**SERVER PROGRAM**

/\* SERVER PROGRAM

Author:Shrikrishna Bhat

\*/

//All Header Files

#include<stdio.h>

#include<limits.h>

#include<stdlib.h>

#include<string.h>

#include<unistd.h>

#include<sys/types.h>

#include<sys/socket.h>

#include<netinet/in.h>

#include<arpa/inet.h>

#include<netdb.h>

//Define Port number

#define PORT 6000

//Define Max buffer

#define MAX 1024

//Define a structure for users

struct users{

int id;

char name[10];

int status;

}cusers[10];

//Set counter flag

static int counter=0;

//Function Prototypes

int check\_user(char uname[MAX]);

int validate\_user(int sockfd);

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

void connection\_accept(fd\_set \*master, int \*fdmax, int sockfd, struct sockaddr\_in \*client\_addr);

void send\_to\_all(int j, int i, int sockfd, int nbytes\_recvd, char \*recv\_buf, fd\_set \*master,int cmd);

void send\_recv(int i, fd\_set \*master, int sockfd, int fdmax);

//Main Function

int main()

{

//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;

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

int fdmax, i;

//Initalise a socket variable

int sockfd= 0;

struct sockaddr\_in my\_addr, client\_addr;

//Clear master set

FD\_ZERO(&master);

//Clear temp set

FD\_ZERO(&read\_fds);

//Open TCP conn using function

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

//Set the value of maste

FD\_SET(sockfd, &master);

//Give the value of sockfd to fdmax

fdmax = sockfd;

//make it run infinitely

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");

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)){

if (i == sockfd){

//If it refers to socket then call this fn

connection\_accept(&master, &fdmax, sockfd, &client\_addr);

}

else

//For new Socket call this fn

send\_recv(i, &master, sockfd, fdmax);

}

}

}

return 0;

}

/\* Function to Check the user

parameter:uname

return:int

\*/

int check\_user(char uname[MAX])

{

//Initalise a file pointer

FILE \*fp;

//Length of a file

size\_t len=0;

ssize\_t read;

char line[MAX];

//Open file

fp=fopen("user\_data.txt","r");

while ((fread(line,sizeof(line),1,fp))){

if(strncmp(line,uname,(read-1))==0){

fclose(fp);

return 1;

}

//If exists

printf("Inside the Chat-Room\n");

}

fclose(fp);

return 0;

}

/\* Function to validate the user

parameter:sockfd

return:int

\*/

int validate\_user(int sockfd)

{

//Initalise a recieve buffer and uname to store username

char recv\_buf[MAX],uname[MAX];

//Receieve The username from client

if ((recv(sockfd,recv\_buf,MAX,0))<= 0) {

printf("Receive Error\n");

}

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

//Call the check\_user to check if he is existing or not

if(check\_user(recv\_buf)){

//If the user exists. Then let him to chat

printf("Welcome To the Chat-Room %s",recv\_buf);

//Add to the local list of users to monitor.

strcpy(cusers[counter].name,recv\_buf);

//Send the response '1' to the client that client can proceed with chatting process

if (send(sockfd,"1", sizeof("1"), 0) == -1) {

printf("Socket Error");

}

}

else{

//User is not registered.Print it

printf("%s You are not registered!",recv\_buf);

fflush(stdout);

//Stop The user temporarily and send the response to client such that user can register

strcpy(uname,recv\_buf);

if (send(sockfd,"0", sizeof("0"), 0) == -1) {

printf("Send Error");

}

//Terminate a recieve buffer with '\0'

strncpy(recv\_buf,"\0",MAX);

//Receieve the response from client to register

if ((recv(sockfd,recv\_buf,MAX, 0)) <= 0) {

printf("Receive Error\n");

}

//Register The user

else{

//Add the user to the file and make him as a registerd user

FILE \*fp;

fp=fopen("user\_data.txt","a");

fwrite(uname,1,sizeof(uname),fp);

/\*fwrite("\n",1,sizeof("\n"),fp);\*/

fclose(fp);

printf("User Registered Successfully....\n");

//Keep track of the user in the counter

strcpy(cusers[counter].name,uname);

}

}

strncpy(recv\_buf,"\0",MAX);

fflush(stdout);

/\*printf("uname=%s\n",uname);\*/

//If the user is online give his status as 1

cusers[counter].status=1;

//Assign his id with the socket id

cusers[counter].id=sockfd;

//He is online.So print his name status

printf("\nName %d:%s\nStatus %d:%d\nId %d:%d\n",counter,cusers[counter].name,counter,cusers[counter].status,counter,cusers[counter].id);

//Increment The counter for multiple users

counter++;

}

/\* 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 \*my\_addr)

{

//Variable if connect Exists

int yes = 1;

//Open Socket

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

perror("Socket");

exit(1);

}

//Connect Socket

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

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

my\_addr->sin\_addr.s\_addr = INADDR\_ANY;

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

if (setsockopt(\*sockfd, SOL\_SOCKET, SO\_REUSEADDR, &yes, sizeof(int)) == -1) {

perror("setsockopt");

exit(1);

}

//Bind The Socket

if (bind(\*sockfd, (struct sockaddr \*)my\_addr, sizeof(struct sockaddr)) == -1) {

perror("Unable to bind");

exit(1);

}

//Listen to The Connection

if (listen(\*sockfd, 10)==-1) {

perror("listen");

exit(1);

}

printf("\nTCP Server Waiting for client on port 6000\n");

fflush(stdout);

}

/\* Function To accept Connection from Client

Parameters:fd\_set \*master, int \*fdmax, int sockfd, struct sockaddr\_in \*client\_addr

return:void

\*/

void connection\_accept(fd\_set \*master, int \*fdmax, int sockfd, struct sockaddr\_in \*client\_addr)

{

//Length of address

socklen\_t addrlen;

//Socket Variable

int newsockfd;

//Length of socket structure

addrlen = sizeof(struct sockaddr\_in);

//Accept Connection

if((newsockfd = accept(sockfd, (struct sockaddr \*)client\_addr, &addrlen)) == -1) {

perror("Accept Error");

exit(1);

}

else {

//Set the new socket and assign it a connection

FD\_SET(newsockfd, master);

if(newsockfd > \*fdmax){

\*fdmax = newsockfd;

}

printf("New connection Accepted: %s on Port number: %d \n",inet\_ntoa(client\_addr->sin\_addr), ntohs(client\_addr->sin\_port));

}

//Check For the validity of the user passed for the new connection

validate\_user(newsockfd);

}

/\* Function To send The message to all the users

Parameters:int j,int i,int sockfd,int bytes,char\* recv\_buf,fd\_set \*master,int cmd

return:void

\*/

void send\_to\_all(int j, int i, int sockfd, int nbytes\_recvd, char \*recv\_buf, fd\_set \*master,int cmd)

{

//Check if all the file descriptors are set correctly

if (FD\_ISSET(j,master)){

if(cmd==1){

//Check for the value if single client is present

if(i==j){

if (send(j,recv\_buf,nbytes\_recvd,0) == -1) {

perror("Send Error");

}

}

}

//For multiple Clients

else{

if (j != sockfd && j!=i) {

if (send(j, recv\_buf, nbytes\_recvd, 0) == -1) {

perror("Send Error");

}

}

}

}

}

/\* Function: To send and receive messagaes

Parameters:int i,fd\_set \*master,int sockfd,int fdmax

Return:void

\*/

void send\_recv(int i, fd\_set \*master, int sockfd, int fdmax)

{

//Initialise to receive bytes

int nbytes\_recvd, j,k;

char recv\_buf[MAX], buf[MAX];

//Initalise to check the sender

int sender;

//Check if the message is sent or not

if ((nbytes\_recvd = recv(i,recv\_buf,MAX,0))<=0) {

if (nbytes\_recvd == 0) {

printf("Socket %d Hung\n", i);

//Check for the Users who left the Chat-Room

for(k=0;k<=counter;k++){

if(i==cusers[k].id && cusers[k].status==1){

cusers[k].status=0;

printf("\n User: %s has left Chat-Room\nStatus:%d\n",cusers[k].name,cusers[k].status);

}

}

}

else {

perror("Receive Error");

}

//Give The value to the current sender

sender=i;

//close the connection

close(i);

//Clear the i flag and master

FD\_CLR(i, master);

}

else {

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

//printf("server received %s.",recv\_buf);\*/

//Check User and store in db

if(strncmp(recv\_buf,"$cu",(nbytes\_recvd-1))==0){

printf("Check user began!");

/\*fflush(stdout);\*/

//Send The list

printf("Sending list\n");

for(j =4; j<=fdmax; j++){

/\*printf("Sending it to id:%d",cusers[k].id);\*/

//Give the value of counter flag to each user

printf("Counter value:%d\nId no.%d is sending\nvalue of j=%d\n",counter,sender,j);

for(k=0;k<counter;k++){

if(cusers[k].status==1)

//Send message to all the users

send\_to\_all(j, i,sockfd, sizeof(cusers[k].name),cusers[k].name, master,1);

}

}

//Flush the screen

fflush(stdout);

printf("Completed Sending list\n");

}

//Send message to all the users

else

for(j = 0; j <= fdmax; j++){

send\_to\_all(j, i, sockfd, nbytes\_recvd, recv\_buf, master,0);

}

}

}