

Assignment 2  
Of  
Network & Distributed System Lab (CS2051)  
Masters of Technology in Computer Science And Engineering

submitted to  
Dr Sujoy Saha  
Assistant Professor  
&  
Dr Suvrojit Das  
Associate Professor  
Dept. of CSE



National Institute of Technology, Durgapur

submitted by  
Arghya Bandyopadhyay  
RollNo. 20CS4103

19 June 2021

Write TCP Chat Program.

**Answer.**

```
1 //This is the Server side implementation of TCP Chat
2
3 #include<string.h>
4 #include<sys/socket.h>
5 #include<netinet/in.h>
6 #include<unistd.h>
7 #include<stdio.h>
8 #include <stdbool.h>
9
10
11 int TCP_ChatServer(int client_desc)
12 {
13     const int BUFFER_SIZE = 4096;
14     char msg[BUFFER_SIZE];
15     int msg_len = 0;
16
17     while((msg_len = read(client_desc, msg, BUFFER_SIZE)) != 0)
18     {
19         printf("%s %d %s", "[User", client_desc, "] Msg Received :");
20         fflush(stdout);
21
22         write(fileno(stdout), msg, msg_len);
23
24         if(msg[0] == 'b' && msg[1] == 'y' && msg[2] == 'e')
25             return 0;
26
27         printf("%s %d %s", "[User", client_desc, "] Enter Response :");
28
29         fflush(stdout);
30
31         msg_len = read(fileno(stdin), msg, BUFFER_SIZE);
32
33         write(client_desc, msg, msg_len);
34
35         if(msg[0] == 'b' && msg[1] == 'y' && msg[2] == 'e')
36             return 0;
37     }
38
39     return 0;
40 }
41
42
43 int main()
44 {
45     //Create Socket
46     int server_desc = socket(AF_INET, SOCK_STREAM, 0);
47
48     //Create and Fill Address Structure for this Server
49     struct sockaddr_in server_addr;
```

```

50 server_addr.sin_family      = AF_INET;      //Address Family (AF_INET, AF_INET6, AF_LOCAL, ...)
51     server_addr.sin_addr.s_addr = INADDR_ANY; //Internet Address (INADDR_ANY-> Accept connection at any IP Address)
52     server_addr.sin_port      = htons(9000); //Port Number (htons -> h.HOST t.TO n.NETWORK s.SHORT , Ensures proper byte ordering)
53
54 //Bind Socket Descriptor and Address Structure together
55 int result = bind(server_desc, (struct sockaddr*) &server_addr, sizeof(server_addr));
56
57 //Start Listioning (Tell kernel to accept connections directed towards this socket) (Puts socket into passive mode)
58 listen(server_desc,4);
59
60
61
62
63 //Server Loop
64 bool RunServer = true;
65 while(RunServer)
66 {
67     //Accept a Connection (Puts process in sleep mode if Connection Queue is Empty)
68     int client_desc;
69     client_desc = accept(server_desc,NULL,NULL);          //Listening Socket
70
71     //Create Child Process to handle connection
72     int pid = fork();
73
74     if(pid > 0)                                           //Parent Process
75     {
76         //Close Client Socket
77         close(client_desc);
78         continue;
79     }
80     else
81     if(pid == 0)                                           //Child Process
82     {
83         //Close Listening Socket
84         close(server_desc);
85
86         TCP_ChatServer(client_desc);
87
88         //Close Connection
89         close(client_desc);
90         break;                                           //Work Done! Exit Child Process.
91     }
92     else
93     {
94         printf("%s ", "fork() Error!!!");
95         break;
96     }
97 }
98
99
100 return 0;
101 }

```

```

1 //This is the client side implementation of TCP Chat
2
3 #include<arpa/inet.h>
4 #include<sys/socket.h>
5 #include<sys/wait.h>
6 #include<netinet/in.h>
7 #include <stdbool.h>
8 #include<string.h>
9 #include<time.h>
10 #include<stdio.h>
11 #include<unistd.h>
12 #include<stdlib.h>
13 #include<string.h>
14 #include<signal.h>
15 #include<errno.h>
16
17
18 int TCP_ChatClient(int server_desc)
19 {
20     const int BUFFER_SIZE = 4096;
21     char msg[BUFFER_SIZE];
22     int len = 0;
23
24     while(true)
25     {
26         printf("%s ", "Input:"); fflush(stdout);
27         len = read(fileno(stdin), msg, BUFFER_SIZE);
28
29         if(len == 0) //EOF
30             return 0;
31
32         len = write(server_desc, msg, len);
33
34         if(msg[0] == 'b' && msg[1] == 'y' && msg[2] == 'e')
35             return 0;
36
37         len = read(server_desc, msg, BUFFER_SIZE);
38
39         printf("%s ", "Response from server :"); fflush(stdout);
40         len = write(fileno(stdout), msg, len);
41
42         if(msg[0] == 'b' && msg[1] == 'y' && msg[2] == 'e')
43             return 0;
44     }
45     return 0;
46 }
47
48
49 int main()
50 {
51     int sock = socket(AF_INET, SOCK_STREAM, 0);
52
53     struct sockaddr_in server_addr;

```

```
54  server_addr.sin_family      = AF_INET;
55  server_addr.sin_addr.s_addr = inet_addr("127.0.0.1");
56  server_addr.sin_port       = htons(9000);
57
58  int result = connect(sock, (struct sockaddr*)&server_addr, sizeof(server_addr));
59
60  TCP_ChatClient(sock);
61
62
63  close(sock);
64  return 0;
65 }
```

```
arghya@Delton: /media/arghya/Development/Github...  
arghya@Delton: /media/arghya/Development/Github Repo/MTechAssignments/Network and Distributed System LaboratoryAssignments/Assignment2/CCode/tcp$ ./tcpserver  
[User 4 ] Msg Received :hello world  
[User 4 ] Enter Response :hello world too  
[User 4 ] Msg Received :bye  
□
```

(a) TCPServer

```
arghya@Delton: /media/arghya/Development/Github Repo/MTechAssignments/Network and Distributed System LaboratoryAssignments/Assignment2/CCode/tcp$ ./tcpclient  
Input: hello world  
Response from server : hello world too  
Input: bye  
arghya@Delton: /media/arghya/Development/Github Repo/MTechAssignments/Network and Distributed System LaboratoryAssignments/Assignment2/CCode/tcp$ □
```

(b) TCPClient

Figure 1: Output:TCP