EXPERIMENT - IX
MULTI USER CHAT SERVER USING TCP

April 3, 2020

ADITHYA D RAJAGOPAL
ROLL NO : 9
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
COLLEGE OF ENGINEERING TRIVANDRUM

AIM

To implement a multi user chat server using TCP as transport layer protocol.

THEORY

TCP (Transmission Control Protocol) works with the Internet Protocol (IP), which defines how computers send packets of data to each other. Together, TCP and IP are the basic rules defining the Internet. It is a connection-oriented protocol, which means that a connection is established and maintained until the application programs at each end have finished exchanging messages.

Server

In a simple multi user chat system, the server usually has the role to receive the messages sent by the clients and send it to all other clients. So basically, he handles the routing of the messages sent by one client to all the other clients.

Client

The client here acts from the side of the user. He sends the messages to the server, and the server sends this message to all the other clients to simulate a simple multi-user chat system.

ALGORITHM

Algorithm 1 Algorithm for the client

```
1: START
2: If argc<2, ask for address and exit.
3: Create the socket using socket().
4: Configure sockaddr_in struct.
5: Connect the socket to server using function connect().
6: Send the message to the server using sendto().
7: Receive the response from the server using recvfrom().
8: Print the response.
9: STOP
```

Algorithm 2 Algorithm for the server

```
1: START
2: Create the TCP socket.
3: Configure sockaddr_in struct.
4: Bind the address struct to the socket using bind().
5: Accept incoming connections using accept().
6: Create child processes for these connections.
7: Let these child processes handle each connected client.
8: Until termination of the infinite loop, repeat steps 8-10.
9: Receive the data using recvfrom().
10: Send response using sendto().
11: Print the received data.
12: STOP
```

SOURCE CODE

Client

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

void main(int argc,char** argv)
{
 char *addr,buffer[MAX],n;
 int sockfd,connfd;
 struct sockaddr_in servaddr;
 if(argc<2)
 {
  printf("No address was passed...\n");
  exit(0);
 }
 addr=argv[1];
 sockfd=socket(AF_INET,SOCK_STREAM,0);
 if(sockfd<0)
 {
  printf("Socket creation failed...\n");
  exit(0);
 }
 else
 printf("Socket successfully created...\n");
 memset(&servaddr,0,sizeof(servaddr));
 servaddr.sin_family=AF_INET;
 servaddr.sin_addr.s_addr=inet_addr(addr);
 servaddr.sin_port=htons(PORT);
 connfd=connect(sockfd,(SA*)&servaddr,sizeof(servaddr));
 if(connfd==0)
 printf("Connected to the server...\n");
 else
 {
  printf("Connection to the server failed...\n");
  exit(0);
 }
 while(1)
 {
  printf("Enter the message to server : ");
  fgets(buffer,MAX,stdin);
  sendto(sockfd,buffer,MAX,0,(SA*)&servaddr,sizeof(servaddr));
  if(strcmp(buffer,"exit\n")==0)
  {
   printf("Client exit...\n");
   break;
  }
  n=recvfrom(sockfd,buffer,MAX,0,NULL,NULL);
  buffer[n]='\0';
  printf("From server : %s",buffer);
 }
 close(sockfd);
}
```

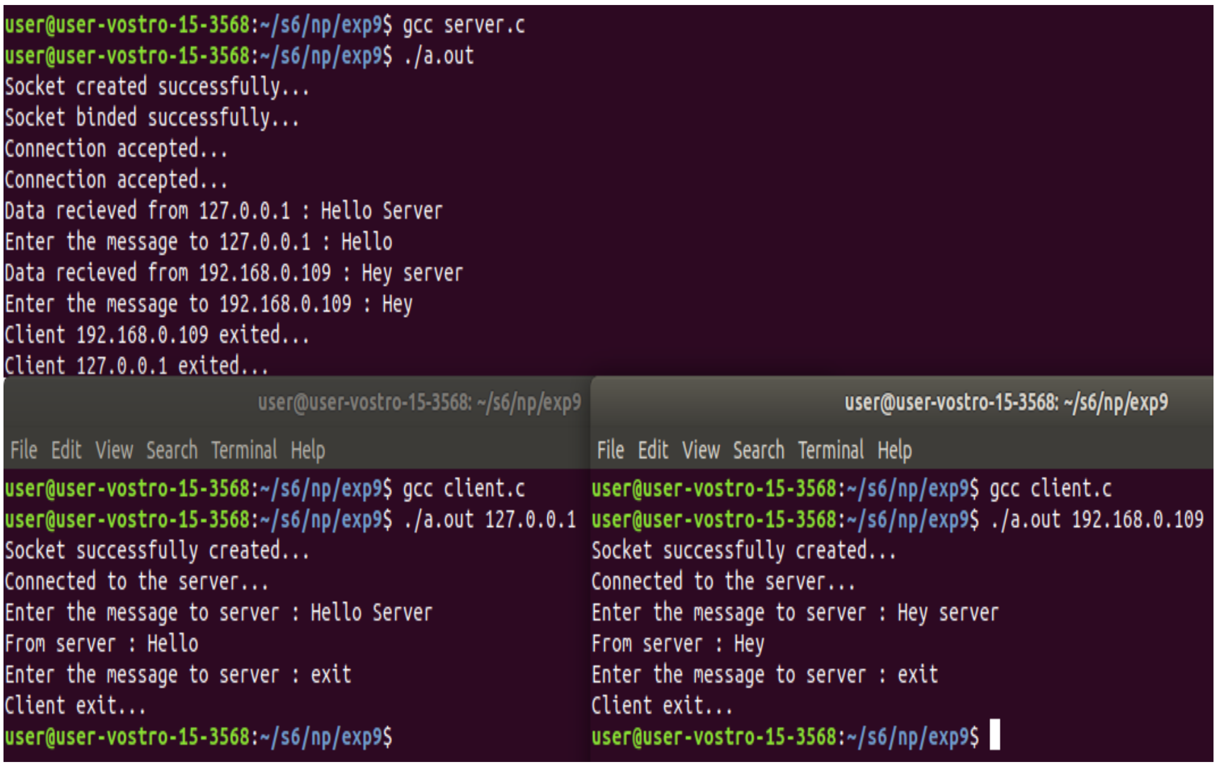
Server

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

```
#define SA struct sockaddr
#define MAX 80

void main()
{
 int sockfd,bindfd,connfd,len,n;
 struct sockaddr_in servaddr,cliaddr;
 char clientAddr[80],buffer[MAX];
 pid_t child;
 sockfd=socket(AF_INET,SOCK_STREAM,0);
 if(sockfd<0)
 {
  printf("Socket creation failed...\n");
  exit(0);
 }
 else
 printf("Socket created successfully...\n");
 memset(&servaddr,0,sizeof(servaddr));
 servaddr.sin_family=AF_INET;
 servaddr.sin_addr.s_addr=INADDR_ANY;
 servaddr.sin_port=htons(PORT);
 bindfd=bind(sockfd,(SA*)&servaddr,sizeof(servaddr));
 if(bindfd<0)
 {
  printf("Socket bind failed...\n");
  exit(0);
 }
 else
 printf("Socket binded successfully...\n");
 listen(sockfd,5);
 while(1)
 {
  len=sizeof(cliaddr);
  connfd=accept(sockfd,(SA*)&cliaddr,&len);
  if(connfd<0)
  {
    printf("Error accpeting connection...\n");
    exit(0);
  }
  else
  printf("Connection accepted...\n");
  inet_ntop(AF_INET,&(cliaddr.sin_addr),clientAddr,80);
  if((child=fork())==0)
  {
    close(sockfd);
    while(1)
    {
      memset(buffer,0,MAX);
      n=recvfrom(connfd,buffer,MAX,0,(SA*)&cliaddr,&len);
      buffer[n]='\0';
      if(strcmp(buffer,"exit\n")==0)
      {
        n=-1;
        printf("Client %s exited...\n",clientAddr);
        break;
      }
      printf("Data recieved from %s : %s",clientAddr,buffer);
      printf("Enter the message to %s : ",clientAddr);
      fgets(buffer,MAX,stdin);
      sendto(connfd,buffer,MAX,0,(SA*)&cliaddr,len);
    }
  }
  close(connfd);
  if(n==-1)
  break;
 }
}
```

OUTPUT



```
user@user-vostro-15-3568:~/s6/np/exp9$ gcc server.c
user@user-vostro-15-3568:~/s6/np/exp9$ ./a.out
Socket created successfully...
Socket binded successfully...
Connection accepted...
Connection accepted...
Data recieved from 127.0.0.1 : Hello Server
Enter the message to 127.0.0.1 : Hello
Data recieved from 192.168.0.109 : Hey server
Enter the message to 192.168.0.109 : Hey
Client 192.168.0.109 exited...
Client 127.0.0.1 exited...
```

user@user-vostro-15-3568: ~/s6/np/exp9
File Edit View Search Terminal Help

```
user@user-vostro-15-3568:~/s6/np/exp9$ gcc client.c
user@user-vostro-15-3568:~/s6/np/exp9$ ./a.out 127.0.0.1
Socket successfully created...
Connected to the server...
Enter the message to server : Hello Server
From server : Hello
Enter the message to server : exit
Client exit...
user@user-vostro-15-3568:~/s6/np/exp9$
```

user@user-vostro-15-3568: ~/s6/np/exp9
File Edit View Search Terminal Help

```
user@user-vostro-15-3568:~/s6/np/exp9$ gcc client.c
user@user-vostro-15-3568:~/s6/np/exp9$ ./a.out 192.168.0.109
Socket successfully created...
Connected to the server...
Enter the message to server : Hey server
From server : Hey
Enter the message to server : exit
Client exit...
user@user-vostro-15-3568:~/s6/np/exp9$
```

RESULT

Implemented a multi user chat server using TCP as transport layer protocol using C.