**Client Program**

/\* CLIENT PROGRAM

Author:Shrikrishna Bhat

\*/

//All Header Files

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

#include<sys/types.h>

#include<sys/socket.h>

#include<netinet/in.h>

#include<arpa/inet.h>

#include<unistd.h>

#include<errno.h>

//Max Buffer Length

#define MAX 1024

//Initalise a flag to monitor the status of logins

static int flag=0;

//Initialise a user array to get the user name

char users[MAX];

//Function Declarations

void connect\_request(int \*sockfd, struct sockaddr\_in \*server\_addr);

void send\_user(int sockfd);

void send\_recv(int i, int sockfd);

//Main Function

int main()

{

//Initialise a socket variable,file descriptor and a local variable

int sockfd,fdmax,i;

//Initialise a structure variable for socket

struct sockaddr\_in server\_addr;

//Initialise a variable to keep control of logins and send information to respective file descriptors

fd\_set master;

//Initialise a temporary file descriptor for select()

fd\_set read\_fds;

//Open TCP socket and connect by calling function

connect\_request(&sockfd, &server\_addr);

//Clear master set

FD\_ZERO(&master);

//Clear temp set

FD\_ZERO(&read\_fds);

//Set the value of master

FD\_SET(0, &master);

FD\_SET(sockfd, &master);

//Give the value of sockfd to fdmax

fdmax = sockfd;

//Enter the client name

printf("\n Enter User Name:");

scanf("%s",users);

getchar();

//call function to send user to server

send\_user(sockfd);

//Make it run infinitey

for(;;){

//Make a copy of master since select changes the value of master

read\_fds = master;

//Select system call is used such that it can monitor multiple file descriptors

if(select(fdmax+1, &read\_fds, NULL, NULL, NULL) == -1){

perror("Select Error: ");

exit(1);

}

//File Descriptors are monitored

for(i=0; i <= fdmax; i++ )

//Checks if the file descriptor is set

if(FD\_ISSET(i,&read\_fds)){

/\*printf("%s:\n",users);\*/

//Call a function

send\_recv(i,sockfd);

}

}

//Make client to exit

printf("Client has quited Successfully\n");

//close socket

close(sockfd);

return 0;

}

/\* Function to open a TCP Socket

Parameters:int \*sockfd and struct sockaddr\_in \*server\_addr

return:void

\*/

void connect\_request(int \*sockfd, struct sockaddr\_in \*server\_addr)

{

//Open a socket

if ((\*sockfd = socket(AF\_INET, SOCK\_STREAM, 0)) == -1) {

perror("Socket Error: ");

exit(1);

}

server\_addr->sin\_family = AF\_INET;

server\_addr->sin\_port = htons(6000);

server\_addr->sin\_addr.s\_addr = inet\_addr("127.0.0.1");

memset(server\_addr->sin\_zero, '\0', sizeof server\_addr->sin\_zero);

//Connect to server

if(connect(\*sockfd, (struct sockaddr \*)server\_addr, sizeof(struct sockaddr)) == -1) {

perror("Connect Error: ");

exit(1);

}

}

/\* Function: send\_user

parameters: int sockfd

return:void

\*/

void send\_user(int sockfd)

{

//Initalise a receive buffer to recieve from server

char recv\_buf[MAX];

//Initalise values for response and choice

int response,choice;

/\*printf("sending name");\*/

//Send user to server

send(sockfd, users, strlen(users), 0);

//Receive ack from server

recv(sockfd, recv\_buf, sizeof(recv\_buf), 0);

//Note the response given by server

response=atoi(recv\_buf);

if(response==0){

printf("You are Not Registered\n");

printf("Do you want to enter the Chatroom?\n1:Yes\n2:No\n");

scanf("%d",&choice);

if(choice==1){

//Enter user to chat by sending to server

send(sockfd, users, strlen(users), 0);

response=1;

}

else{

//If other options exit chat room

printf("Exitting Chat-Room");

exit(0);

}

}

//If entered chat-room

if(response==1)

printf("Welcome to the Chat-Room: %s\n",users);

//flush the screen

fflush(stdout);

}

/\* Function:send and receive

Parameters: variable i and sockfd-socket\_descriptor

returns:void

\*/

void send\_recv(int i, int sockfd)

{

//Everytime increment the flag for each terminal

flag++;

//Initalise buffers to send and recieve

char send\_buf[MAX];

char recv\_buf[MAX];

//Initalise values to check bytes recieved

int nbyte\_recvd;

//A temporary user array

char users1[MAX];

//For every new terminal to exit if user enters quit

if (i == 0){

/\*printf("\nU:");\*/

//get user input from chat

fgets(send\_buf,MAX,stdin);

//if entered quit

if (strcmp(send\_buf,"quit\n") == 0

|| strcmp(send\_buf,"exit\n")==0) {

exit(0);

}

//Else other message

else if (strcmp(send\_buf,"$cu\n") == 0) {

/\*printf("client sending %s\n","$cu");\*/

//Send to socket by checking current message typed

send(sockfd,"$cu\n", strlen("$cu\n"), 0);

/\*fflush(stdout);

//printf("Completed sending");

//fflush(stdout);

\*/

}

//To display message as "uname: "

else{

//Copy present user to temporary user1 array

strcpy(users1,users);

//Append with:

strcat(users1,":");

//Concatenate with send\_buf

strcat(users1,send\_buf);

//copy users1 to send\_buf

strcpy(send\_buf,users1);

//Send to server

send(sockfd, send\_buf, strlen(send\_buf), 0);

//End with null character

strcpy(users1,"\0");

}

}

//If not store in a recieve buffer and print on screen

else {

nbyte\_recvd = recv(sockfd, recv\_buf, MAX, 0);

recv\_buf[nbyte\_recvd] = '\0';

printf("%s\n",recv\_buf);

fflush(stdout);

}

}