# МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

**Національний університет “Львівська політехніка”**



**Інститут післядипломної освіти**

**ЗВІТ**

**Про виконання лабораторної роботи №6**

**«Файли, що відображаються в пам'ять в ОС Windows»**

**з дисципліни «Операційні системи»**

Виконав:

слухач групи ПЗС-11

Гринчук Тарас

Прийняв:

доц. Яковина В.С.

« »\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2014 р.

∑ \_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_

ЛЬВІВ – 2014

**Тема роботи**: Файли, що відображаються в пам'ять в ОС Windows.

**Мета роботи**: Ознайомитися з відображенням файлів в оперативну пам’ять в ОС Windows. Навчитися реалізовувати відображення файлів в оперативну пам’ять.

## 1. Теоретичні відомості

Операції з файлами - це те, що рано чи пізно доводиться робити практично у всіх програмах, і завжди це викликає масу проблем. Чи повинен додаток просто відкрити файл, зробити розрахунок і закрити його, або відкрити, скопіювати фрагмент в буфер і перезаписати його в іншу частину файлу? У Windows багато з цих проблем вирішуються дуже витончено - за допомогою проектованих в пам'ять файлів (memory - mapped files).

Як і віртуальна пам'ять, проектовані файли дозволяють резервувати регіон адресного простору і передавати йому фізичну пам'ять. Різниця між цими механізмами полягає в тому, що в останньому випадку фізична пам'ять не виділяється зі сторінкового файлу, а береться з файлу, який вже знаходиться на диску. Як тільки файл спроектовано в пам'ять, до нього можна звертатися так, ніби він цілком в неї завантажений.

Проектовані в пам’ять файли застосовуються для:

* Завантаження і виконання EXE - і DLL – файлів. Це дозволяє істотно економити як на розмірі сторінкового файлу, так і на часі, необхідному для підготовки додатка до виконання;
* Доступу до файлу даних, розміщеному на диску. Це дозволяє обійтися без операцій файлового вводу-виводу і буферизації його вмісту;
* Розділення даних між декількома процесами, виконуваними на одній машині (в Windows є й інші методи для спільного доступу різних процесів до спільних даних - але всі вони так чи інакше реалізовані на основі проектованих в пам'ять файлів).

Операційна система дозволяє проектувати на адресний простір процесу файл даних. Це дуже зручно при маніпуляціях з великими потоками даних.

Для використання проектованих в пам'ять файлів потрібно виконати три операції:

1. Створити або відкрити об'єкт ядра "файл", що ідентифікує дисковий файл, який Ви хочете використовувати як проектований в пам'ять.
2. Створити об'єкт ядра "проекція файлу", щоб повідомити системі розмір файла і спосіб доступу до нього.
3. Вказати системі, як спроектувати в адресний простір Вашого процесу об'єкт «проекція файлу» - цілком або частково.

Закінчивши роботу з проектованим в пам'ять файлом, слід виконати теж три операції:

1. Повідомити системі про скасування проектування адресного простору процесу об'єкта ядра "проекція файлу".
2. Закрити цей об'єкт .
3. Закрити об'єкт ядра "файл".

Розглянемо змістовну частину програми, що здійснює проектування файлу в пам'ять і роботу з його проекцією. Будемо вважати , що файл з ім'ям 1.dat складається з 1000 чисел типу int (і займає на диску простір 4000 байт).

*/ \* Операція відкриття існуючого файлу \* /*

*hFile = CreateFile ( " 1.dat " , GENERIC\_READ ,*

*0 , NULL , OPEN\_EXISTING , 0 , NULL) ;*

*/ \* Створення в пам'яті об'єкта "проекція файлу" \* /*

*HANDLE hMem = CreateFileMapping*

*( hFile , NULL , PAGE\_READONLY , 0,0 , NULL) ;*

*/ \* Відображення лінійних адрес на проекцію файлу \* /*

*int \* ptr = ( int \* ) MapViewOfFile ( hMem , FILE\_MAP\_READ , 0,0,0 ) ;*

*/ \* Читання ділянок файлу \* /*

*int x = ptr [ 0 ] ;/ / Читання першого елемента з файлу*

*int y = ptr [ 999 ] ;/ / Читання останнього елемента з файлу*

У наведеному прикладі файл був відкритий тільки для читання (параметр *GENERIC\_READ* функції *CreateFile ( )*) . Відповідно, такий же атрибут отримала проекція файлу у фізичній пам'яті (параметр *PAGE\_READONLY*), а також і виділена область лінійних адрес (параметр *FILE\_MAP\_READ*) . Якщо в проектується файл передбачається виконувати запис нових даних, тоді для відповідних параметрів викликаються функцій слід вказати значення *GENERIC\_READ | GENERIC\_WRITE*, *PAGE\_READWRITE* і *FILE\_MAP\_ALL\_ACCESS*.

Файл з даними може складатися з даних будь-якого типу. Тому функція *MapViewOfFile()*, яка повинна виконувати свою роботу для будь-яких типів даних, повертає у відповідності зі своїм прототипом узагальнений вказівник типу LPVOID. При використанні повертається цією функцією значення його необхідно перетворити у вказівник на конкретний тип даних, що містяться у файлі. У нашому випадку файл заповнений числами типу *int* , і повертається значення явним чином перетвориться в тип *int*\*. Якби файл був заповнений, наприклад, байтовими змінними типу char або BYTE, то і вказівник на проекцію файлу слід було отримати такого ж типу:

*BYTE \* ptr = ( BYTE \* ) MapViewOfFile (...) ;*

Четвертий і п'ятий параметри функції CreateFileMapping( ) дозволяють задати розмір проекції файлу. Вказівка ​​в якості параметрів нульових значень змушує Windows спроектувати весь файл.

Функція *MapViewOfFile( )* дозволяє при необхідності отримати відображення на лінійні адреси будь-якої частини файлу, для чого слід вказати зсув відображення від початку файлу (третій і четвертий параметр функції), а також розмір відображення (п'ятий параметр). Якщо потрібно спроектувати весь файл (як у нашому прикладі), на місці всіх цих трьох параметрів вказуються нулі.

## 2. Хід роботи

## Завдання (варіант №5).

1. Реалізувати відображення файлу в оперативну пам'ять.
2. Виконати завдання, з використанням цього відображення. Створити масив N елементів і відсортувати його елементи у порядку зростання за допомогою методу «вибірки». Елементи масиву згенерувати випадковим чином за допомогою вбудованих функцій. Результуючий масив вивести у зовнішній файл через відображення цього файлу в оперативній пам’яті.
3. Результати виконання роботи відобразити у звіті.
4. Відображаємо файл в оперативній пам'ять у функції *main()* нашої програми наступним чином:

*int main() {*

*......................*

*HANDLE hFile, hMap; //дескрипотори файлу та його проекції*

*LPBYTE pData = NULL; //вказівник на початок файлу*

*//Відкриваємо файл, перезаписуючи його вміст*

*hFile = CreateFile(L"d:\\MMFTest.txt", GENERIC\_WRITE | GENERIC\_READ, 0, NULL, CREATE\_ALWAYS, FILE\_ATTRIBUTE\_NORMAL, NULL);*

*if (hFile != INVALID\_HANDLE\_VALUE) {*

*//Створюємо проекцію всього файлу*

*hMap = CreateFileMapping(hFile, NULL, PAGE\_READWRITE, 0, size, NULL);*

*if (hMap) {*

*//Отримуємо його розмір та закриваємо дескриптор*

*DWORD dwSize = GetFileSize(hFile, NULL);*

*CloseHandle(hFile);*

*//Проектуємо весь файл в адресний простір*

*pData = (LPBYTE)MapViewOfFile(hMap, FILE\_MAP\_WRITE, 0, 0, 0);*

*if (pData) {*

*......................*

*//запис даних у файл*

*......................*

*}*

*//Закриваємо проекцію файла у адресному просторі*

*UnmapViewOfFile(pData);*

*}*

*//Закриваємо дескриптор проекції файла*

*CloseHandle(hMap);*

*}*

*......................*

*return 0;*

*}*

1. У функції *sort(…) -* буде виконуватись алгоритм сортування, що має вигляд:

*//функція сортування*

*void sort(int arr[], int n) {*

*for (int i = 0; i < n; i++) {*

*// Знайдемо мінімальний елемент на*

*// проміжку індексів [i; n);*

*// спочатку його індекс рівний i*

*int minValueIndex = i;*

*// Перебираємо елементи, що залишилися на проміжку*

*for (int j = i + 1; j < n; j++) {*

*// Якщо елемент в позиції j менший*

*// елемента в позиції minValueIndex, то*

*// необхідно обновити значення індекса*

*if (arr[j] < arr[minValueIndex]) {*

*minValueIndex = j;*

*}*

*}*

*// Міняємо поточний елемент з мінімальним*

*int temp = arr[i];*

*arr[i] = arr[minValueIndex];*

*arr[minValueIndex] = temp;*

*}*

*}*

Початкове заповнення вхідного масиву випадковими числами та розрахунок розміру файлу в байтах, виглядає так:

*int main() {*

*......................*

*//заповнимо вхідний масив випадковими числами 0 <= a[i] < N\*10*

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

*int size = 0; //розрахуємо розмір файлу в байтах*

*for(int i = 0; i < N; i++) {*

*a[i] = rand()%(N\*10);*

*//розіб'ємо число на масив char[] і розрахуємо к-сть символів*

*//p - вказівник на першу цифру*

*for(p = itoa(a[i], foo, 10); \*p; p++, size++);*

*}*

*size += N; //потрібно N байт для '\t' після кожного числа*

*......................*

*return 0;*

*}*

1. Кінцевий текст програми міститься в розділі 3, а результати виконання див. розділ 4.

## Текст програми

#include <iostream>

#include <cstdlib>

#include <stdio.h>

#include <conio.h>

#include <math.h>

#include <time.h>

#include <windows.h>

using namespace std;

//==============================================================

//глобальні константи

const int N = 30; //к-сть елементів масиву

int a[N]; //вхідний масив

//==============================================================

//вивід одновимірного масиву на екран

void show(int arr[], int n) {

for(int i = 0; i < n; i++)

printf("%d\t", arr[i]);

}

//==============================================================

//функція сортування

void sort(int arr[], int n) {

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

// Знайдемо мінімальний елемент на

// проміжку індексів [i; n);

// спочатку його індекс рівний i

int minValueIndex = i;

// Перебираємо елементи, що залишилися на проміжку

for (int j = i + 1; j < n; j++) {

// Якщо елемент в позиції j менший

// елемента в позиції minValueIndex, то

// необхідно обновити значення індекса

if (arr[j] < arr[minValueIndex]) {

minValueIndex = j;

}

}

// Міняємо поточний елемент з мінімальним

int temp = arr[i];

arr[i] = arr[minValueIndex];

arr[minValueIndex] = temp;

}

}

//==============================================================

int main() {

char \*p; //вказівник на результат ф-ї itoa()

char foo[2];

setlocale(LC\_ALL,"Ukrainian");

//заповнимо вхідний масив випадковими числами 0 <= a[i] < N\*10

srand((unsigned)time(NULL));

int size = 0; //розрахуємо розмір файлу в байтах

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

a[i] = rand()%(N\*10);

//розіб'ємо число на масив char[] і розрахуємо к-сть символів

//p - вказівник на першу цифру

for(p = itoa(a[i], foo, 10); \*p; p++, size++);

}

size += N; //потрібно N байт для '\t' після кожного числа

printf("Масив a[%d] до сортування:\n", N);

show(a, N);

//сортуємо масив

sort(a, N);

HANDLE hFile, hMap; //дескрипотори файлу та його проекції

LPBYTE pData = NULL; //вказівник на початок файлу

//Відкриваємо файл, перезаписуючи його вміст

hFile = CreateFile(L"d:\\MMFTest.txt", GENERIC\_WRITE | GENERIC\_READ, 0, NULL, CREATE\_ALWAYS, FILE\_ATTRIBUTE\_NORMAL, NULL);

if (hFile != INVALID\_HANDLE\_VALUE) {

//Створюємо проекцію всього файлу

hMap = CreateFileMapping(hFile, NULL, PAGE\_READWRITE, 0, size, NULL);

if (hMap) {

//Отримуємо його розмір та закриваємо дескриптор

DWORD dwSize = GetFileSize(hFile, NULL);

CloseHandle(hFile);

//Проектуємо весь файл в адресний простір

pData = (LPBYTE)MapViewOfFile(hMap, FILE\_MAP\_WRITE, 0, 0, 0);

if (pData) {

//зміщення в байтах від першого запису файла

int offset = 0;

for(int i = 0; i < N; i++, offset++) { //зміщуємо для '\t' після кожного числа

for(p = itoa(a[i], foo, 10); \*p; p++, offset++)

\*(pData+offset) = \*p; //записуємо чергову цифру числа у файл

\*(pData+offset) = '\t'; //записуємо розділювач між числами у файл

}

//Закриваємо проекцію файла у адресному просторі

UnmapViewOfFile(pData);

}

//Закриваємо дескриптор проекції файла

CloseHandle(hMap);

}

}

printf("\nРезультуючий масив знаходиться у файлi: d:\\MMFTest.txt");

\_getch();

return 0;

}

## Результат виконання програми

Запустимо програму на виконання з ***N = 30*** (рис. 4.1).

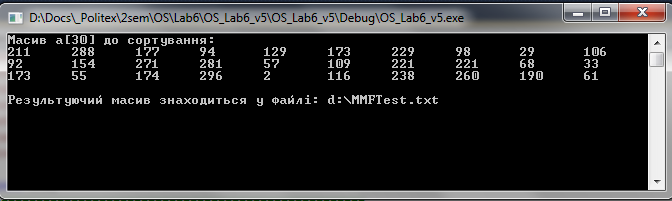


Рис. 4.1. Результат запуску програми

Відкриємо файл ***MMFTest.txt*** у блокноті та перевіримо результат (рис. 4.2).

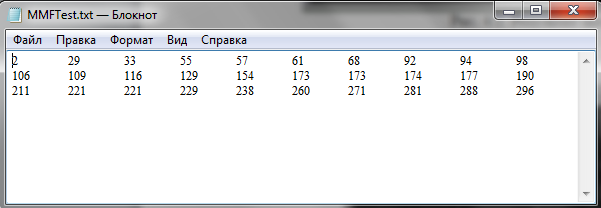


Рис. 4.2. Результуючий файл *MMFTest.txt*

Як бачимо, вихідний масив відсортований правильно.

## ВИСНОВКИ

На даній лабораторній роботі я ознайомився з відображенням файлів в оперативну пам’ять в ОС Windows. Навчився реалізовувати відображення файлів в оперативну пам’ять, зокрема: створювати і відкривати об'єкт ядра "файл", що ідентифікує дисковий файл, створювати об'єкт ядра "проекція файлу", щоб повідомити системі розмір файла і спосіб доступу до нього, вказувати системі, як спроектувати в адресний простір нашого процесу об'єкта «проекція файлу» - цілком або частково.