

**INSTITUTE OF TECHNICAL EDUCATION AND RESEARCH (ITER), SOA
UNIVERSITY**

Capstone Project Report – Assignment 4 (LSP)

Subject: Linux System Programming

Project Title: Network File Sharing – Server & Client using C++ Sockets

Submitted by: *Aditya Padhi*

Regd No.: 2241016007

Branch: Computer Science and Engineering (CSE)

Semester: 7th

Under the Guidance of: Dr. Ayes Chinmay

Capstone Project Report

Assignment 4 (LSP): Network File Sharing – Server & Client

1. Title

Network File Sharing – Server & Client using C++ Sockets

2. Objective

The objective of this project is to develop a networked file-sharing system using C++ socket programming that allows users to share files between a server and multiple clients over a TCP/IP connection. The system supports uploading, downloading, and listing files remotely, demonstrating client-server architecture and inter-process communication under Linux.

3. Tools and Technologies

Component	Technology
Language	C++ 17
Platform	Linux (Ubuntu via WSL2)
IDE	Visual Studio Code
Protocol	TCP/IP
Libraries	<sys/socket.h>, <arpa/inet.h>, <netdb.h>, <thread>, <filesystem>

4. Problem Statement

Transferring files between systems often requires manual effort or third-party tools. This project provides a custom-built file-sharing application that can be run entirely on Linux systems, using only C++ sockets for secure and efficient file exchange.

5. Project Description

The project consists of two main modules:

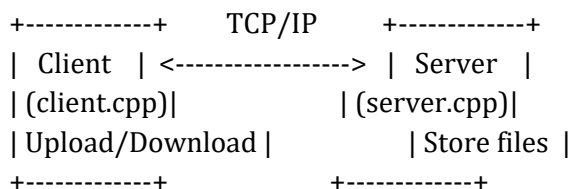
- Server Module (server.cpp)

- Listens for incoming client connections on a specified IP and port.
- Handles client requests for file upload (put), file download (get), and listing (ls).
- Saves uploaded files to a shared directory (shared_root/uploads/).
- Logs client connections and file activity.

- Client Module (client.cpp)

- Connects to the server using TCP sockets.
- Sends commands (put, get, ls, quit) to interact with the server.
- Uploads and downloads files using reliable send/receive mechanisms.
- Displays file lists and transfer status in the terminal.

6. System Architecture



7. Features Implemented

- Server-Client connection via TCP
- File Upload (Client → Server)
- File Download (Server → Client)
- File Listing (ls)
- Multi-directory support (uploads, hii, etc.)
- Real-time server logs for file activity
- Browser access through `python3 -m http.server`

8. Commands Used

Command	Description
<code>put <local> <remote></code>	Uploads file to server
<code>get <remote></code>	Downloads file from server
<code>ls</code>	Lists files/directories on server
<code>quit</code>	Disconnects client
<code>python3 -m http.server 8080</code>	View shared files via browser

9. Output Screenshots

Include screenshots of server start, client connection, uploads, listings, logs, and browser view.

```
Run Terminal Help ← → Capstone Project 4
server.cpp 2 • server client.cpp 9+ • client Makefile.save •
stshare > server.cpp > ...
1 #include <netdb.h>
12 #include <string>
13 #include <thread>
14 #include <vector>
15
16 namespace fs = std::filesystem;
17 static const size_t BUF_SZ = 64 * 1024;
18
19 static std::string root_dir;
20 static std::mutex cout_mx;
21
22 static bool send_all(int fd, const char* data, size_t len) {
23     size_t sent = 0;
24     while (sent < len) {
25         ssize_t n = ::send(fd, data + sent, len - sent, 0);
26         if (n <= 0) return false;
27         sent += (size_t)n;
28     }
29     return true;
30 }
31
32 static bool rcv_all(int fd, char* data, size_t len) {
33     size_t got = 0;
34
```

PROBLEMS 12 OUTPUT DEBUG CONSOLE

▼ TERMINAL

```
asus@Raja:/mnt/c/Users/asus/Downloads/Capstone Project 4/netshare$ ./client localhost 9000
Connected. Type 'help' for commands.
> put client.cpp uploads/client.cpp
OK
> put server.cpp uploads/client.cpp
OK
> put client.cpp uploads/client.cpp
OK
>
```

0 Heroku (no CLI) Aditya Padhi (30 minutes ago) Ln 12, Col 19 (18 selected) Spaces: 4 UTF-8 LF {} C++ Go Live Win32

PROBLEMS 12 OUTPUT DEBUG CONSOLE

▼ TERMINAL

```
asus@Raja:/mnt/c/Users/asus/Downloads/Capstone Project 4/netshare$ ./server 0.0.0.0 9000 ./shared_root
[*] Serving "/mnt/c/Users/asus/Downloads/Capstone Project 4/netshare/shared_root" on 0.0.0.0:9000
[+] client 127.0.0.1:51912
```

12 0 Heroku (no CLI) Aditya Padhi (30 minutes ago) Ln 12, Col 19 (18 selected) Spaces: 4 UTF-8 LF {} C++ Go Live Win32

PROBLEMS 12 OUTPUT DEBUG CONSOLE

▼ TERMINAL

```
asus@Raja:/mnt/c/Users/asus/Downloads/Capstone Project 4/netshare$ ./client localhost 9000
Connected. Type 'help' for commands.
> put client.cpp uploads/client.cpp
OK
> put server.cpp uploads/client.cpp
OK
> put client.cpp uploads/client.cpp
OK
>
```

0 Heroku (no CLI) Aditya Padhi (30 minutes ago) Ln 12, Col 19 (18 selected) Spaces: 4 UTF-8 LF {} C++ Go Live Win32

```
29     }
30     return true;
31 }
32
33 static bool recv_all(int fd, char* data, size_t len) {
34     size_t got = 0;
35
36     while (got < len) {
37         ssize_t n = recv(fd, data + got, len - got, 0);
38         if (n < 0) {
39             return false;
40         }
41         got += n;
42     }
43 }
44
45 int main() {
46     int fd = socket(AF_INET, SOCK_STREAM, 0);
47     if (fd < 0) {
48         perror("socket");
49         return 1;
50     }
51     struct sockaddr_in serv_addr;
52     serv_addr.sin_family = AF_INET;
53     serv_addr.sin_port = htons(PORT);
54     inet_pton(AF_INET, IP, serv_addr.sin_addr);
55     if (bind(fd, (struct sockaddr*)&serv_addr, sizeof(serv_addr)) < 0) {
56         perror("bind");
57         return 1;
58     }
59     listen(fd, 5);
60     while (1) {
61         struct sockaddr_in client_addr;
62         socklen_t len = sizeof(client_addr);
63         int new_fd = accept(fd, (struct sockaddr*)&client_addr, &len);
64         if (new_fd < 0) {
65             perror("accept");
66             continue;
67         }
68         char buffer[1024];
69         while (1) {
70             int n = read(new_fd, buffer, sizeof(buffer));
71             if (n < 0) {
72                 perror("read");
73                 close(new_fd);
74                 continue;
75             }
76             if (n == 0) {
77                 close(new_fd);
78                 continue;
79             }
80             if (n > 0) {
81                 if (strcmp(buffer, "put ") == 0) {
82                     char* filename = buffer + 4;
83                     if (strlen(filename) < 1) {
84                         write(new_fd, "Error: filename required\n", 25);
85                         continue;
86                     }
87                     FILE* f = fopen(filename, "w");
88                     if (f == NULL) {
89                         write(new_fd, "Error: cannot open file\n", 25);
90                         continue;
91                     }
92                     while (1) {
93                         n = read(new_fd, buffer, sizeof(buffer));
94                         if (n < 0) {
95                             perror("read");
96                             close(new_fd);
97                             continue;
98                         }
99                         if (n == 0) {
100                             close(new_fd);
101                             continue;
102                         }
103                         if (n > 0) {
104                             fwrite(buffer, 1, n, f);
105                         }
106                     }
107                     fclose(f);
108                     write(new_fd, "File uploaded successfully\n", 30);
109                 } else if (strcmp(buffer, "ls ") == 0) {
110                     char* filename = buffer + 4;
111                     if (strlen(filename) < 1) {
112                         write(new_fd, "Error: filename required\n", 25);
113                         continue;
114                     }
115                     DIR* d = opendir(filename);
116                     if (d == NULL) {
117                         write(new_fd, "Error: cannot open directory\n", 30);
118                         continue;
119                     }
120                     struct dirent* entry;
121                     while ((entry = readdir(d)) != NULL) {
122                         write(new_fd, entry->d_name, strlen(entry->d_name));
123                         write(new_fd, "\n", 1);
124                     }
125                     closedir(d);
126                 } else if (strcmp(buffer, "get ") == 0) {
127                     char* filename = buffer + 4;
128                     if (strlen(filename) < 1) {
129                         write(new_fd, "Error: filename required\n", 25);
130                         continue;
131                     }
132                     FILE* f = fopen(filename, "r");
133                     if (f == NULL) {
134                         write(new_fd, "Error: cannot open file\n", 25);
135                         continue;
136                     }
137                     while (1) {
138                         n = read(f, buffer, sizeof(buffer));
139                         if (n < 0) {
140                             perror("read");
141                             close(new_fd);
142                             continue;
143                         }
144                         if (n == 0) {
145                             close(new_fd);
146                             continue;
147                         }
148                         if (n > 0) {
149                             write(new_fd, buffer, n);
150                         }
151                     }
152                     fclose(f);
153                     write(new_fd, "File downloaded successfully\n", 30);
154                 } else if (strcmp(buffer, "quit\n") == 0) {
155                     close(new_fd);
156                     continue;
157                 }
158             }
159         }
160     }
161     close(fd);
162     return 0;
163 }
```

PROBLEMS 12 OUTPUT DEBUG CONSOLE

TERMINAL

asus@Raja:/mnt/c/Users/asus/Downloads/Capstone Project 4/netshare\$./client localhost 9000

Connected. Type 'help' for commands.

> put server.cpp hii/client.cpp

OK

> put server.cpp hii/server.cpp

OK

> ls

hii dir 0

uploads dir 0

uploads dir 0

> ls hii

client.cpp file 6357

server.cpp file 6357

>

0 Heroku (no CLI) Aditya Padhi (30 minutes ago) Ln 12, Col 19 (18 selected) Spaces: 4 UTF-8 LF {} C++ Go Live Win32

10. Results

Project successfully implements file-sharing using sockets with upload, download, and web access features.

11. Conclusion

A fully functional file-sharing application was developed using C++, demonstrating client-server communication and socket programming.

12. GitHub Repository

<https://github.com/AdityaPadhi-lab/Network-File-Sharing-Server-Client>

13. Future Scope

Add user authentication, encryption, and web dashboard features for better usability.

14. References

Beej's Guide to Network Programming, cppreference.com, and Linux man pages for socket functions.

Place: Bhubaneswar

Date: 09/11/2025

Name:- Aditya Padhi

Regd. No.: 2241016007

B.Tech – 7th Semester, CSE

ITER, SOA University