ECEN 602

NETWORK SIMULATION ASSIGNMENT – 02

TEAM 17

Mohammad Faisal Khan

Amiya Ranjan Panda

INTRODUCTION

We have implemented a client and server for a simple chat service.

It has been successfully compiled, executed and tested on gcc compiler (part of standard LINUX).

Common Errors and Catches:

-If data is not input correctly on the command line as per the ordering given below, it throws a segmentation error.

-This is NOT to be assumed as an error.

-If data is missing from the command line, it throws segmentation error too.

-If a user with a name same as an existing active chat room user joins, he will be required to reenter chat room using a valid name.

-Username can be any size until 512 characters.

-Maximum number of clients active on chat room = argv[3]. If argv[3]+1 client joins, it will throw an ERROR on server and client console.

-IMPORTANT: Please do not use same serverport when running server second time, until the first server process is killed (NOT SUSPENDED).

-If a client disconnects using keyword 'quit', a smooth disconnect happens and resources are deallocated. If not, the other clients enter into hang state.

-This is an iterative server, no use of fork() in this code.

Usage of MAKEFILE:

make -f makefile_server

make -f makefile_client

Then, EXECUTE using the following commands:

./team17_server <ip> <port> <max user>

Example - ./team17_server 127.0.0.1 50001 12

./team17_client <user name> <ip> <port>

Example - ./team17_client user1 127.0.0.1 50001

Usage:

All commands are executed on standard Terminal of LINUX which supports gcc compiler commands.

Special uses:

- SELECT() function has been used in both client and server codes to determine the action.

- SEND() and RECV() are used to send data and receive data respectively into the structs.

Architecture: Modular (Functional) Programming

-We have used modular approach to C Programming in this code.

-Specific functions like sending JOIN, FWD and SEND messages are implemented separetely in the code for higher clarity.

-This code has comments on most lines to fathom the working of the functions.

Data usage: Data Structures (DS)

-Encapsulation of messages into network packets is implemented using DS.

-SBCP frame format for packetization is used.

-As per the requirement, two level encapsulation scheme is employed.

COMMANDS Implemented:

JOIN COMMAND: When a new user connects to the chat room, a JOIN struct is sent across to the server.

This contains the username of the client.

SEND COMMAND: When a user types a chat message, this is encapsulated in the SEND struct and sent to the server.

This contains the username and the chat message of a single client.

FWD COMMAND: Server broadcasts the chat message to the clients using the FWD struct.

ERROR COMMAND: When a client with the same name as a already active client joins the session, Server sends a ERROR command.

This client is disconnected from the server and chat room.

Prints on Server:

-Server starting.

-Listening.

-Shows a connected succesfully message each time a client connects to server IP and server port.

-Shows execution of FWD command.

-Shows execution of ERROR command.

Prints on Client:

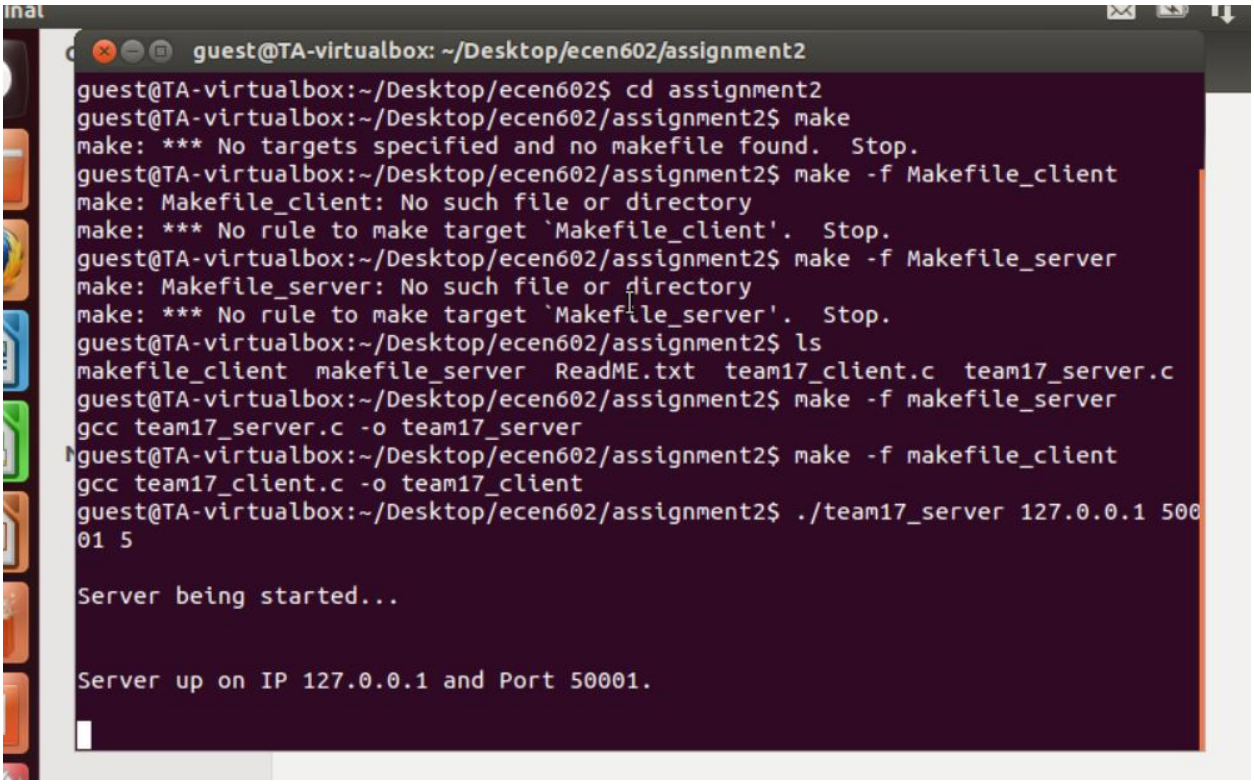
-Client connecting.

-JOIN Message.

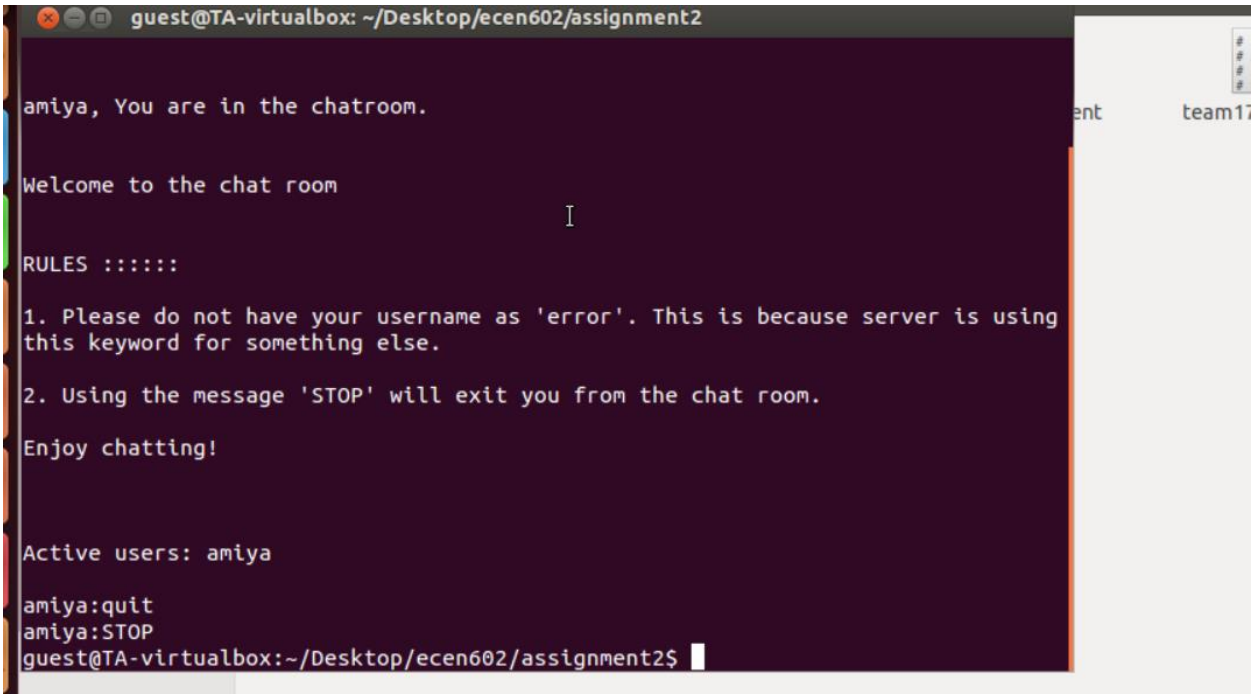
-Chat room rules.

-Active users on the chat room.

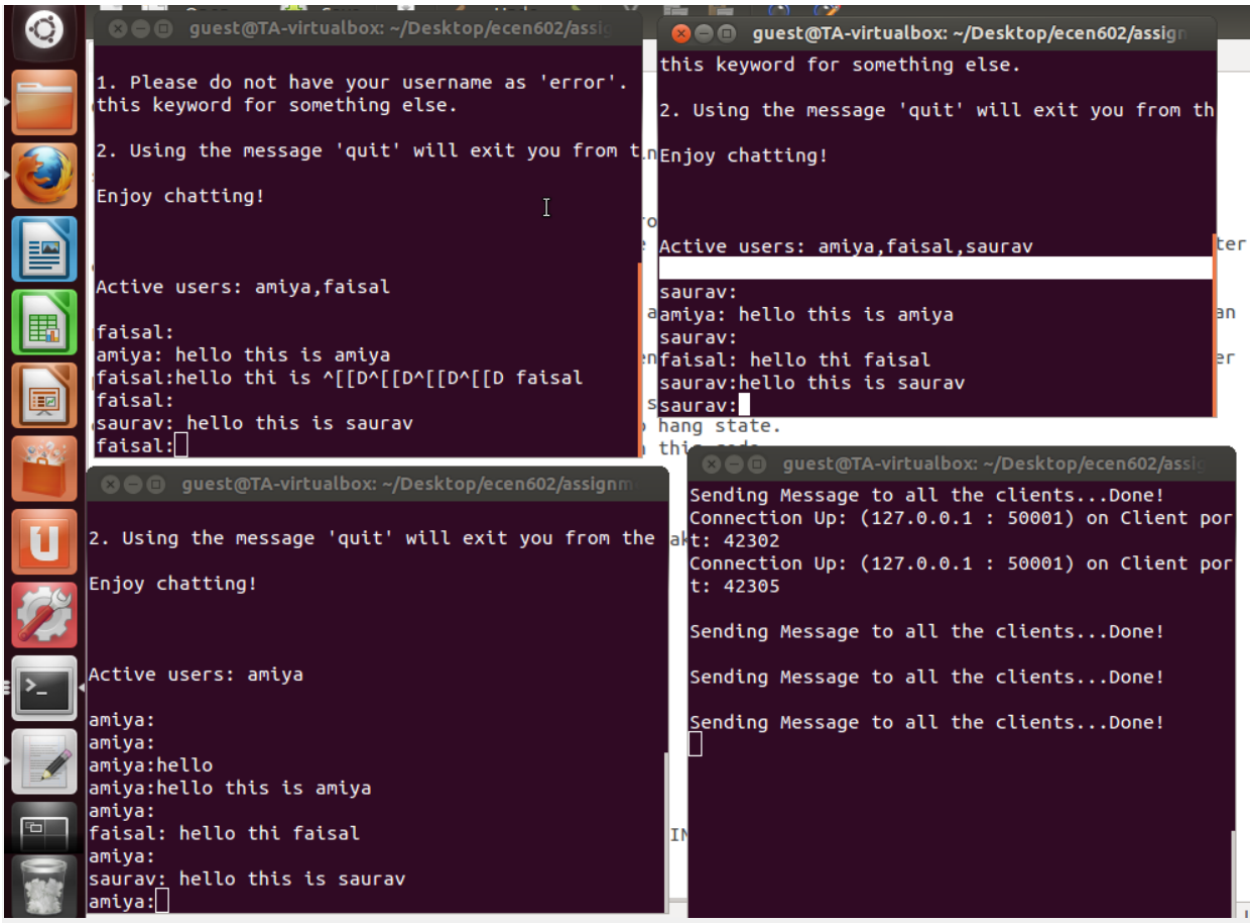
-ERROR when client uses the same username as an already active client.

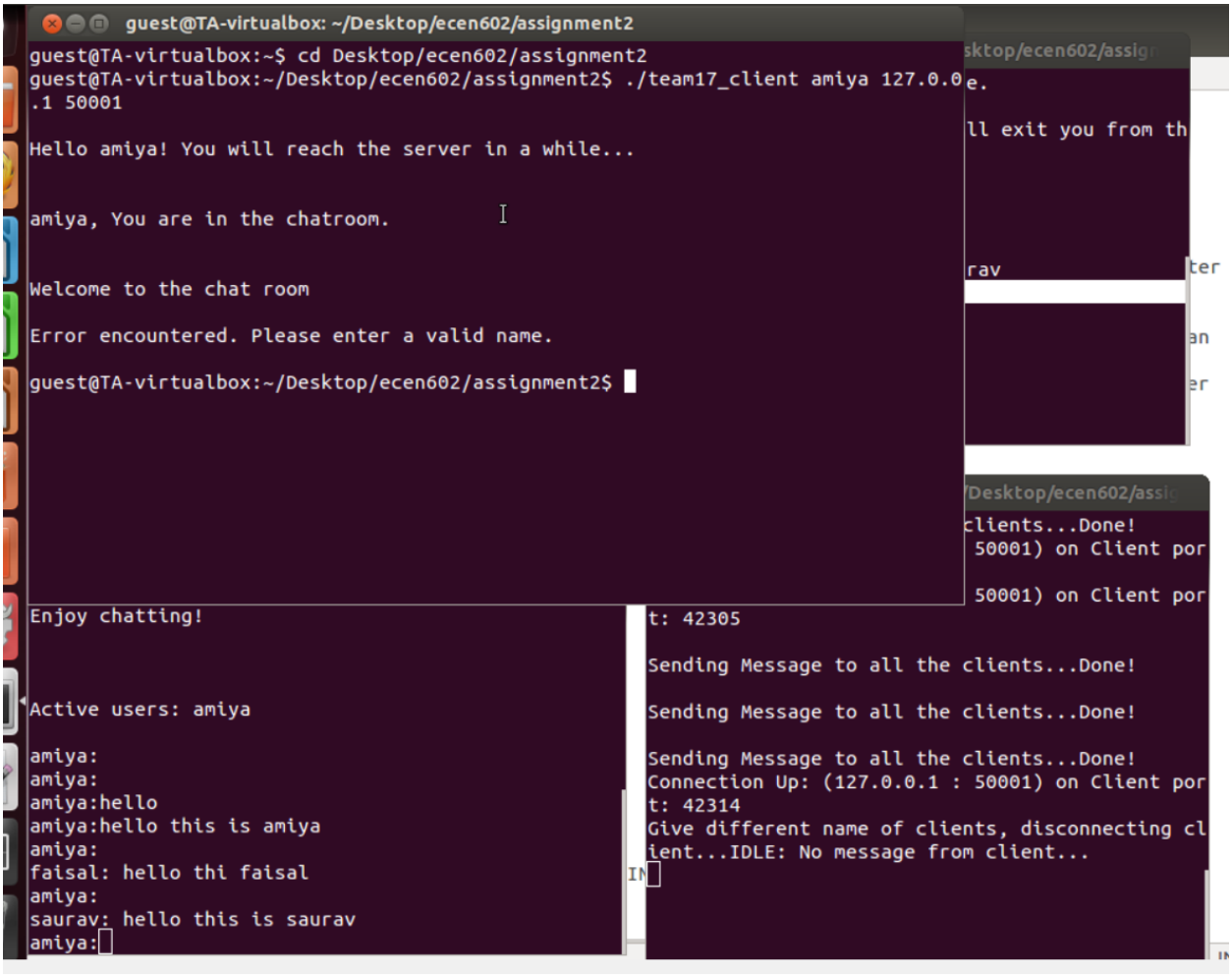

guest@TA-virtualbox: ~/Desktop/ecen602/assignment2
guest@TA-virtualbox:~/Desktop/ecen602$ cd assignment2
guest@TA-virtualbox:~/Desktop/ecen602/assignment2$ make
make: *** No targets specified and no makefile found. Stop.
guest@TA-virtualbox:~/Desktop/ecen602/assignment2$ make -f Makefile_client
make: Makefile_client: No such file or directory
make: *** No rule to make target `Makefile_client'. Stop.
guest@TA-virtualbox:~/Desktop/ecen602/assignment2$ make -f Makefile_server
make: Makefile_server: No such file or directory
make: *** No rule to make target `Makefile_server'. Stop.
guest@TA-virtualbox:~/Desktop/ecen602/assignment2$ ls
makefile_client makefile_server ReadME.txt team17_client.c team17_server.c
guest@TA-virtualbox:~/Desktop/ecen602/assignment2$ make -f makefile_server
gcc team17_server.c -o team17_server
guest@TA-virtualbox:~/Desktop/ecen602/assignment2$ make -f makefile_client
gcc team17_client.c -o team17_client
guest@TA-virtualbox:~/Desktop/ecen602/assignment2$ ./team17_server 127.0.0.1 500
01 5
Server being started...
Server up on IP 127.0.0.1 and Port 50001.

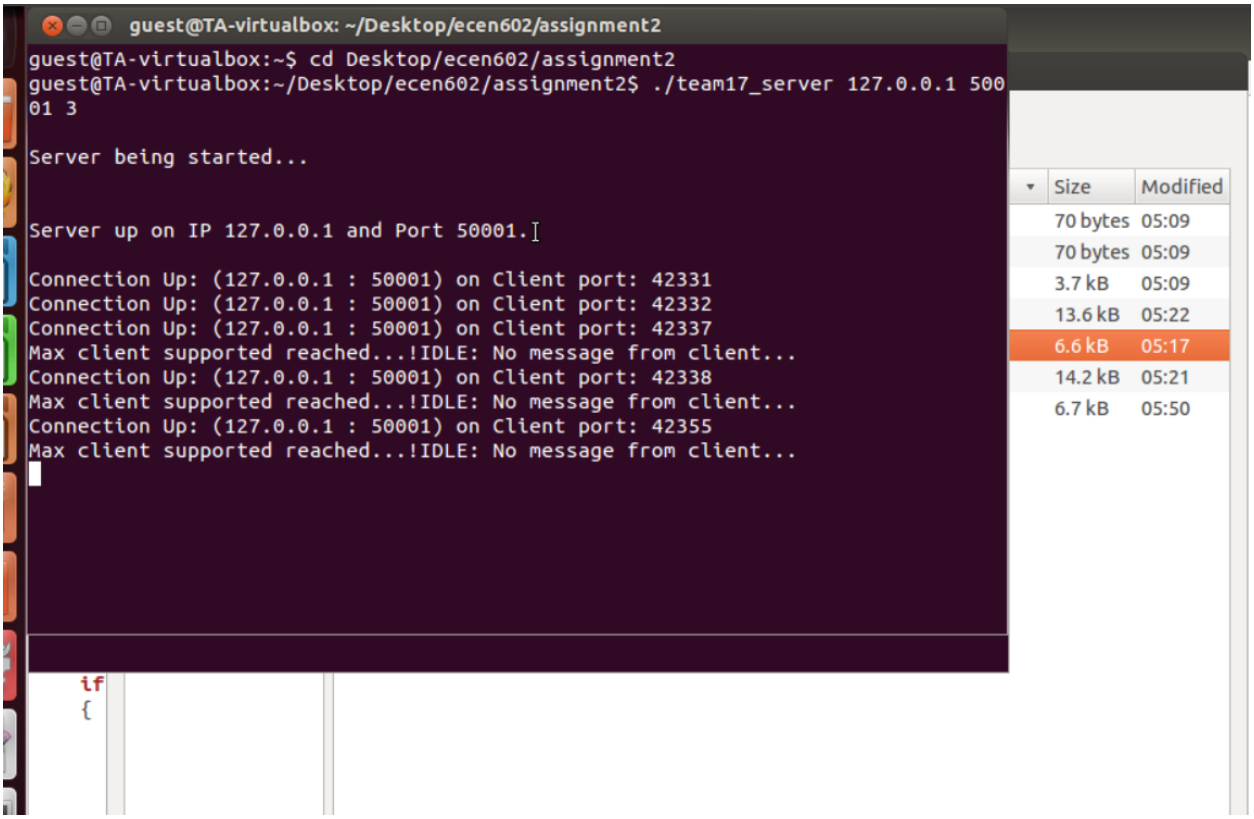

guest@TA-virtualbox:~/Desktop/ecen602/assignment2$ ./team17_server 127.0.0.1
Segmentation fault (core dumped)
guest@TA-virtualbox:~/Desktop/ecen602/assignment2$ ./team17_server 50001
Segmentation fault (core dumped)
guest@TA-virtualbox:~/Desktop/ecen602/assignment2$ ./team17_server 127.0.0.1 50001
Segmentation fault (core dumped)
guest@TA-virtualbox:~/Desktop/ecen602/assignment2$ ./team17_server 127.0.0.1 50001 10
Server being started...
Server up on IP 127.0.0.1 and Port 50001.


guest@TA-virtualbox: ~/Desktop/ecen602/assignment2
amiya, You are in the chatroom.
Welcome to the chat room
RULES ::::::
1. Please do not have your username as 'error'. This is because server is using this keyword for something else.
2. Using the message 'STOP' will exit you from the chat room.
Enjoy chatting!
Active users: amiya
amiya:quit
amiya:STOP
guest@TA-virtualbox:~/Desktop/ecen602/assignment2$

```
guest@TA-virtualbox:~$ cd Desktop/ecen602/assignment2
guest@TA-virtualbox:~/Desktop/ecen602/assignment2$ ./team17_client

Hello (null)! You will reach the server in a while...

./team17_client: Segmentation fault (core dumped)
guest@TA-virtualbox:~/Desktop/ecen602/assignment2$ ./team17_client 127.0.0.1

Hello 127.0.0.1! You will reach the server in a while...

Segmentation fault (core dumped)
guest@TA-virtualbox:~/Desktop/ecen602/assignment2$ ./team17_client 127.0.0.1 50001
Segmentation fault (core dumped)
guest@TA-virtualbox:~/Desktop/ecen602/assignment2$ ./team17_client amiya 127.0.0.1 50001

Hello amiya! You will reach the server in a while...


amiya, You are in the chatroom.


Welcome to the chat room


RULES ::::::

1. Please do not have your username as 'error'. This is because server is using this keyword for some
 else.

2. Using the message 'quit' will exit you from the chat room.

Enjoy chatting!



Active users: amiya

amiya:
```


guest@TA-virtualbox: ~/Desktop/ecen602/assi
1. Please do not have your username as 'error'.
this keyword for something else.
2. Using the message 'quit' will exit you from t
Enjoy chatting!
Active users: amiya,faisal
faisal:
amiya: hello this is amiya
faisal:hello thi is ^[[D^[[D^[[D^[[D faisal
faisal:
saurav: hello this is saurav
faisal:
guest@TA-virtualbox: ~/Desktop/ecen602/assign
this keyword for something else.
2. Using the message 'quit' will exit you from th
Enjoy chatting!
Active users: amiya,faisal,saurav
saurav:
amiya: hello this is amiya
saurav:
faisal: hello thi faisal
saurav:hello this is saurav
saurav:
guest@TA-virtualbox: ~/Desktop/ecen602/assignm
2. Using the message 'quit' will exit you from the
Enjoy chatting!
Active users: amiya
amiya:
amiya:
amiya:hello
amiya:hello this is amiya
amiya:
faisal: hello thi faisal
amiya:
saurav: hello this is saurav
amiya:
guest@TA-virtualbox: ~/Desktop/ecen602/assi
Sending Message to all the clients...Done!
Connection Up: (127.0.0.1 : 50001) on Client por
t: 42302
Connection Up: (127.0.0.1 : 50001) on Client por
t: 42305
Sending Message to all the clients...Done!
Sending Message to all the clients...Done!
Sending Message to all the clients...Done!


guest@TA-virtualbox: ~/Desktop/ecen602/assignment2
guest@TA-virtualbox:~$ cd Desktop/ecen602/assignment2
guest@TA-virtualbox:~/Desktop/ecen602/assignment2$ ./team17_client amiya 127.0.0.1 50001
Hello amiya! You will reach the server in a while...
amiya, You are in the chatroom.
Welcome to the chat room
Error encountered. Please enter a valid name.
guest@TA-virtualbox:~/Desktop/ecen602/assignment2$
Enjoy chatting!
Active users: amiya
amiya:
amiya:
amiya:hello
amiya:hello this is amiya
amiya:
faisal: hello thi faisal
amiya:
saurav: hello this is saurav
amiya:
Sending Message to all the clients...Done!
Sending Message to all the clients...Done!
Sending Message to all the clients...Done!
Connection Up: (127.0.0.1 : 50001) on Client port: 42314
Give different name of clients, disconnecting client...IDLE: No message from client...


guest@TA-virtualbox: ~/Desktop/ecen602/assignment2
guest@TA-virtualbox:~$ cd Desktop/ecen602/assignment2
guest@TA-virtualbox:~/Desktop/ecen602/assignment2$ ./team17_server 127.0.0.1 500
01 3
Server being started...
Server up on IP 127.0.0.1 and Port 50001.
Connection Up: (127.0.0.1 : 50001) on Client port: 42331
Connection Up: (127.0.0.1 : 50001) on Client port: 42332
Connection Up: (127.0.0.1 : 50001) on Client port: 42337
Max client supported reached...!IDLE: No message from client...
Connection Up: (127.0.0.1 : 50001) on Client port: 42338
Max client supported reached...!IDLE: No message from client...
Connection Up: (127.0.0.1 : 50001) on Client port: 42355
Max client supported reached...!IDLE: No message from client...
Size
Modified
70 bytes 05:09
70 bytes 05:09
3.7 kB 05:09
13.6 kB 05:22
6.6 kB 05:17
14.2 kB 05:21
6.7 kB 05:50
if
{

TCP CHAT SERVER CODE ::

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

/* Global Structures */
int count;
void reply_join(int socket_fd1, int argv3);
char users[1024];

struct attr_payload
{
    char client_usrname[16];
    char message[512];
    char reason[32];
    int client_count;
}a_payload;

struct attr_sbcp
{
    int attr_type;
    int attr_length;
    struct attr_payload sbcp_attr_payload;
}msg_username, msg, msg_reason;

struct msg_sbcp
{
    int ver;
    int msg_type;
    int msg_length;
    struct attr_sbcp msg_payload[4];
}final_msg, join_msg;

/* Declarations of used functions */
void connect_server(int *socket_fd,struct sockaddr_in *serv_addr,int
argv2, int argv3, char argv1[4]);
void accept_new_client(fd_set *main_set, int *max_fd, int socket_fd,
struct sockaddr_in *client_addr, char argv2[10], int argv3);
void send_receive_data(int num, fd_set *main_set, int socket_fd, int
max_fd);
void broadcast_data (int count, int num, int socket_fd, int
recvd_bytes, struct msg_sbcp receive_buffer, fd_set *main_set);
```

```
void delete_client(fd_set *main_set, int *max_fd, int socket_fd,
struct sockaddr_in *client_addr, char argv2[10], int argv3);

int main(int argc, char *argv[])
{
    struct sockaddr_in serv_addr, client_addr;
    fd_set main_set;
    fd_set temp_set;
    int max_fd;
    int i;
    int argv2=atoi(argv[2]);
    int argv3=atoi(argv[3]);
    int socket_fd=0;

    FD_ZERO(&main_set);
    FD_ZERO(&temp_set);

    connect_server(&socket_fd, &serv_addr,argv2,argv3,argv[1]);
    FD_SET(socket_fd, &main_set);
    max_fd=socket_fd;

    fflush(stdout);
    printf("\nServer being started...\n\n");
    fflush(stdout);
    sleep(1);
    fflush(stdout);
    printf("\nServer up on IP %s and Port %s.\n\n",argv[1], argv[2]);
    fflush(stdout);

    while(1)
    {
        temp_set=main_set;

        if(select(max_fd+1,&temp_set,NULL,NULL,NULL)==-1)
        {
            error("Error in select()..!");
            exit(4);
        }

        for(i=0;i<=max_fd;i++)
        {
            if(FD_ISSET(i, &temp_set))
            {
                if(i==socket_fd)
                {

accept_new_client(&main_set,&max_fd,socket_fd,&client_addr,argv[2],arg
v3);
                }

                else
                    send_receive_data(i,&main_set,socket_fd,max_fd);
```

```
            }
        }
    }

    return 0;
}

void connect_server(int *socket_fd,struct sockaddr_in *serv_addr, int
argv2,int argv3, char argv1[4])
{
    if((*socket_fd = socket(AF_INET,SOCK_STREAM,0))==-1)
        error("Error in socket()..!");

    serv_addr->sin_family      = AF_INET;
    serv_addr->sin_port        = htons(argv2);
    serv_addr->sin_addr.s_addr = inet_addr(argv1);

    memset(serv_addr->sin_zero, '\0', sizeof serv_addr->sin_zero);
    int flag=1;


if(setsockopt(*socket_fd,SOL_SOCKET,SO_REUSEADDR,&flag,sizeof(int))==-
1)
    {
        error("Error in socket opt()...\n");
        exit(1);
    }

    if(bind(*socket_fd, (struct sockaddr *)serv_addr, sizeof (struct
sockaddr))==-1)
    {
        error("Error in bind()...\n");
        exit(1);
    }

    if(listen(*socket_fd,argv3) == -1)
    {
        error("Error in listen()...\n");
        exit(1);
    }
}

void accept_new_client(fd_set *main_set, int *max_fd, int socket_fd,
struct sockaddr_in *client_addr,char argv2[10], int argv3)
{
    int new_socket_fd;
    socklen_t length;
    length=sizeof(struct sockaddr_in);

    if((new_socket_fd=accept(socket_fd,(struct
sockaddr*)client_addr,&length))==-1)
    {
```

```
        error("Client error... Cannot accept...");
        exit(1);
    }

    else
    {
        FD_SET(new_socket_fd,main_set);

        if(new_socket_fd>*max_fd)
        {
           *max_fd=new_socket_fd;
        }

        printf("Connection Up: (%s : %s) on Client port:
%d\n",inet_ntoa(client_addr->sin_addr),argv2,ntohs(client_addr-
>sin_port));
        int new_socket_fd1=new_socket_fd;
        reply_join(new_socket_fd1,argv3);
    }
}

void send_receive_data(int num, fd_set *main_set, int socket_fd, int
max_fd)
{

    int recvd_bytes;

    memset(&final_msg, 0, sizeof(final_msg));

    if((recvd_bytes= recv(num, &final_msg, sizeof(final_msg), 0)) > 0)
    {
           int count;
        for(count=0;count<=max_fd;count++)
        {

broadcast_data(count,num,socket_fd,recvd_bytes,final_msg,main_set);
        }

           fflush(stdout);
        printf("\nSending Message to all the clients...Done!\n");
        fflush(stdout);
    }

    else
    {
        if(recvd_bytes==0)
        {
            printf("IDLE: No message from client...\n", num);
        }

        else
        {
```

```
            error("Error in data sent...");
        }

        close(num);
        FD_CLR(num,main_set);
    }
}

void reply_join(int socket_fd1, int argv3)
{
    int recvd_bytes;
    memset(&join_msg, 0, sizeof(join_msg));

    if((recvd_bytes= recv(socket_fd1, &join_msg, sizeof(join_msg), 0))
<=0)
    {
        if(recvd_bytes==0)
        {
            printf("Socket not replying...", socket_fd1);
        }

           else
        {
            error("Error in received bytes...");
        }

        close(socket_fd1);
    }

    char name[16];

strcpy(name,join_msg.msg_payload[0].sbcp_attr_payload.client_usrname);

    if(strstr(users,name))
    {
        fflush(stdout);
        printf("Give different name of clients, disconnecting
client...");
        fflush(stdout);

        char error[]="error_name";
        send(socket_fd1,error,strlen(error),0);
        memset(&join_msg, 0, sizeof(join_msg));
    }

    else
    {
        if(count<argv3-1)
        {

      strcat(users,join_msg.msg_payload[0].sbcp_attr_payload.client_usr
name);
```

```
            int m=strlen(users);

            send(socket_fd1,users,strlen(users),0);
            users[m]=',';
            memset(&join_msg, 0, sizeof(join_msg));
            count++;
        }

        else
        {
            printf("Max client supported reached...!");
            fflush(stdout);

            char error[]="error_max";
            send(socket_fd1,error,strlen(error),0);
            memset(&join_msg, 0, sizeof(join_msg));
        }
    }
}

void broadcast_data (int count, int num, int socket_fd, int
recvd_bytes, struct msg_sbcp final_msg, fd_set *main_set)
{
    if(FD_ISSET(count,main_set))
    {
        if(count!=socket_fd && count!=num)
        {
            if(send(count,&final_msg,recvd_bytes,0)==-1)
            {
                error("Error in send()...");
            }

            memset(&final_msg, 0, sizeof(final_msg));
        }
    }
}
```

TCP CHAT CLIENT CODE –

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


void cli_conn(int *serv_filedes, struct sockaddr_in *servaddr, int
argv3, char argv2[4]);
void sendrecv_info(int i, int serv_filedes, char argv1[16]);
void separate(char buffer[256], char username[16], int serv_filedes);
void consolidate(int serv_filedes, char argv1[16]);
void join(int serv_filedes1, char username[16]);


int main(int argc, char *argv[])
{

    struct sockaddr_in servaddr;

    int serv_filedes=0;
    int max_fd, i;


    int argv3=atoi(argv[3]);




    fd_set main_set, temp_set;

    fflush(stdout);
    printf("\nHello %s! You will reach the server in a
while...\n\n",argv[1]);


    cli_conn(&serv_filedes, &servaddr, argv3, argv[2]);
    sleep(1);
    fflush(stdout);

    printf("\n%s, You are in the chatroom.\n\n", argv[1]);
```

```
    sleep(1);
    printf("\nWelcome to the chat room\n\n");
    fflush(stdout);
    sleep(1);



    sleep(1);
    int serv_filedes1=serv_filedes;
    join(serv_filedes1,argv[1]);

    char user_list[1024]={'\0'};

    int l;



    l=recv(serv_filedes1,user_list,1024,0);

    if(strcmp(user_list,"error_name")==0)
    {
        printf("Error encountered. Please enter a different
name.\n\n");
        exit(0);
    }

    if(strcmp(user_list,"error_max")==0)
    {
        printf("Error encountered. Maximum users already present in
chatroom.\n\n");
        exit(0);
    }


    printf("\nRULES ::::::\n\n");
    printf("1. Please do not have your username as 'error'. This is
because server is using this keyword for something else.\n\n");
    printf("2. Using the message 'STOP' will exit you from the chat
room.\n\n");

    printf("Enjoy chatting!\n\n");

    printf("\n\nActive users: %s\n\n",user_list);

    FD_ZERO(&main_set);
    FD_ZERO(&temp_set);

    FD_SET(0,&main_set);
    FD_SET(serv_filedes,&main_set);

    max_fd=serv_filedes;
    fflush(stdout);
```

```
    printf("%s:",argv[1]);
    fflush(stdout);


    while(1)
    {

        temp_set=main_set;

        if(select(max_fd+1,&temp_set,NULL,NULL,NULL)==-1)
        {
            error("Cannot select()");
            exit(4);
        }


        for(i=0;i<=max_fd;i++)
        {
            if(FD_ISSET(i,&temp_set))
            {
                sendrecv_info(i,serv_filedes,argv[1]);
            }
        }
    }


    return 0;
}







void separate(char buffer[256], char username[16], int serv_filedes)
{

    int buff_len=strlen(buffer);
    int user_len=strlen(username);

    struct attr_payload
    {

        char client_username[16];
        char message[512];
        char reason[32];
        int client_count;

    };
```

```
    struct sbcp_attribute
    {

    int attribute_type;
    int attribute_length;
    struct attr_payload sbcp_payload;

    }msg_username, msg, msg_reason;

    struct sbcp_message
    {

        int ver;
        int msg_type;
        int msg_length;
        struct sbcp_attribute msg_payload[4];

    }final_msg,join_msg;

    memset(&final_msg, 0, sizeof(final_msg));

    strcpy(msg_username.sbcp_payload.client_username,username);
    msg_username.attribute_type=2;
    msg_username.attribute_length=user_len+2+2;

    strcpy(msg.sbcp_payload.message,buffer);
    msg.attribute_type=4;
    msg.attribute_length=buff_len+2+2;

    final_msg.ver=3;
    final_msg.msg_type=4;
    final_msg.msg_length=sizeof(msg_username)+sizeof(msg)+2+2;

    final_msg.msg_payload[0]=msg_username;
    final_msg.msg_payload[1]=msg;

    send(serv_filedes, &final_msg,sizeof(final_msg),0);
    memset(&final_msg, 0, sizeof(final_msg));
    fflush(stdout);
    printf("%s:",username);
    fflush(stdout);
}


void sendrecv_info(int i, int serv_filedes, char argv1[16])
{

    char send_buffer[256];
    char receive_buffer[256];
```

```
    int received_bytes;
    if(i==0)
    {

        fgets(send_buffer,256,stdin);
        if(strcmp(send_buffer,"STOP\n")!=0)
                       separate(send_buffer,argv1,serv_filedes);
         else
           exit(0);
            fflush(stdout);
    }

    else
    {

        consolidate(serv_filedes, argv1);

    }
}


void consolidate(int serv_filedes, char argv1[16])
{

    struct attr_payload
    {

        char client_username[16];
        char message[512];
        char reason[32];
        int client_count;

    };


   struct sbcp_attribute
    {

        int attribute_type;
        int attribute_length;
        struct attr_payload sbcp_payload;

    };

    struct sbcp_message
    {

        int ver;
        int msg_type;
        int msg_length;
        struct sbcp_attribute msg_payload[4];
```

```
    }final_msg;

    int received_bytes;
    char receive_buffer[256];

    received_bytes=recv(serv_filedes, &final_msg,
sizeof(final_msg),0);


strcpy(receive_buffer,final_msg.msg_payload[1].sbcp_payload.message);
    receive_buffer[received_bytes]='\0';

    printf("\n%s:
%s",final_msg.msg_payload[0].sbcp_payload.client_username,receive_buff
er);
    fflush(stdout);
    printf("%s:",argv1);
    fflush(stdout);
    memset(&final_msg, 0, sizeof(final_msg));

    fflush(stdout);

}


void cli_conn(int *serv_filedes, struct sockaddr_in *servaddr, int
argv3, char argv2[4])
{
    if((*serv_filedes=socket(AF_INET,SOCK_STREAM,0))==-1)
    {
        error("Cannot create socket.");
        exit(1);
    }

    servaddr->sin_family=AF_INET;
    servaddr->sin_port=htons(argv3);
    servaddr->sin_addr.s_addr=inet_addr(argv2);


    memset(servaddr->sin_zero,'\0',sizeof servaddr->sin_zero);

    if(connect(*serv_filedes,(struct sockaddr *) servaddr,
sizeof(struct sockaddr))==-1)
    {
        error("Failed to connect to socket.");
        exit(1);
    }
}


void join(int serv_filedes1, char username[16])
{
```

```
    int user_len=strlen(username);

    struct attr_payload
    {

        char client_username[16];
        char message[512];
        char reason[32];
        int client_count;

    };


    struct sbcp_attribute
    {

        int attribute_type;
        int attribute_length;
        struct attr_payload sbcp_payload;

    }msg_username, msg, msg_reason;

    struct sbcp_message
    {

        int ver;
        int msg_type;
        int msg_length;
        struct sbcp_attribute msg_payload[4];

    }join_msg;

    memset(&join_msg, 0, sizeof(join_msg));

    strcpy(msg_username.sbcp_payload.client_username,username);
    msg_username.attribute_type=2;
    msg_username.attribute_length=user_len+2+2;


    join_msg.ver=3;
    join_msg.msg_type=2;
    join_msg.msg_length=sizeof(msg_username)+2+2;

    join_msg.msg_payload[0]=msg_username;

    send(serv_filedes1, &join_msg,sizeof(join_msg),0);

    memset(&join_msg, 0, sizeof(join_msg));
}
```