**CLASS CODE FOR LOGGER.H :-**

#ifndef LOGGER\_H

#define LOGGER\_H

#include <iostream>

#include <fstream>

#include <string>

#include <mutex>

class Logger {

private:

std::ofstream log\_file;

std::mutex log\_mutex;

public:

Logger(const std::string &file\_name) {

log\_file.open(file\_name, std::ios::app);

if (!log\_file.is\_open()) {

std::cerr << "Failed to open log file: " << file\_name << std::endl;

}

}

~Logger() {

if (log\_file.is\_open()) {

log\_file.close();

}

}

void log(const std::string &message) {

std::lock\_guard<std::mutex> guard(log\_mutex);

std::cout << message << std::endl;

if (log\_file.is\_open()) {

log\_file << message << std::endl;

}

}

};

#endif

**SERVER CODE WITH LOGGER.H :-**

#include <iostream>

#include <fstream>

#include <cstring>

#include <sys/socket.h>

#include <netinet/in.h>

#include <unistd.h>

#include <thread>

#include "logger.h" // Include the logger

Logger logger("server\_log.txt"); // Create a logger instance for the server

void handle\_client(int client\_socket) {

char buffer[1024];

// Log client connection

logger.log("Client connected");

// Receive file info

ssize\_t bytes\_received = recv(client\_socket, buffer, sizeof(buffer) - 1, 0);

if (bytes\_received <= 0) {

logger.log("Error receiving file info");

close(client\_socket);

return;

}

buffer[bytes\_received] = '\0';

std::string file\_info(buffer);

size\_t comma\_pos = file\_info.find(',');

std::string filename = file\_info.substr(0, comma\_pos);

size\_t filesize = std::stoul(file\_info.substr(comma\_pos + 1));

// Open file to write received data

std::ofstream output\_file(filename, std::ios::binary);

if (!output\_file) {

logger.log("Error opening file");

close(client\_socket);

return;

}

// Receive file data

while (filesize > 0) {

ssize\_t bytes\_received = recv(client\_socket, buffer, sizeof(buffer), 0);

if (bytes\_received <= 0) {

logger.log("Error receiving file data");

break;

}

output\_file.write(buffer, bytes\_received);

filesize -= bytes\_received;

}

output\_file.close();

logger.log("Received file: " + filename);

close(client\_socket);

}

int main() {

int server\_fd = socket(AF\_INET, SOCK\_STREAM, 0);

if (server\_fd < 0) {

logger.log("Socket creation error");

return 1;

}

sockaddr\_in server\_addr {};

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

server\_addr.sin\_addr.s\_addr = INADDR\_ANY;

server\_addr.sin\_port = htons(65432);

if (bind(server\_fd, (struct sockaddr\*)&server\_addr, sizeof(server\_addr)) < 0) {

logger.log("Bind failed");

close(server\_fd);

return 1;

}

if (listen(server\_fd, 5) < 0) {

logger.log("Listen failed");

close(server\_fd);

return 1;

}

logger.log("Server listening on port 65432");

while (true) {

sockaddr\_in client\_addr {};

socklen\_t client\_addr\_len = sizeof(client\_addr);

int client\_socket = accept(server\_fd, (struct sockaddr\*)&client\_addr, &client\_addr\_len);

if (client\_socket < 0) {

logger.log("Accept failed");

continue;

}

std::thread(handle\_client, client\_socket).detach();

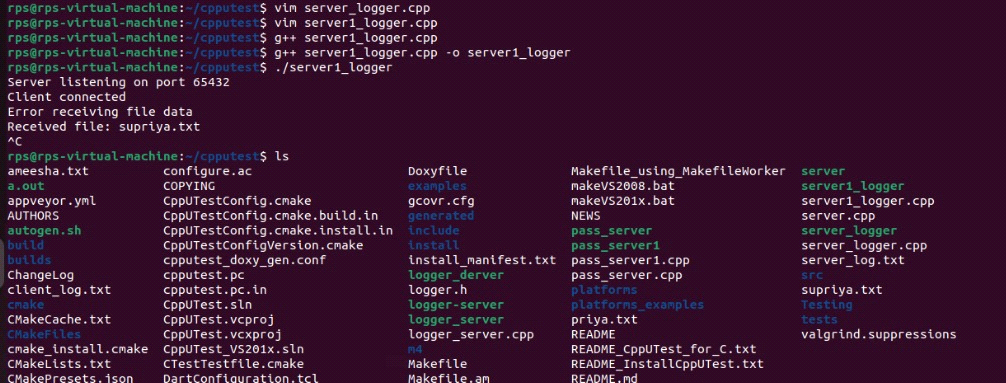
}

close(server\_fd);

return 0;

}

**OUTPUT FOR SERVER CODE WITH LOGGER.H :-**

****

**CLIENT CODE WITH LOGGER.H :-**

#include <iostream>

#include <fstream>

#include <cstring>

#include <sys/socket.h>

#include <netinet/in.h>

#include <unistd.h>

#include "logger.h" // Include the logger

Logger logger("client\_log.txt"); // Create a logger instance for the client

void send\_file(const std::string& filename, const std::string& server\_ip, uint16\_t server\_port) {

int client\_socket = socket(AF\_INET, SOCK\_STREAM, 0);

if (client\_socket < 0) {

logger.log("Socket creation error");

return;

}

sockaddr\_in server\_addr {};

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

server\_addr.sin\_port = htons(server\_port);

if (inet\_pton(AF\_INET, server\_ip.c\_str(), &server\_addr.sin\_addr) <= 0) {

logger.log("Invalid address/ Address not supported");

close(client\_socket);

return;

}

if (connect(client\_socket, (struct sockaddr\*)&server\_addr, sizeof(server\_addr)) < 0) {

logger.log("Connection failed");

close(client\_socket);

return;

}

std::ifstream input\_file(filename, std::ios::binary | std::ios::ate);

if (!input\_file) {

logger.log("Error opening file");

close(client\_socket);

return;

}

std::streamsize filesize = input\_file.tellg();

input\_file.seekg(0, std::ios::beg);

std::string file\_info = filename + "," + std::to\_string(filesize);

send(client\_socket, file\_info.c\_str(), file\_info.size(), 0);

char buffer[1024];

while (input\_file.read(buffer, sizeof(buffer)) || input\_file.gcount() > 0) {

send(client\_socket, buffer, input\_file.gcount(), 0);

}

input\_file.close();

logger.log("File sent successfully");

close(client\_socket);

}

int main() {

std::string filename;

std::cout << "Enter the filename to send: ";

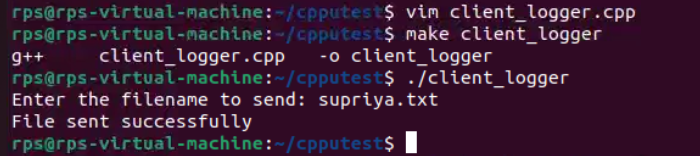
std::cin >> filename;

send\_file(filename, "127.0.0.1", 65432);

return 0;

}

**OUTPUT FOR CLIENT WITH LOGGER.H :-**

****