sin_family=AF_INET;
servaddr.sin_addr.s_addr=htonl(INADDR_ANY);
servaddr.sin_port=htons(33345);
bind(sd,(structsockaddr*)&servaddr,sizeof(servaddr));
listen(sd,5);
clilen=sizeof(cliaddr);
```

```
nsd=accept(sd,(structsockaddr*)&cliaddr,&clilen);
printf("\messages:");
do
{
recv(nsd,recvmsg,20,0);
printf("%s\n",recvmsg);
fgets(sendmsg,20,stdin);
len=strlen(sendmsg);
sendmsg[len-1]='\0';
send(nsd,sendmsg,20,0);
wait(20);
}
while(strcmp(sendmsg,”bye”)!=0);
}
```

OUTPUT

CLIENT SIDE

```
qq@aj-Desk: ~/jithin/btechNPunix
qq@aj-Desk: ~/jithin/btechNPunix    qq@aj-Desk: ~/jithin/btechNPunix
qq@aj-Desk: ~/jithin/btechNPunix 80x22
qq@aj-Desk:~/jithin/btechNPunix$ gcc clientTcpBi.c -o c
qq@aj-Desk:~/jithin/btechNPunix$ ./c

 messages:
hai server
hai client
bye


bye
qq@aj-Desk:~/jithin/btechNPunix$
```

SERVER SIDE

```
qq@aj-Desk: ~/jithin/btechNPunix
qq@aj-Desk: ~/jithin/btechNPunix    qq@aj-Desk: ~/jithin/btechNPunix
qq@aj-Desk: ~/jithin/btechNPunix 79x22
qq@aj-Desk:~/jithin/btechNPunix$ gcc serverTcpBi.c -o s
qq@aj-Desk:~/jithin/btechNPunix$ ./s
hai client
messages:hai server

bye

bye


qq@aj-Desk:~/jithin/btechNPunix$
qq@aj-Desk:~/jithin/btechNPunix$
```

RESULT

The program is compiled and output is obtained successfully.

Experiment No:3	UNIDIRECTIONAL CHAT USING UDP
Date:13/12/2018	

AIM

To write a program to implement unidirectional chat using udp.

ALGORITHM

Step1: Start

Step2: Create structure members cliaddr and servaddr for sockaddr_in on both server and client.

Step3: create a UDP client socket on client and UDP server socket on server.

Step4: Assign sin_family, sin_addr and sin_port values for cliaddr on client and servaddr on sever.

Step5: Find the server socket.

Step6: Repeat step 7 to step 13.

Step7: Read input from user on client side.

Step8: Send this message to server using sendto() function.

Step9: On server, receive the message using recvfrom() function.

Step10: Display the message on screen.

Step11: End.

PROGRAM

CLIENTSIDE

```
#include<stdio.h>
#include<sys/types.h>
#include<netinet/in.h>
#include<string.h>
#include<sys/socket.h>
main()
{
int csd,cport,len,len1;
charsendmsg[30],recvmsg[20];
structsockaddr_incliaddr,servaddr;

csd=socket(AF_INET,SOCK_DGRAM,0);

cliaddr.sin_family=AF_INET;
cliaddr.sin_addr.s_addr=htonl(INADDR_ANY);
cliaddr.sin_port=htons(12345);
len1=sizeof(cliaddr);
printf("\nclient says:\n");
fgets(sendmsg,20,stdin);
len=strlen(sendmsg);
sendmsg[len-1]='\0';
sendto(csd,sendmsg,len,0,(structsockaddr*)&cliaddr,len1);
```

```
close(csd);
}
```

SERVERSIDE

```
#include<stdio.h>
#include<sys/types.h>
#include<netinet/in.h>
#include<sys/socket.h>
#include<string.h>
main()
{
intsd,sport,len;
charsendmsg[30],recvmsg[20];
structsockaddr_inservaddr,cliaddr;
sd=socket(AF_INET,SOCK_DGRAM,0);
servaddr.sin_family=AF_INET;
servaddr.sin_addr.s_addr=htonl(INADDR_ANY);
servaddr.sin_port=htons(12345);
bind(sd,(structsockaddr*)&servaddr,sizeof(servaddr));
len=sizeof(cliaddr);
printf("\nClient says:");
wait(20);
recvfrom(sd,recvmsg,20,0,(structsockaddr*)&cliaddr,&len);
printf("%s",recvmsg);
}
```

OUTPUT

CLIENT SIDE

```
user2@user-HP-15-Notebook-PC: ~/Desktop
user2@user-HP-15-Notebook-PC:~$ cd Desktop
user2@user-HP-15-Notebook-PC:~/Desktop$ gcc uc.c -o n
user2@user-HP-15-Notebook-PC:~/Desktop$ ./n

client says:
HAI FRIENDS !
user2@user-HP-15-Notebook-PC:~/Desktop$
```

SERVER SIDE

```
user2@user-HP-15-Notebook-PC: ~/Desktop
user2@user-HP-15-Notebook-PC:~$ cd Desktop
user2@user-HP-15-Notebook-PC:~/Desktop$ gcc us.c -o m
user2@user-HP-15-Notebook-PC:~/Desktop$ ./m

Client says:HAI FRIENDS !user2@user-HP-15-Notebook-PC:~/Desktop$
```

RESULT

The program is compiled and output is obtained successfully.

Experiment no:4	BIDIRECTIONAL CHAT USING UDP
Date: 13/12/2018	

AIM

Write a UNIX program to implement bidirectional chat using UDP.

ALGORITHM

Step1: Start

Step2: Create structure members cliaddr and servaddr for sockaddr_in on both server and client

Step3: Create a UDP client socket on client and UDP server socket onserver.

Step4: Assign sin_family, sin_addr, sin_port values for cliaddr on client and servaddr on server.

Step5: Bind the server socket.

Step6: Repeat step7 to step13.

Step7: Read input from user on client side.

Step8: Send this message to server using sendto() function.

Step9: On server receive the message using recvfrom() function.

Step10: Display the message on screen.

Step11: Read input from user on server side.

Step12: Send this message to client using sendto() function.

Step13: On client receive this message using recvfrom() function and display it on screen.

Step14: Stop.

PROGRAM

CLIENTSIDE

```
#include<stdio.h>
#include<sys/types.h>
#include<netinet/in.h>
#include<string.h>
#include<sys/socket.h>
main()
{
int csd,cport,len,len1,len2;
charsendmsg[30],recvmsg[20];
structsockaddr_incliaddr,servaddr;
csd=socket(AF_INET,SOCK_DGRAM,0);
cliaddr.sin_family=AF_INET;
cliaddr.sin_addr.s_addr=htonl(INADDR_ANY);
cliaddr.sin_port=htons(25478);
len1=sizeof(cliaddr);
len2=sizeof(servaddr);
do
{
printf("\nclient says:\n");
fgets(sendmsg,20,stdin);
len=strlen(sendmsg);
```

```
sendmsg[len-1]='\0';
sendto(csd,sendmsg,len,0,(structsockaddr*)&cliaddr,len1);
printf("\nserver says:\n");
recvfrom(csd,recvmsg,20,0,(structsockaddr*)&servaddr,&len2);
printf("%s",recvmsg);
}
while(strcmp(recvmsg,"bye")!=0);
close(csd);
}
```

SERVERSIDE

```
#include<stdio.h>
#include<sys/types.h>
#include<netinet/in.h>
#include<sys/socket.h>
#include<string.h>
main()
{
int sd,sport,len,len1,len2;
charsendmsg[30],recvmsg[20];
structsockaddr_inservaddr,cliaddr;
sd=socket(AF_INET,SOCK_DGRAM,0);
servaddr.sin_family=AF_INET;
servaddr.sin_addr.s_addr=htonl(INADDR_ANY);
servaddr.sin_port=htons(25478);
```

```
inti=bind(sd,(structsockaddr*)&servaddr,sizeof(servaddr));
len2=sizeof(cliaddr);
len1=sizeof(servaddr);
do
{
printf("\nClient says:");
recvfrom(sd,recvmsg,20,0,(structsockaddr*)&cliaddr,&len2);
printf("%s",recvmsg);
printf("\nServer says:\n");
fgets(sendmsg,20,stdin);
len=strlen(sendmsg);
sendmsg[len-1]='\0';
sendto(sd,sendmsg,len,0,(structsockaddr*)&cliaddr,len1);
close(sd);
}
While(strcmp(recvmsg,"bye")!=0);
}
```

OUTPUT

SERVERSIDE

```
cse@cse-desktop:~/lab$ ./s

Client says:hai
Server says:
hello

Client says:bye
Server says:
bye
```

CLIENTSIDE

```
cse@cse-desktop:~/lab$ ./o
bash: ./o: No such file or directory
cse@cse-desktop:~/lab$ ./c

client says:
hai

server says:
hello
client says:
bye

server says:
bye
client says:
```

RESULT

The program is compiled and output is obtained successfully