

**МИНОБРНАУКИ РОССИИ**  
**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**  
**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**  
**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**  
**Кафедра Вычислительной техники**

**ОТЧЕТ**  
**по лабораторной работе № 2.2**  
**по дисциплине «Операционные системы»**  
**Тема: «Управление файловой системой»**

Студент гр. 3311 Пасечный Л.В.\_\_\_\_\_

Преподаватель Тимофеев А. В.\_\_\_\_\_

Санкт-Петербург

2025

## **Введение**

### **Цель работы:**

Исследовать механизмы управления виртуальной памятью

1)

### **Постановка задачи:**

1. Создайте консольное приложение с меню (каждая выполняемая функция и/или операция должна быть доступна по отдельному пункту меню), которое выполняет:
  - получение информации о вычислительной системе (функция Win32 API – GetSystemInfo);
  - определение статуса виртуальной памяти (функция Win32 API – GlobalMemoryStatus);
  - определение состояния конкретного участка памяти по заданному с клавиатуры адресу (функция Win32 API – VirtualQuery);
  - раздельное резервирование региона и передачу ему физической памяти в автоматическом режиме и в режиме ввода адреса начала региона (функция Win32 API – VirtualAlloc, VirtualFree);
  - одновременное резервирование региона и передача ему физической памяти в автоматическом режиме и в режиме ввода адреса начала региона (функция Win32 API – VirtualAlloc, VirtualFree);
  - запись данных в ячейки памяти по заданным с клавиатуры адресам;
  - установку защиты доступа для заданного (с клавиатуры) региона памяти и ее проверку (функция Win32 API – VirtualProtect).
2. Запустите приложение и проверьте его работоспособность на нескольких наборах вводимых данных. Запротоколируйте результаты в отчет. Дайте свои комментарии в отчете относительно выполнения функций Win32 API.
2. Подготовьте итоговый отчет с развернутыми выводами по заданию.

### **Результаты:**

---MENU---

- 1 - Get information about calculate system
- 2 - Get status of virtual memory
- 3 - Get status of certain area of memory
- 4 - Separate reservations
- 5 - Simultaneous reservations
- 0 - Exit

Enter the option: 1

Number of processors: 16

Type of processor: 8664

Memory page size: 4096

---MENU---

- 1 - Get information about calculate system
- 2 - Get status of virtual memory
- 3 - Get status of certain area of memory
- 4 - Separate reservations
- 5 - Simultaneous reservations
- 0 - Exit

Enter the option: 2

Procent of using memory: 0%

Total memory size: 68853669675

Can use memory size: 8192

Total virtual memory size: 0

Can use virtual memory size: 1787493

Enter the option: 3

Basic adress: 0x10000000

Basic adress of block of the memory: 0

Protect of the block of memory: 0

Size of virtual memory: 1878917120

Status of the virtual memory: 65536

Protect of the virtual memory: 1

Type of the virtual memory: 0

```
---MENU---
1 - Get information about calculate system
2 - Get status of virtual memory
3 - Get status of certain area of memory
4 - Separate reservations
5 - Simultaneous reservations
0 - Exit
Enter the option: 4
Do you want to automatic adress or handle?(1/0) : 1
Memory is reserved
Memory is commit
Memory allocated at adress: 0x1b466510000
Do you want to record the data?(1/0): 1
Memory adress is: 0x1b466510000
Size of commit region: 4096

Enter the offset (0 to 4095) or -1 for exit : 20
Enter the digit (0 to 255): 25
Value 25 success record to adress 0x1b466510000
Check the value: 25

Enter the offset (0 to 4095) or -1 for exit : -1
Do you want to change protect of the memory to read only?(1/0): 1
Memory protecting changed to read only
Do you want to change protect of the memory to read and write?(1/0): 1
Memory protecting changed to read and write
Memory is free
```

```
---MENU---
1 - Get information about calculate system
2 - Get status of virtual memory
3 - Get status of certain area of memory
4 - Separate reservations
5 - Simultaneous reservations
0 - Exit
Enter the option: 4
Do you want to automatic adress or handle?(1/0) : 0
Enter the prefferred starting adress (in hexadecimal 0x10000 for example): 0x10000
Memory is reserved
Memory is commit
Memory allocated at adress: 0x10000
Do you want to record the data?(1/0): 0
Memory is free
```

```

---MENU---
1 - Get information about calculate system
2 - Get status of virtual memory
3 - Get status of certain area of memory
4 - Separate reservations
5 - Simultaneous reservations
0 - Exit
Enter the option: 5
Do you want to automatic adress or handle?(1/0) : 1
Memory is reserved and commit at adress: 0x1dbb3ce0000
Do you want to record the data?(1/0): 1
Memory adress is: 0x1dbb3ce0000
Size of commit region: 4096

Enter the offset (0 to 4095) or -1 for exit : 35
Enter the digit (0 to 255): 25
Value 25 success record to adress 0x1dbb3ce0000
Check the value: 25

Enter the offset (0 to 4095) or -1 for exit : -1
Do you want to change protect of the memory to read only?(1/0): 1
Memory protecting changed to read only
Do you want to change protect of the memory to read and write?(1/0): 1
Memory protecting changed to read and write
Memory is free

```

## Заключение

Программа демонстрирует работу с системной и виртуальной памятью Windows, используя API Win32. Включены функции для получения информации о системе, состоянии памяти, а также для резервирования, выделения, защиты и освобождения участков виртуальной памяти.

## Код программы

```

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

void get_sys_info(){
    SYSTEM_INFO si;
    GetSystemInfo(&si);

    cout << "Number of processors: " << si.dwNumberOfProcessors << endl;
    cout << "Type of processor: " << si.dwProcessorType << endl;
    cout << "Memory page size: " << si.dwPageSize << endl;
    system("pause");
}

void menu(){
    cout << "---MENU---" << endl;
    cout << "1 - Get information about calculate system" << endl;
    cout << "2 - Get status of virtual memory" << endl;
    cout << "3 - Get status of certain area of memory" << endl;
    cout << "4 - Separate reservations" << endl;
}

```

```

    cout << "5 - Simultaneous reservations" << endl;
    cout << "0 - Exit" << endl;
}

void gl_mem_satus(){
    MEMORYSTATUSEX ms;
    ms.dwLength = sizeof(MEMORYSTATUS);
    GlobalMemoryStatusEx(&ms);

    cout << "Procent of using memory: " << ms.dwMemoryLoad << "%" << endl;
    cout << "Total memory size: " << ms.ullTotalPhys / (1024 * 1024) << endl;
    cout << "Can use memory size: " << ms.ullAvailPhys / (1024 * 1024) << endl;
    cout << "Total virtual memory size: " << ms.ullTotalVirtual / (1024 * 1024) << endl;
    cout << "Can use virtual memory size: " << ms.ullAvailVirtual / (1024 * 1024) << endl;
}

void virt_quary(){
    MEMORY_BASIC_INFORMATION mbi;
    size_t size = VirtualQuery((LPVOID)0x10000000, &mbi, sizeof(mbi));

    if (size == 0){
        cerr << "Error: " << GetLastError() << endl;
        exit(EXIT_FAILURE);
    }else{
        cout << "Basic adress: " << mbi.BaseAddress << endl;
        cout << "Basic adress of block of the memory: " << mbi.AllocationBase << endl;
        cout << "Protect of the block of memory: " << mbi.AllocationProtect << endl;
        cout << "Size of virtual memory: " << mbi.RegionSize << endl;
        cout << "Status of the virtual memory: " << mbi.State << endl;
        cout << "Protect of the virtual memory: " << mbi.Protect << endl;
        cout << "Type of the virtual memory: " << mbi.Type << endl;
    }
    system("pause");
}

void record_data(LPVOID addr, size_t region_size){
    int flag;
    cout << "Memory adress is: " << addr << endl;
    cout << "Size of commit region: " << region_size << endl;

    while (true){
        size_t offset;
        int value;

        cout << "\nEnter the offset (0 to " << (region_size - 1) << ") or -1 for exit : ";
        cin >> offset;

        if (offset == (size_t)-1){
            break;
        }

        if (offset >= region_size){
            cerr << "Error: too big offset" << endl;
            continue;
        }

        cout << "Enter the digit (0 to 255): ";
        cin >> value;
        if (value < 0 || value > 255){
            cerr << "Incorrect value, try again" << endl;

```

```

        continue;
    }

    BYTE* ptr = reinterpret_cast<BYTE*>(addr) + offset;
    *ptr = static_cast<BYTE>(value);

    cout << "Value " << value << " success record to adress " << addr << endl;

    cout << "Check the value: " << static_cast<int>(*ptr) << endl;
}

cout << "Do you want to change protect of the memory to read only?(1/0): ";
cin >> flag;

if (flag == 1){
    DWORD oldProtect;
    if(!VirtualProtect(addr, region_size, PAGE_READONLY, &oldProtect)){
        cerr << "Error Virtual Protect: " << GetLastError() << endl;
        return;
    }

    cout << "Memory protecting changed to read only" << endl;
}

cout << "Do you want to change protect of the memory to read and write?(1/0): ";
cin >> flag;

if (flag == 1){
    DWORD oldProtect;
    if(!VirtualProtect(addr, region_size, PAGE_READWRITE, &oldProtect)){
        cerr << "Error Virtual Protect: " << GetLastError() << endl;
        return;
    }

    cout << "Memory protecting changed to read and write" << endl;
}
}

void sep_automatic_alloc(){
    size_t size = 4096;
    DWORD flProtect = PAGE_READWRITE;
    DWORD fdwAlloc = MEM_RESERVE;

    LPVOID lpAdress = VirtualAlloc(nullptr, size, fdwAlloc, flProtect);

    cout << "Memory is reserved" << endl;

    lpAdress = VirtualAlloc(lpAdress, size, MEM_COMMIT, flProtect);
    cout << "Memory is commit" << endl;

    if (lpAdress == nullptr){
        cerr << "Virtual alloc failed: " << GetLastError() << endl;
        return;
    }

    cout << "Memory allocated at adress: " << lpAdress << endl;

    int flag;
    cout << "Do you want to record the data?(1/0): ";
    cin >> flag;
}

```

```

    if (flag == 1){
        record_data(lpAdress, size);
    }

    if (!VirtualFree(lpAdress, 0, MEM_RELEASE)){
        cerr << "VirtualFree error: " << GetLastError() << endl;
        return;
    }

    cout << "Memory is free" << endl;
    system("pause");
}

void sep_handle_alloc(){
    LPVOID prefferedAddress;
    cout << "Enter the preffered starting adress (in hexadecimal 0x10000 for example): ";
    string input;
    cin >> hex >> input;
    prefferedAddress = (LPVOID)stoull(input, nullptr, 16);

    size_t size = 4096;
    DWORD flProtect = PAGE_READWRITE;
    DWORD fdwAlloc = MEM_RESERVE;

    LPVOID lpAdress = VirtualAlloc(prefferedAddress, size, fdwAlloc, flProtect);

    cout << "Memory is reserved" << endl;

    lpAdress = VirtualAlloc(lpAdress, size, MEM_COMMIT, flProtect);
    cout << "Memory is commit" << endl;

    if (lpAdress == nullptr){
        cerr << "Error Virtual alloc: " << GetLastError() << endl;
        return;
    }

    cout << "Memory allocated at adress: " << lpAdress << endl;

    int flag;
    cout << "Do you want to record the data?(1/0): ";
    cin >> flag;

    if (flag == 1){
        record_data(lpAdress, size);
    }

    if (!VirtualFree(lpAdress, 0, MEM_RELEASE)){
        cerr << "Error VirtualFree: " << GetLastError() << endl;
        return;
    }

    cout << "Memory is free" << endl;
    system("pause");
}

void sep_semult_auto(){
    size_t size = 4096;

    LPVOID addr = VirtualAlloc(NULL, size, MEM_COMMIT | MEM_RESERVE,
    PAGE_READWRITE);

```



```

    if (addr == nullptr){
        cerr << "Error of VirtualAlloc: " << GetLastError() << endl;
        return;
    }

    cout << "Memory is reserved and commit at adress: " << addr << endl;

    int flag;
    cout << "Do you want to record the data?(1/0): ";
    cin >> flag;

    if (flag == 1){
        record_data(addr, size);
    }

    if (!VirtualFree(addr, 0, MEM_RELEASE)){
        cerr << "Error of VirtualFree: " << GetLastError() << endl;
        return;
    }

    cout << "Memory is free" << endl;
    system("pause");
}

void sep_semult_handle(){
    LPVOID prefferedAddress;
    size_t size = 4096;
    cout << "Enter the preffered starting adress (in hexadecimal 0x10000 for example): ";
    string input;
    cin >> hex >> input;

    LPVOID addr = VirtualAlloc(prefferedAddress, size, MEM_RELEASE | MEM_COMMIT,
    PAGE_READWRITE);

    if (addr == nullptr){
        cerr << "Error VirtualAlloc: " << GetLastError() << endl;
        return;
    }

    cout << "Memory release and commit at adress: " << addr << endl;

    int flag;
    cout << "Do you want to record the data?(1/0): ";
    cin >> flag;

    if (flag == 1){
        record_data(addr, size);
    }

    if (!VirtualFree(addr, 0, MEM_RELEASE)){
        cerr << "Error VirtualFree : " << GetLastError() << endl;
        return;
    }

    cout << "Memory is free" << endl;
    system("pause");
}

int main(){
    int choice = -1;
    int flag;

```

```

do{
    menu();
    cout << "Enter the option: ";
    cin >> choice;
    switch (choice){
        case 1:
            get_sys_info();
            break;
        case 2:
            gl_mem_satus();
            break;
        case 3:
            virt_quary();
            break;
        case 4:
            cout << "Do you want to automatic adress or handle?(1/0) : ";
            cin >> flag;
            if (flag == 1){
                sep_automatic_alloc();
                break;
            }else if (flag == 0){
                sep_handle_alloc();
                break;
            }else{
                cout << "Incorrect answer, try again";
                break;
            }
        case 5:
            cout << "Do you want to automatic adress or handle?(1/0) : ";
            cin >> flag;
            if (flag == 1){
                sep_semult_auto();
                break;
            }else if (flag == 0){
                sep_semult_handle();
                break;
            }else{
                cout << "Incorrect answer, try again" << endl;
                break;
            }
        case 0:
            cout << "You choose exit" << endl;
            break;
        default:
            cout << "Please try again" << endl;
            break;
    }
} while (choice != 0);
}

```

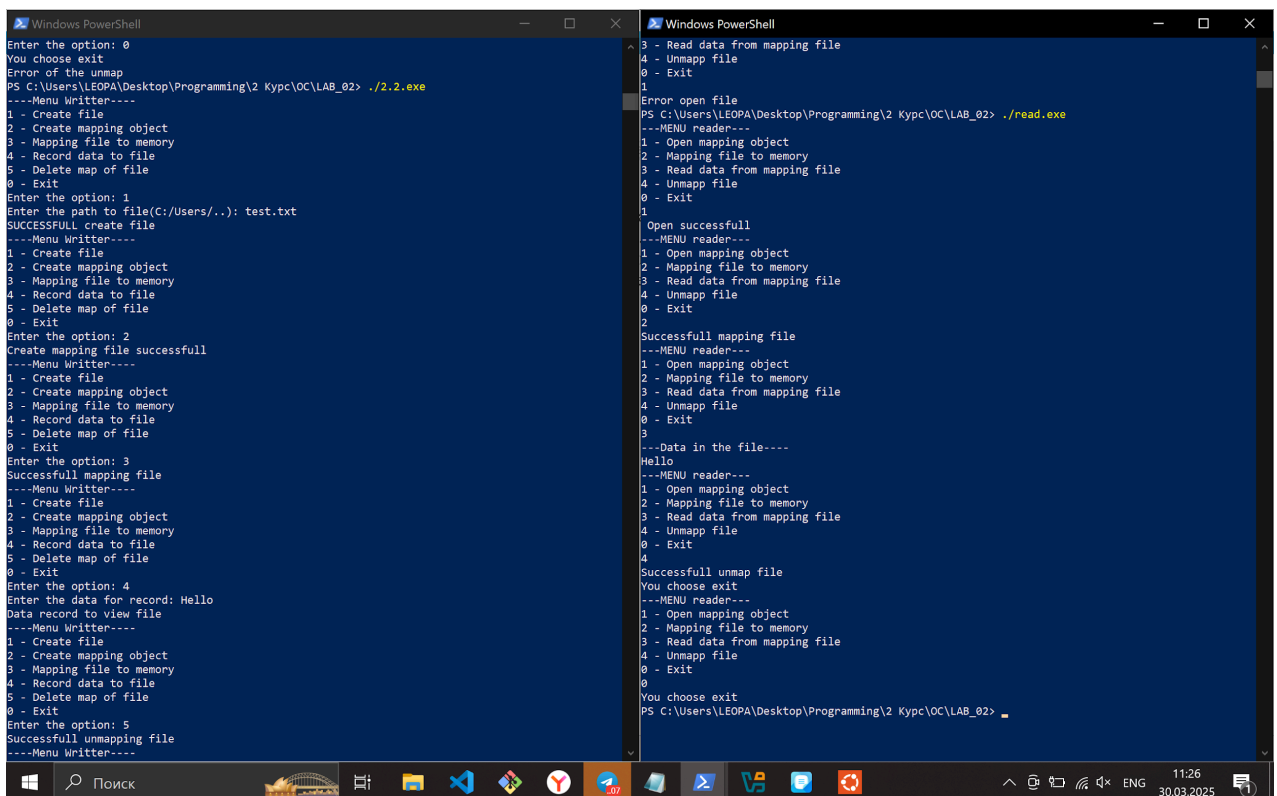
2)

### **Постановка задачи:**

1. Создайте два консольных приложения с меню (каждая выполняемая функция и/или операция должна быть доступна по отдельному пункту меню), которые выполняют:

- приложение-писатель создает проецируемый файл (функции Win32 API – CreateFile, CreateFileMapping), проецирует фрагмент файла в память (функции Win32 API – MapViewOfFile, UnmapViewOfFile), осуществляет ввод данных с клавиатуры и их запись в спроецированный файл;
  - приложение-читатель открывает проецируемый файл (функция Win32 API – OpenFileMapping), проецирует фрагмент файла в память (функции Win32 API – MapViewOfFile, UnmapViewOfFile), считывает содержимое из спроецированного файла и отображает на экран.
2. Запустите приложения и проверьте обмен данными между процессами, удостоверьтесь в надлежащем выполнении задания. Запротоколируйте результаты в отчет. Дайте свои комментарии в отчете относительно выполнения функций Win32 API.
3. Подготовьте итоговый отчет с развернутыми выводами по заданию.

## Результаты:



```
Windows PowerShell
Enter the option: 0
You choose exit
Error of the unmap
PS C:\Users\LEOPA\Desktop\Programming\2 Курс\OC\LAB_02> ./.2.2.exe
----Menu Writer----
1 - Create file
2 - Create mapping object
3 - Mapping file to memory
4 - Record data to file
5 - Delete map of file
0 - Exit
Enter the option: 1
Enter the path to file(C:/Users/.): test.txt
SUCCESSFULL create file
----Menu Writer----
1 - Create file
2 - Create mapping object
3 - Mapping file to memory
4 - Record data to file
5 - Delete map of file
0 - Exit
Enter the option: 2
Create mapping file successfull
----Menu Writer----
1 - Create file
2 - Create mapping object
3 - Mapping file to memory
4 - Record data to file
5 - Delete map of file
0 - Exit
Enter the option: 3
Successfull mapping file
----Menu Writer----
1 - Create file
2 - Create mapping object
3 - Mapping file to memory
4 - Record data to file
5 - Delete map of file
0 - Exit
Enter the option: 4
Enter the data for record: Hello
Data record to view file
----Menu Writer----
1 - Create file
2 - Create mapping object
3 - Mapping file to memory
4 - Record data to file
5 - Delete map of file
0 - Exit
Enter the option: 5
Successfull unmapping file
----Menu Writer----

Windows PowerShell
3 - Read data from mapping file
4 - Unmap file
0 - Exit
1
Error open file
PS C:\Users\LEOPA\Desktop\Programming\2 Курс\OC\LAB_02> ./read.exe
---MENU reader---
1 - Open mapping object
2 - Mapping file to memory
3 - Read data from mapping file
4 - Unmap file
0 - Exit
1
Open successfull
---MENU reader---
1 - Open mapping object
2 - Mapping file to memory
3 - Read data from mapping file
4 - Unmap file
0 - Exit
2
Successfull mapping file
---MENU reader---
1 - Open mapping object
2 - Mapping file to memory
3 - Read data from mapping file
4 - Unmap file
0 - Exit
3
---Data in the file---
Hello
---MENU reader---
1 - Open mapping object
2 - Mapping file to memory
3 - Read data from mapping file
4 - Unmap file
0 - Exit
4
Successfull unmap file
You choose exit
---MENU reader---
1 - Open mapping object
2 - Mapping file to memory
3 - Read data from mapping file
4 - Unmap file
0 - Exit
0
You choose exit
PS C:\Users\LEOPA\Desktop\Programming\2 Курс\OC\LAB_02>
```

## Заключение:

В ходе выполнения задания были созданы два консольных приложения, реализующих межпроцессный обмен данными с использованием технологии отображения файлов в память (Memory-Mapped Files) средствами Win32 API.

Приложение-писатель создает файл и отображение файла в память (через CreateFile, CreateFileMapping, MapViewOfFile), принимает ввод с клавиатуры и записывает данные непосредственно в область памяти, связанную с файлом.

Приложение-читатель открывает уже существующее отображение файла (OpenFileMapping), отображает его в свою адресную память (MapViewOfFile), считывает содержимое и выводит на экран.

Проверка работы программ показала, что данные, введенные в приложении-писателе, успешно считываются приложением-читателем без необходимости использования дополнительных файлов или каналов передачи данных. Обмен происходит быстро и эффективно, что подтверждает удобство и производительность этого метода межпроцессного взаимодействия.

Функции Win32 API, используемые для создания, открытия и управления отображениями файла, продемонстрировали стабильную работу. Важно правильно управлять правами доступа, размерами отображений и корректно освобождать ресурсы (UnmapViewOfFile, CloseHandle).

В целом, задание позволило освоить механизм отображения файлов в память для обмена данными между независимыми процессами, что является важным методом реализации высокоэффективного IPC (межпроцессного взаимодействия) в Windows.

**Вопрос: Напишите подробный вывод, как вы поняли механизм проецируемых файлов, как на уровне ОС получается, что процессы, находящиеся в разных ВАП, работают с общей памятью.**

Проецируемые файлы (memory-mapped files) позволяют отобразить содержимое файла или выделенной области памяти в виртуальное адресное пространство процесса. Для этого создается специальный объект ядра — секция.

Объект секции (Section Object) — это специальный объект ядра Windows, который представляет область памяти, потенциально связанную с файлом на диске или просто выделенную в оперативной памяти.

Он служит связующим звеном между физической памятью и виртуальными адресными пространствами разных процессов.

Когда один процесс создает или открывает именованный объект секции (CreateFileMapping), а другой процесс подключается к нему (OpenFileMapping), оба они отображают один и тот же участок физической памяти в свои разные виртуальные адресные пространства (MapViewOfFile).

В результате, несмотря на разные виртуальные адреса, процессы работают с общей физической памятью: все изменения, сделанные одним процессом, сразу видны другому.

На уровне ОС это реализуется через объект секции, который связывает виртуальные адреса разных процессов с одними и теми же физическими страницами памяти. Такой механизм обеспечивает быстрый и эффективный обмен данными без копирования и пересылок.

**Код:**

1) **Писатель**

```

#include<Windows.h>
#include<iostream>
#include<string>

#define FILE_SIZE 1024

using namespace std;

HANDLE hFile = INVALID_HANDLE_VALUE;
HANDLE hMapping = nullptr;
LPVOID pView = nullptr;
void menu(){
    wcout << L"----Menu Writer----" << endl;
    wcout << L"1 - Create file" << endl;
    wcout << L"2 - Create mapping object" << endl;
    wcout << L"3 - Mapping file to memory" << endl;
    wcout << L"4 - Record data to file" << endl;
    wcout << L"5 - Delete map of file" << endl;
    wcout << L"0 - Exit" << endl;
}

void create_file(){
    wstring path;
    wcout << "Enter the path to file(C:/Users/..): ";
    wcin.ignore();
    getline(wcin, path);

    hFile = CreateFileW(path.c_str(),
                        GENERIC_READ | GENERIC_WRITE,
                        0,
                        NULL,
                        OPEN_EXISTING,
                        FILE_ATTRIBUTE_NORMAL,
                        NULL);

    if (hFile == INVALID_HANDLE_VALUE){
        wcerr << L"Error openning file: " << GetLastError() << endl;
        CloseHandle(hFile);
        exit(EXIT_FAILURE);
    }

    cout << "SUCCESSFULL create file" << endl;
}

void create_mapping_file(){
    hMapping = CreateFileMappingW(hFile, NULL, PAGE_READWRITE, 0, FILE_SIZE,
    L"MyMappingObj");

    if (hMapping == NULL){
        cerr << "File mapping error: " << GetLastError() << endl;
        exit(EXIT_FAILURE);
    }

    cout << "Create mapping file successfull" << endl;
}

void MapViewOfFilefunc(){
    if (hMapping == nullptr){
        cerr << "ERROR of open hMapping file" << endl;
    }
}

```

```

        exit(EXIT_FAILURE);
    }

    if (pView != nullptr){
        cerr << "This file already view" << endl;
        exit(EXIT_FAILURE);
    }

    pView = MapViewOfFile(hMapping, FILE_MAP_ALL_ACCESS, 0, 0, FILE_SIZE);

    if (pView == nullptr){
        cerr << "ERROR map view if file" << endl;
        exit(EXIT_FAILURE);
    }

    cout << "Successfull mapping file" << endl;
}

void writeData(){
    if (pView == nullptr){
        cerr << "File not view" << endl;
        return;
    }

    cout << "Enter the data for record: ";
    string input;
    cin.ignore();
    getline(cin, input);

    if (input.length() >= FILE_SIZE) {
        cerr << "Too big input data" << endl;
        return;
    }

    memcpy(pView, input.c_str(), input.length() + 1);
    cout << "Data record to view file" << endl;
}

void unmapView(){
    if (pView == nullptr){
        cerr << "File not view" << endl;
        return;
    }

    if (UnmapViewOfFile(pView) == 0){
        cerr << "Error of the unmap" << endl;
        return;
    }else{
        cout << "Successfull unmapping file" << endl;
        pView = nullptr;
    }
}

void closeHandle(){
    if (pView != nullptr){
        unmapView();
    }

    if (hMapping != nullptr){
        CloseHandle(hMapping);
        hMapping = nullptr;
    }
}

```

```

    }

    if (hFile != nullptr){
        CloseHandle(hFile);
        hFile = nullptr;
    }
}

int main(){
    int choice;

    do{
        menu();
        cout << "Enter the option: ";
        cin >> choice;
        switch(choice){
            case 1:
                create_file();
                break;
            case 2:
                create_mapping_file();
                break;
            case 3:
                MapViewOfFilefunc();
                break;
            case 4:
                writeData();
                break;
            case 5:
                unmapView();
                break;
            case 0:
                cout << "You choose exit" << endl;
                break;
            default:
                cout << "Try again." << endl;
                break;
        }
    }while(choice != 0);

    closeHandle();
    return 0;
}

```

## 2) Читатель

```

#include <windows.h>
#include <iostream>
#include <string>
using namespace std;

#define FILE_SIZE 1024

HANDLE hFileMapping = nullptr;
LPVOID pView = NULL;

void menu(){
    cout << "---MENU reader---" << endl;
    cout << "1 - Open mapping object" << endl;
    cout << "2 - Mapping file to memory" << endl;
    cout << "3 - Read data from mapping file" << endl;
}

```

```

    cout << "4 - Unmapp file" << endl;
    cout << "0 - Exit" << endl;
}

void OpenFileMappingfunc() {
    if (hFileMapping != nullptr) {
        cerr << "File already open" << endl;
        exit(EXIT_FAILURE);
    }

    hFileMapping = OpenFileMappingW(
        FILE_MAP_READ,
        FALSE,
        L"MyMappingObj"
    );

    if (hFileMapping == nullptr) {
        cerr << "Error open file" << endl;
        exit(EXIT_FAILURE);
    } else {
        cout << " Open successfull" << endl;
    }
}

void mapView() {
    if (hFileMapping == nullptr) {
        cerr << "File not mapping" << endl;
        exit(EXIT_FAILURE);
    }

    if (pView != nullptr) {
        cerr << "File already view" << endl;
        exit(EXIT_FAILURE);
    }

    pView = MapViewOfFile(hFileMapping, FILE_MAP_READ, 0, 0, FILE_SIZE);

    if (pView == nullptr) {
        cerr << "Error of view map file" << endl;
        exit(EXIT_FAILURE);
    } else {
        cout << "Successfull mapping file" << endl;
    }
}

void readData() {
    if (pView == nullptr) {
        cerr << "File not view" << endl;
        exit(EXIT_FAILURE);
    }

    char* data = static_cast<char*>(pView);
    cout << "---Data in the file----" << endl;
    cout << data << endl;
}

void unmapView() {
    if (pView == nullptr) {
        cerr << "File not view" << endl;
        exit(EXIT_FAILURE);
    }
}

```



```

    if (UnmapViewOfFile(pView) == 0){
        cerr << "Error of file unmapping" << endl;
        exit(EXIT_FAILURE);
    }else{
        cout << "Successfull unmap file" << endl;
        pView = nullptr;
    }
}

void closeHandle(){
    if (pView != nullptr){
        unmapView();
    }
    if (hFileMapping != nullptr){
        CloseHandle(hFileMapping);
        hFileMapping = nullptr;
    }
}

int main(){
    int option;

    do{
        menu();
        cin >> option;
        switch(option){
            case 1:
                OpenFileMappingfunc();
                break;
            case 2:
                mapView();
                break;
            case 3:
                readData();
                break;
            case 4:
                unmapView();
            case 0:
                cout << "You choose exit" << endl;
                break;
            default:
                cout << "Try again" << endl;
        }
    }while (option != 0);

    closeHandle();
    return 0;
}

```