**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”**

Інститут: **ІКНІ**

Кафедра: **ПЗ**



**ЗВІТ**

До лабораторної роботи **№ 11**

**Організація взаємодії між процесами**

**Лектор:**

ст. викл. каф. ПЗ

Грицай О.Д.

**Виконав:**

студ. групи ПЗ-25

Масюк О. Я.

**Прийняв:**

ст. викл. каф. ПЗ

Горечко О. М.

«\_\_\_» \_\_\_\_\_\_ 2021 р.

∑ = \_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_

Львів – 2021

***Тема:***  Організація взаємодії між процесами

***Мета:*** Ознайомитися зі способами міжпроцесної взаємодії. Ознайомитися з класичним прикладом взаємодії між процесами на прикладі задачі «виробник – споживач». Навчитися працювати із процесами з використанням способів міжпроцесної взаємодії, синхронізувати їхню роботу.

**Теоретичні відомості**

**Функції socket()**

Функція socket() створює сокет, який прив’язаний до певного мережевого провайдера для передачі даних.

SOCKET WSAAPI socket(int af, int type, int protocol);

Функція реалізована в бібліотеці WS2tcpip.h і приймає три параметри. Перший параметр - сімейство адресів. AF\_INET - інтернет протокол ipv4/

Другий параметр - тип сокета. SOCK\_STREAM - параметр для використання TCP.

Третій параметр - протокол.

**Функція connect()**

Функція connect() встановлює зв’язок із вказаним сокетом.

int WSAAPI connect(SOCKET s, const sockaddr \*name, int namelen);

Перший параметр - дескриптор, який вказує непід'єднаний сокет.

Другий параметр - вказівник на об’єкт структури sockaddr.

Третій параметр - розмір у байтах об’єкта структури socaddr.

**Функція bind()**

Функція bind() встановлює зв’язок з адресою і сокетом.

int bind(SOCKET s, const sockaddr \*addr,int namelen);

Перший параметр - дескриптор, який вказує незв'язаний сокет.

Другий параметр - вказівник на об’єкт структури sockaddr.

Третій параметр - розмір у байтах значення, на яке вказує параметр name.

**Функція accept()**

Функція accept() приймає запит вхідного сигналу на підключення сокета.

SOCKET WSAAPI accept(SOCKET s,sockaddr \*addr, int\* addrlen);

Перший параметр - дескриптор, який ідентифікує сокет, який переведено в стан прослуховування за допомогою функції прослуховування.

Другий параметр - опціональний покажчик на буфер, який отримує адресу об’єкта з’єднання, відомий комунікаційному шару.

Третій параметр - опціональний покажчик на ціле число, що містить довжину структури, на яку вказує параметр addr.

**Функція send()**

Функція send() надсилає дані на підключений сокет.

int WSAAPI send(SOCKET s, const char \*buf, int len, int flags);

Перший параметр - дескриптор, що ідентифікує підключений сокет.

Другий параметр - вказівник на буфер, що містить дані для передачі.

Третій параметр - довжина в байтах даних у буфері, на яку вказує параметр buf.

Четвертий параметр - набір прапорів, які визначають спосіб здійснення виклику.

**Функція recv()**

Функція recv() отримує дані з підключеного сокета або зв'язаного сокета без з'єднання.

int recv(SOCKET s, char \*buf, int len, int flags);

Перший параметр - дескриптор, який ідентифікує підключений сокет.

Другий параметр - вказівник на буфер для отримання вхідних даних.

Третій параметр - довжина в байтах буфера, на який вказує параметр buf.

Четвертий параметр - набір прапорів, що впливає на поведінку цієї функції.

**Структура sockaddr\_in**

Структура sockaddr\_in залежить від вибраного протоколу. Крім параметра сімейства Sin , вміст SOCKADDR виявляється у порядку мережевого байта.

Функції Winsock, які використовують SOCKADDR, не суворо обробляються як покажчики структуру SOCKADDR. Структура інтерпретується по-різному у різних сімейств адрес. Єдина вимога полягає в тому, що перший u - короткий - це сімейство адрес, а загальний розмір буфера пам'яті в байтах - namelen.

Структура сховища SOCKADDR також зберігає відомості про адресу сокета, і структура досить велика для зберігання відомостей про IPv4 або IPv6. Використання структури сховища SOCKADDR сприяє використанню сімейства протоколів і незалежної від версії протоколу та полегшує розробку. Рекомендується використовувати структуру зберігання SOCKADDR замість структури SOCKADDR. структура сховища SOCKADDR підтримується на Windows Server 2003 та пізніших версій.

struct sockaddr\_in {

short sin\_family;

u\_short sin\_port;

struct in\_addr sin\_addr;

char sin\_zero[8];

};

**Структура WSAData**

Структура WSAData містить інформацію про реалізацію Windows Sockets.

**Структура fd\_set**

Структура fd\_set використовується різними функціями Windows Sockets і постачальниками послуг, такими як функція select, для розміщення сокетів у «набір» для різних цілей, наприклад для перевірки даного сокета на читабельність за допомогою параметра readfds функції вибору.

typedef struct fd\_set {

u\_int fd\_count;

SOCKET fd\_array[FD\_SETSIZE];

} fd\_set, FD\_SET, \*PFD\_SET, \*LPFD\_SET;

**Функція getnameinfo()**

Функція getnameinfo() забезпечує незалежне від протоколу розділення імен від адреси до імені хоста ANSI і від номера порту до імені служби ANSI.

INT WSAAPI getnameinfo(

[in] const SOCKADDR \*pSockaddr,

[in] socklen\_t SockaddrLength,

[out] PCHAR pNodeBuffer,

[in] DWORD NodeBufferSize,

[out] PCHAR pServiceBuffer,

[in] DWORD ServiceBufferSize,

[in] INT Flags

);

**Функція closesocket()**

Функція closesocket() закриває існуючий сокет.

int closesocket( [in] SOCKET s);

Де s - дескриптор, що ідентифікує сокет, який потрібно закрити.

**Завдання**

1. Реалізувати алгоритм моделювання заданої задачі за допомогою окремих процесів згідно індивідуального завдання.

2. Реалізувати синхронізацію роботи процесів.

3. Забезпечити зберігання результатів виконання завдання.

4. Результати виконання роботи відобразити у звіті.

**Варіант - 7**

Реалізувати, використовуючи сокети, міжпроцесорну взаємодію , де один із процесів має бути сервером, який дозволяє процесам клієнтам підписатись/відписатись на один із серверів розсилки (щогодинний прогноз погоди, щохвилинний курс акцій, щоденний курс валют).

**Розподіл завдання**

Масюк Олег – реалізовував багатопоточність, для того, щоб кожен клієнт мав свою особисту функцію розсилки, за часом життя конкретного зв’язку клієнта зі сервером, рядки з 28 – 72, 173 -182, 194, 14-17, 24-25, файл серверу

Бірак Денис – реалізовував роботу сервера, реалізовував під’єднання з’єднання клієнтів до серверу і їхню синхронізацію за принципом Multiple connections, рядки з 19-26, 75-171, 188-201 файл серверу

Янович Марко – реалізовував створення клієнтів, реалізовував зовнішнє під’єднання конкретних клієнтів до сервера, 1-48, файл клієнту

Білик Тарас – реалізовував вибір типу розсилку між користувачем там сервером, реалізовував прийом типу розсилки на стороні клієнта для подальшого опрацювання, з 50 – 114, файл клієнту

Я реалізував вибір типу розсилку між користувачем там сервером, реалізовував прийом типу розсилки на стороні клієнта для подальшого опрацювання.

Реалізована мною частина коду виділена **жирним шрифтом**.

**Код програми**

server.cpp

#include <iostream>

#include <WS2tcpip.h>

#include <string>

#include <vector>

#include <processthreadsapi.h>

#include <Windows.h>

#include <time.h>

#include <stdlib.h>

#pragma comment (lib, "ws2\_32.lib")

using namespace std;

**struct MailingInfo {**

**SOCKET sock;**

**int type;**

**};**

**struct Info {**

**sockaddr\_in client;**

**int clientSize;**

**char host[NI\_MAXHOST]; // Client's remote name**

**char service[NI\_MAXSERV]; // Service (i.e. port) the client is connect on**

**HANDLE th;**

**DWORD th\_id;**

**};**

**DWORD WINAPI SendData(LPVOID lpParameter) {**

**MailingInfo\* p = (MailingInfo\*)lpParameter;**

**string isContinue;**

**switch (p->type) {**

**case 1: {**

**int hour = 0;**

**while (true) {**

**srand(time(NULL));**

**string msg = "Prediction on hour: " + to\_string(hour) + " is " + to\_string(rand() % 40) + " C^";**

**send(p->sock, msg.c\_str(), msg.size() + 1, 0);**

**hour++;**

**Sleep(5000);**

**}**

**}**

**case 2: {**

**int minutes = 0;**

**while (true) {**

**srand(time(NULL));**

**string msg = "Prediction on minute: " + to\_string(minutes) + " is " + to\_string(rand() % 30) + "." + to\_string(rand() % 30) + " eth.";**

**send(p->sock, msg.c\_str(), msg.size() + 1, 0);**

**minutes++;**

**Sleep(5000);**

**}**

**}**

**case 3: {**

**int day = 0;**

**while (true) {**

**srand(time(NULL));**

**string msg = "Prediction on day: " + to\_string(day) + " is " + to\_string(rand() % 30) + "." + to\_string(rand() % 30) + " usd.";**

**send(p->sock, msg.c\_str(), msg.size() + 1, 0);**

**day++;**

**Sleep(5000);**

**}**

**}**

**}**

**return 0;**

**}**

int SocketClientsCounter = 0;

int main()

{

//Init winsock

WSADATA wsData;

WORD ver = MAKEWORD(2, 2); //request winsock 2.2 version

int wsOk = WSAStartup(ver, &wsData);

if (wsOk != 0)

{

cerr << "Can't Initialize winsock! Quitting" << endl;

exit(3); //3 - exit code to error init winsock

}

// Create a socket

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

//AF\_INET is ised to communicate by ipv4

//TCP - SOCK\_STREAM

//UDP - SOCK\_DGRAM

if (listening == INVALID\_SOCKET)

{

cerr << "Can't create a socket! Quitting" << endl;

exit(4); //4 - exit code to error creating socket

}

// Bind the ip address and port to a socket

sockaddr\_in hint; //socket for IP protocols

hint.sin\_family = AF\_INET;

hint.sin\_port = htons(54000); //port number to be accessed by process

hint.sin\_addr.S\_un.S\_addr = INADDR\_ANY; //address to link socket with(INADDR\_ANY - all adresses from local host - 0.0.0.0)

bind(listening, (sockaddr\*)&hint, sizeof(hint)); //binds socket to port and adress

//places a socket in state of listening for incoming connection

listen(listening, SOMAXCONN);

fd\_set master; //set of incoming sockets clients

FD\_ZERO(&master); //clears set

FD\_SET(listening, &master); //sets first FD set element as server socket

vector<Info> clientsInfo; //info about each client socket

while (true) {

fd\_set copy = master;

int socketCount = select(0, &copy, nullptr, nullptr, nullptr);

for (int i = 0; i < socketCount; i++) {

SOCKET sock = copy.fd\_array[i];

if (sock == listening) {

clientsInfo.push\_back({});

clientsInfo[i].clientSize = sizeof(clientsInfo[i].client);

ZeroMemory(clientsInfo[i].host, NI\_MAXHOST);

ZeroMemory(clientsInfo[i].service, NI\_MAXSERV);

//accepts incoming connection

SOCKET client = accept(listening, (sockaddr\*)&clientsInfo[i].client, &clientsInfo[i].clientSize);

FD\_SET(client, &master);

if (getnameinfo((sockaddr\*)&clientsInfo[i].client, sizeof(clientsInfo[i].client), clientsInfo[i].host, NI\_MAXHOST, clientsInfo[i].service, NI\_MAXSERV, 0) == 0)

{

cout << clientsInfo[i].host << " connected on port " << clientsInfo[i].service << endl;

}

else

{

inet\_ntop(AF\_INET, &clientsInfo[i].client.sin\_addr, clientsInfo[i].host, NI\_MAXHOST);

cout << clientsInfo[i].host << " connected on port " << ntohs(clientsInfo[i].client.sin\_port) << endl;

}

}

else {

char buf[4096];

ZeroMemory(buf, 4096);

int bytesIn = recv(sock, buf, 4096, 0);

if (bytesIn == SOCKET\_ERROR) {

closesocket(sock);

FD\_CLR(sock, &master);

}

else {

if (!strcmp(buf, "1") || !strcmp(buf, "2") || !strcmp(buf, "3")) {

for (int i = 0; i < master.fd\_count; i++) {

SOCKET outSock = master.fd\_array[i];

if (outSock != listening && outSock == sock) {

string SubsciptionAccepted = "Sucessfully subscribed to mailing service...";

int oper = atoi(buf);

cout << to\_string(oper) + " option of mailing was chosed by client " + clientsInfo[i - 1].host + " on port: " + clientsInfo[i - 1].service << endl;

send(outSock, SubsciptionAccepted.c\_str(), SubsciptionAccepted.size() + 1, 0);

**MailingInfo\* mailing = new MailingInfo;**

**mailing->sock = outSock;**

**mailing->type = oper;**

**clientsInfo[i - 1].th = CreateThread(NULL, 0, &SendData, (void\*)mailing, 0, &clientsInfo[i - 1].th\_id);**

**if (clientsInfo[i - 1].th == NULL) {**

**cerr << "Error creating thread..." << endl;**

**exit(5);**

**}**

**break;**

**}**

**}**

**}**

else if (!strcmp(buf, "N")) {

for (int i = 0; i < master.fd\_count; i++) {

SOCKET outSock = master.fd\_array[i];

if (outSock != listening && outSock == sock) {

**TerminateThread(clientsInfo[i - 1].th, 4);**

string SubsciptionCanceled = "Your subscription was cancelled.";

send(outSock, SubsciptionCanceled.c\_str(), SubsciptionCanceled.size() + 1, 0);

break;

}

}

}

}

}

}

}

// Cleanup winsock

WSACleanup();

system("pause");

return 0;

}

client.cpp

#include <iostream>

#include <string>

#include <WS2tcpip.h>

#pragma comment(lib, "ws2\_32.lib")

using namespace std;

void main()

{

string ipAddress = "127.0.0.1"; // IP Address of the server

int port = 54000; // Listening port # on the server

// Initialize WinSock

WSAData data;

WORD ver = MAKEWORD(2, 2);

int wsResult = WSAStartup(ver, &data);

if (wsResult != 0)

{

cerr << "Can't start Winsock..." << endl;

exit(3);

}

// Create socket

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

if (sock == INVALID\_SOCKET)

{

cerr << "Can't create socket..." << endl;

WSACleanup();

exit(4);

}

// Fill in a hint structure

sockaddr\_in hint;

hint.sin\_family = AF\_INET;

hint.sin\_port = htons(port);

inet\_pton(AF\_INET, ipAddress.c\_str(), &hint.sin\_addr);

// Connect to server

int connResult = connect(sock, (sockaddr\*)&hint, sizeof(hint));

if (connResult == SOCKET\_ERROR)

{

cerr << "Can't connect to server..." << endl;

closesocket(sock);

WSACleanup();

exit(5);

}

cout << "Connecting...\n";

while (true) {

char buf[4096];

string userInput;

string chosedOper;

cout << "Subsciribe to:\n\n";

cout << "1. Hourly weather prediction\n2. Minutely stock exchange rate\n3. Daily exchange rate\n\n--> ";

cin >> chosedOper;

int sendResult = send(sock, chosedOper.c\_str(), chosedOper.size() + 1, 0);

if (sendResult != SOCKET\_ERROR)

{

ZeroMemory(buf, 4096);

int bytesReceived = recv(sock, buf, 4096, 0);

if (bytesReceived > 0)

{

cout << "[SERVER]> " << string(buf, 0, bytesReceived) << endl;

}

}

int k = 1;

string isContinue;

while (true) {

if (k % 5 == 0) {

cout << "[SERVER]> " << "Do you want to continue subsciption: Y | N: ";

cin >> isContinue;

int continueResult = send(sock, isContinue.c\_str(), isContinue.size() + 1, 0);

if (continueResult == SOCKET\_ERROR) exit(5);

ZeroMemory(buf, 4096);

int bytesReceived = recv(sock, buf, 4096, 0);

if (bytesReceived > 0)

{

cout << "[SERVER]> " << string(buf, 0, bytesReceived) << endl;

}

if (!isContinue.compare("N")) {

break;

}

}

else {

ZeroMemory(buf, 4096);

int bytesReceived = recv(sock, buf, 4096, 0);

if (bytesReceived > 0)

{

cout << "[SERVER]> " << string(buf, 0, bytesReceived) << endl;

}

}

k++;

}

}

// Gracefully close down everything

closesocket(sock);

WSACleanup();

}

**Виконання програми**

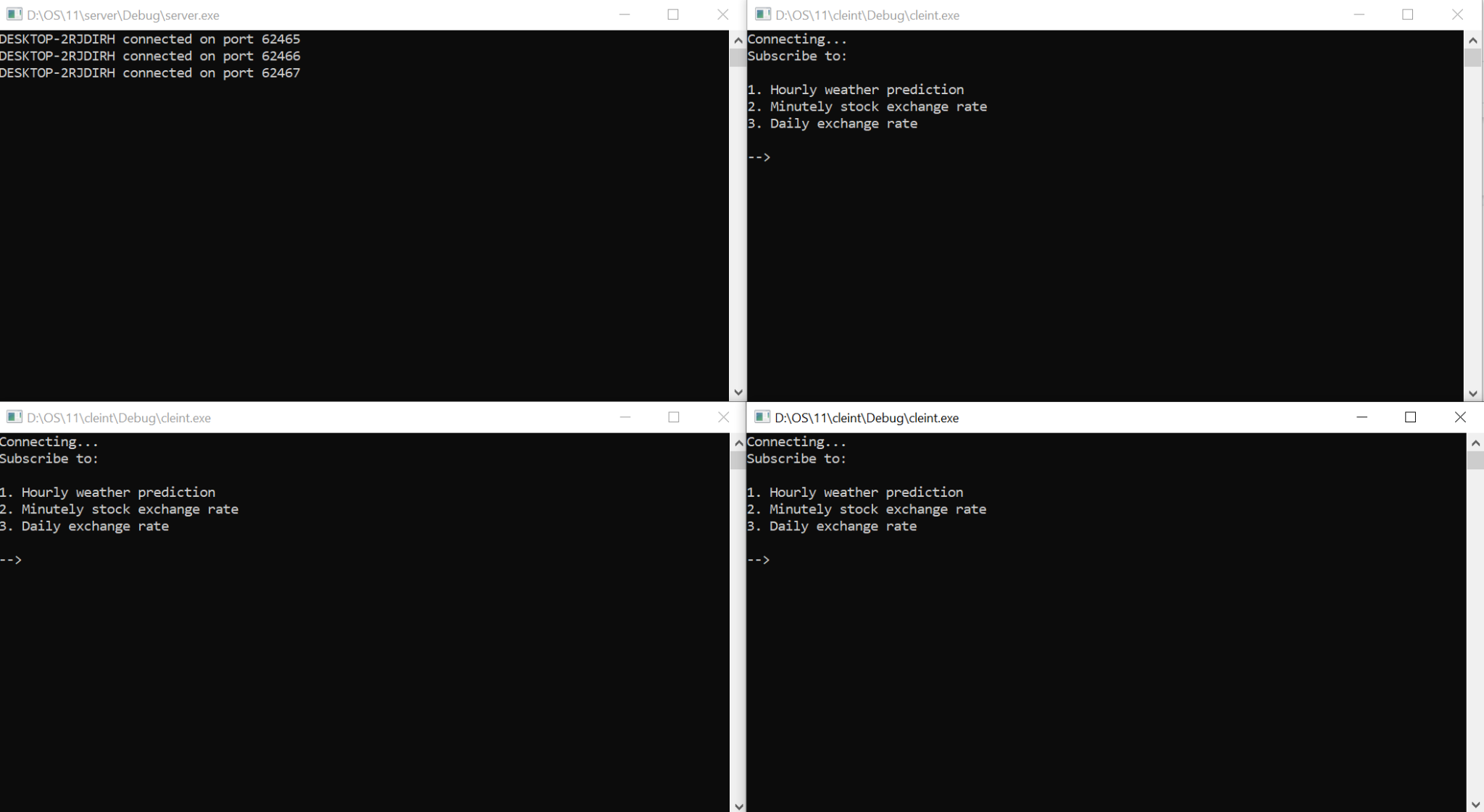


Рис 1. Підключення декількох клієнтів до одного сервера

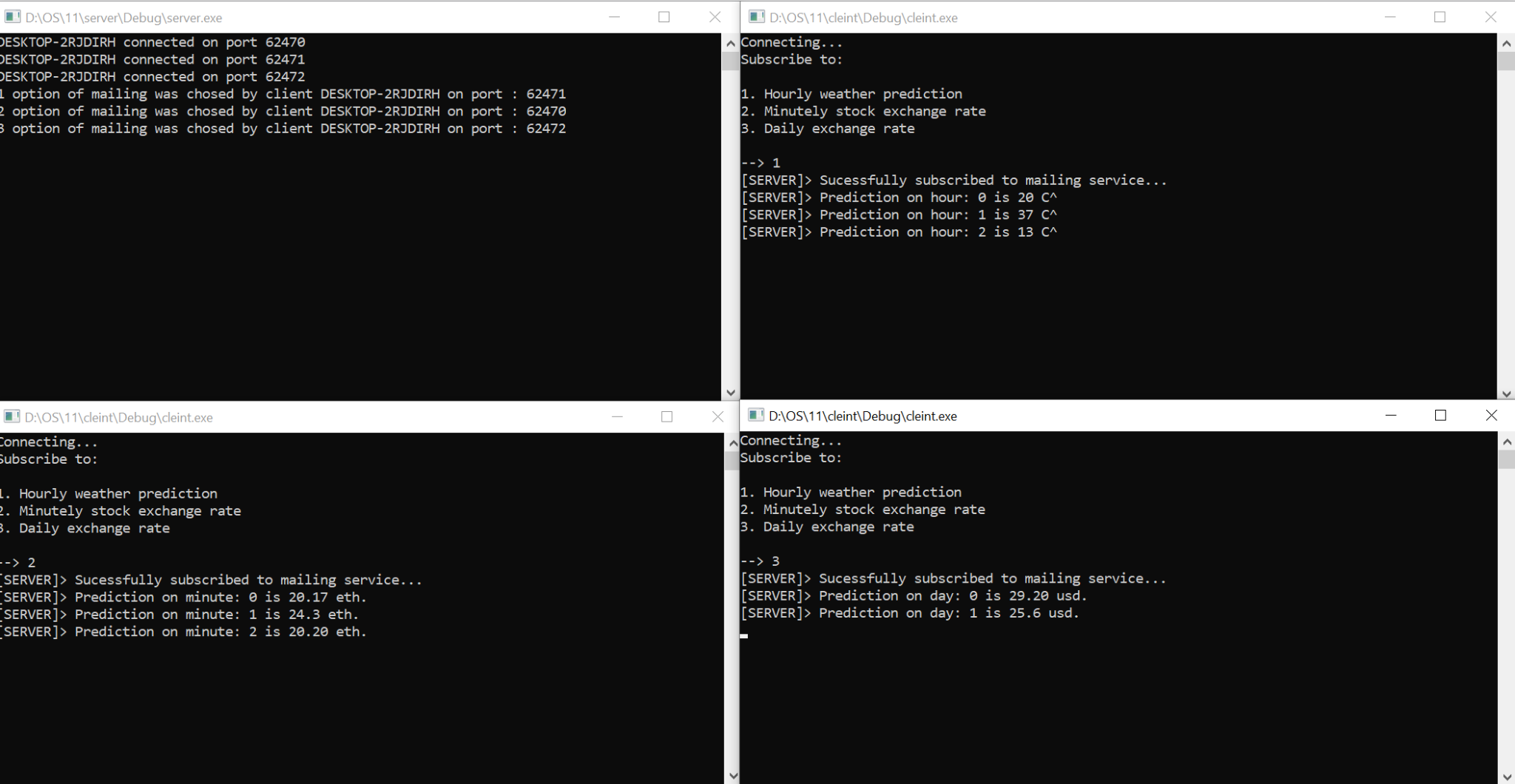


Рис 2. Запуск розсилки

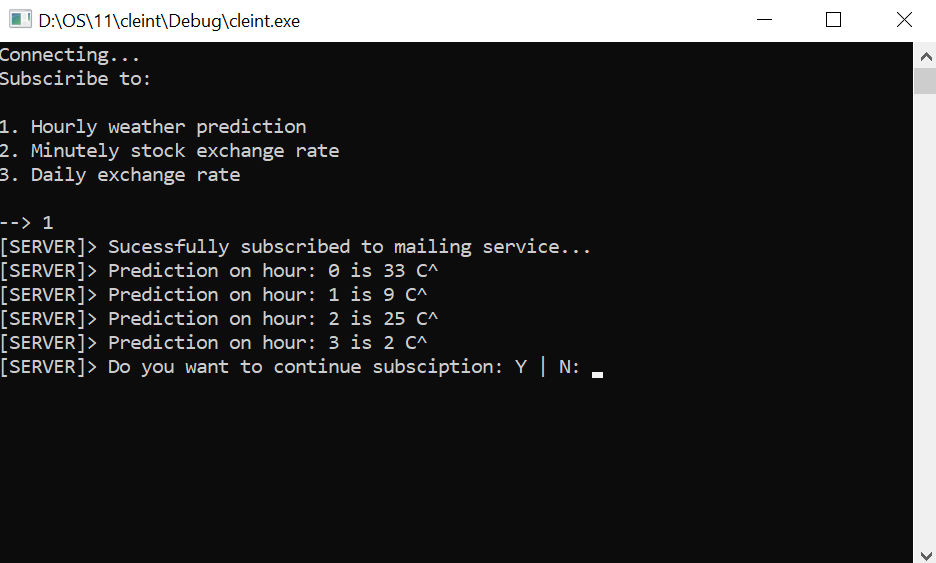


Рис 3. Запит на продовження підписки на розсилку

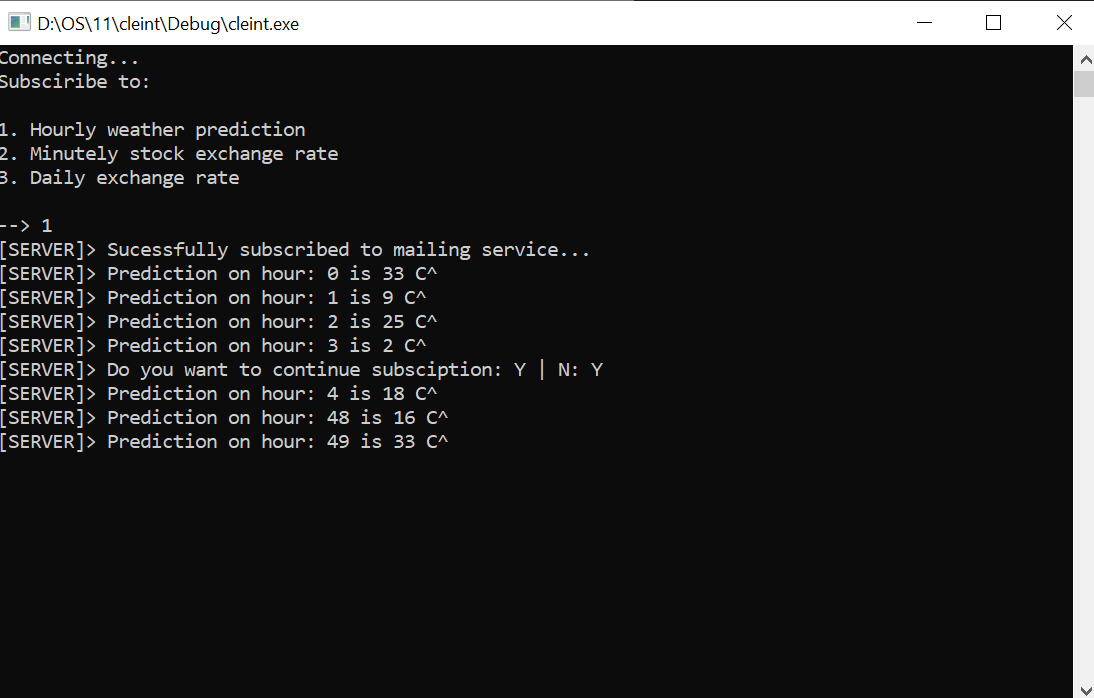


Рис 4. Продовження підписки на розсилку

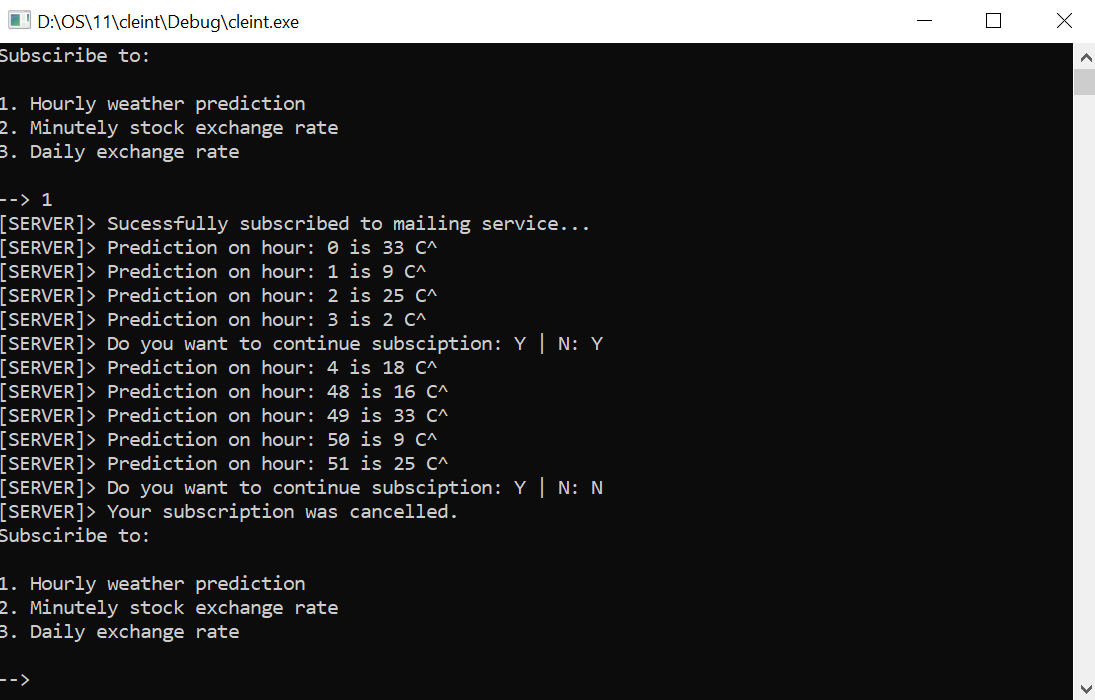


Рис 5. Відписка від розсилки

**Висновки**

Під час виконання лабораторної роботи ми ознайомилися зі способами міжпроцесної взаємодії. Ознайомилися з класичним прикладом взаємодії між процесами на прикладі задачі «виробник – споживач». Навчилися працювати із процесами з використанням способів міжпроцесної взаємодії, синхронізувати їхню роботу.

Також розвинули навички роботи у команді, комунікації між одногрупниками. При розробці врахували думку кожного під час розробки програми. Навчилися висловлювати власну думку у команді та приймати критику з боку колег.

Поділили наші обов’язки та завдання, щоб успішно реалізувати лабораторну роботу, проаналізували свої помилки та виправили їх.