KARNATAKA LAW SOCIETY’S

GOGTE INSTITUTE OF TECHNOLOGY

UDYAMBAG, BELAGAVI-590008

(An Autonomous Institution under Visvesvaraya Technological University, Belagavi)

**(APPROVED BY AICTE, NEW DELHI)**

Department of Computer Science Engineering



*Course Activity Report*

**TITLE: Implementation of simple file server using sockets.**

*Submitted in partial fulfillment of the academic requirement of*

***VII Semester B.E.***

***In***

***Network Programming Laboratory***

*Submitted by*

|  |  |
| --- | --- |
| **NIKITA CHAVAN** | **2GI18CS081** |
| **MRIDULA PATIL** | **2GI18CS074** |
| **AKSHATA CHAVAN** | **2GI18CS184** |

**GUIDE**

Prof. Naitik Suryavanshi

Department of Computer Science

**2021 - 2022**

# KLS Gogte Institute of Technology, Belagavi

Department of Computer Science and Engineering

**COURSE PROJECT REPORT (Academic Year 2021 - 2022)**

**SUBJECT:** Network Programming Laboratory **SUBJECT CODE:** 18CSL77

**TOPIC:** Implementation of simple file server using sockets. **Date:** 25th December 2021

**Team Members Details:**

|  |  |  |
| --- | --- | --- |
| **S.No.** | **USN** | **Student Name** |
| **1** | **2GI18CS081** | **NIKITA CHAVAN** |
| **2** | **2GI18CS074** | **MRIDULA PATIL** |
| **3** | **2GI18CS184** | **AKSHATA CHAVAN** |

**Marks allocation:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Batch No. : | | | | | |
| 1. | Project Title: **Implementation of simple file server using sockets** | Marks Range | **USN** | | | |
| **2GI18CS081** | **2GI18CS074** | **2GI18CS184** |
| 2. | Problem statement (PO2) | 0-1 |  |  |  |
| 3. | Objectives of Defined Problem statement (PO1,PO2) | 0-2 |  |  |  |
| 4. | Design / Algorithm/Flowchart/Methodology (PO3) | 0-3 |  |  |  |
| 5. | Implementation details/Function/Procedures/Classes and Objects (Language/Tools) (PO1,PO3,PO4,PO5) | 0-4 |  |  |  |
| 6. | Working model of the final solution (PO3,PO12) | 0-5 |  |  |  |
| 7. | Report and Oral presentation skill (PO9,PO10) | 0-5 |  |  |  |
|  | Total | 20 |  |  |  |

**\* 20 marks is converted to 10 marks for CGPA calculation.**

**Signature of Staff in Charge**

**Problem Statement:**

Implement simple file server using sockets. The file server should be able to take the request from any client and return the requested file to client or return error message, status to client. Consider all the possible inputs for the file server. Implement using programming. Compare this result with FTP by using suitable tools.

**Introduction:**

TCP refers to the Transmission Control Protocol, which is a highly efficient and reliable protocol designed for end-to-end data transmission over an unreliable network. A TCP connection uses a three-way handshake to connect the client and the server. It is a process that requires both the client and the server to exchange synchronization (SYN) and acknowledge (ACK) packets before the data transfer takes place. Some important features of TCP:

• It’s a connection-oriented protocol.

• It provides error-checking and recovery mechanisms.

• It helps in end-to-end communication.

**Theory:**

Project structure

The project is divided into two files:

1. client.c

2. server.c

The client.c file contains the code for the client-side, which read the text file and sends it to the server and the server.c file receives the data from the client and saves it in a text file.

• Client

The client performs the following functions.

1. Start the program

2. Declare the variables and structures required.

3. A socket is created and the connect function is executed.

4. The file is opened.

5. The data from the file is read and sent to the server.

6. The socket is closed.

7. The program is stopped.

• Server

The server performs the following functions.

1. Start the program.

2. Declare the variables and structures required.

3. The socket is created using the socket function.

4. The socket is bound to the specific port.

5. Start listening for the connections.

6. Accept the connection from the client.

7. Creates a child process to handle request client among multiple clients.

8. Close server socket descriptor

9. Create a new file.

10. Receives the data from the client.

11. Write the data into the file.

12. The program is stopped.

**Source Code:**

server.c

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <arpa/inet.h>

#define SIZE 1024

void write\_file(int sockfd,char \*outputFile){

int n;

FILE \*fp;

char \*filename = outputFile;

char buffer[SIZE];

fp = fopen(filename, "a");

printf("\n Data sent to created output file is: ");

while (1) {

n = recv(sockfd, buffer, SIZE, 0);

printf("%s",buffer);

if (n <= 0){

break;

}

fprintf(fp, "%s", buffer);

bzero(buffer, SIZE);

}

fclose(fp);

return;

}

int main(int argc, char \*\*argv){

char \*ip = "127.0.0.1";

int port = 8080;

int e;

int listenfd, connfd, n;

pid\_t childpid;

socklen\_t clilen;

int sockfd, new\_sock;

struct sockaddr\_in server\_addr, new\_addr;

socklen\_t addr\_size;

char buffer[SIZE];

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

if(sockfd < 0) {

perror("Error in socket");

exit(1);

}

printf("Server socket created successfully.\n");

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

server\_addr.sin\_port = port;

server\_addr.sin\_addr.s\_addr = inet\_addr(ip);

e = bind(sockfd, (struct sockaddr\*)&server\_addr, sizeof(server\_addr));

if(e < 0) {

perror("Error in bind");

exit(1);

}

printf("Binding successfull.\n");

if(listen(sockfd, 10) == 0){

printf("Listening....\n");

}else{

perror("Error in listening");

exit(1);

}

int k=0;

for(;;)

{

k++;

addr\_size = sizeof(new\_addr);

new\_sock = accept(sockfd, (struct sockaddr\*)&new\_addr, &addr\_size);

if ( (childpid = fork ()) == 0 ) {

printf ("\n\nChild created for dealing with client %d request",k);

//close listening socket

close (listenfd);

write\_file(new\_sock,argv[1]);

printf("\nData written in the file successfully.\n");

}

}

return 0;

}

client.c

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <string.h>

#include <arpa/inet.h>

#define SIZE 1024

void send\_file(FILE \*fp, int sockfd){

int n;

char data[SIZE] = {0};

while(fgets(data, SIZE, fp) != NULL) {

if (send(sockfd, data, sizeof(data), 0) == -1) {

perror("Error in sending file.");

exit(1);

}

bzero(data, SIZE);

}

}

int main(int argc, char\*\* argv){

char \*ip = "127.0.0.1";

int port = 8080;

int e;

int sockfd;

struct sockaddr\_in server\_addr;

FILE \*fp;

char \*filename = argv[1];

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

if(sockfd < 0) {

perror("Error in socket");

exit(1);

}

printf("Server socket created successfully.\n");

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

server\_addr.sin\_port = port;

server\_addr.sin\_addr.s\_addr = inet\_addr(ip);

e = connect(sockfd, (struct sockaddr\*)&server\_addr, sizeof(server\_addr));

if(e == -1) {

perror("Error in socket");

exit(1);

}

printf("Connected to Server.\n");

fp = fopen(filename, "r");

if (fp == NULL) {

perror("Error in reading file.");

exit(1);

}

send\_file(fp, sockfd);

printf("File data sent successfully.\n");

printf("Closing the connection.\n");

close(sockfd);

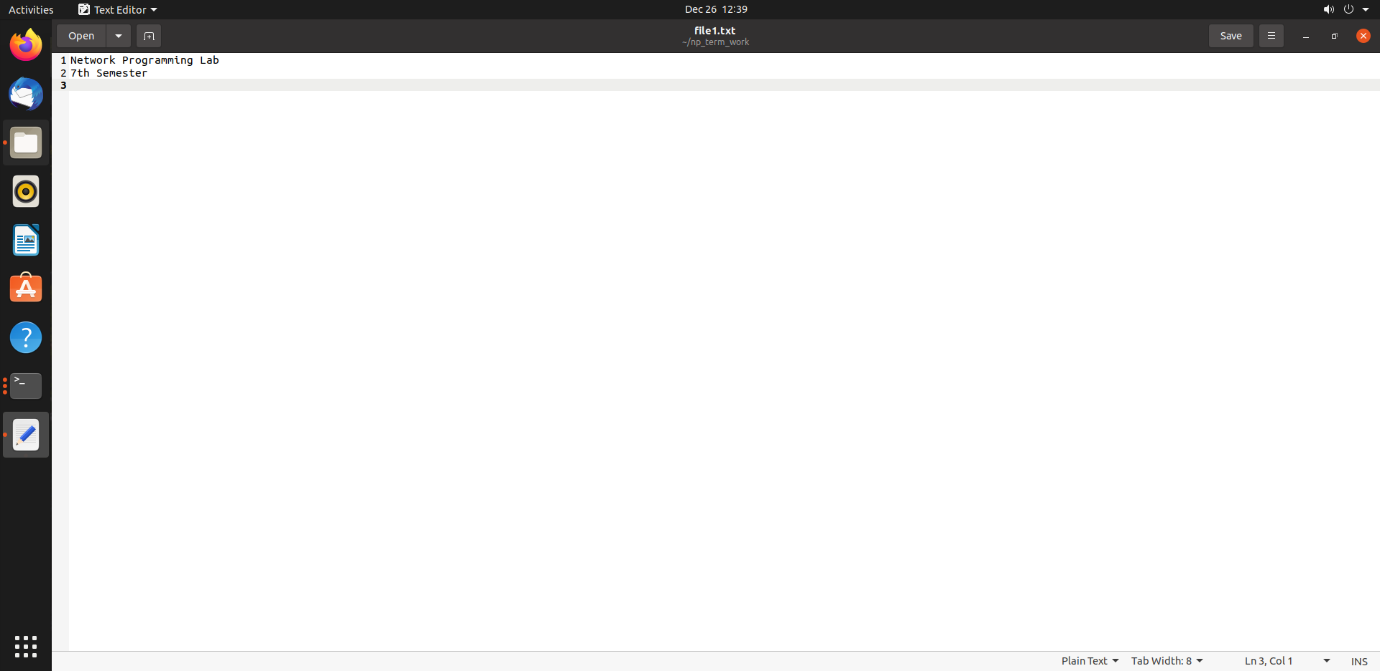
return 0;

}

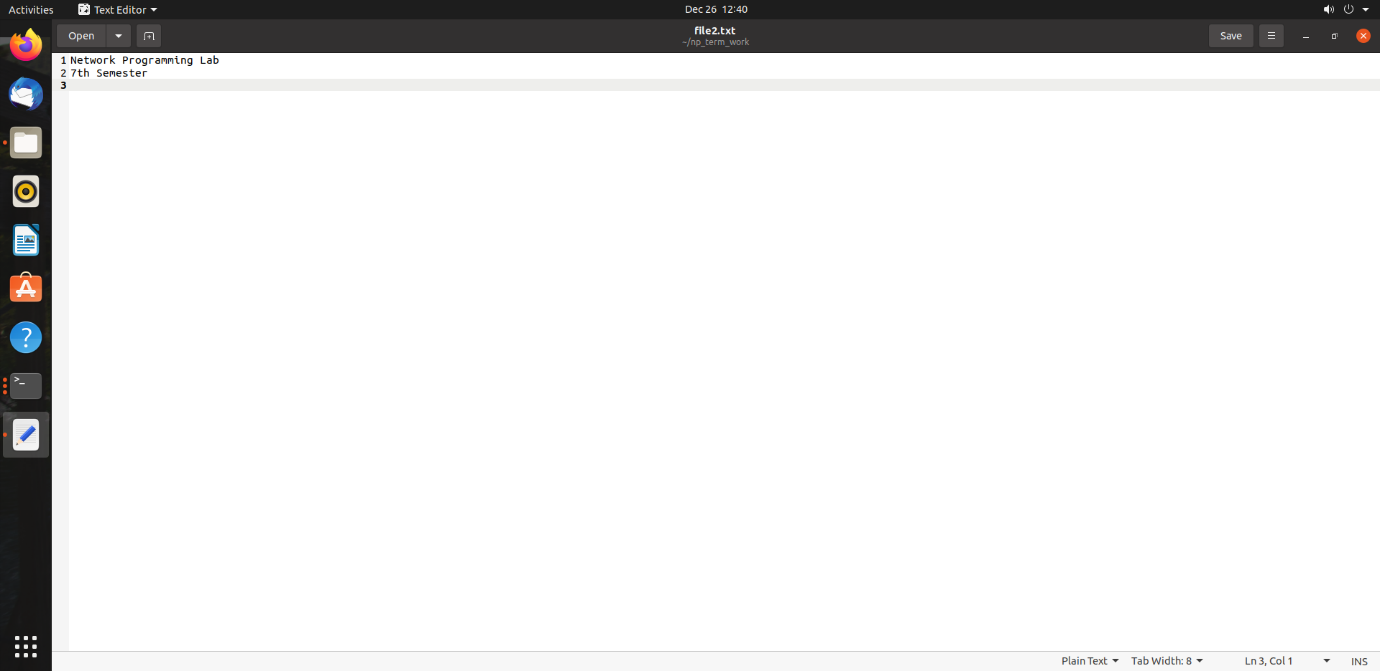
**Output:**

Server containing files as file1.txt and file2.txt

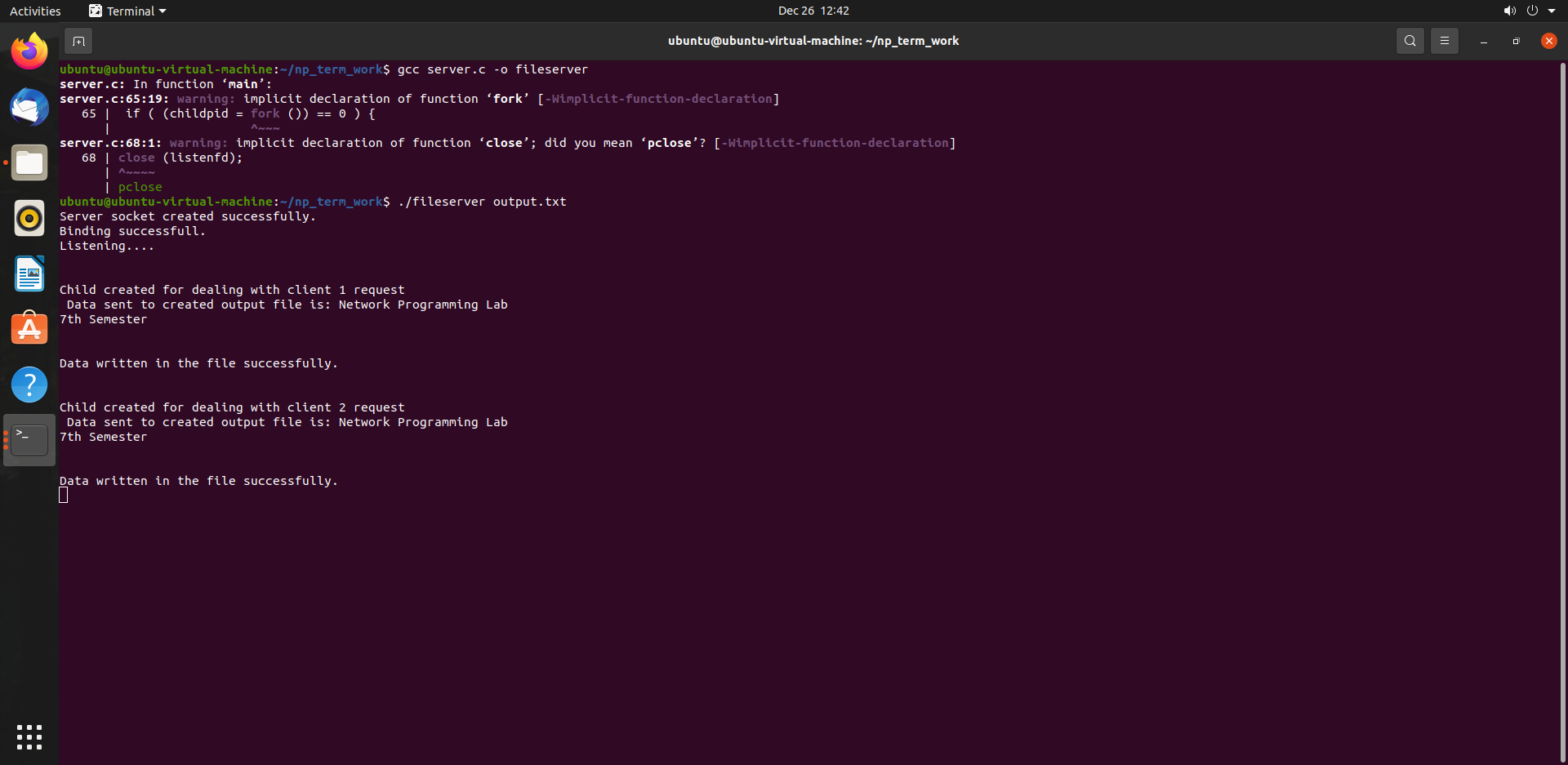
file1.txt

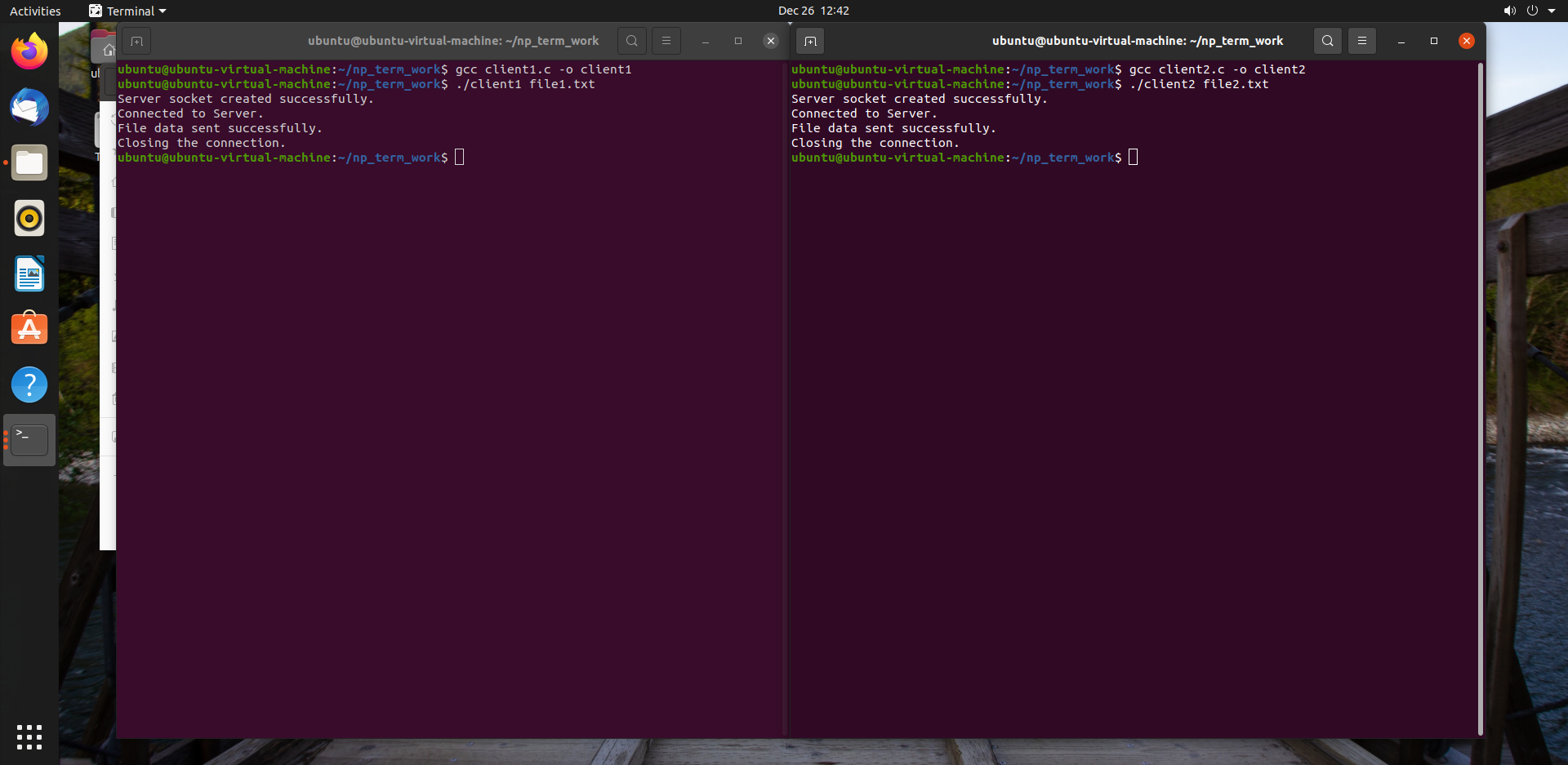


file2.txt

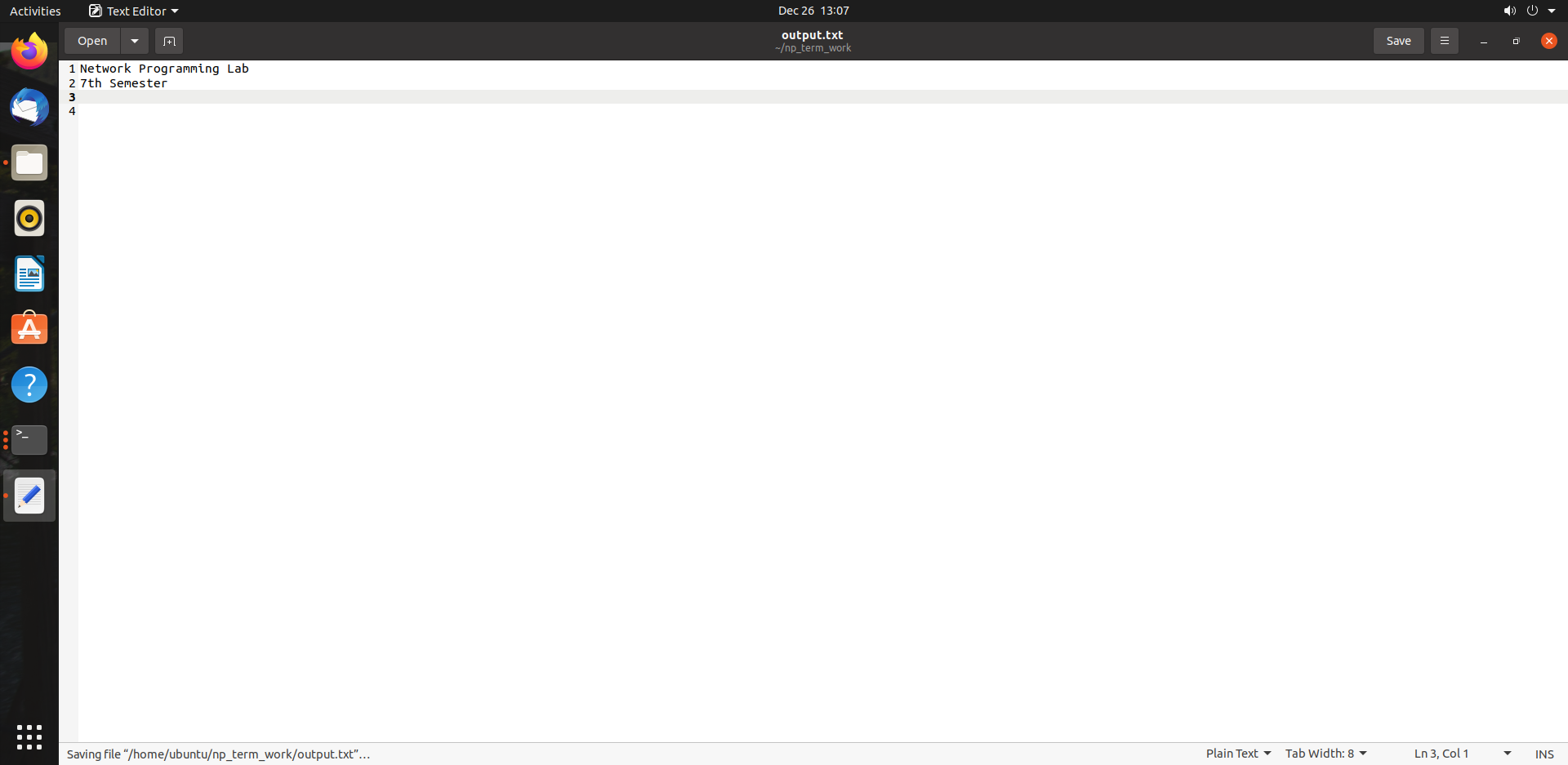


File server

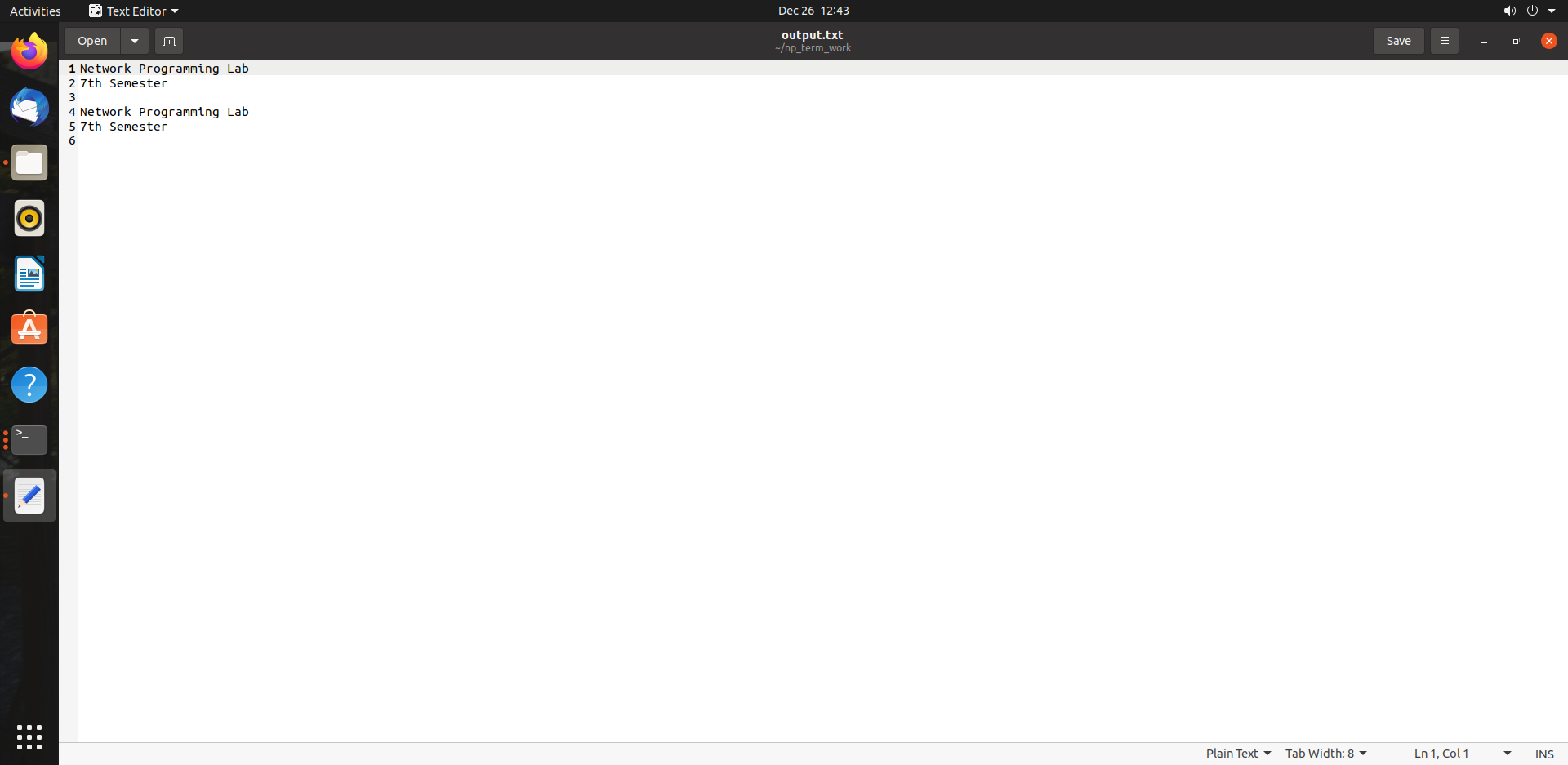


Two clients (client1 and client2 execution)  
  


Client1 appends its message into output.txt file from Server.



Client2 appends its message into output.txt file from Server.



**Conclusion:**

In this project, we implemented File server using socket programming to handle multiple client requests to access files from server. We understood how Inter process communication works with socket programming and steps involved in communication. We also understood Concurrent Server concept to handle multiple client requests.

**References:**

https://www.geeksforgeeks.org/tcp-server-client-implementation-in-c