**Network and Communication CSE1004**

**Single Client and Multiple Client Chatting**

**Name:** Ayush Sharma

**Reg. No:** 15BCE1335

**Faculty:** Dr. T. Subbulakshmi

**SINGLE CLIENT**

**SERVER**

#include <stdio.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <sys/types.h>

#include <string.h>

#include <unistd.h>

#define PORT 1248

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

{

char buffer[20];

int sockfd,a,connfd,len,pid;

struct sockaddr\_in servaddr,cliaddr;

sockfd=socket(AF\_INET,SOCK\_STREAM,0);

if(sockfd==-1)

printf("Error creating socket\n");

bzero(&servaddr,sizeof(servaddr));

servaddr.sin\_family=AF\_INET;

servaddr.sin\_port=htons(PORT);

servaddr.sin\_addr.s\_addr=htonl(INADDR\_ANY);

bind(sockfd,(struct sockaddr\*)&servaddr,sizeof(servaddr));

if((a=listen(sockfd,5))<0)

printf("Error in LISTEN function");

while(1)

{

len=sizeof(cliaddr);

connfd=accept(sockfd,(struct sockaddr\*)&cliaddr,&len);

pid=fork();

if(pid==0)

{ close(sockfd);

while(1)

{

strcpy(buffer,"");

read(connfd,buffer,sizeof(buffer));

printf("Msg received is :%s",buffer);

printf("Enter your msg : ");

fgets(buffer,sizeof(buffer),stdin);

write(connfd,buffer,sizeof(buffer));

}

}

else

close(connfd);

}

}

**CLIENT**

#include <stdio.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <sys/types.h>

#include <arpa/inet.h>

#include <string.h>

#include <unistd.h>

#define PORT 1248

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

{

int sockfd;

struct sockaddr\_in serv\_addr;

char buff[20];

sockfd = socket(AF\_INET, SOCK\_STREAM, 0);

memset(&serv\_addr,'0', sizeof(serv\_addr));

serv\_addr.sin\_family = AF\_INET;

serv\_addr.sin\_port = htons(PORT);

serv\_addr.sin\_addr.s\_addr=inet\_addr("127.0.0.1");

if( connect(sockfd, (struct sockaddr \*)&serv\_addr, sizeof(serv\_addr)) < 0)

{

printf("\n Error : Connect Failed \n");

return 1;

}

while(1)

{

printf("Enter the string to echo: ");

fgets(buff,sizeof(buff),stdin);

write(sockfd,buff,sizeof(buff));

strcpy(buff,"");

printf("Msg received is: ");

read(sockfd,buff,sizeof(buff));

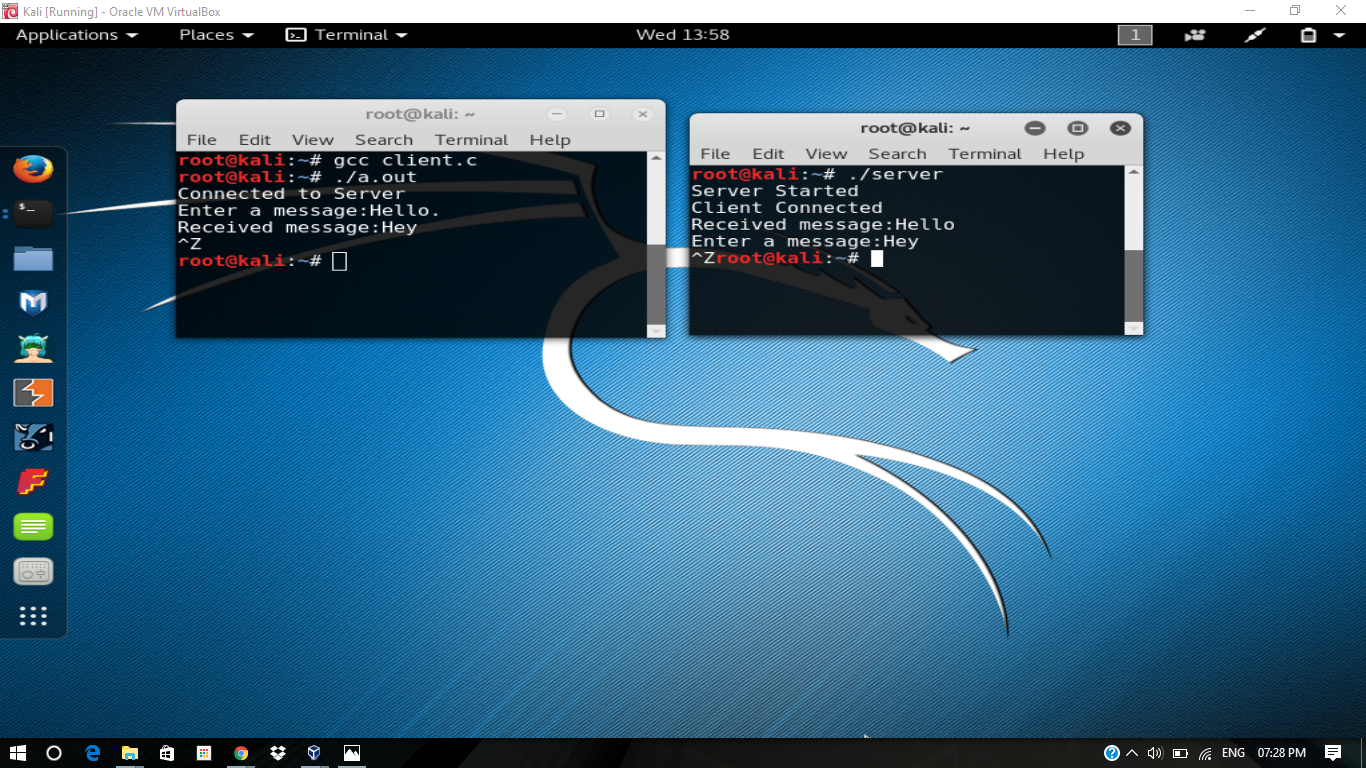
fputs(buff,stdout);

}

close(sockfd);

return 0;

}



**MULITPLE CLIENT**

**SERVER**

import java.io.DataInputStream;

import java.io.PrintStream;

import java.io.IOException;

import java.net.Socket;

import java.net.ServerSocket;

public class MultiThreadChatServer {

private static ServerSocket serverSocket = null;

private static Socket clientSocket = null;

private static final int maxClientsCount = 10;

private static final clientThread[] threads = new clientThread[maxClientsCount];

public static void main(String args[]) {

int portNumber = 6000;

if (args.length < 1) {

System.out

.println("Usage: java MultiThreadChatServer <portNumber>\n"

+ "Now using port number=" + portNumber);

} else {

portNumber = Integer.valueOf(args[0]).intValue();

}

try {

serverSocket = new ServerSocket(portNumber);

} catch (IOException e) {

System.out.println(e);

}

while (true) {

try {

clientSocket = serverSocket.accept();

int i = 0;

for (i = 0; i < maxClientsCount; i++) {

if (threads[i] == null) {

(threads[i] = new clientThread(clientSocket, threads)).start();

break;

}

}

if (i == maxClientsCount) {

PrintStream os = new PrintStream(clientSocket.getOutputStream());

os.println("Server too busy. Try later.");

os.close();

clientSocket.close();

}

} catch (IOException e) {

System.out.println(e);

}

}

}

}

**CLIENT**

import java.io.DataInputStream;

import java.io.PrintStream;

import java.io.IOException;

import java.net.Socket;

import java.net.ServerSocket;

class clientThread extends Thread {

private DataInputStream is = null;

private PrintStream os = null;

private Socket clientSocket = null;

private final clientThread[] threads;

private int maxClientsCount;

public clientThread(Socket clientSocket, clientThread[] threads) {

this.clientSocket = clientSocket;

this.threads = threads;

maxClientsCount = threads.length;

}

public void run() {

int maxClientsCount = this.maxClientsCount;

clientThread[] threads = this.threads;

try {

/\*

\* Create input and output streams for this client.

\*/

is = new DataInputStream(clientSocket.getInputStream());

os = new PrintStream(clientSocket.getOutputStream());

os.println("Enter your name.");

String name = is.readLine().trim();

os.println("Hello " + name

+ " to our chat room.\nTo leave enter /quit in a new line");

for (int i = 0; i < maxClientsCount; i++) {

if (threads[i] != null && threads[i] != this) {

threads[i].os.println("\*\*\* A new user " + name

+ " entered the chat room !!! \*\*\*");

}

}

while (true) {

String line = is.readLine();

if (line.startsWith("/quit")) {

break;

}

for (int i = 0; i < maxClientsCount; i++) {

if (threads[i] != null) {

threads[i].os.println("<" + name + "&gr; " + line);

}

}

}

for (int i = 0; i < maxClientsCount; i++) {

if (threads[i] != null && threads[i] != this) {

threads[i].os.println("\*\*\* The user " + name

+ " is leaving the chat room !!! \*\*\*");

}

}

os.println("\*\*\* Bye " + name + " \*\*\*");

/\*

\* Clean up. Set the current thread variable to null so that a new client

\* could be accepted by the server.

\*/

for (int i = 0; i < maxClientsCount; i++) {

if (threads[i] == this) {

threads[i] = null;

}

}

/\*

\* Close the output stream, close the input stream, close the socket.

\*/

is.close();

os.close();

clientSocket.close();

} catch (IOException e) {

}

}

}

**SCREENSHOT**

