

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**  
**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»**  
**Інститут комп'ютерних наук та інформаційних технологій**  
**Кафедра програмного забезпечення**



**ЗВІТ**

До лабораторної роботи № 11

**На тему:** “*Організація взаємодії між процесами*”

**З дисципліни:** “*Операційні системи*”

**Лектор:**

Старший викладач  
кафедри ПЗ  
Грицай О.Д.

**Виконав:**

ст. гр. ПЗ-26  
Галайко С.В.

**Прийняв:**

Доцент  
кафедри ПЗ  
Горечко М.О

« \_\_\_\_ » \_\_\_\_\_ 2021 р.

$\Sigma$  = \_\_\_\_ .....

**Львів – 2021**

**Тема роботи:** Організація взаємодії між процесами.

**Мета роботи:** : 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**

Існує досить великий клас засобів ОС, за допомогою яких забезпечується взаємна синхронізація процесів і потоків. Потреба в синхронізації потоків виникає тільки в мультипрограмній ОС і залежить від спільного використання апаратних та інформаційних ресурсів обчислювальної системи. Синхронізація потрібна для запобігання перегонам та безвиході під час обміну даними між потоками, поділу даних, доступу до процесора і пристроїв введення-виведення. У багатьох ОС ці засоби називаються засобами міжпроцесної взаємодії – Inter Process Communications (IPC), що відображає історичну первинність поняття процес відносно поняття потік. Зазвичай до засобів IPC належать не тільки засоби міжпроцесної синхронізації, але й засоби обміну даними. Будь-яка взаємодія процесів або потоків залежить від їх синхронізації, яка полягає в узгодженні їх швидкостей через припинення потоку до настання деякої події й подальшої його активізації під час настання цієї події. Синхронізація лежить в основі будь-якої взаємодії потоків, незалежно від того, чи пов'язана ця взаємодія з розподілом ресурсів або з обміном даними. Наприклад, потік-одержувач повинен звертатися за даними тільки після того, як вони поміщені в буфер потоком-відправником. Якщо ж потік-одержувач звернувся до даних до моменту їх надходження в буфер, то він має бути припинений. Синхронізація також потрібна у разі спільного використання апаратних ресурсів. Наприклад, коли активному потоку потрібен доступ до послідовного порту, а з цим портом у монопольному режимі працює інший потік, який перебуває у стані очікування, то ОС припиняє активний потік і не активізує його доти, доки потрібний йому порт не звільниться. Часто потрібна також синхронізація з подіями, які не належать до обчислювальної системи, наприклад реакція на натискання

комбінації клавіш Ctrl+C. Для синхронізації потоків прикладних програм програміст може використовувати як власні засоби та прийоми синхронізації, так і засоби ОС.

Наприклад, два потоки одного прикладного процесу можуть координувати свою роботу за допомогою доступної для них обох глобальної логічної змінної, яка набуває значення одиниці при здійсненні деякої події, наприклад вироблення одним потоком даних, потрібних для продовження роботи іншого. Однак у багатьох випадках більш ефективними або навіть єдино можливими є засоби синхронізації, що надаються ОС у формі системних викликів. Так, потоки, що належать різним процесам, не мають можливості втручатися будь-яким чином у роботу один одного. Без посередництва ОС вони не можуть призупинити один одного або сповістити про подію, що відбулася. Засоби синхронізації використовуються ОС не тільки для синхронізації прикладних процесів, але й для її внутрішніх потреб. Зазвичай розробники ОС надають у розпорядження прикладних і системних програмістів широкий спектр засобів синхронізації. Ці засоби можуть утворювати ієрархію, коли на основі більш простих засобів будуються більш складні, а також бути функціонально спеціалізованими, наприклад засоби для синхронізації потоків одного процесу, засоби для синхронізації потоків різних процесів під час обміну даними і т. ін. Часто функціональні можливості різних системних викликів синхронізації перекриваються, тому для вирішення одного завдання програміст може скористатися кількома викликами залежно від особистих переваг.

## **Умова 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.

## **Індивідуальне завдання**

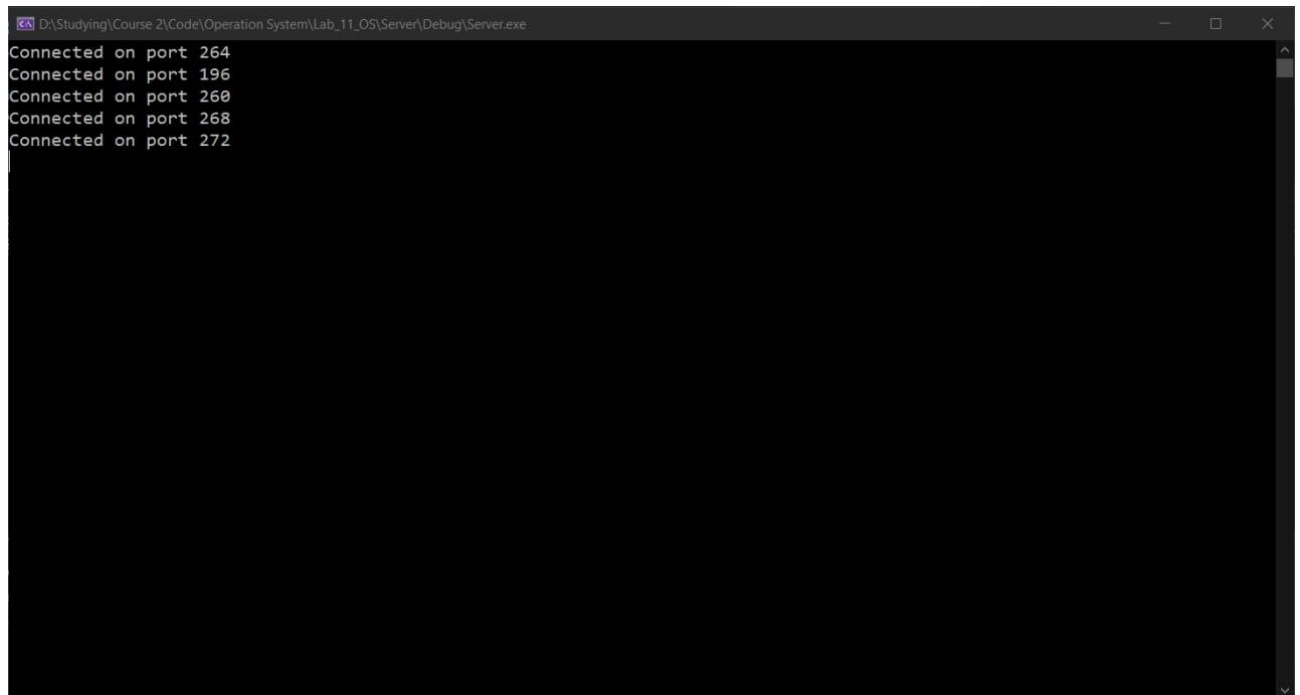
### **Варіант-6**

Створити програму, що моделює наступну ситуацію: Процес-науковий керівник проекту пропонує виконавців проекту-дочірні процеси. Процес-керівник створює додаток-віртуальну дошку (файл), де можна генерувати ідеї для проекту. Процеси-виконавці генерують ідеї, записуючи їх на спільну дошку. На виконання даного завдання вони мають 3 хвилини, після чого процес-керівник призупиняє їхню роботу і виводить на екран усі згенеровані ідеї, нумеруючи кожен з них. Процеси-виконавці голосують за три найкращі ідеї. Після чого процес-керівник записує на дошку три найкращі ідеї і закриває роботу додатку-віртуальної дошки, зберігаючи її вміст. Реалізувати дану модель, використовуючи б) файли, що проектуються у пам'ять; сокети (робота в межах однієї системи)

## ХІД РОБОТИ

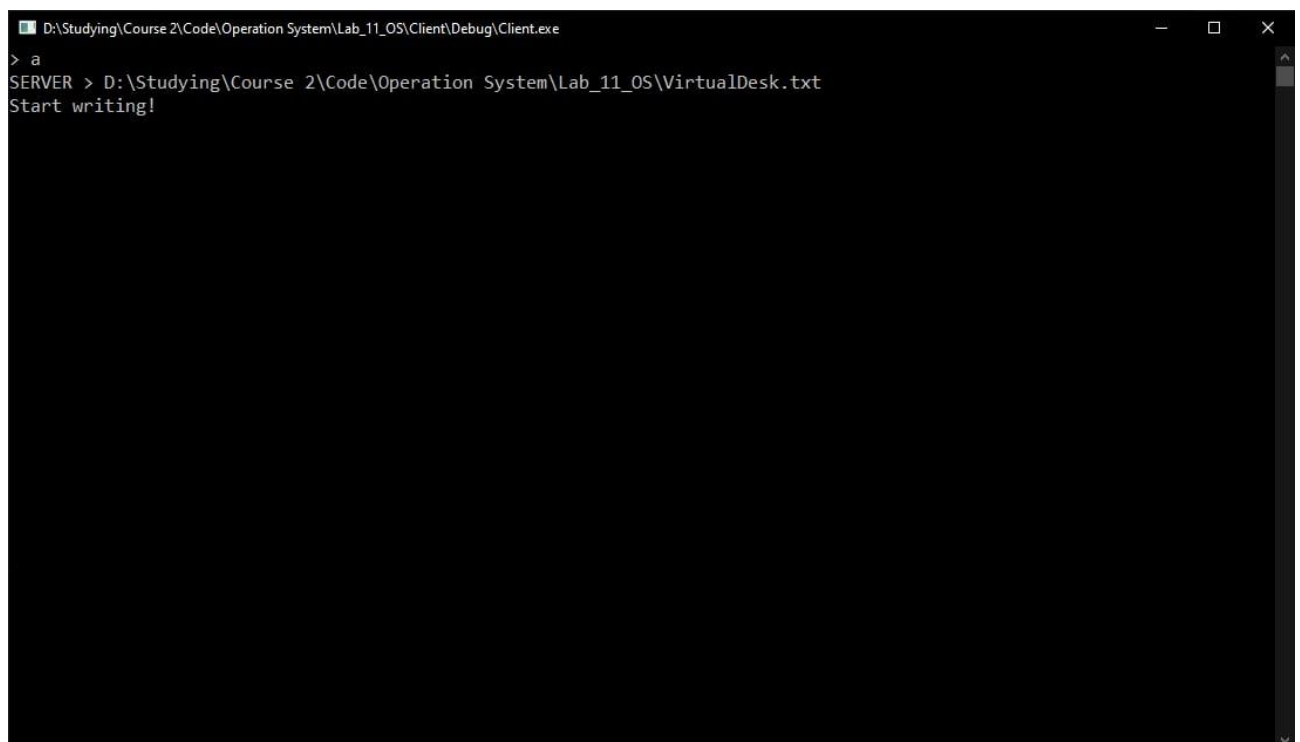
1. Зібрали команду для виконання проекту, у складі : Войцеховський Владислав, Галайко Степан, Жиляков Владислав, Закаляк Роман.
2. Створили GitHub проект для зручної командної роботи.
3. Зробили розподіл завдань. А саме:
  - Галайко Степан - розробляє частину головного процесу, який створює дошку (файл) і запускає інші процеси реалізація алгоритму визначення найкращих ідей CountingSearch()
  - Закаляк Роман - розробляє процеси-виконавці, що генерують ідеї та записують на спільну дошку
  - Жиляков Владислав - розробляє голосування процесів-виконавців та готує звіт про виконання завдання
  - Войцеховський Владислав - розробляє вивід результатів голосування на дошку, готує презентацію виконаного завдання
4. У команді провели дослідження предметної області.
5. Зреалізували кожен відповідну частину проекту.

## Результати виконання роботи



```
D:\Studying\Course 2\Code\Operation System\Lab_11_OS\Server\Debug\Server.exe
Connected on port 264
Connected on port 196
Connected on port 260
Connected on port 268
Connected on port 272
```

*Рис 1. Підключення до серверу*



```
D:\Studying\Course 2\Code\Operation System\Lab_11_OS\Client\Debug\Client.exe
> a
SERVER > D:\Studying\Course 2\Code\Operation System\Lab_11_OS\VirtualDesk.txt
Start writing!
```

*Рис 2. Початок роботи клієнта*

```
D:\Studying\Course 2\Code\Operation System\Lab_11_OS\Server\Debug\Server.exe
Connected on port 264
Connected on port 196
Connected on port 260
Connected on port 268
Connected on port 272

Time is up! Close file
1. Start your own blog
2. Offer consulting services
3. Create a popular social media channel
4. Write product reviews
5. Offer consulting services
6. Become a translator
7. App tester
8. Start a chatbot agency
9. Create a popular social media channel
10. Data entry specialist
11. Offer consulting services
12. Data entry specialist
13. Create a popular social media channel
14. Offer online courses
15. Create a popular social media channel
16. Start your own blog
17. Create a popular social media channel
18. Offer consulting services
19. App tester
20. Start a chatbot agency
21. Start a chatbot agency
22. Offer online courses
23. Start your own blog
24. Start a chatbot agency
25. Write product reviews
26. Create a popular social media channel
27. Offer online courses
28. Start your own blog
29. Offer consulting services
30. Start your own blog
Start voting!
```

*Рис 3. Завершення часу очікування серверу та зчитування ідей з файлу*

```
D:\Studying\Course 2\Code\Operation System\Lab_11_OS\Client\Debug\Client.exe
> a
SERVER > D:\Studying\Course 2\Code\Operation System\Lab_11_OS\VirtualDesk.txt
Start writing!
Stop writing! File is closed!
I vote for 6 idea!
I vote for 4 idea!
I vote for 11 idea!
```

*Рис 4. Голосування клієнта*

```
D:\Studying\Course 2\Code\Operation System\Lab_11_OS\Server(Debug)\Server.exe
Connected on port 256
Connected on port 272

Time is up! Close file
1. Start your own blog
2. Offer consulting services
3. Create a popular social media channel
4. Write product reviews
5. Offer consulting services
6. Become a translator
7. App tester
8. Start a chatbot agency
9. Create a popular social media channel
10. Data entry specialist
11. Offer consulting services
12. Data entry specialist
13. Create a popular social media channel
14. Offer online courses
15. Create a popular social media channel
16. Start your own blog
17. Create a popular social media channel
18. Offer consulting services
Start voting!

Voting ideas: 3 9 8 8 6 9 6 4 11 4 7 12 4 1 1
Best ideas:
1. Start your own blog
4. Write product reviews
6. Become a translator
Write to virtual desk!
```

*Рис 5. Завершення голосування та вибір трьох найкращих ідей*

```
VirtualDesk.txt
1 Start your own blog
2 Offer consulting services
3 Create a popular social media channel
4 Write product reviews
5 Offer consulting services
6 Become a translator
7 App tester
8 Start a chatbot agency
9 Create a popular social media channel
10 Data entry specialist
11 Offer consulting services
12 Data entry specialist
13 Create a popular social media channel
14 Offer online courses
15 Create a popular social media channel
16 Start your own blog
17 Create a popular social media channel
18 Offer consulting services
19
20 Best ideas:
21 1. Start your own blog
22 4. Write product reviews
23 6. Become a translator
24

D:\Studying\Course 2\Code\Operation System\Lab_11_OS\VirtualDesk.txt
Рд 20, стлп 13 100% Unix (LF) UTF-8
```

*Рис 6. Вміст вихідного файлу*



## Код (фрагмент виділений жирним це моя робота по коду)

**Server.cpp**

```
#include <iostream>
#include <string>
#include <vector>
#include <fstream>
#include <ws2tcpip.h>
#pragma comment (lib, "ws2_32.lib")
using namespace std;

int findMin(vector<int>&v) {
    int min = v[0];

    for (int i = 0; i < v.size(); i++) {
        if (v[i] < min) min = v[i];
    }

    return min;
}

int findMax(vector<int>&v) {
    int max = v[0];

    for (auto val : v) {
        if (val > max) max = val;
    }

    return max;
}

void CountingSearch(vector<int> v) {
    int min = findMin(v);
    int max = findMax(v);
    int best_ideas[3];
    int range = max - min + 1;

    vector<int> table(range);

    for (int i = 0; i < v.size(); i++) {
        table[v[i] - min]++;
    }

    for (int i = 0; i < v.size(); i++) {
        cout << table[i] << "\n";
    }

    for (int i = 0; i < 3; i++) {
        max = 0;
        best_ideas[i] = 0;
        for (int j = 0; j < table.size(); j++) {
            if (table[j] != 0) {
                if (table[max] < table[j]) {
                    max = j;
                }
            }
        }
        table[max] = 0;
        best_ideas[i] = max + min;
    }
    cout << "\n\n";
}
```

```

        for (int i = 0; i < 3; i++) {
            cout << best_ideas[i] << "\n";
        }
    }
HANDLE file;
char path[] = "D:\\Studying\\Course 2\\Code\\Operation System\\Lab_11_OS\\VirtualDesk.txt";
string bestIdeas;
bool voting = false;

void TimeIsUp() {
    Sleep(3000);
    cout << "Time is up! Close file" << endl;

    SetFileAttributesA(path, FILE_ATTRIBUTE_READONLY);
    file = CreateFileA(path, GENERIC_READ, FILE_SHARE_READ | FILE_SHARE_WRITE, NULL, OPEN_EXISTING,
FILE_ATTRIBUTE_READONLY, NULL);
    if (file == INVALID_HANDLE_VALUE) {
        cout << "Cannot open file!";
        getchar();
        return;
    }
    int counter = 1;
    char ReadBuf[1];
    string output;
    output.clear();
    DWORD dwRead;

    ifstream in(path);
    if (in.is_open())
    {
        while (getline(in, output))
        {
            cout << to_string(counter) << ". " << output << endl;
            counter++;
        }
    }
    in.close();

    cout << "Start voiting!";
    voting = true;
    getchar();

    CloseHandle(file);
}

int main()
{
    //Create file
    file = CreateFileA(path, GENERIC_WRITE | GENERIC_READ, FILE_SHARE_READ | FILE_SHARE_WRITE, NULL,
CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL);
    SetFileAttributesA(path, FILE_ATTRIBUTE_NORMAL);
    //CloseHandle(file);

    //Initialize winsock
    WSADATA wsData;
    WORD ver = MAKEWORD(2, 2);

    int wsOk = WSASStartup(ver, &wsData);
    if (wsOk != 0) {

```

```

    cerr << "Can't initialize winsock!" << endl;
    return 1;
}

//Create socket
SOCKET listening = socket(AF_INET, SOCK_STREAM, 0);
if (listening == INVALID_SOCKET) {
    cerr << "Can't create a socket!" << endl;
    return 1;
}

//Bind ip address and port to a socket
sockaddr_in hint;
hint.sin_family = AF_INET;
hint.sin_port = htons(54000);
hint.sin_addr.S_un.S_addr = INADDR_ANY; //Could also use inet_pton

bind(listening, (sockaddr*)&hint, sizeof(hint));

//Tell winsock the socket is for listening
listen(listening, SOMAXCONN);

fd_set master;
FD_ZERO(&master);

FD_SET(listening, &master);

clock_t start, stop;
double result = 0;
start = clock();
int countTime = 0;
while (1) {
    fd_set copy = master;
    int count = 1;
    int socketCount = select(0, &copy, nullptr, nullptr, nullptr);
    socketCount = 2;

    for (int i = 0; i < socketCount; i++) {
        SOCKET sock = copy.fd_array[i];
        if (sock == listening) {
            //Accept a new connction
            SOCKET client = accept(listening, nullptr, nullptr);

            //Add the new connection to the list of connected clients
            FD_SET(client, &master);

            //Send a welcome message to the connected client
            string welcomeMsg = "D:\\Studying\\Course 2\\Code\\Operation System\\Lab_11_OS\\VirtualDesk.txt";
            send(client, welcomeMsg.c_str(), welcomeMsg.size() + 1, 0);
            cout << "Connected on port " << client << endl;
        }
        else {
            //Recieve message
            char buf[4096];
            ZeroMemory(buf, 4096);
            int bytesIn = recv(sock, buf, 4096, 0);
            if (countTime == 0 && buf[0] == 'b') {
                countTime++;
                TimeIsUp();
            }
        }
    }
}

```

```

    }

//-----
    for (int i = 0; i < strlen(buf); i++) {
        if (buf[i] >= 48 && buf[i] <= 57) {
            int numb = buf[i] - 48;

            int tempCount = 1;
            ifstream in(path);
            if (in.is_open())
            {
                while (getline(in, bestIdeas))
                {
                    int numb = buf[i] - 48;
                    if (tempCount == numb) {
                        cout << endl << to_string(tempCount) << ". " << bestIdeas << endl;
                    }
                    tempCount++;
                }
            }
            in.close();
        }
    }

//-----

    if (bytesIn <= 0) {
        //Drop the client
        closesocket(sock);
        FD_CLR(sock, &master);
    }
    else {
        //Send message to other clients
        for (int i = 0; i < master.fd_count; i++) {
            SOCKET outSock = master.fd_array[i];
            if (outSock != listening && outSock != sock) {
                send(outSock, buf, bytesIn, 0);
            }
        }
    }
}

}

cout << "FUCK";

//Cleanup winsock
WSACleanup();

system("pause");
}

```

### Client.cpp

```

#include <iostream>
#include <vector>
#include <ws2tcpip.h>
#include <string>
#pragma comment (lib, "ws2_32.lib")
using namespace std;

```

```

int main()
{
    vector<string> ideas;
    ideas.push_back("Start a chatbot agency\n");
    ideas.push_back("Become a translator\n");
    ideas.push_back("Data entry specialist\n");
    ideas.push_back("App tester\n");
    ideas.push_back("Write product reviews\n");
    ideas.push_back("Start your own blog\n");
    ideas.push_back("Offer online courses\n");
    ideas.push_back("Create a popular social media channel\n");
    ideas.push_back("Offer consulting services\n");
    ideas.push_back("Become an online reseller\n");

    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 initialize winsock! " << wsResult << endl;
        return 1;
    }

    //Create socket
    SOCKET sock = socket(AF_INET, SOCK_STREAM, 0);
    if (sock == INVALID_SOCKET) {
        cerr << "Can't create a socket! " << WSAGetLastError() << endl;
        WSACleanup();
        return 1;
    }

    //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! " << WSAGetLastError() << endl;
        closesocket(sock);
        WSACleanup();
        return 1;
    }

    //Do-while loop to send and receive data
    char buf[4096];
    string userInput;
    int countOfIdeas = 0;

    do {
        //Prompt the user for some text
        cout << "> ";
        getline(cin, userInput);

        if (userInput.size() > 0) {

```

```

//Send the text
int sendResult = send(sock, userInput.c_str(), userInput.size() + 1, 0);
if (sendResult != SOCKET_ERROR) {
    //Wait for response
    ZeroMemory(buf, 4096);
    int bytesReceived = recv(sock, buf, 4096, 0);
    if (bytesReceived > 0) {
        //Echo response to console
        cout << "SERVER > " << string(buf, 0, bytesReceived) << endl;

        //Writing to file
        //char test[] = "D:\\Studying\\Course 2\\Code\\Operation System\\Lab_11_OS\\VirtualDesk.txt";

        HANDLE file = CreateFileA(buf, GENERIC_WRITE | GENERIC_READ | TRUNCATE_EXISTING, FILE_SHARE_READ |
FILE_SHARE_WRITE, NULL, OPEN_EXISTING, NULL, NULL);

        if (file == INVALID_HANDLE_VALUE) {
            cout << "Cannot open file!";
            getchar();
            return 1;
        }
        SetFileAttributesA(buf, FILE_ATTRIBUTE_NORMAL);
        cout << "Start writing!" << endl;
        while (1) {
            if (GetFileAttributesA(buf) == FILE_ATTRIBUTE_READONLY) {
                //WriteFile(file, (void*)"&", 1, 0, NULL);
                cout << "Stop writing! File is closed!" << endl;
                //getchar();
                break;
            }
            countOfIdeas++;
            int randIdea = rand() % 9;
            //SetFilePointer(file, 2, NULL, FILE_END);
            WriteFile(file, (void*)ideas[randIdea].c_str(), ideas[randIdea].length(), 0, NULL);
            send(sock, "b", 1, 0);
            Sleep(500);
        }

        vector<int> checkVect;
        bool check = true;
        for (int i = 0; i < 3; i++) {
            int numb = rand() % countOfIdeas;

            for (int j = 0; j < checkVect.size(); j++) {
                if (numb == checkVect[j]) {
                    i--;
                    check = false;
                    break;
                }
            }
            if (check) {
                checkVect.push_back(numb);
                string votes = to_string(numb+1);
                send(sock, votes.c_str(), 1, MSG_DONTROUTE);
                cout << "I vote for " << votes << " idea!" << endl;
            }
            check = true;
        }

        getchar();
        CloseHandle(file);

```

```

        }
    }
} while (userInput.size() > 0);

//Close everything
closesocket(sock);
WSACleanup();
//CloseHandle(file);
}

```

ManyClients.cpp

```

#include <Windows.h>
#include <iostream>
using namespace std;

```

```

STARTUPINFOA arrSi[1000];
PROCESS_INFORMATION arrPi[1000];

```

```

int main(){
    cout << "Put a num of clients: ";
    int countOfProcess;
    cin >> countOfProcess;
    string file;
    for(int i = 0; i < countOfProcess; i++){
        ZeroMemory(&arrSi[i], sizeof(STARTUPINFO));
        arrSi[i].cb=sizeof(STARTUPINFO);
        ZeroMemory(&arrPi[i], sizeof(PROCESS_INFORMATION));

        // file = "D:\\Studying\\Course 2\\Code\\Operation System\\mergeSort.exe";
        // char* path = new char[1000];

        // strcpy(path, file.c_str());
        CreateProcessA(NULL, "D:\\Studying\\Course 2\\Code\\Operation System\\Lab_11_OS\\Client\\Debug\\Client.exe",
        NULL, NULL, TRUE,CREATE_NEW_CONSOLE, NULL, NULL, &arrSi[i], &arrPi[i]);
    }

}

```

## **Висновки**

Я ознайомився зі способами міжпроцесної взаємодії. Також я ознайомився з класичним прикладом взаємодії між процесами на прикладі задачі «виробник – споживач». Навчився працювати із процесами з використанням способів міжпроцесної взаємодії, синхронізувати їхню роботу.