Assignment- 13

ELP - 718 Telecom Software Laboratory

Ch Krishna Chaitanya
2019JTM2674
2019-21

A report on Socket Programming using C

Bharti School of
Telecommunication Technology and Management
IIT Delhi
India

November 6, 2019

Contents

1 Problem Statement -1 **2**
1.1 Problem Satement 2
1.2 Algorithm and Implementation 2
1.3 Flowchart 3
1.4 Test Cases 4
1.4.1 Input 4
1.4.2 Output 4
1.5 Screenshots 4
1.5.1 Screenshot1 4

2 Problem Statement -2 **5**
2.1 Problem Satement 5
2.2 Algorithm and Implementation 5
2.3 Flowchart 6
2.4 Test Cases 7
2.4.1 Input 7
2.4.2 Output 7
2.5 Screenshots 7
2.5.1 Screenshot1 7
2.5.2 Screenshot2 7

Appendices **8**

References **18**

Objective Statement

To test our understanding on Socket programming using C.

1 Problem Statement -1

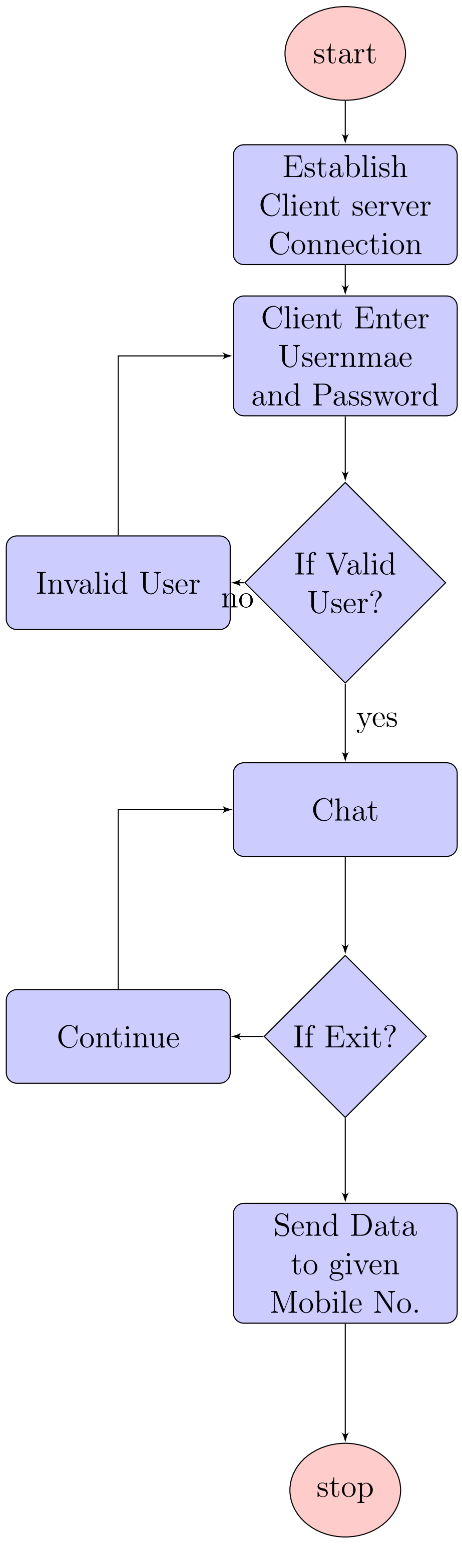
To Create a server that is capable of handling multiple clients using TCP communication sockets

1.1 Problem Satement

1.2 Algorithm and Implementation

- Create File with username and password
- Create a Server capable of connecting to multiple clients
- Validate User from Client Side
- After finishing chat, ask for mobile number to send messages to particular mobile number
- If user is not present store it in a file
- After availability of user, send the stored chat
- Delete the file after sending chat information.

1.3 Flowchart


start
Establish Client server Connection
Client Enter Usernmae and Password
If Valid User?
no
Invalid User
yes
Chat
If Exit?
Continue
Send Data to given Mobile No.
stop

1.4 Test Cases

1.4.1 Input

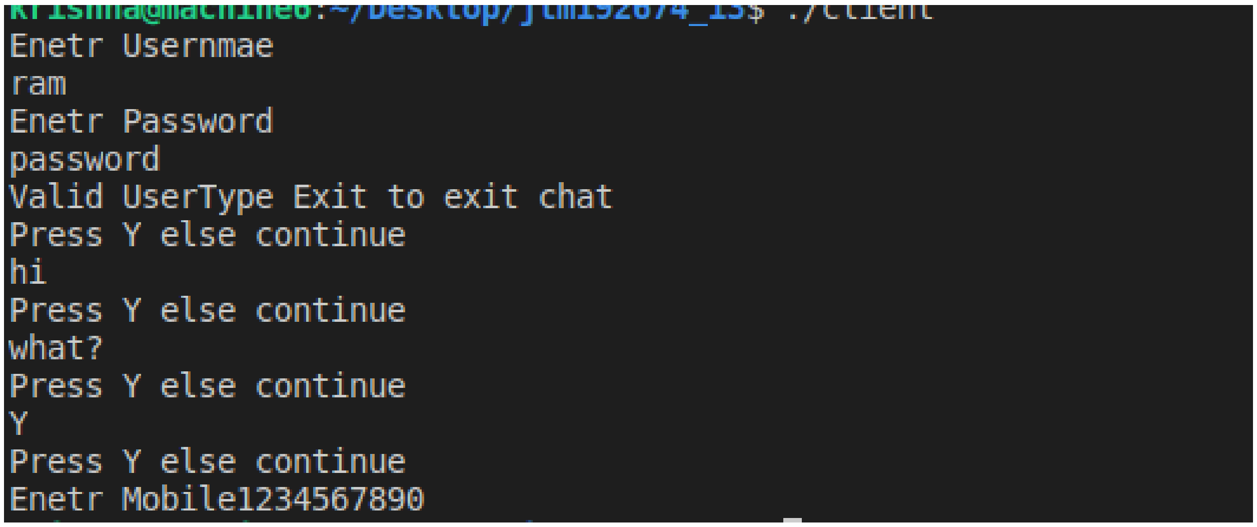
Enter User name
Enter Password

1.4.2 Output

start Chat

1.5 Screenshots

1.5.1 Screenshot1



```
krishna@machine6:~/Desktop/jtm192674_13$ ./client
Enetr Usernmae
ram
Enetr Password
password
Valid UserType Exit to exit chat
Press Y else continue
hi
Press Y else continue
what?
Press Y else continue
Y
Press Y else continue
Enetr Mobile1234567890
```

2 Problem Statement -2

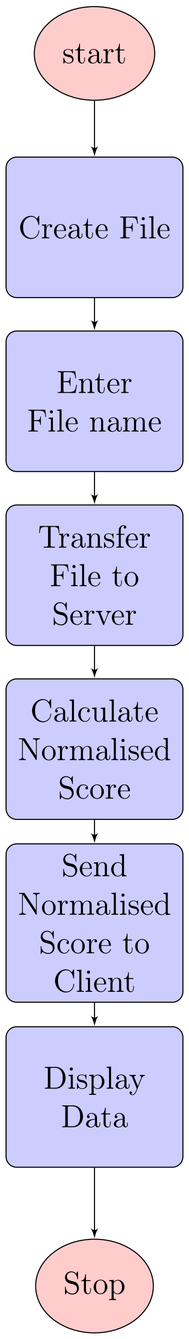
To Create a normalised score rating application.

2.1 Problem Satement

2.2 Algorithm and Implementation

- Create File with name and other information
- Input File Name
- Send file data from client side to server side
- On server side, assign the rating as given
- Calculate Normalised Percentage
- Send it to client
- Display Information

2.3 Flowchart



2.4 Test Cases

2.4.1 Input

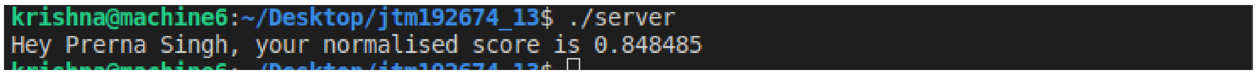
Enter File Nmae
file1.txt

2.4.2 Output

Hey Prerna Singh, your normalised score is 0.848485

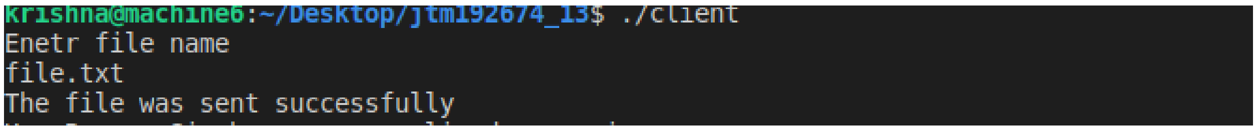
2.5 Screenshots

2.5.1 Screenshot1



```
krishna@machine6:~/Desktop/jtm192674_13$ ./server
Hey Prerna Singh, your normalised score is 0.848485
```

2.5.2 Screenshot2



```
krishna@machine6:~/Desktop/jtm192674_13$ ./client
Enetr file name
file.txt
The file was sent successfully
```

Appendices

Problem 1

code:

```
// Server side C/C++ program to demonstrate Socket programming
#include <unistd.h>
#include <stdio.h>
#include <sys/socket.h>
#include <stdlib.h>
#include <netinet/in.h>
#include <string.h>
#define PORT 8080
int main(int argc, char const *argv[])
{
int server_fd, new_socket, valread;
    int serverReceive;
struct sockaddr_in address;
int opt = 1;
int addrlen = sizeof(address);
char buffer[1024] = {0};
char *hello = "Hello from server";

// Creating socket file descriptor
if ((server_fd = socket(AF_INET, SOCK_STREAM, 0)) == 0)
{
perror("socket failed");
exit(EXIT_FAILURE);
}

// Forcefully attaching socket to the port 8080
if (setsockopt(server_fd, SOL_SOCKET, SO_REUSEADDR | SO_REUSEPORT,
&opt, sizeof(opt)))
{
perror("setsockopt");
exit(EXIT_FAILURE);
}
address.sin_family = AF_INET;
address.sin_addr.s_addr = INADDR_ANY;
address.sin_port = htons( PORT );
```

```
// Forcefully attaching socket to the port 8080
if (bind(server_fd, (struct sockaddr *)&address,
sizeof(address))<0)
{
perror("bind failed");
exit(EXIT_FAILURE);
}
if (listen(server_fd, 3) < 0)
{
perror("listen");
exit(EXIT_FAILURE);
}
if ((new_socket = accept(server_fd, (struct sockaddr *)&address,
(socklen_t*)&addrlen))<0)
{
perror("accept");
exit(EXIT_FAILURE);
}



    char user[30], pass[30],data[1024],option[30]={0};
    char f_user[30], f_pass[30],f_mob[30],mobile_no[20];
    char message[30]="Valid User",invalid[30]="Invalid User";
    FILE *fp;
    int valid=0;
    //Receive Username
serverReceive = read( new_socket , buffer, 1024);
strcpy(user,buffer);
bzero(buffer,sizeof(buffer));


//Recieve Password
serverReceive = read( new_socket , buffer, 1024);
strcpy(pass,buffer);
    bzero(buffer,sizeof(buffer));

    printf("%s %s",user,pass);
    fp =fopen("user.txt","r");
    if ( fp == NULL )
{
puts ( "Cannot open file" ) ;
exit(0) ;
}
while (fscanf ( fp, "%s %s %s", f_user,f_mob,f_pass ) != EOF )
{

         if(strcmp(f_pass,pass)==0&&strcmp(f_user,user)==0 )

         valid=1;
}
fclose ( fp ) ;

    if(valid==1){
        send(new_socket , message , strlen(message) , 0 );
    }
    else
    {
        send(new_socket , invalid , strlen(invalid) , 0 );
    }


    while(1){
        serverReceive = read( new_socket , buffer, 1024);
        //printf("%s",buffer);
        int z = strcmp(buffer,"Y");
if(z==10){
         serverReceive = read( new_socket , buffer, 1024);
            strcpy(mobile_no,buffer);
            printf("%s",mobile_no);
break;
}
    }
    // serverReceive = read( new_socket , buffer, 1024);
    // printf("%s",buffer);
    // bzero(buffer,sizeof(buffer));
return 0;
}
```

Problem 2

Server code:

```
// Server side C/C++ program to demonstrate Socket programming
```

```
#include <unistd.h>
#include <stdio.h>
#include <sys/socket.h>
#include <stdlib.h>
#include <netinet/in.h>
#include <string.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <netdb.h>
#include<ctype.h>
#include<stdlib.h>
#define PORT 6001
void delay()  //Delay function to avoid overlapping of data
{
   int c, d;

   for (c = 1; c <= 32767; c++)
       for (d = 1; d <= 32767; d++)
       {}

}
int main(int argc, char const *argv[])
{
int server_fd, new_socket, valread;
struct sockaddr_in address;
int opt = 1;
int addrlen = sizeof(address);
char buffer[1024] = {0};
char *hello = "Hello from server";
char k[10]="user";
// Creating socket file descriptor
if ((server_fd = socket(AF_INET, SOCK_STREAM, 0)) == 0)
{
perror("socket failed");
exit(EXIT_FAILURE);
}

// Forcefully attaching socket to the port 8080
if (setsockopt(server_fd, SOL_SOCKET, SO_REUSEADDR | SO_REUSEPORT,
&opt, sizeof(opt)))
{
perror("setsockopt");
```

```
exit(EXIT_FAILURE);
}
address.sin_family = AF_INET;
address.sin_addr.s_addr = INADDR_ANY;
address.sin_port = htons( PORT );

// Forcefully attaching socket to the port 8080
if (bind(server_fd, (struct sockaddr *)&address,
sizeof(address))<0)
{
perror("bind failed");
exit(EXIT_FAILURE);
}
if (listen(server_fd, 3) < 0)
{
perror("listen");
exit(EXIT_FAILURE);
}
if ((new_socket = accept(server_fd, (struct sockaddr *)&address,
(socklen_t*)&addrlen))<0)
{
perror("accept");
exit(EXIT_FAILURE);
}


    FILE *fp;
    char ch;
    char m[30];
    char first_name[50],last_name[50],ten[20],inter[20],grad[20],work_exp[20];
    char s_first[50],s_last[50];
    char f[20],l[20],t[20],i[20],g[20],w[20];
    int ten_p,inter_p,grad_p;
    float work;
    int bi;

    float a,b,c,d,norm_score=0,s_norm;
    fp = fopen("new_file.txt","w+");
    if ( fp == NULL )
{
puts ( "Cannot open file" ) ;
exit(0) ;
```

```
}
    while((bi=recv(new_socket,buffer,1024,0))>0){
        //read(new_socket,buffer,1024);
        fprintf(fp,"%s\t",buffer);

        // ch=fgetc(fp);
    }

    fseek(fp,0,SEEK_SET);
    //send(new_socket , k , strlen(k) , 0 );
    while(fscanf ( fp, "%s %s %s %s %s %s %s %s %s %s %s %s",  f,first_name,l,last_name,
        ten_p = atoi(ten);
        inter_p=atoi(inter);
        grad_p=atoi(grad);
        if(ten_p>90)
            a=10;
        else if(ten_p>80 && ten_p<=90)
            a=8;
        else if(ten_p>70 && ten_p<=80)
            a=5;
        else if(ten_p>60 && ten_p<=70)
            a=3;
        else if(ten_p>55 && ten_p<=50)
            a=2;
        else if(ten_p<=55)
            a=1;


        if(inter_p>90)
            b=10;
        else if(inter_p>80 && inter_p<=90)
            b=8;
        else if(inter_p>70 && inter_p<=80)
            b=5;
        else if(inter_p>60 && inter_p<=70)
            b=3;
        else if(inter_p>55 && inter_p<=50)
            b=2;
        else if(inter_p<=55)
            b=1;
        if(grad_p>85)
            c=10;
        else if(grad_p>75 && grad_p<=85)
            c=8;
        else if(grad_p>70 && grad_p<=75)
            c=5;
        else if(grad_p>65 && grad_p<=70)
            c=3;
        else if(grad_p>60 && grad_p<=65)
            c=2;
        else if(grad_p<=60)
            c=1;


        if(strcmp(work_exp,"NIL")==0){
            d=0;
            // printf("Here");
        }
        if(strcmp(work_exp,"NIL")!=0){
            work = atof(work_exp);
            // printf("work %f",work);
            if(work>36)
                d=1;
            else if(work>12 && work<=36){
                d= 1 + (work-12)/24;
                }
            else if(work<=12){
                d=1;
                }
        }
        // printf("%f\t%f\t%f\t%f",a,b,c,d);
        norm_score = (a+b+c+d)/33;

        s_norm=norm_score;
        sprintf(buffer,"%f",s_norm);
        //printf("str %s",buffer);
        //send(new_socket , buffer , strlen(buffer) , 0 );
        printf("Hey %s %s, your normalised score is %f\n",first_name,last_name,norm_scor


    }
    fclose(fp);
```

```
    // sprintf(buffer,"%f",s_norm);
    // printf("str %s",buffer);
    // send(new_socket , buffer , strlen(buffer) , 0 );
return 0;
}
```

Client code:

```
// Client side C/C++ program to demonstrate Socket programming
#include <stdio.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <netdb.h>
#include<ctype.h>
#include<stdlib.h>

#define PORT 6001

int main(int argc, char const *argv[])
{
int sock = 0, valread;
struct sockaddr_in serv_addr;
char *hello = "Hello from client";
char buffer[1024] = {0};
    char file_name[50];
if ((sock = socket(AF_INET, SOCK_STREAM, 0)) < 0)
{
printf("\n Socket creation error \n");
return -1;
}

serv_addr.sin_family = AF_INET;
serv_addr.sin_port = htons(PORT);

// Convert IPv4 and IPv6 addresses from text to binary form
if(inet_pton(AF_INET, "127.0.0.1", &serv_addr.sin_addr)<=0)
{
printf("\nInvalid address/ Address not supported \n");
return -1;
}
```

```
if (connect(sock, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) < 0)
{
printf("\nConnection Failed \n");
return -1;
}

    printf("Enetr file name\n");
    scanf("%s",file_name);
    FILE *fp = fopen(file_name,"r+");
    char ch;
    float norm;
    char first_name[50],last_name[50];
    char f[50],l[50],j[20];
    int p;
    if(fp == NULL){
        perror("Error opening file");
    }

    while(ch != EOF){
        fscanf(fp,"%s",buffer);
        write(sock,buffer,1024);
        ch = fgetc(fp);
    }
    printf("The file was sent successfully\n");
    bzero(buffer,sizeof(buffer));
    fseek(fp,0,SEEK_SET);

    while(fscanf ( fp, "%s %s",f,l) != EOF){
        if(strcmp(f,"FirstName:")==0){
            strcpy(first_name,l);
        }
        if(strcmp(f,"LastName:")==0){
            strcpy(last_name,l);
        }
    }
    fclose(fp);
    //read(sock,&norm,sizeof(float));
    //p = read(sock , buffer, 1024);
    printf("%s",buffer);
    bzero(buffer,sizeof(buffer));
    //p = read(sock , buffer, 1024);
    printf("Hey %s %s, your normalised score is %s\n",first_name,last_name,buffer);
return 0;
}
```

References

[1] Flowchart using Latex
Kjell Magne Fauske
http://www.texample.net/tikz/examples/simple-flow-chart/

[2] Socket Programming
https://www.geeksforgeeks.org/socket-programming-cc/

[3] Socket Tutorial
http://www.linuxhowtos.org/C_C++/socket.htm

[4] Network Programming
https://beej.us/guide/bgnet/

[5] Sockets Tutorial
https://www.cs.rpi.edu/~moorthy/Courses/os98/Pgms/socket.html