

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.

Виконали:

студ. групи ПЗ-25

Бірак Д.Ф., Білик Т.П., Масюк О.Я., Янович М.А.

Прийняв:

ст. викл. каф. ПЗ

Горечко О. М.

«___» _____ 2021 р.

Σ = _____

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 WINAPI 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 WINAPI 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 WINAPI 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 WINAPI 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 структури `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 WINAPI 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).

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 main()
{
    //Init winsock
    WSADATA wsData;
    WORD ver = MAKEWORD(2, 2); //request winsock 2.2 version

    int wsOk = WSASStartup(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 addresses 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
SubscriptionAccepted = "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,
SubscriptionAccepted.c_str(), SubscriptionAccepted.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
SubscriptionCanceled = "Your subscription was cancelled.";
send(outSock,
SubscriptionCanceled.c_str(), SubscriptionCanceled.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 = WSASStartup(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
subscription: 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();
}

```

Фрагмент коду написаний студентом Біраком Денисом

```
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;
};

int main()
{
    //Init winsock
    WSADATA wsData;
    WORD ver = MAKEWORD(2, 2); //request winsock 2.2 version

    int wsOk = WSASStartup(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 used 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
                    {

```

[illegible]


```

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

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

                                string
SubscriptionCanceled = "Your subscription was cancelled.";
                                send(outSock,
SubscriptionCanceled.c_str(), SubscriptionCanceled.size() + 1, 0);

                                break;
                                }
                                }
                                }

```

Фрагмент коду написаний студентом Біликом Тарасом

```

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
subscription: 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();

```

Фрагмент коду написаний студентом Масюком Олегом

```

#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;
};

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;

```



```

// Initialize WinSock
WSAData data;
WORD ver = MAKEWORD(2, 2);
int wsResult = WSASStartup(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";

```

Виконання програми

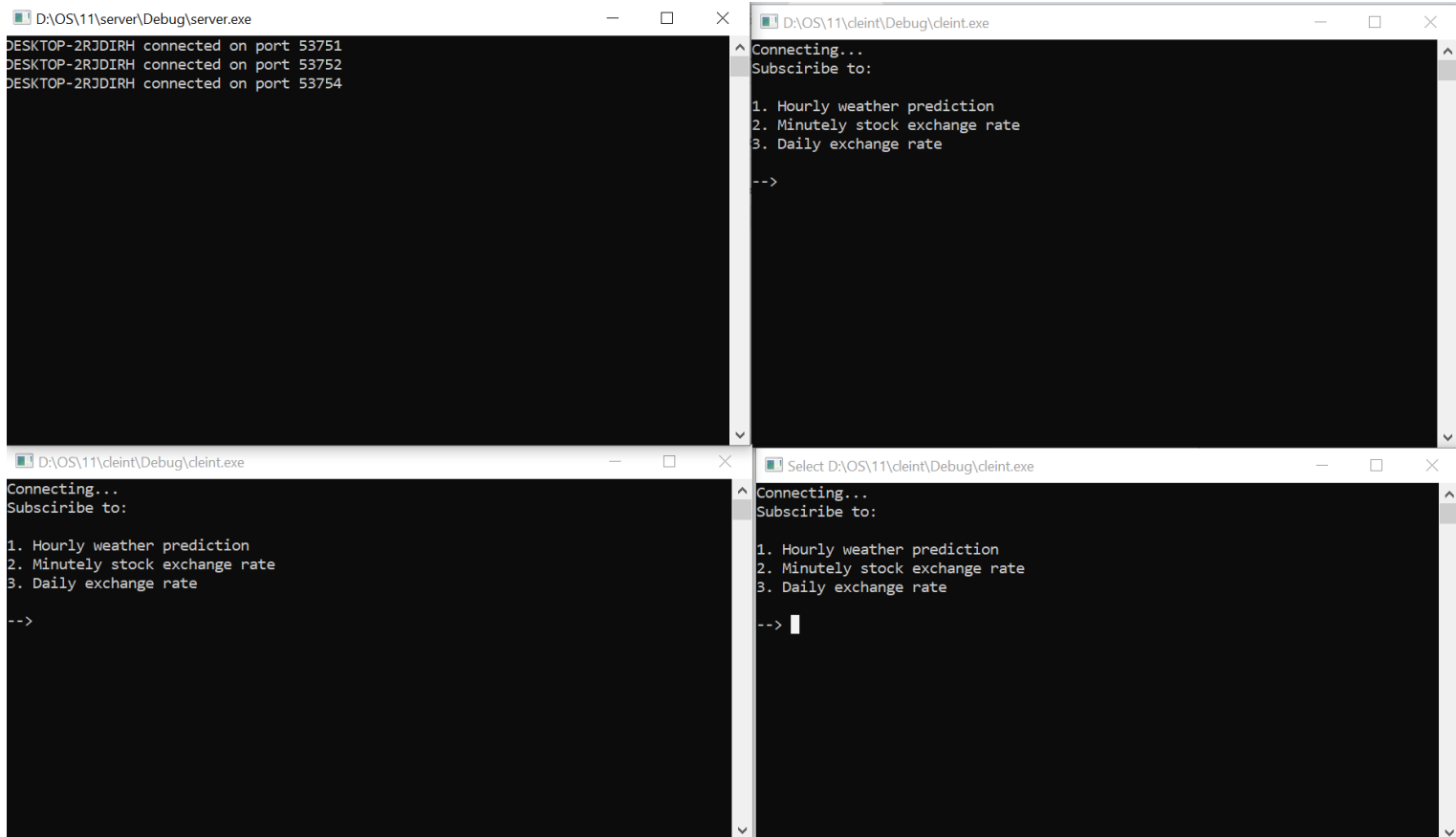


Рис 1. Підключення декількох клієнтів до одного сервера

The image displays four terminal windows arranged in a 2x2 grid, showing the interaction between a server and multiple clients. The top-left window, titled 'D:\OS\11\server\Debug\server.exe', shows the server receiving connections from 'DESKTOP-2RJDIRH' on ports 49890, 49891, and 49892. It lists three subscription options: 1. Hourly weather prediction, 2. Minutely stock exchange rate, and 3. Daily exchange rate. The top-right window, titled 'D:\OS\11\cleint\Debug\cleint.exe', shows a client connecting and selecting option 1. The server responds with hourly temperature predictions for hours 0, 1, and 2. The bottom-left window, also titled 'D:\OS\11\cleint\Debug\cleint.exe', shows a client selecting option 2. The server responds with minutely stock exchange rate predictions for minutes 0 and 1. The bottom-right window, also titled 'D:\OS\11\cleint\Debug\cleint.exe', shows a client selecting option 3. The server responds with daily stock exchange rate predictions for days 0 and 1.

```
D:\OS\11\server\Debug\server.exe
DESKTOP-2RJDIRH connected on port 49890
DESKTOP-2RJDIRH connected on port 49891
DESKTOP-2RJDIRH connected on port 49892
1 option of mailing was chosed by client DESKTOP-2RJDIRH on port: 49892
2 option of mailing was chosed by client on port:
3 option of mailing was chosed by client on port:

D:\OS\11\cleint\Debug\cleint.exe
Connecting...
Subscribire to:
1. Hourly weather prediction
2. Minutely stock exchange rate
3. Daily exchange rate
--> 1
[SERVER]> Sucessfully subscribed to mailing service...
[SERVER]> Prediction on hour: 0 is 33 C^
[SERVER]> Prediction on hour: 1 is 9 C^
[SERVER]> Prediction on hour: 2 is 25 C^

D:\OS\11\cleint\Debug\cleint.exe
Connecting...
Subscribire to:
1. Hourly weather prediction
2. Minutely stock exchange rate
3. Daily exchange rate
--> 2
[SERVER]> Sucessfully subscribed to mailing service...
[SERVER]> Prediction on minute: 0 is 2.26 eth.
[SERVER]> Prediction on minute: 1 is 28.12 eth.

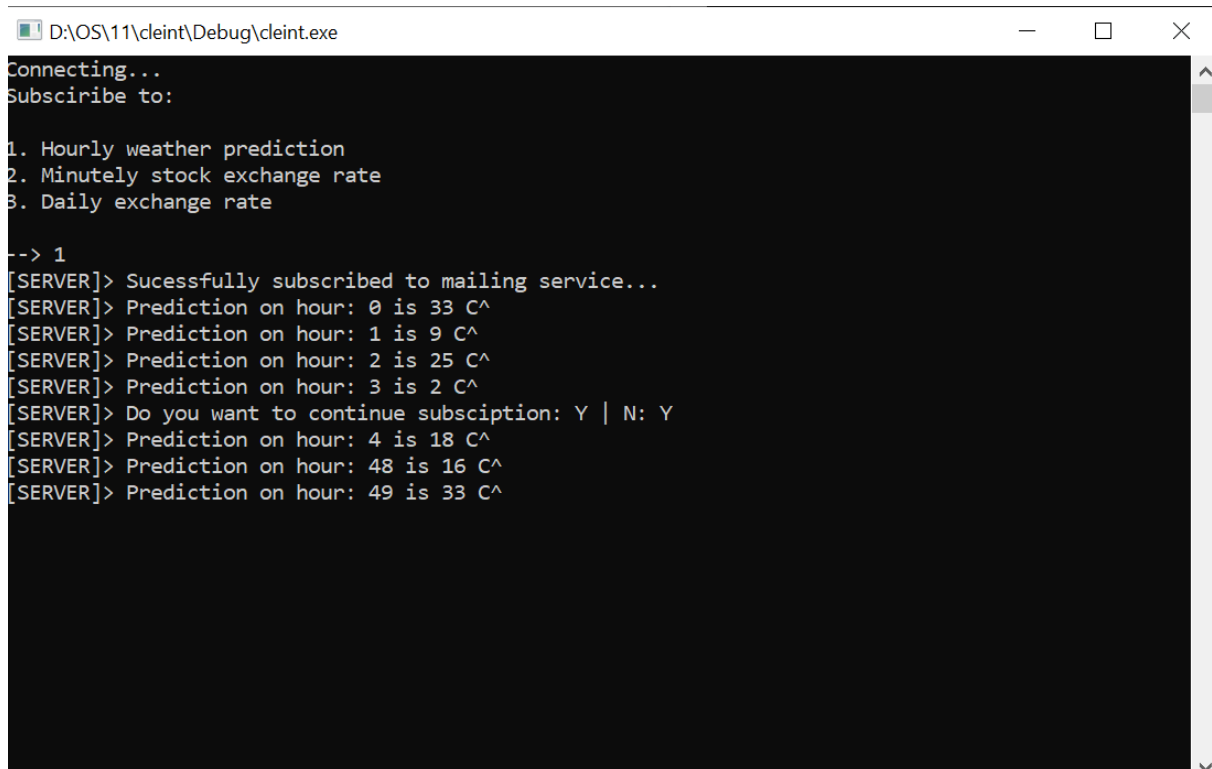
D:\OS\11\cleint\Debug\cleint.exe
Connecting...
Subscribire to:
1. Hourly weather prediction
2. Minutely stock exchange rate
3. Daily exchange rate
--> 3
[SERVER]> Sucessfully subscribed to mailing service...
[SERVER]> Prediction on day: 0 is 11.2 usd.
[SERVER]> Prediction on day: 1 is 15.19 usd.
```

Рис 2. Запуск розсилки

The image shows a single terminal window titled 'D:\OS\11\cleint\Debug\cleint.exe'. It displays the same subscription menu as in Figure 2. The user has selected option 1. The server has provided hourly temperature predictions for hours 0, 1, and 2. Now, the server prompts the user with '[SERVER]> Do you want to continue subscption: Y | N: ' followed by a cursor.

```
D:\OS\11\cleint\Debug\cleint.exe
Connecting...
Subscribire to:
1. Hourly weather prediction
2. Minutely stock exchange rate
3. Daily exchange rate
--> 1
[SERVER]> Sucessfully subscribed to mailing service...
[SERVER]> Prediction on hour: 0 is 33 C^
[SERVER]> Prediction on hour: 1 is 9 C^
[SERVER]> Prediction on hour: 2 is 25 C^
[SERVER]> Prediction on hour: 3 is 2 C^
[SERVER]> Do you want to continue subscption: Y | N: 
```

Рис 3. Запит на продовження підписки на розсилку

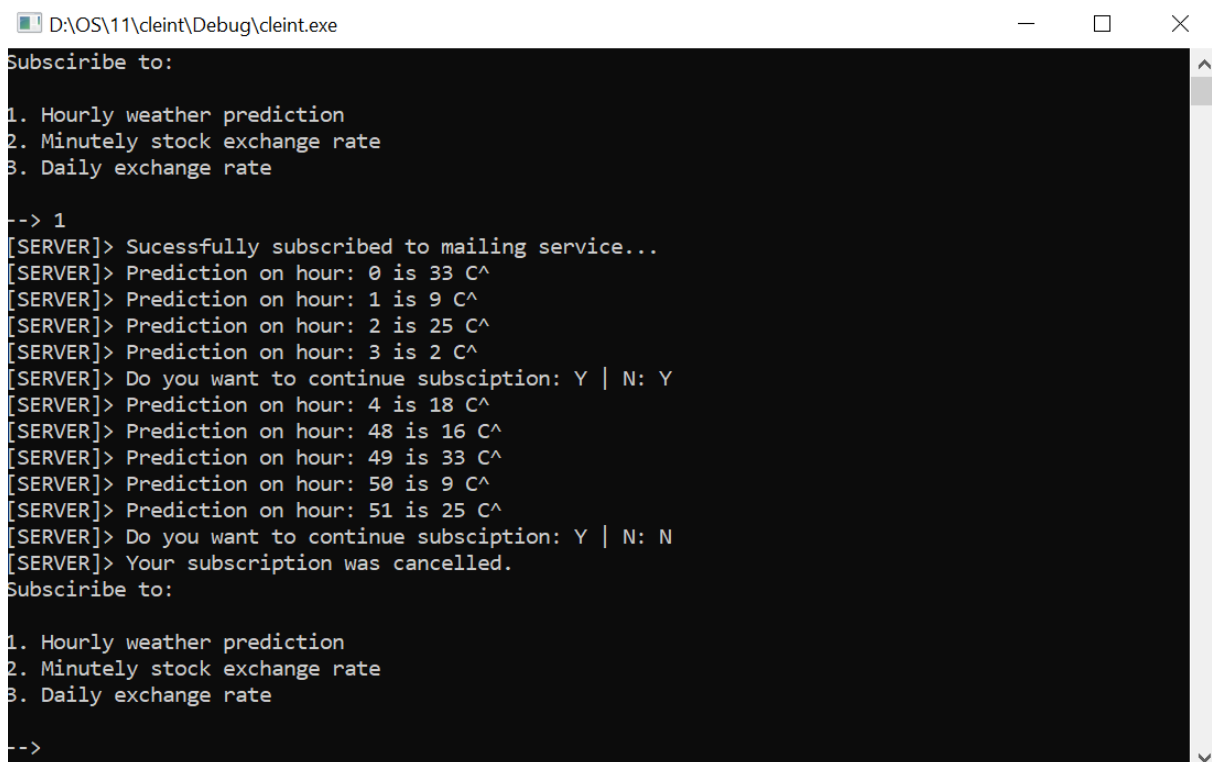


```
D:\OS\11\cleint\Debug\cleint.exe
Connecting...
Subsctirbe to:

1. Hourly weather prediction
2. Minutely stock exchange rate
3. Daily exchange rate

--> 1
[SERVER]> Sucessfully subscribed to mailing service...
[SERVER]> Prediction on hour: 0 is 33 C^
[SERVER]> Prediction on hour: 1 is 9 C^
[SERVER]> Prediction on hour: 2 is 25 C^
[SERVER]> Prediction on hour: 3 is 2 C^
[SERVER]> Do you want to continue subscription: Y | N: Y
[SERVER]> Prediction on hour: 4 is 18 C^
[SERVER]> Prediction on hour: 48 is 16 C^
[SERVER]> Prediction on hour: 49 is 33 C^
```

Рис 4. Продовження підписки на розсилку



```
D:\OS\11\cleint\Debug\cleint.exe
Subsctirbe to:

1. Hourly weather prediction
2. Minutely stock exchange rate
3. Daily exchange rate

--> 1
[SERVER]> Sucessfully subscribed to mailing service...
[SERVER]> Prediction on hour: 0 is 33 C^
[SERVER]> Prediction on hour: 1 is 9 C^
[SERVER]> Prediction on hour: 2 is 25 C^
[SERVER]> Prediction on hour: 3 is 2 C^
[SERVER]> Do you want to continue subscription: Y | N: Y
[SERVER]> Prediction on hour: 4 is 18 C^
[SERVER]> Prediction on hour: 48 is 16 C^
[SERVER]> Prediction on hour: 49 is 33 C^
[SERVER]> Prediction on hour: 50 is 9 C^
[SERVER]> Prediction on hour: 51 is 25 C^
[SERVER]> Do you want to continue subscription: Y | N: N
[SERVER]> Your subscription was cancelled.
Subsctirbe to:

1. Hourly weather prediction
2. Minutely stock exchange rate
3. Daily exchange rate

-->
```

Рис 5. Відписки 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 однокласниками. 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.