**TASK 13**

UDP Socket Programming in C

Create a basic UDP program in C:

Implement both client and server functionalities within the program.

The client sends a message "Hello from Client" to the server.

The server receives the message, prints it, and responds with "Hello from Server."

The client receives the server's response and displays it.

**UDP CLIENT CODE**

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <string.h>

#include <arpa/inet.h>

#define PORT 8080

#define MAXLINE 1024

int main() {

    int sockfd;

    char buffer[MAXLINE];

    char \*hello = "Hello from Client";

    struct sockaddr\_in servaddr;

    // Create socket

    if ((sockfd = socket(AF\_INET, SOCK\_DGRAM, 0)) < 0) {

        perror("Socket creation failed");

        exit(EXIT\_FAILURE);

    }

    memset(&servaddr, 0, sizeof(servaddr));

    servaddr.sin\_family = AF\_INET;

    servaddr.sin\_port = htons(PORT);

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

    // Send message to server

    sendto(sockfd, (const char \*)hello, strlen(hello), MSG\_CONFIRM,

           (const struct sockaddr \*)&servaddr, sizeof(servaddr));

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

    int n, len;

    len = sizeof(servaddr);

    // Receive response

    n = recvfrom(sockfd, (char \*)buffer, MAXLINE, MSG\_WAITALL,

                 (struct sockaddr \*)&servaddr, &len);

    buffer[n] = '\0';

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

    close(sockfd);

    return 0;

}

**FILE**

****

**UDP SERVER CODE**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <arpa/inet.h>

#define PORT 8080

#define MAXLINE 1024

int main() {

    int sockfd;

    char buffer[MAXLINE];

    char \*hello = "Hello from Server";

    struct sockaddr\_in servaddr, cliaddr;

    // Create socket

    if ((sockfd = socket(AF\_INET, SOCK\_DGRAM, 0)) < 0) {

        perror("Socket creation failed");

        exit(EXIT\_FAILURE);

    }

    // Fill server info

    memset(&servaddr, 0, sizeof(servaddr));

    memset(&cliaddr, 0, sizeof(cliaddr));

    servaddr.sin\_family = AF\_INET;

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

    servaddr.sin\_port = htons(PORT);

    // Bind

    if (bind(sockfd, (const struct sockaddr \*)&servaddr, sizeof(servaddr)) < 0) {

        perror("Bind failed");

        exit(EXIT\_FAILURE);

    }

    int len, n;

    len = sizeof(cliaddr);

    // Receive message

    n = recvfrom(sockfd, (char \*)buffer, MAXLINE, MSG\_WAITALL,

                 (struct sockaddr \*)&cliaddr, &len);

    buffer[n] = '\0';

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

    // Send reply

    sendto(sockfd, (const char \*)hello, strlen(hello), MSG\_CONFIRM,

           (const struct sockaddr \*)&cliaddr, len);

    printf("Hello message sent.\n");

    close(sockfd);

    return 0;

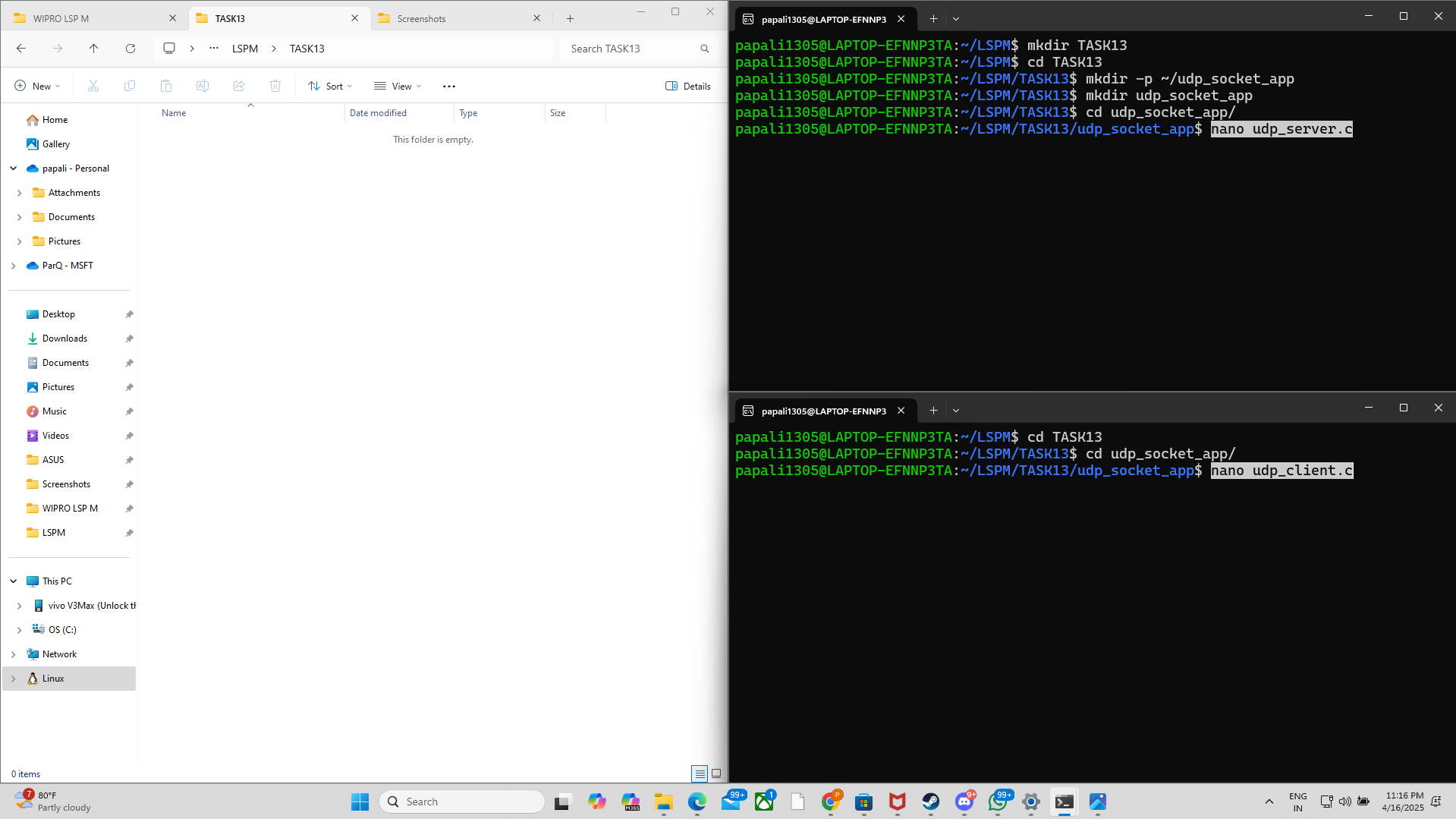
}

**FILE**

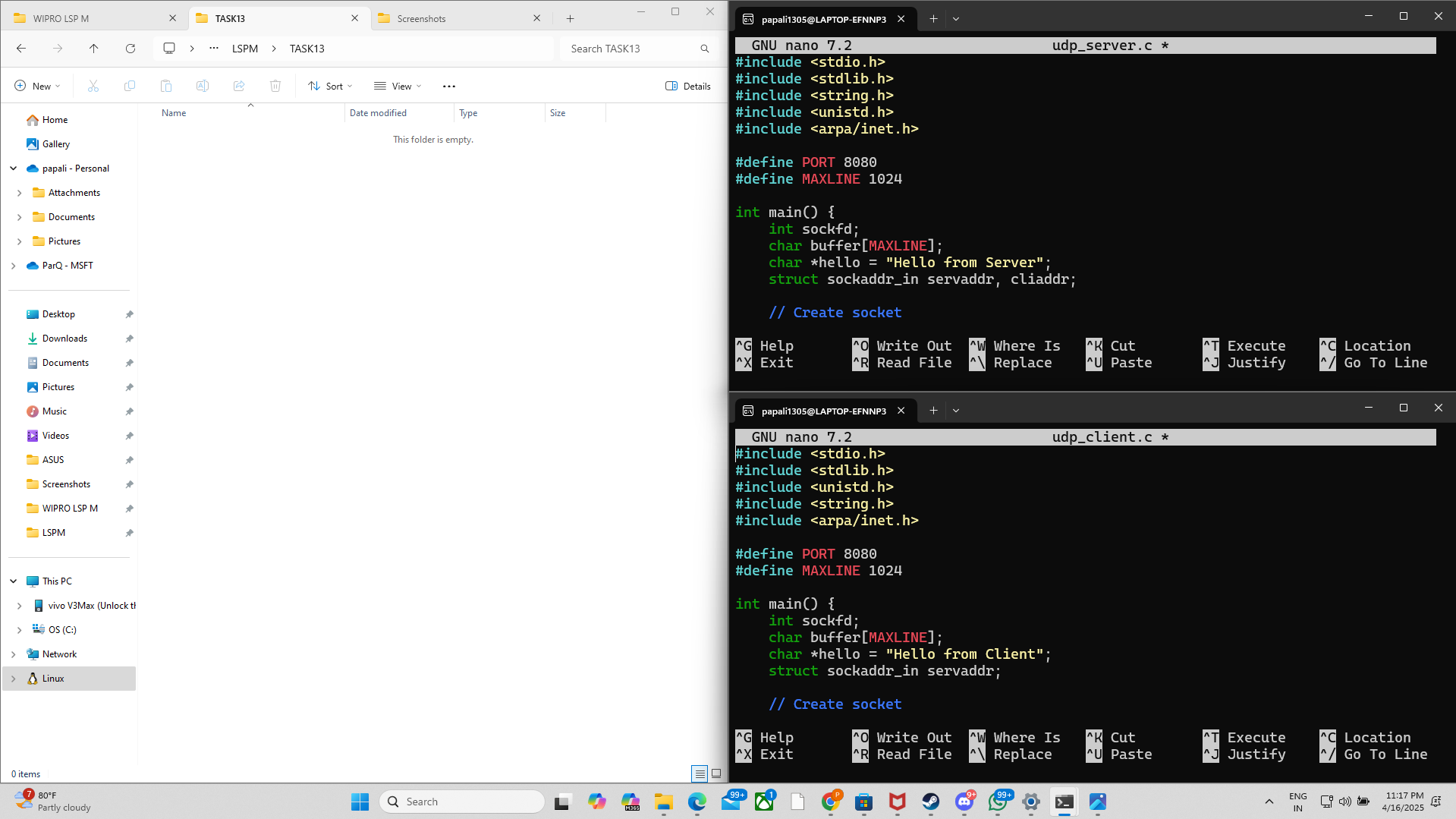
****

**OUTPUT & SCREENSHOT**

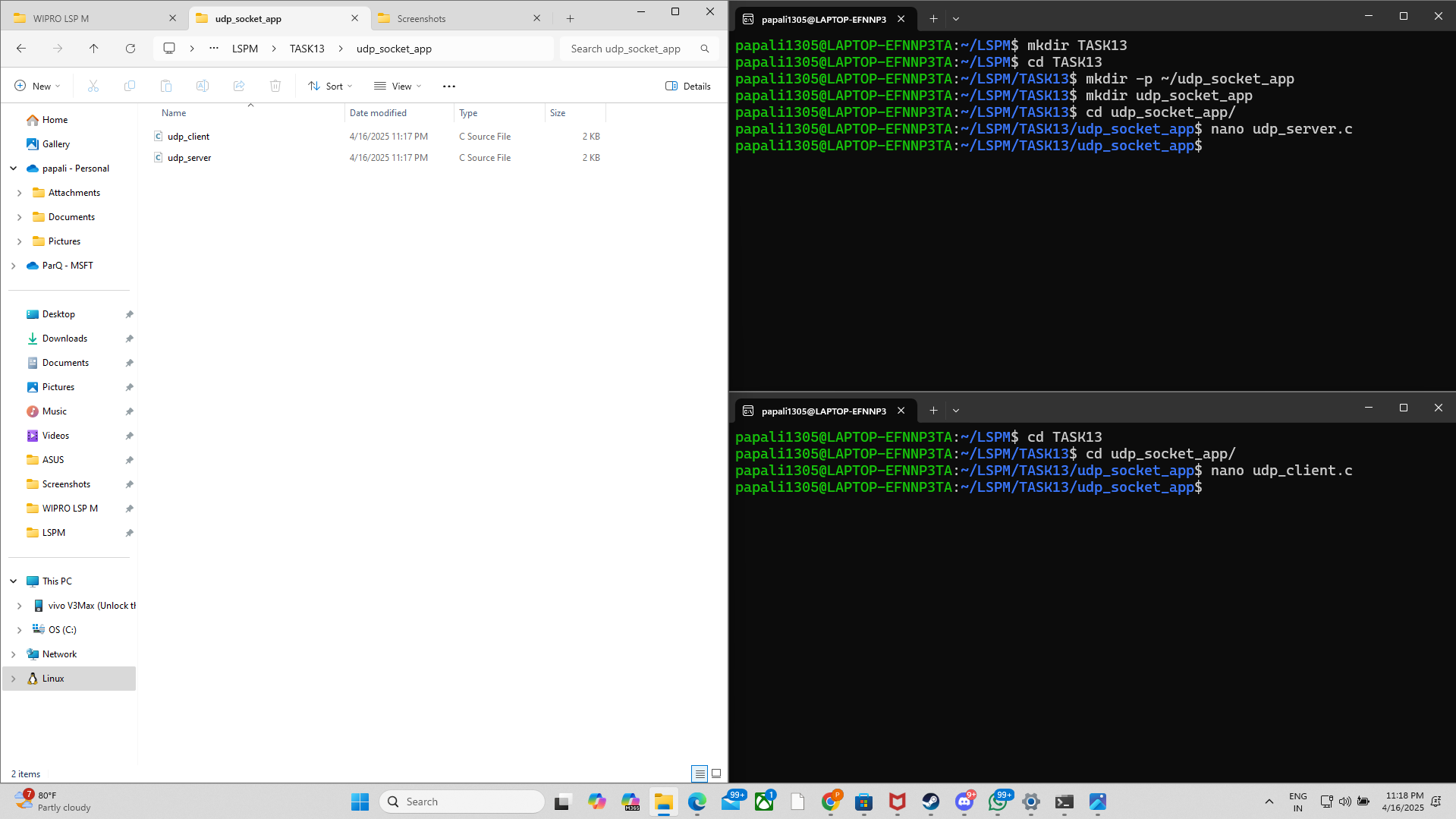
**01).**



**02).**



**03).**



**04).**

