**Server cpp:**

#include <iostream>

#include <fstream>

#include <cstring>

#include <sys/socket.h>

#include <netinet/in.h>

#include <unistd.h>

#include "Logger.h"

// Constants

const int PORT = 8080;

const int BUFFER\_SIZE = 1024;

// Function to Handle File Reception

void receiveFile(int clientSocket, Logger& logger) {

std::ofstream file("received\_file", std::ios::binary);

if (!file) {

logger.log("Error creating file for receiving.");

return;

}

char buffer[BUFFER\_SIZE];

logger.log("Starting file reception.");

ssize\_t bytesReceived;

while ((bytesReceived = recv(clientSocket, buffer, BUFFER\_SIZE, 0)) > 0) {

if (std::strncmp(buffer, "FILE\_END", 8) == 0) {

break;

}

file.write(buffer, bytesReceived);

}

logger.log("File received successfully.");

}

// Function to Handle Messages

void receiveMessage(int clientSocket, Logger& logger) {

char buffer[BUFFER\_SIZE] = {0};

recv(clientSocket, buffer, BUFFER\_SIZE, 0);

logger.log("Message received: " + std::string(buffer));

}

int main() {

Logger logger("log.txt");

int serverFd, clientSocket;

struct sockaddr\_in serverAddr, clientAddr;

socklen\_t clientAddrLen = sizeof(clientAddr);

// Create socket

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

if (serverFd == 0) {

logger.log("Socket creation error.");

return -1;

}

logger.log("Socket created successfully.");

// Bind socket

serverAddr.sin\_family = AF\_INET;

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

serverAddr.sin\_port = htons(PORT);

if (bind(serverFd, (struct sockaddr\*)&serverAddr, sizeof(serverAddr)) < 0) {

logger.log("Bind failed.");

return -1;

}

logger.log("Bind successful.");

// Listen for connections

if (listen(serverFd, 3) < 0) {

logger.log("Listen failed.");

return -1;

}

logger.log("Server is listening for connections.");

// Accept connection

clientSocket = accept(serverFd, (struct sockaddr\*)&clientAddr, &clientAddrLen);

if (clientSocket < 0) {

logger.log("Accept failed.");

return -1;

}

logger.log("Connection accepted from client.");

// Handle client interaction

char buffer[BUFFER\_SIZE] = {0};

while (true) {

recv(clientSocket, buffer, BUFFER\_SIZE, 0);

if (std::strncmp(buffer, "FILE\_START", 10) == 0) {

receiveFile(clientSocket, logger);

} else if (std::strncmp(buffer, "exit", 4) == 0) {

logger.log("Client chose to exit.");

break;

} else {

receiveMessage(clientSocket, logger);

}

}

close(clientSocket);

logger.log("Client socket closed.");

close(serverFd);

logger.log("Server shut down.");

    return 0;

}