9 Multi user chat server using TCP

9.1 Aim

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

9.2 Theory

- **Server**: A server is a program that processes requests from client programs and replies to the requests accordingly
- **Client**:A client is a program that requests services from the server.The client program sends a request in a predefined format to the server and the server replies accordingly to the request.
- **Socket**:A socket is a way to communicate in a connection using file descriptors.They act as endpoints in a connection.
- **Thread**:A thread is a subprocess that executes a part of the program concurrently with the main thread.A program can have multiple threads running simultaneously.

9.3 Algorithm

Algorithm 1 Server

```
procedure userCallBack(void * socketfd)
    for true do
        int n =
        read((long)socketfd,buffer,sizeof(buffer))
        strcpy(clientM.buffer,buffer)
        clientM.messagerId = (long)socketfd
        for int i = 0;i<socketList.size();i++ do
            if socketList[i] != (long)socketfd then
                write(socketList[i],(void *)&clientM,sizeof(clientM))
            end if
        end for
        if strcmp(buffer,"exit") == 0 then
            break
        end if
    end for
    pthread_exit(NULL);
end procedure
procedure Main Procedure
    socketfd = socket(AF_INET,SOCK_STREAM,0);
    if bind(socketfd,(sockaddr *)&server,sizeof(server))!=0 then
        printf("Socket binding failed !");
        exit(0);
    end if
    if listen(socketfd,5) <0 then
        printf("Listening failed !");
        exit(0);
    end if
    for true do
        temp = accept(socketfd,(sockaddr *)&client,(socklen_t *)&len)
        if temp<0 then
        else
            pthread_create(&userThread,NULL,userCallBack,(void *)temp)
            socketList.push_back(temp)
        end if
    end for
    close(socketfd);
end procedure
```

Algorithm 2 Client

```
procedure Main Procedure
    socketfd = socket(AF_INET,SOCK_STREAM,0)
    if socketfd<0 then
        printf("Socket creation failed !")
        exit(0)
    end if
    if connect(socketfd,(sockaddr *)&server,sizeof(server))!= 0 then
        printf("Connection failed ");
        exit(0);
    end if
    for true do
        scanf("%s",buffer);
        write(socketfd,buffer,sizeof(buffer));
        if strcmp(buffer,"exit") == 0 then
            printf("Client exiting...",socketfd)
            break
        end if
        read(socketfd,(void *)&serverM,sizeof(serverM))
        printf("Client #%d says : %s",serverM.messagerId,serverM.buffer)
    end for
end procedure
```

9.4 Code

Server

```
////////////////////////////////////////
//                                    //
//Server For Multiuser chat using tcp //
//                                    //
////////////////////////////////////////
#include<sys/socket.h>
#include<sys/types.h>
#include<arpa/inet.h>
#include<unistd.h>
#include<stdlib.h>
#include<stdio.h>
#include<pthread.h>
#include<vector>
#include<string.h>
#define PORT 8080
#define MAX 1024
using namespace std;
vector<int> socketList;

struct message{
    char buffer[MAX];
    int messagerId;
};
typedef struct message message;
```

```
void *userCallBack( void * socketfd){
    char buffer[MAX];
    char serverMessage[] = "Welcome to MUCS! \nEnjoy your stay !\n"
    ;
    for(;;){
    int n = read((long)socketfd,buffer,sizeof(buffer));
    printf("Client #%d: %s\n",socketfd,buffer);
    message clientM;
    strcpy(clientM.buffer,buffer);
    clientM.messagerId = (long)socketfd;
    for(int i = 0;i<socketList.size();i++){
        if(socketList[i] != (long)socketfd){
            write(socketList[i],(void *)&clientM,sizeof(clientM));
        }
    }
    if(strcmp(buffer,"exit") == 0)
        break;
    }
    printf("Client #%d exiting.....\n",(long)socketfd);
    pthread_exit(NULL);
}

int main(){
    int socketfd,temp;
    struct sockaddr_in server,client;
    memset(&server,0,sizeof(server));
    socketfd = socket(AF_INET,SOCK_STREAM,0);
    if(socketfd <0){
        printf("Socket creation failed !\n");
        exit(0);
    }
    printf("Socket  creation successful !\n");

    server.sin_addr.s_addr = htonl(INADDR_ANY);
    server.sin_family = AF_INET;
    server.sin_port = htons(PORT);
    if(bind(socketfd,(sockaddr *)&server,sizeof(server))!=0){
        printf("Socket binding failed !\n");
        exit(0);
    }
    printf("Socket binding successful\n");
    if(listen(socketfd,5) <0){
        printf("Listening failed !\n");
        exit(0);
    }
    printf("Listening successful !\n");
    for(;;){
    unsigned len = sizeof(client);
    temp = accept(socketfd,(sockaddr *)&client,(socklen_t *)&len);
    if(temp < 0){
        printf("Client connection failed !\n");
    }else{
        pthread_t userThread;
        pthread_create(&userThread,NULL,userCallBack,(void *)temp);
        socketList.push_back(temp);
        printf("Client connected !\n");
        temp = -1;
```

```
    }
    }
    close(socketfd);

    return 0;
}
```

Client

```
////////////////////////////////////////////////
//                                            //
//     Client for MultiUserChat Server        //
//                                            //
////////////////////////////////////////////////
#include<sys/socket.h>
#include<sys/types.h>
#include<arpa/inet.h>
#include<netinet/in.h>
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<string.h>

#define PORT 8080
#define MAX 1024

struct message{
    char buffer[MAX];
    int messagerId;
};
typedef struct message message;
int main(){
    int socketfd;
    struct sockaddr_in server;
    memset(&server,0,sizeof(server));
    socketfd = socket(AF_INET,SOCK_STREAM,0);
    if(socketfd < 0){
        printf("Socket creation failed !\n");
        exit(0);
    }
    printf("Socket creation successful !\n");

    server.sin_addr.s_addr = inet_addr("127.0.0.1");
    server.sin_family = AF_INET;
    server.sin_port = htons(PORT);

    if(connect(socketfd,(sockaddr *)&server,sizeof(server))!= 0){
        printf("Connection failed \n");
        exit(0);
    }
    printf("Connection successful !\n");
    char buffer[MAX];
    for(;;){
    scanf("%s",buffer);

```

```
47      write(socketfd,buffer,sizeof(buffer));
48      printf("Data send to server!\n");
49      if(strcmp(buffer,"exit") == 0){
50          printf("Client exiting...\n",socketfd);
51          break;
52      }
53      message serverM;
54      read(socketfd,(void *)&serverM,sizeof(serverM));
55      printf("Client #%d says : %s\n",serverM.messagerId,serverM.
        buffer);
56
57      }
58      close(socketfd);
59      return 0;
60  }
```

9.5 Output

- **Server**

```
Socket  creation successful !
Socket binding successful
Listening successful !
Client connected !
Client connected !
Client #4: helloworld
Client #5: heyman
Client #4: howareyou
Client #5: hi
Client #5: exit
Client #5 exiting......
Client #4: exit
Client #4 exiting......
```

- **Client #5**

```
Socket creation successful !
Connection successful !
helloworld
Data send to server!
Client #5 says : heyman
howareyou
Data send to server!
Client #5 says : hi
exit
```

```
Data send to server!
Client  exiting...
```

- **Client #4**

```
Connection successful !
heyman
Data send to server!
Client #4 says : helloworld
hi
Data send to server!
Client #4 says : howareyou
exit
Data send to server!
Client  exiting...
```

9.6 Result

Server and client was implemented in C++.A socket is created for the server and binded to a particular port.The server then establishes a connection through which the clients can communicate with the server.The server then accepts the client and saves the file descriptor of the socket in an array.When a client sends a message the server broadcasts that message to all the file descriptors in the array.