

// Write-ups

## Assignment No. 6

- Title: TCP Sockets
- Problem Statement: Write a program using TCP socket for wired network for following
  - a) Say Hello to each other (for all students)
  - b) File Transfer (for all students)
  - c) Calculator Arithmetic (50% students)
  - d) Calculator Trigonometry (50% students)
- Objective: To,
  - 1) learn TCP socket programming for wired network using TCP socket API.
  - 2) learn the difference between TCP and UDP.
- Outcomes: I will be able to,
  - 1) Implement TCP socket Programming
  - 2) Differentiate between TCP & UDP
- Software and Hardware requirements:  
Fedora 20/Windows 10, 1GB RAM, 120 GB HDD,  
monitor, keyboard, mouse, Eclipse/Visual studio
- Theory:
  - a) TCP Socket Programming for wired network:  
The two key classes from the java.net package used in creation of server and client programs are

ServerSocket & Socket. A server program creates a specific type of socket that is used to listen for client requests. In the case of connection request, the program creates a new socket through which it will exchange data with the client using input and output streams. The socket abstraction is very similar to the file concept: developers have to open a socket, perform I/O and close it.

#### b) File Transfer

A TCP client initiates the communication with a server which is waiting for the connection. TCP is connection oriented and UDP is connectionless, which means that UDP sockets do not need to be connected before being used.

A TCP listener is created and starts listening to the specified port. Again the buffer size is set to 1024 bytes. A TCP listener can ~~precheck~~ check to see if there are any connections pending before calling the `acceptTcpClient` method. It returns true if there are any pending connections.

#### • A simple Server Program in Java

The steps for creating a simple server program are:

- 1) Open the Server Socket: `ServerSocket server = new ServerSocket(PORT);`
- 2) Wait for the Client Request: `Socket client = server.accept();`



- 3) Create I/O streams for communicating to the client:- `DataInputStream is = new DataInputStream (client.getInputStream());`  
`DataOutputStream os = new DataOutputStream (client.getOutputStream());`
- 4) Perform communication with client receive from client:  
`String line = is.readLine();` Send to client: `os.writeBytes ("Hello\n");`
- 5) Close socket: `client.close();`

A simple Client Program in Java

The steps for creating a simple client program are:

- 1) Create a socket object: `Socket client = new Socket (server, port_id);`
- 2) Create I/O streams for communicating with the server `is = new DataInputStream (client.getInputStream());`
- 3) Perform I/O or communication with the server.  
 Receive data from the server: `String line = is.readLine();`
- 4) Close the socket when done: `client.close();`

Test Cases:

	Input	Expected o/p	Actual o/p	Result
1)	selected opt to chat with server client → hello	client > hello Expecting server output server > hello expecting client output	Client > hello Expecting server output server > hello expecting client output	Success
2)	select opt 2 for file transfer Name of file: abc.txt	File transferred successfully	File transferred successfully	success
3)	select opt 3 for calculator input: 25/6	h	4	Success
<p>Conclusion:</p> <p>Thus, we successfully implemented the TCP Socket programming for wired network using TCP socket and learnt the difference between TCP and UDP.</p>				

// Sample Code

-----server.c

#include <stdio.h>

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

```
void error(const char *msg) {
    perror(msg);
    exit(1);
}
```

```
int main(int argc, char const *argv[]) {
    if(argc<2){
        fprintf(stderr,"Port number not provided.\n");
        exit(1);
    }
    int sockfd, newsockfd, portno, n;
    char buffer[255]; // The data to be sent to the server and received from it

    struct sockaddr_in serv_addr, cli_addr;
    socklen_t clilen;

    sockfd = socket(AF_INET, SOCK_STREAM, 0);    //SOCK_STREAM for TCP
    if(sockfd<0){
        error("Error opening the server socket");
    }
```

```

bzero((char *)&serv_addr,sizeof(serv_addr));

portno = atoi(argv[1]);


serv_addr.sin_family = AF_INET;
serv_addr.sin_addr.s_addr = INADDR_ANY;
serv_addr.sin_port = htons(portno);  // Host to network short


if(bind(sockfd,(struct sockaddr *)&serv_addr,sizeof(serv_addr))<0){
    error("Binding Failed.");
}


listen(sockfd, 5); // Waiting for client to connect
clilen = sizeof(cli_addr);


newsockfd = accept(sockfd,(struct sockaddr *)&cli_addr,&clilen);  // Client accepted


if(newsockfd < 0)
error("Error accepting");
if(portno==9890){  // Chat App from server to client and vice versa
    while(1){
        bzero(buffer,255);
        n = read(newsockfd,buffer,255);  // Reading from client
        if(n<0)
            error("Error on reading");
        printf("Client: %s\n",buffer);
        bzero(buffer,255);
        fgets(buffer,255,stdin); // Reading from the server terminal
        n = write(newsockfd,buffer,strlen(buffer));  // Sending to Client
        if(n<0)
            error("Error on writing\n");
        int i = strncmp("bye",buffer,3);
    }
}

```

```

        if (i==0)
            break;
    }
}

if(portno==9891){    // File Transfer
    FILE *fp;

    int ch =0;

    fp= fopen("receivedtextfile.txt","a");    // Append if the file already exists or else create a new
one
    int words;

    read(newsockfd,&words,sizeof(int));    // Read the word count

    while (ch!=words) {
        read(newsockfd,buffer,255);    // Read a word

        fprintf(fp,"%s ",buffer);    // Write that word to the file

        ch++;
    }

    printf("The file was received");
}

if(portno==9892){
    int num1,num2,answer,choice;

    char choices[5][15]={"Addition","Subtraction","Multiplication","Division","Exit"};

```

S:     n = write(newsockfd,"Enter number 1: ",strlen("Enter number 1: "));     // Sending a message  
to ask for num1

```

    if(n<0)
        error("Error on writing\n");

    read(newsockfd,&num1,sizeof(int));    // Reading num1 sent from client

    printf("Client number 1 is: %d\n",num1);

    n = write(newsockfd,"Enter number 2: ",strlen("Enter number 2: "));    // Sending a message
to ask for num2

    if(n<0)
        error("Error on writing\n");

```

```

read(newsockfd,&num2,sizeof(int)); // Reading num2 sent from client

printf("Client number 2 is: %d\n",num2);

n = write(newsockfd,"1. Addition\n2. Subtraction\n3. Multiplication\n4. Division\n5. Exit\n",
          strlen("1. Addition\n2. Subtraction\n3. Multiplication\n4. Division\n5. Exit\n")); //Sending
request for choice

if(n<0)
    error("Error on writing\n");

read(newsockfd,&choice,sizeof(int)); // Reading choice for operation

printf("Client operation is: %s\n",choices[choice-1]); // Fetching the opearation name from
string array

switch (choice) {
    case 1:
        answer = num1+num2; //addition
        break;
    case 2:
        answer = num1-num2; //Subtraction
        break;
    case 3:
        answer = num1*num2; //Multiplication
        break;
    case 4:
        answer = num1/num2; //Division
        break;
    case 5:
        goto Q; //Exit
        break;

}

write(newsockfd,&answer,sizeof(int));

if(choice!=5){ //Exit Case
    goto S;
}

```



```
}
```

```
}
```

```
Q: close(newsockfd);
```

```
close(sockfd);
```

```
return 0;
```

```
}
```

```
-----client.c
```

```
/*
```

```
filename server_ipaddress portno
```

```
*/
```

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
#include <string.h>
```

```
#include <unistd.h>
```

```
#include <sys/types.h>
```

```
#include <sys/socket.h>
```

```
#include <netinet/in.h>
```

```
#include <netdb.h>
```

```
#include <ctype.h>
```

```
void error(const char *msg) {
```

```
    perror(msg);
```

```
    exit(1);
```

```
}
```

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

```
    int sockfd,portno,n;
```

```
    struct sockaddr_in serv_addr;
```

```
    struct hostent *server;
```

```
    char buffer[255];
```

```
    if(argc<3){
```

```
        fprintf(stderr, "usage %s hostname port\nport 9890 for chat\nport 9891 for file transfer\nport 9892 for calculator\n",argv[0]);
```

```
        exit(1);
```

```
    }
```

```
    portno = atoi(argv[2]);  // String to integer
```

```
    sockfd = socket(AF_INET,SOCK_STREAM,0);
```

```
    if(sockfd < 0)
```

```
        error("Error opening Socket");
```

```

server = gethostbyname(argv[1]);

if(server == NULL)

fprintf(stderr,"Error, no such host");

bzero((char *)&serv_addr,sizeof(serv_addr));

serv_addr.sin_family = AF_INET;

bcopy((char*)server->h_addr,(char *)&serv_addr.sin_addr.s_addr,server->h_length);

serv_addr.sin_port = htons(portno); // host to network short

if(connect(sockfd,(struct sockaddr *)&serv_addr,sizeof(serv_addr))<0){ // Connecting to Server
    error("Connection failed");
}

if(portno==9890){ // Chat App
    while(1){
        bzero(buffer,255);

        fgets(buffer,255,stdin); // Reading from client terminal

        int j = strncmp("bye",buffer,3);

        if (j==0)

            break;

        n = write(sockfd,buffer,strlen(buffer)); // Sending to server

        if(n<0)

            error("Error on writing");

        bzero(buffer,255);

        n = read(sockfd,buffer,255); // Reading from server

        if(n<0)

            error("Error on reading");

        printf("Server says: %s\n",buffer); // Printing server's sent message

        int i = strncmp("bye",buffer,3);

        if (i==0)

            break;

    }
}

if(portno==9891){ // File Transfer

```

```

FILE *f;

int words = 0;

char c;

f = fopen("textfile.txt","r");    // Opening an already existing file to send its data to server
while((c= getc(f))!= EOF){        // Counting the number of words
    fscanf(f,"%s",buffer);
    if(isspace(c) || c=='\t')
        words++;
}

write(sockfd,&words,sizeof(int));    // Sending the number of words

rewind(f);    // Setting the file pointer at initial word


char ch;

while(ch!=EOF){
    fscanf(f,"%s",buffer);
    write(sockfd,buffer,255);    // Sending words one by one
    ch = fgetc(f);
}

printf("The file has been sent.\n");

}

if(portno==9892){
    while(1){
        int num1,num2,choice,answer;

        bzero(buffer,255);

        n = read(sockfd,buffer,255);    // Reading first request from server
        if(n<0)
            error("Error reading");

        printf("Server- %s",buffer);    // printing first request
        scanf("%d", &num1);    // getting num1 from client terminal
        n = write(sockfd,&num1,sizeof(int));    // sending num1 to server
    }
}

```



```

if (n<0) {
    error("Error writing\n");
}
bzero(buffer,255);
n = read(sockfd,buffer,255);    // Reading first request from server
if(n<0)
    error("Error reading");
printf("Server- %s",buffer);    // printing first request
scanf("%d", &num2);            // getting num1 from client terminal
n = write(sockfd,&num2,sizeof(int));    // sending num2 to server
if (n<0) {
    error("Error writing\n");
}
bzero(buffer,255);
n = read(sockfd,buffer,255);
if(n<0)
    error("Error reading");
printf("Server- %s\n",buffer);
scanf("%d", &choice);          // Reading operation choice from client

n = write(sockfd,&choice,sizeof(int));
if (n<0) {
    error("Error writing\n");
}
if(choice==5){    // Exit case
    goto E;
    break;
}
printf("Answer: ");
n = read(sockfd,&answer,sizeof(int));    // Getting the answer.
printf("%d\n",answer );                // printing the answer

```

```

    }
}

E: close(sockfd);

return 0;
}

```

## // Outputs

The image shows two terminal windows side-by-side, both titled 'durvesh@predator: ~/31139/SEMV/CNL/Assignment6'. The left window shows the server's execution, and the right window shows the client's execution.

**Left Terminal (Server):**

```

durvesh@predator:~/31139/SEMV/CNL/Assignment6$ gcc server.c -o server
durvesh@predator:~/31139/SEMV/CNL/Assignment6$ ./server 9890
Client: Hello

Hey
Client: This is a message from the client
Well this is a server from the server!!
Client:
bye
durvesh@predator:~/31139/SEMV/CNL/Assignment6$

```

**Right Terminal (Client):**

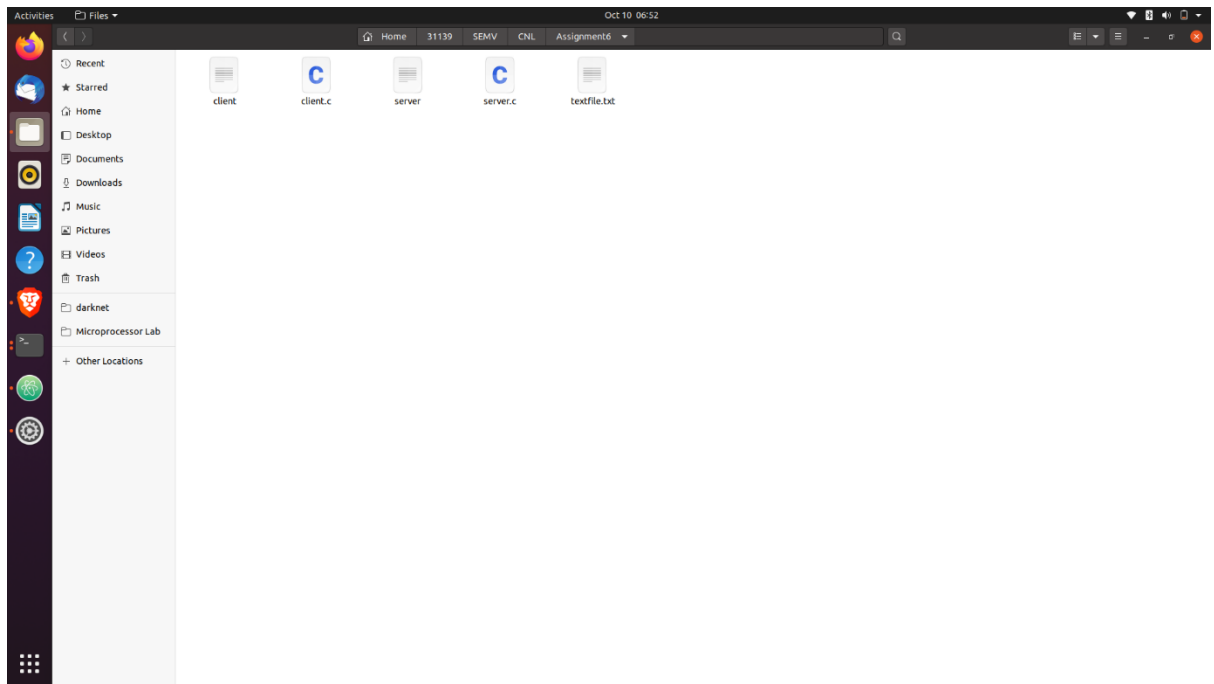
```

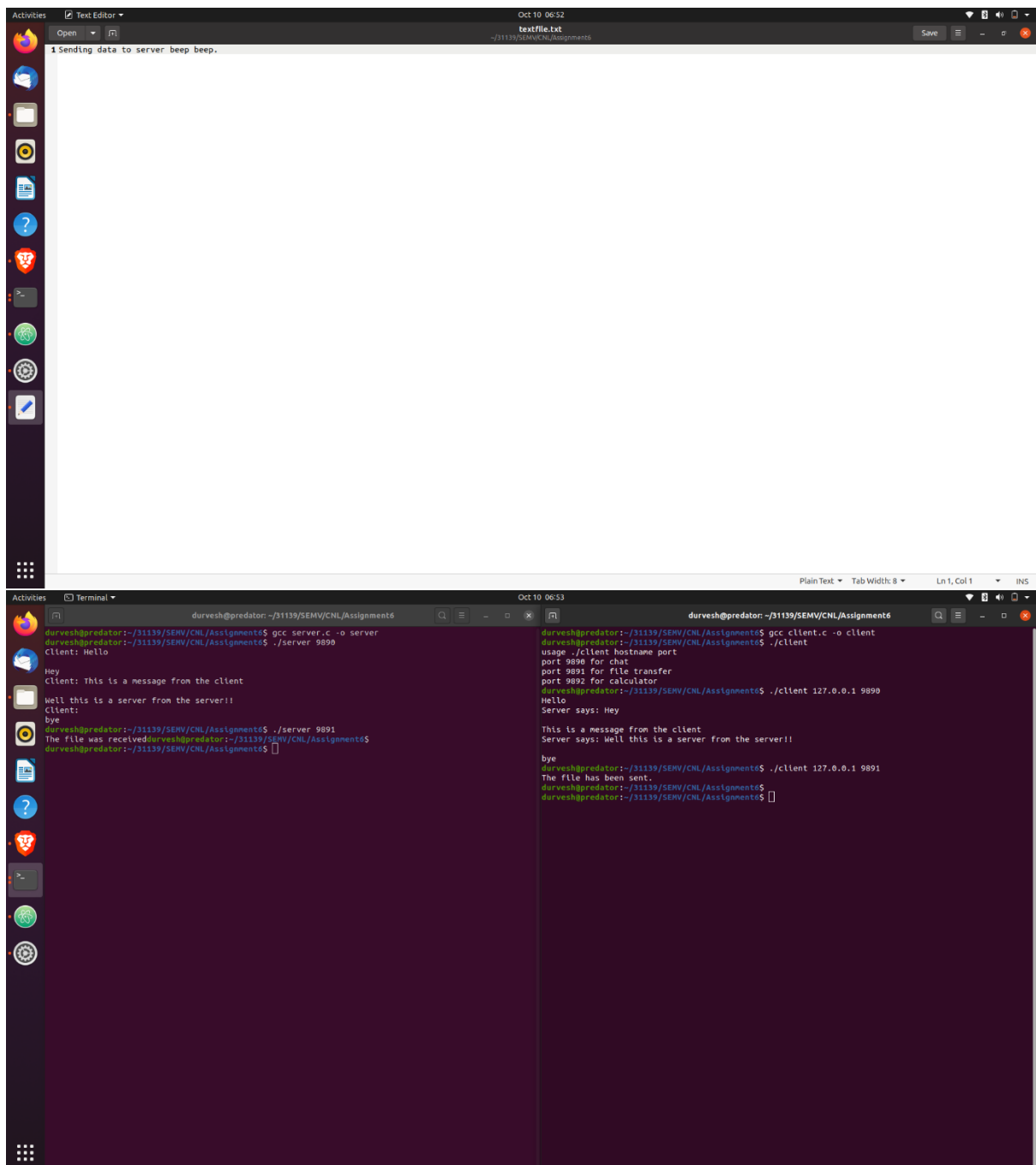
durvesh@predator:~/31139/SEMV/CNL/Assignment6$ gcc client.c -o client
durvesh@predator:~/31139/SEMV/CNL/Assignment6$ ./client
usage: ./client hostname port
port 9890 for chat
port 9891 for file transfer
port 9892 for calculator
durvesh@predator:~/31139/SEMV/CNL/Assignment6$ ./client 127.0.0.1 9890
Hello
Server says: Hey

This is a message from the client
Server says: Well this is a server from the server!!

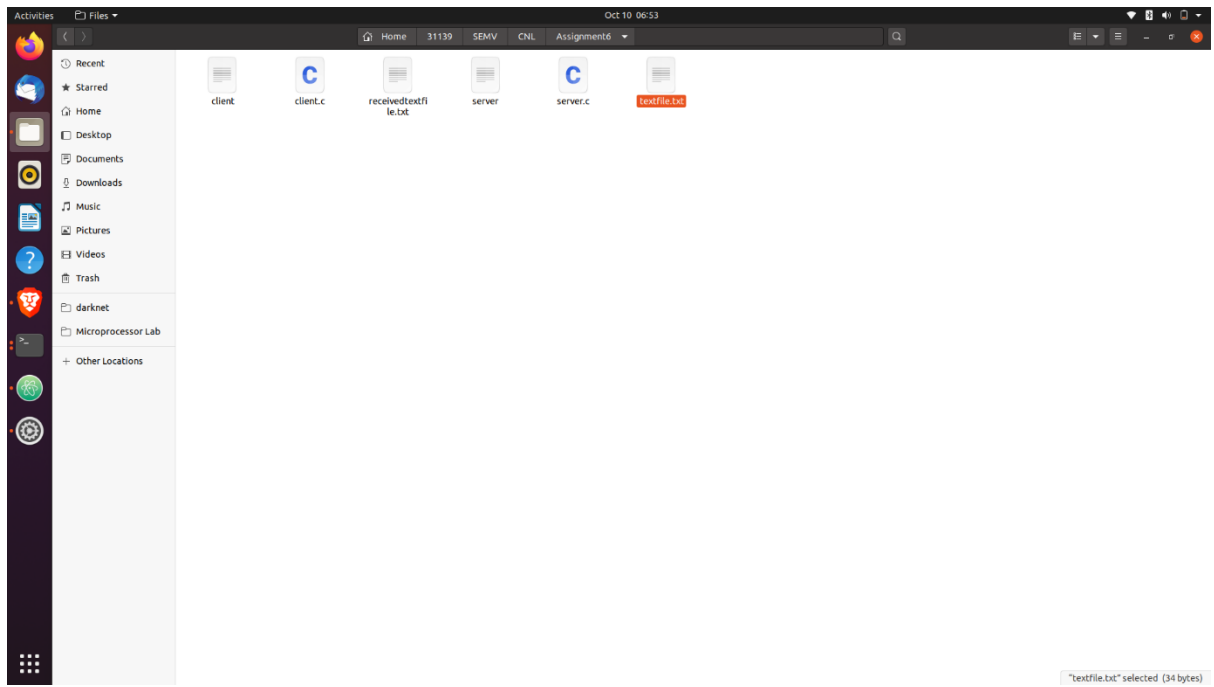
bye
durvesh@predator:~/31139/SEMV/CNL/Assignment6$

```









The image shows a Linux desktop environment with a text editor and a terminal window.

**Text Editor Window:**

- server.c:**

```
1 #include <stdio.h>
2 #include <unistd.h>
3 #include <sys/types.h>
4 #include <sys/socket.h>
5 #include <string.h>
6 #include <arpa/inet.h>
7
8 int main()
9 {
10     int server_fd, new_socket;
11     struct sockaddr_in address;
12     int port = 9890;
13     int n;
14     char buffer[1024];
15     int addrlen = sizeof(address);
16
17     // Creating socket on IPv4 and TCP domain
18     server_fd = socket(AF_INET, SOCK_STREAM, 0);
19
20     // Binding the socket with the IP and port
21     address.sin_addr = inet_addr("127.0.0.1");
22     address.sin_port = htons(9890);
23     bind(server_fd, (struct sockaddr *)&address, sizeof(address));
24
25     // Listening the socket
26     listen(server_fd, 5);
27
28     // Accepting incoming connections
29     new_socket = accept(server_fd, (struct sockaddr *)&address, &addrlen);
30
31     // Infinite loop to handle incoming connections
32     while(1)
33     {
34         n = read(new_socket, buffer, 1024);
35         if(n < 0)
36             continue;
37         printf("Client: %s\n", buffer);
38         write(new_socket, "Well this is a server from the server!!", 30);
39     }
40 }
```
- client.c:**

```
1 #include <stdio.h>
2 #include <unistd.h>
3 #include <sys/types.h>
4 #include <sys/socket.h>
5 #include <string.h>
6 #include <arpa/inet.h>
7
8 int main()
9 {
10     int client_fd;
11     struct sockaddr_in address;
12     int port = 9890;
13     int addrlen = sizeof(address);
14
15     // Creating socket on IPv4 and TCP domain
16     client_fd = socket(AF_INET, SOCK_STREAM, 0);
17
18     // Binding the socket with the IP and port
19     address.sin_addr = inet_addr("127.0.0.1");
20     address.sin_port = htons(9890);
21
22     // Connecting the client to the server
23     connect(client_fd, (struct sockaddr *)&address, sizeof(address));
24
25     // Sending data to the server
26     printf("Client: Hello\n");
27     write(client_fd, "Hey\n", 5);
28     printf("Client: This is a message from the client\n");
29     write(client_fd, "Well this is a server from the server!!\n", 30);
30     printf("Client: bye\n");
31     write(client_fd, "bye\n", 5);
32
33     // Receiving data from the server
34     char buffer[1024];
35     read(client_fd, buffer, 1024);
36     printf("Server: %s\n", buffer);
37
38     // Sending data to the server
39     write(client_fd, "The file has been sent.\n", 15);
40
41     // Receiving data from the server
42     read(client_fd, buffer, 1024);
43     printf("Server: %s\n", buffer);
44
45     // Sending data to the server
46     write(client_fd, "Server- Enter number 1: 26\n", 25);
47
48     // Receiving data from the server
49     read(client_fd, buffer, 1024);
50     printf("Server: %s\n", buffer);
51
52     // Sending data to the server
53     write(client_fd, "Server- Enter number 2: 4\n", 25);
54
55     // Receiving data from the server
56     read(client_fd, buffer, 1024);
57     printf("Server: %s\n", buffer);
58
59     // Sending data to the server
60     write(client_fd, "Server- 1. Addition\n", 20);
61
62     // Receiving data from the server
63     read(client_fd, buffer, 1024);
64     printf("Server: %s\n", buffer);
65
66     // Sending data to the server
67     write(client_fd, "Server- 2. Subtraction\n", 25);
68
69     // Receiving data from the server
70     read(client_fd, buffer, 1024);
71     printf("Server: %s\n", buffer);
72
73     // Sending data to the server
74     write(client_fd, "Server- 3. Multiplication\n", 25);
75
76     // Receiving data from the server
77     read(client_fd, buffer, 1024);
78     printf("Server: %s\n", buffer);
79
80     // Sending data to the server
81     write(client_fd, "Server- 4. Division\n", 20);
82
83     // Receiving data from the server
84     read(client_fd, buffer, 1024);
85     printf("Server: %s\n", buffer);
86
87     // Sending data to the server
88     write(client_fd, "Server- 5. Exit\n", 15);
89
90     // Receiving data from the server
91     read(client_fd, buffer, 1024);
92     printf("Server: %s\n", buffer);
93
94     // Sending data to the server
95     write(client_fd, "Server- 6\n", 10);
96
97     // Receiving data from the server
98     read(client_fd, buffer, 1024);
99     printf("Server: %s\n", buffer);
100
101     // Sending data to the server
102     write(client_fd, "Server- Enter number 1: 3\n", 25);
103
104     // Receiving data from the server
105     read(client_fd, buffer, 1024);
106     printf("Server: %s\n", buffer);
107
108     // Sending data to the server
109     write(client_fd, "Server- Enter number 2: 7\n", 25);
110
111     // Receiving data from the server
112     read(client_fd, buffer, 1024);
113     printf("Server: %s\n", buffer);
114
115     // Sending data to the server
116     write(client_fd, "Server- 1. Addition\n", 20);
117
118     // Receiving data from the server
119     read(client_fd, buffer, 1024);
120     printf("Server: %s\n", buffer);
121
122     // Sending data to the server
123     write(client_fd, "Server- 2. Subtraction\n", 25);
124
125     // Receiving data from the server
126     read(client_fd, buffer, 1024);
127     printf("Server: %s\n", buffer);
128
129     // Sending data to the server
130     write(client_fd, "Server- 3. Multiplication\n", 25);
131
132     // Receiving data from the server
133     read(client_fd, buffer, 1024);
134     printf("Server: %s\n", buffer);
135
136     // Sending data to the server
137     write(client_fd, "Server- 4. Division\n", 20);
138
139     // Receiving data from the server
140     read(client_fd, buffer, 1024);
141     printf("Server: %s\n", buffer);
142
143     // Sending data to the server
144     write(client_fd, "Server- 5. Exit\n", 15);
145
146     // Receiving data from the server
147     read(client_fd, buffer, 1024);
148     printf("Server: %s\n", buffer);
149
150     // Sending data to the server
151     write(client_fd, "Server- 6\n", 10);
152
153     // Receiving data from the server
154     read(client_fd, buffer, 1024);
155     printf("Server: %s\n", buffer);
156
157     // Sending data to the server
158     write(client_fd, "Server- Enter number 1: 3\n", 25);
159
160     // Receiving data from the server
161     read(client_fd, buffer, 1024);
162     printf("Server: %s\n", buffer);
163
164     // Sending data to the server
165     write(client_fd, "Server- Enter number 2: 7\n", 25);
166
167     // Receiving data from the server
168     read(client_fd, buffer, 1024);
169     printf("Server: %s\n", buffer);
170
171     // Sending data to the server
172     write(client_fd, "Server- 1. Addition\n", 20);
173
174     // Receiving data from the server
175     read(client_fd, buffer, 1024);
176     printf("Server: %s\n", buffer);
177
178     // Sending data to the server
179     write(client_fd, "Server- 2. Subtraction\n", 25);
180
181     // Receiving data from the server
182     read(client_fd, buffer, 1024);
183     printf("Server: %s\n", buffer);
184
185     // Sending data to the server
186     write(client_fd, "Server- 3. Multiplication\n", 25);
187
188     // Receiving data from the server
189     read(client_fd, buffer, 1024);
190     printf("Server: %s\n", buffer);
191
192     // Sending data to the server
193     write(client_fd, "Server- 4. Division\n", 20);
194
195     // Receiving data from the server
196     read(client_fd, buffer, 1024);
197     printf("Server: %s\n", buffer);
198
199     // Sending data to the server
200     write(client_fd, "Server- 5. Exit\n", 15);
201
202     // Receiving data from the server
203     read(client_fd, buffer, 1024);
204     printf("Server: %s\n", buffer);
205
206     // Sending data to the server
207     write(client_fd, "Server- 6\n", 10);
208
209     // Receiving data from the server
210     read(client_fd, buffer, 1024);
211     printf("Server: %s\n", buffer);
212
213     // Sending data to the server
214     write(client_fd, "Server- Enter number 1: 3\n", 25);
215
216     // Receiving data from the server
217     read(client_fd, buffer, 1024);
218     printf("Server: %s\n", buffer);
219
220     // Sending data to the server
221     write(client_fd, "Server- Enter number 2: 7\n", 25);
222
223     // Receiving data from the server
224     read(client_fd, buffer, 1024);
225     printf("Server: %s\n", buffer);
226
227     // Sending data to the server
228     write(client_fd, "Server- 1. Addition\n", 20);
229
230     // Receiving data from the server
231     read(client_fd, buffer, 1024);
232     printf("Server: %s\n", buffer);
233
234     // Sending data to the server
235     write(client_fd, "Server- 2. Subtraction\n", 25);
236
237     // Receiving data from the server
238     read(client_fd, buffer, 1024);
239     printf("Server: %s\n", buffer);
240
241     // Sending data to the server
242     write(client_fd, "Server- 3. Multiplication\n", 25);
243
244     // Receiving data from the server
245     read(client_fd, buffer, 1024);
246     printf("Server: %s\n", buffer);
247
248     // Sending data to the server
249     write(client_fd, "Server- 4. Division\n", 20);
250
251     // Receiving data from the server
252     read(client_fd, buffer, 1024);
253     printf("Server: %s\n", buffer);
254
255     // Sending data to the server
256     write(client_fd, "Server- 5. Exit\n", 15);
257
258     // Receiving data from the server
259     read(client_fd, buffer, 1024);
260     printf("Server: %s\n", buffer);
261
262     // Sending data to the server
263     write(client_fd, "Server- 6\n", 10);
264
265     // Receiving data from the server
266     read(client_fd, buffer, 1024);
267     printf("Server: %s\n", buffer);
268
269     // Sending data to the server
270     write(client_fd, "Server- Enter number 1: 3\n", 25);
271
272     // Receiving data from the server
273     read(client_fd, buffer, 1024);
274     printf("Server: %s\n", buffer);
275
276     // Sending data to the server
277     write(client_fd, "Server- Enter number 2: 7\n", 25);
278
279     // Receiving data from the server
280     read(client_fd, buffer, 1024);
281     printf("Server: %s\n", buffer);
282
283     // Sending data to the server
284     write(client_fd, "Server- 1. Addition\n", 20);
285
286     // Receiving data from the server
287     read(client_fd, buffer, 1024);
288     printf("Server: %s\n", buffer);
289
290     // Sending data to the server
291     write(client_fd, "Server- 2. Subtraction\n", 25);
292
293     // Receiving data from the server
294     read(client_fd, buffer, 1024);
295     printf("Server: %s\n", buffer);
296
297     // Sending data to the server
298     write(client_fd, "
```