**Міністерство освіти і науки України**

**Національний університет «Львівська політехніка»**

**Інститут комп’ютерних наук та інформаційних технологій**

**Кафедра «Системи штучного інтелекту»**



**Розрахункова робота з курсу «Операційні системи»**

***Виконав:***

*студент групи КН-208*

*Пулик Максим*

***Перевірив:***

*к.т.н., асистент*

*Кривенчук Ю.П.*

***Львів 2019 р.***

**Варіант 1**

Потрібно написати програму для обміну текстовими повідомленнями між екземплярами через що проектуються в пам'ять файли, яка:

1. Запитує у користувача найменування спільної області пам'яті.

2. Намагається відкрити зазначену область функцією **OpenFileMapping ()** ; якщо

області не існує, створює її функцією **CreateFileMapping ().**

Вказівка. Відсутність спільної області пам'яті можна діагностувати, якщо функція

**OpenFileMapping ()** завершилася помилкою з відповідним кодом (який можна

отримати функцією **GetLastError ()**). Дізнатися потрібний код помилки можна по MSDN.

3. Проектує ділянку спільної області пам'яті фіксованого розміру.

4. Запитує у користувача, що слід зробити:

4.1. Записати рядок в спільну область пам'яті. Необхідно запропонувати користувачеві ввести рядок, а потім записати в спроектований ділянку спільної пам'яті.   
4.2. Зчитати рядок в спільну область пам'яті. Необхідно вивести на екран вміст спроектованої ділянки пам'яті як рядок.   
4.3. Завершити роботу. Необхідно закінчити проектування функцією **UnmapViewOfFile ()** і відключитися від відкритої або створеної спільної області пам'яті функцією **CloseHandle ().**

5. Перевірити експериментально і відобразити результати в звіті можливість роботи зі спроектованою областю файлу після того, як завершилася створила цю

область програма (з програм, які відкрили цю область ще при її роботі).

***Приклад роботи програми - процес № 1:***

*Enter mapping name: communication*

*Connecting to `communication '... failed.*

*Creating new mapping `comminication '.*

*Enter 1 for writing, 2 for reading, 3 for exit: 1*

*Enter the message: Hello!*

*Enter 1 for writing, 2 for reading, 3 for exit:*

***Приклад роботи програми - процес № 2:***

*Enter mapping name: communication*

*Connecting to `communication '... done.*

*Enter 1 for writing, 2 for reading, 3 for exit: 2*

*The message is `Hello! '.*

**Теоретичні відомості**

***Відображення файлу або File Mapping*** - це об'єднання вмісту файлу з частиною віртуального адресного простору процесу. Система створює об’єкт відображення файлів (також відомий як об’єкт розділу) для підтримки цієї асоціації. ***Вигляд файлу або*** ***File view*** - це частина віртуального адресного простору, яку використовує процес для доступу до вмісту файлу. Відображення файлів дозволяє процесу використовувати як випадковий вхід і вихід (введення / виведення), так і послідовні введення / виведення. Це також дозволяє ефективно працювати з великими файловими даними, такими як база даних, без необхідності відображати весь файл у пам'ять. Кілька процесів також можуть використовувати файли, зібрані в пам'яті, для обміну даними.

Процеси зчитування та записування у вигляд файлу використовують вказівники, так само, як і з динамічно виділеною пам'яттю. Використання відображення файлів підвищує ефективність, оскільки файл знаходиться на диску, але вигляд файлу знаходиться в пам'яті. Процеси також можуть управляти поданням файлів за допомогою функції ***VirtualProtect***.

Файлом на диску може бути будь-який файл, який ви хочете відобразити в пам'ять, або файл системної сторінки. Об'єкт відображення файлу може складатися з усього файлу або лише його частини . Він підтримується файлом на диску. Це означає, що коли система замінює сторінки об’єкта відображення файлів, будь-які зміни, внесені до об’єкта відображення файлів, записуються у файл. Коли сторінки об’єкта відображення файлів повертаються назад, вони відновлюються з файлу. Це вважається написанням і зчитуванням через оперативну пам’ять.

Вигляд файлу може складатися з усієї або лише частини об'єкта відображення файлів. Процес маніпулює файлом через вигляд файлів. Процес може створити кілька виглядів для об’єкта відображення файлів. Вигляди файлів, створені кожним процесом, знаходяться у віртуальному адресному просторі цього процесу. Коли процес потребує даних з частини файлу, відмінного від того, що знаходиться у поточному вигляді файлу, він може видалити відображення поточного вигляду файлу, а потім створити новий вигляд файлу.

Коли багато процесів використовують один і той же об’єкт відображення файлів для створення представлень для локального файлу, дані є ***когерентними.*** Тобто, перегляди містять однакові копії файлу на диску. Файл не може перебувати на віддаленому комп'ютері, якщо ви хочете ділитися пам'яттю між декількома процесами, тобто потрібна фізична наявність файлу на диску.

***Код програми***

#include <iostream>

#include <windows.h>

#include <stdio.h>

#include <conio.h>

#include <tchar.h>

#include <string>

using namespace std;

#define BUFFER\_SIZE 256

//Задаємо максимальний розмір об'єкта для запису

int main()

{

string mapping\_object\_name;

cout << "Enter the name of maping object" << endl;

cin >> mapping\_object\_name; //назва області пам'яті ( рядок )

boolean was\_created = false; //датчик який перевіряє чи область пам'яті була створена

HANDLE hMapFile; //абстрактне посилання на память

LPCTSTR pBuf; //макрос для декларування рядка

//перетворення string у wstring, а потім в LPCWSTR, щоби задати його як параметр функції OpenFileMapping або CreateFileMappping

//LPCWSTR mapping\_object = transformation(mapping\_object\_name);

wstring stemp = wstring(mapping\_object\_name.begin(), mapping\_object\_name.end());

LPCWSTR mapping\_object = stemp.c\_str();

cout << "Searching " << mapping\_object\_name << endl;

hMapFile = OpenFileMapping(FILE\_MAP\_ALL\_ACCESS, FALSE, mapping\_object); //отримує повний доступ до рядка у спільній пам'яті

if (hMapFile == NULL) //перевірка чи існує дана область пам'яті

//якщо не існує, тоді створюємо дану область пам'яті, і змінюємо датчик

{

cout << "Object |" << mapping\_object\_name << "| wasn't found" << endl;

cout << "Mapping object |" << mapping\_object\_name << "| will be created" << endl;

was\_created = true; //позначає що область пам'яті була створена

hMapFile = CreateFileMapping(INVALID\_HANDLE\_VALUE, NULL, PAGE\_READWRITE, 0, BUFFER\_SIZE, mapping\_object); //створює вказану в параметрах область пам'яті

if (hMapFile == NULL) //перевірка чи створилась область пам'яті

{

cout << "Creating of file mapping object failed(%d).\n" << GetLastError(); //при невдачі повертає помилку

return 1;

}

}

else //якщо існує, тоді показуємо це, і не змінюємо датчик

{

cout << "Mmaping object" << mapping\_object\_name << " has been found" << endl;

was\_created = false; //позначає що область пам'яті вже була створена раніше

}

pBuf = (LPCTSTR)MapViewOfFile(hMapFile, FILE\_MAP\_ALL\_ACCESS, 0, 0, BUFFER\_SIZE); //отримуємо лінк на введений рядок у пам'яті

if (pBuf == NULL) // якщо область пуста, повертає помилку

{

cout << "Mapping view of file failed(%d).\n" << GetLastError();

CloseHandle(hMapFile); //очищуємо посилання на спільну область пам'яті

return 1;

}

int command;

string input;

do //починається робота програми до моменту виходу певною командою

{

cout << "\nAvailable actions with mapping object: " << endl;

cout << "to write enter '1'" << endl;

cout << "to read enter '2'" << endl;

cout << "to exit enter '3'" << endl;

cin >> command; //введення команди

switch (command)

{

case 1:

{

cout << "Enter the string you want to store in the memory" << endl;

cin >> input; //введений рядок для запису

//перехід string в LPCWSTR

//LPCWSTR input\_text = transformation(input);

wstring stemp = wstring(input.begin(), input.end());

LPCWSTR input\_text = stemp.c\_str();

CopyMemory((PVOID)pBuf, input\_text, (\_tcslen(input\_text) \* sizeof(CHAR\*))); //записує рядок в спільну область пам'яті, який можуть бачити інші користувачі/процеси

cout << "Your text was stored in memory" << endl;

was\_created = false; //позначає що область пам'яті вже була створена

break;

}

case 2:

{

if (was\_created == true) //перевіряє чи область пам'яті існує за допомогою датчика

{

cout << "This mmaping object was created recently, so its empty" << endl;

// область пуста, бо у неї ніхто ще не записував дані

break;

}

MessageBox(NULL, pBuf, TEXT("The content of mapping object is: "), MB\_OK); //виводить рядок, який знаходиться в спільній області пам'яті

break;

}

case 3:

{

cout << "Work was ended. Goodbye" << endl;

UnmapViewOfFile(pBuf); //звільняє спільну область пам'яті

CloseHandle(hMapFile); //видаляємо наше посилання на спільну область пам'яті

exit(0); //вихід з програми

break;

}

}

} while (true);

return 0;

}

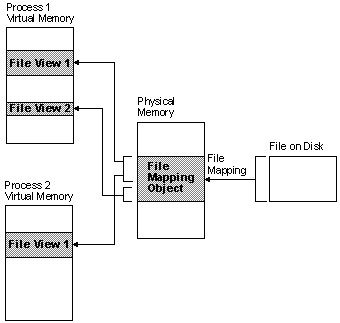
***Пояснення***

Для початку відбувається одночасний запуск двох(або більше) екземплярів програми, що показує кількість потоків для роботи з відображенням файлу. Один із потоків повинен створити нове відображення файлу. Це виконується командою ***CreateFileMapping*(),** або ж ***OpenFileMapping*(),** якщо така область уже існує. Програма використовує рядок формату ***LPCTSTR*** для створення відображення файлу, отже звичайний рядок потрібно перетворити у даний формат. Це виконується перетворенням його у ***wstring*** через кастування, а пізніше у ***LPCTSTR*** командою c\_str(). Даних переходів можна уникнути при використанні багатобайтного кодування замість кодування Юнікоду, що дозволяє відкинути зайві перетворення типів.

Також ми, власне, створюємо вигляд файлу в пам’яті. Це виконується командою ***MapViewOfFile(),*** яка повертає вказівник на віртуальну ділянку пам’яті де буде зберігатися наше відображення файлу і яку ми будемо змінювати. У кожного вигляду файлу є доступ до спільної пам’яті, створеної спочатку, для здійснення операцій зчитування та запису у файл через оперативну пам'ять.

Наступним етапом є запис у вигляд файлу певного рядка вмісту, який ми власне і хочемо передати. Ми використовуємо команду ***CopyMemory(),*** щоб передати вміст бажаного рядка у файл, використовуючи його вигляд файлу, тобто вигляд файлу призначення. Відповідно, ми змінюємо вигляд файлу у цьому потоці, і змінюємо відображення файлу у всій програмі загалом.

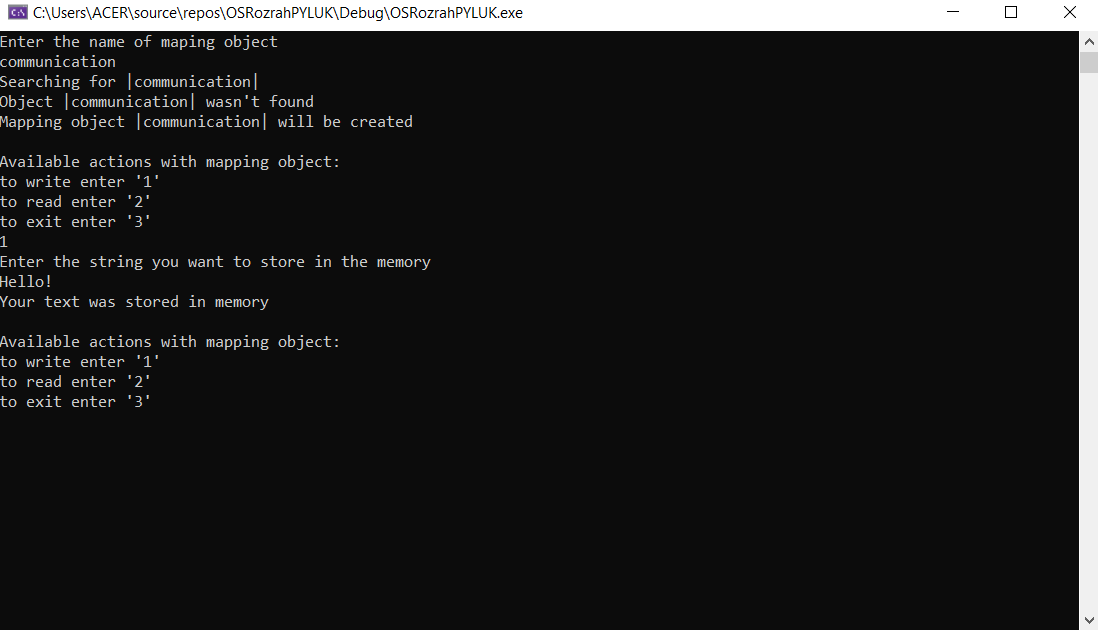
Схожим чином працює зчитування вмісту рядка даним або іншим потоком. Так як через вигляд файлу у попередній дії було змінено відображення файлу в пам’яті, файл уже не є пустим, тобто ми можемо отримати його вміст через вигляд файлу поточного потоку. Це досить яскраво продемонстровано на наступному зображенні:



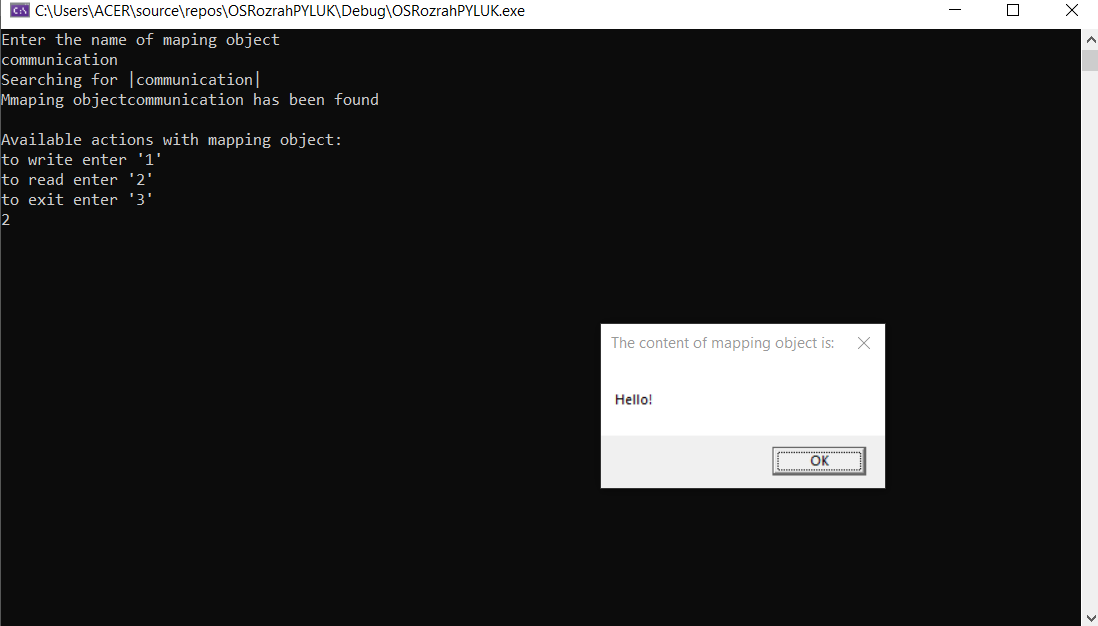
Послідовність даних дій можна нескінченно довго повторювати, записуючи все нові та нові значення у відображення файлу, тим самим роблячи їх доступними для усіх потоків даної програми.

***Результати***

***Приклад роботи 1 потоку***

****

***Приклад роботи 2 потоку***

****

***Висновок:*** Під час виконання даної розрахункової роботи, я значно розширив знання у відображенні файлів через оперативну пам’ять та паралельному доступі до даних всередині одної програми зі сторони багатьох потоків та декількох програм. Це допомогло краще зрозуміти принципи таких дій і знайти пояснення на запитання, що виникли під час схожого завдання раніше.

***Список використаних джерел:***

1. https://docs.microsoft.com/uk-ua/windows/win32/memory/file-mapping
2. https://docs.microsoft.com/uk-ua/windows/win32/memory/creating-a-file-mapping-object
3. https://docs.microsoft.com/uk-ua/windows/win32/memory/creating-a-file-view
4. https://docs.microsoft.com/uk-ua/windows/win32/memory/sharing-files-and-memory
5. [https://docs.microsoft.com/uk-ua/windows/win32/memory/sharing-files-and-memory](https://docs.microsoft.com/en-us/windows/win32/ipc/pipe-functions)
6. https://docs.microsoft.com/uk-ua/windows/win32/memory/using-file-mapping
7. https://docs.microsoft.com/uk-ua/windows/win32/memory/creating-named-shared-memory
8. https://en.wikipedia.org/wiki/Read-write\_memory