**TASK 12**

TCP Socket Programming in C

Develop a simple client-server application using TCP sockets in C:

The server listens on a specified port and accepts a client connection.

Once connected, the server sends a message "Welcome to the Server!" to the client.

The client receives the message and prints it to the console.

**CLIENT CODE**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <netinet/in.h>

#include <sys/socket.h>

#include <arpa/inet.h>

#define PORT 8080

int main() {

    int sock = 0;

    struct sockaddr\_in serv\_addr;

    char buffer[1024] = {0};

    if ((sock = socket(AF\_INET, SOCK\_STREAM, 0)) < 0) {

        printf("Socket creation error\n");

        return -1;

    }

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

    serv\_addr.sin\_port = htons(PORT);

    // Convert IP address to binary

    if (inet\_pton(AF\_INET, "127.0.0.1", &serv\_addr.sin\_addr) <= 0) {

        printf("Invalid address/ Address not supported\n");

        return -1;

    }

    // Connect to server

    if (connect(sock, (struct sockaddr \*)&serv\_addr, sizeof(serv\_addr)) < 0) {

        printf("Connection Failed\n");

        return -1;

    }

    // Read server message

    read(sock, buffer, sizeof(buffer));

    printf("Server says: %s\n", buffer);

    close(sock);

    return 0;

}

**FILES**

****

**SERVER CODE**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <netinet/in.h>

#include <sys/socket.h>

#define PORT 8080

int main() {

    int server\_fd, new\_socket;

    struct sockaddr\_in address;

    int opt = 1;

    int addrlen = sizeof(address);

    char \*message = "Welcome to the Server!";

    // Create socket file descriptor

    if ((server\_fd = socket(AF\_INET, SOCK\_STREAM, 0)) == 0) {

        perror("socket failed");

        exit(EXIT\_FAILURE);

    }

    // Attach socket to the port 8080

    setsockopt(server\_fd, SOL\_SOCKET, SO\_REUSEADDR | SO\_REUSEPORT, &opt, sizeof(opt));

    address.sin\_family = AF\_INET;

    address.sin\_addr.s\_addr = INADDR\_ANY; // listen on all interfaces

    address.sin\_port = htons(PORT);

    // Bind socket

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

        perror("bind failed");

        exit(EXIT\_FAILURE);

    }

    // Listen

    if (listen(server\_fd, 3) < 0) {

        perror("listen failed");

        exit(EXIT\_FAILURE);

    }

    printf("Server is listening on port %d...\n", PORT);

    // Accept a connection

    if ((new\_socket = accept(server\_fd, (struct sockaddr \*)&address, (socklen\_t\*)&addrlen)) < 0) {

        perror("accept failed");

        exit(EXIT\_FAILURE);

    }

    // Send message to client

    send(new\_socket, message, strlen(message), 0);

    printf("Message sent to client.\n");

    close(new\_socket);

    close(server\_fd);

    return 0;

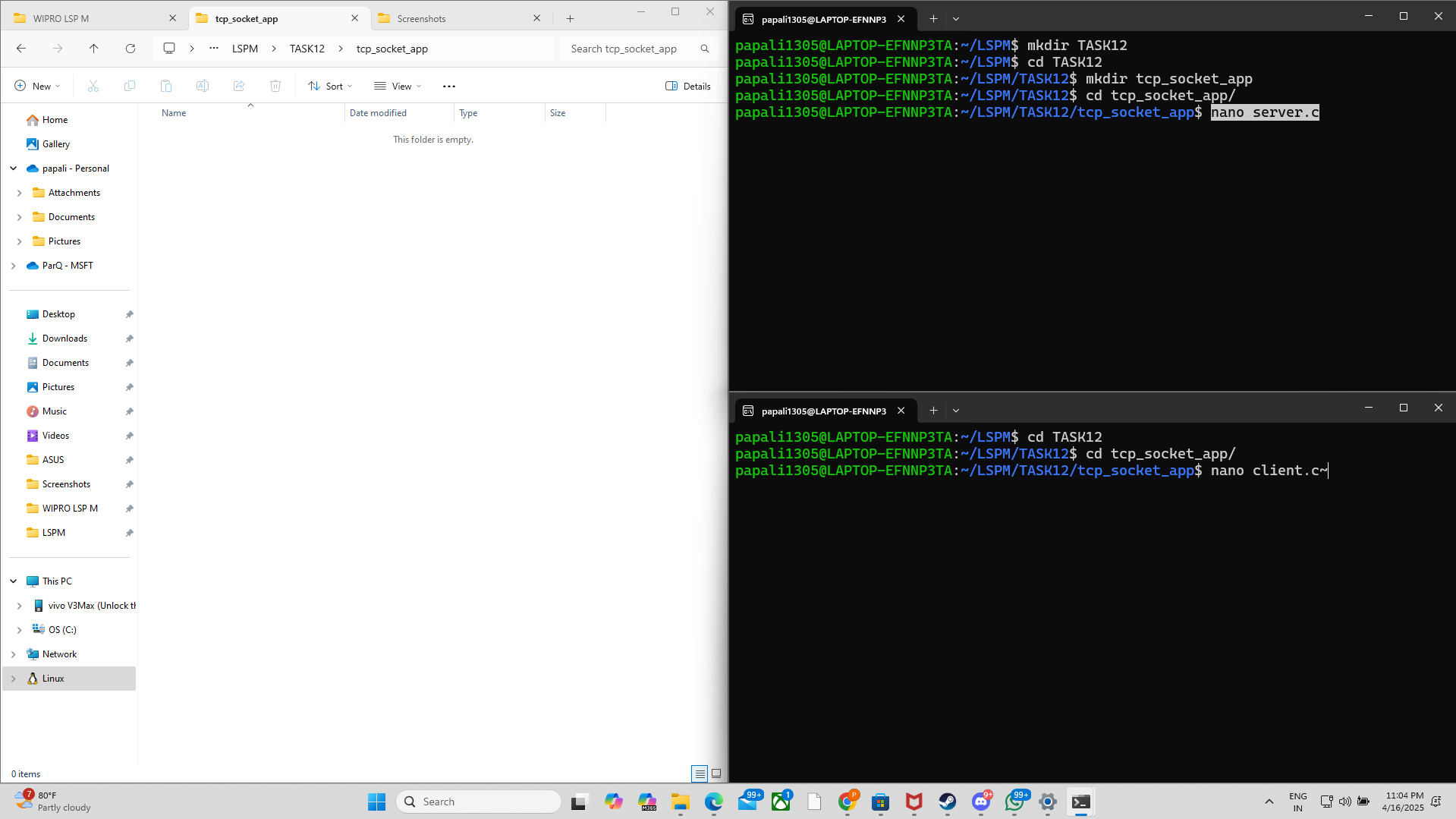
}

**FILES**

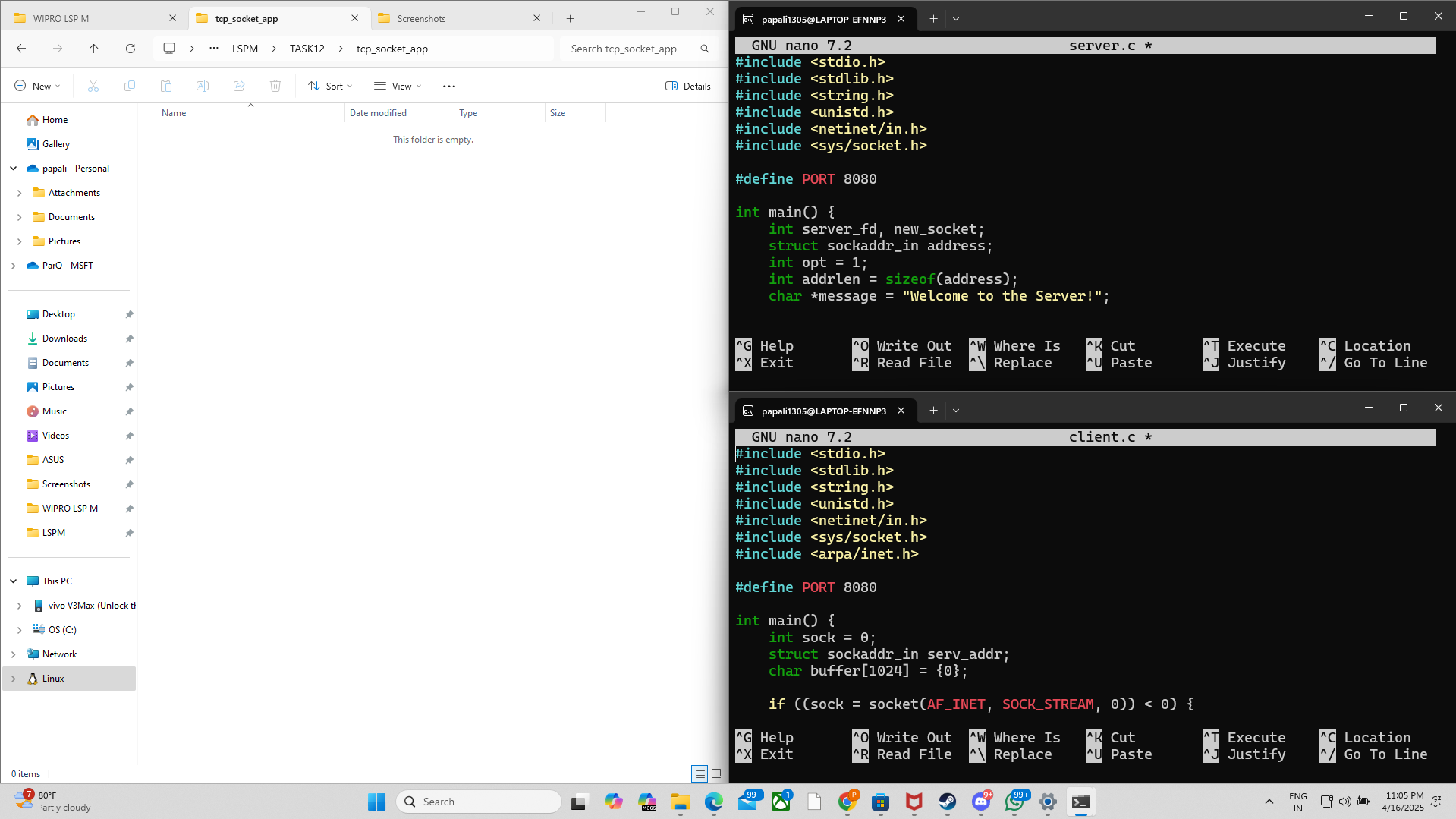
****

**OUTPUT & SCREENSHOTS**

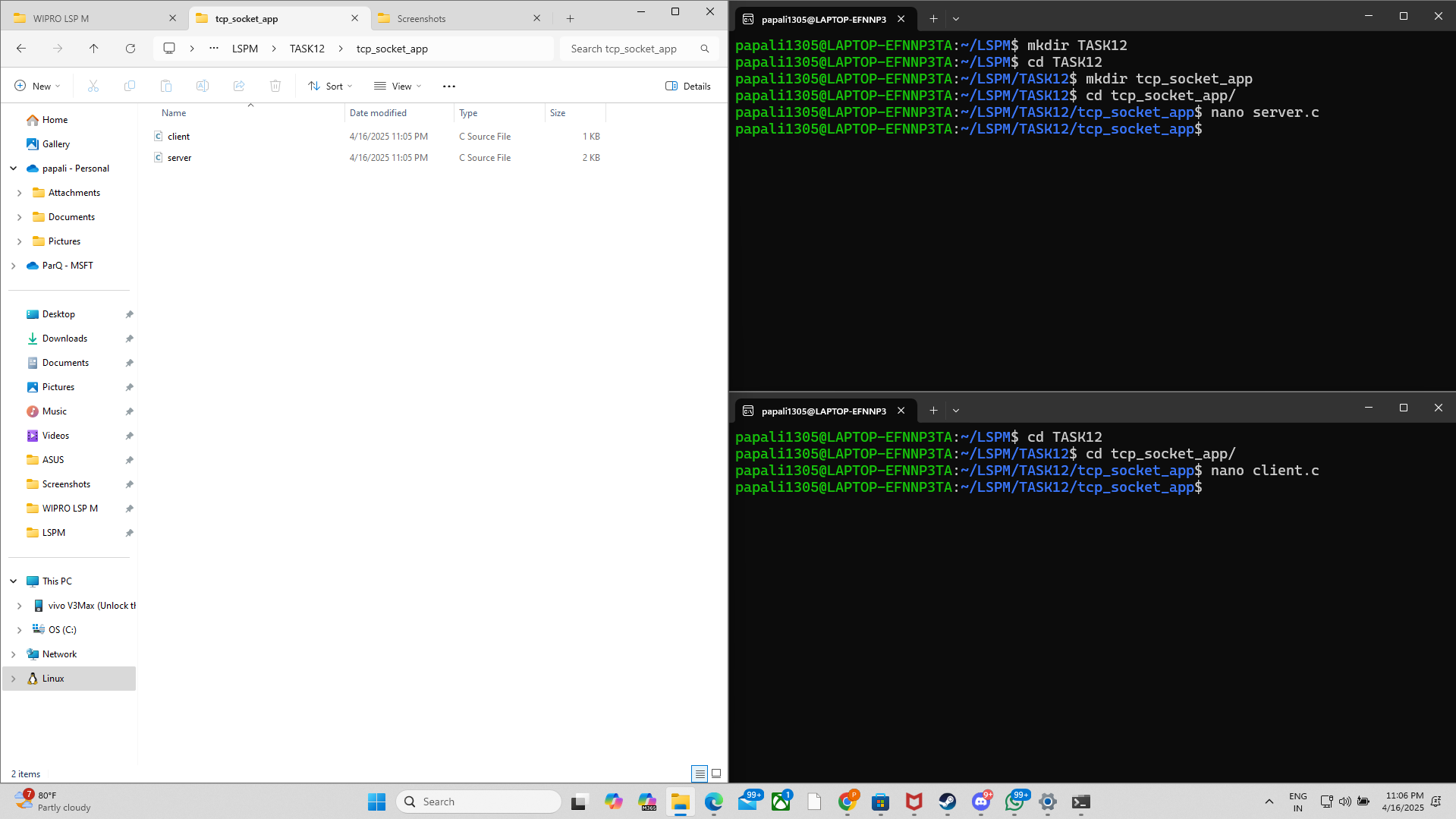
**01).**



02).



03).



04).

