

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

**Український державний університет науки і технологій**

Кафедра «Комп’ютерні інформаційні технології»

**Лабораторна робота №4**

**з дисципліни «Операційні системи»**

**на тему:**

**«Проекція файлу. Завдання синхронізації.»**

Виконав:

Студент гр. ПЗ2011

Кулик С. В.

Прийняв:

Андрющенко В. О.

Дніпро, 2022

**Тема.** Проекція файлу. Завдання синхронізації.

**Теоретичні відомості**

Як і віртуальна пам'ять, файли, що проектуються, 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 з сторінкового файлу, а береться з файлу, який вже знаходиться на диску. Як тільки файл відображений в пам'ять, до нього можна звертатися так, ніби він цілком в неї завантажений. Проекції файлів 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 програми до виконання,

• доступу до файлу даних, розміщених на диску. Це дозволяє обійтися без операцій файлового введення-виведення і буферизації його вмісту,

• поділу даних між декількома процесами, виконуваними па 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).

У Windows багато механізмів, що дозволяють додаткам легко і швидко розділяти будь-які дані. Самий низькорівневий механізм спільного використання даних на одній машині - проектування файлу в пам'ять. Спільне використання даних в цьому випадку відбувається так: два або більше процесів проектують в пам'ять уявлення одного і того ж об'єкта «проекція файлу», тобто ділять одні й ті ж сторінки фізичної пам'яті. В результаті, коли один процес записує дані в уявлення загального об'єкта «проекція файлу», зміни негайно відображаються на уявленнях в інших процесах. Але при цьому всі процеси повинні використовувати однакове ім'я об'єкта «проекція файлу».

**Текст програми**

**Файл Main.cpp**

#include <Windows.h>

#include <iostream>

#include <thread>

const int BUFFER\_SIZE = 10;

HANDLE hSemaphore;

LPVOID lpMapAddress;

HANDLE hFile;

DWORD WINAPI Producer(LPVOID lpParam)

{

while (true)

{

// Wait for the semaphore to be available

WaitForSingleObject(hSemaphore, INFINITE);

// Write data to the shared buffer

DWORD bytesWritten;

int data = rand();

std::cout << "Prodused: " << data << std::endl;

LARGE\_INTEGER li;

li.QuadPart = 0; // Set the file pointer to the beginning of the file

// Move the file pointer to the desired position

SetFilePointerEx(hFile, li, NULL, FILE\_BEGIN);

WriteFile(hFile, &data, sizeof(int), &bytesWritten, NULL);

// Release the semaphore

ReleaseSemaphore(hSemaphore, 1, NULL);

Sleep(1000);

}

return 0;

}

DWORD WINAPI Consumer(LPVOID lpParam)

{

while (true)

{

// Wait for the semaphore to be available

WaitForSingleObject(hSemaphore, INFINITE);

// Read data from the shared buffer

DWORD bytesRead;

int data;

LARGE\_INTEGER li;

li.QuadPart = 0; // Set the file pointer to the beginning of the file

// Move the file pointer to the desired position

SetFilePointerEx(hFile, li, NULL, FILE\_BEGIN);

ReadFile(hFile, &data, sizeof(int), &bytesRead, NULL);

std::cout << "Consumed: " << data << std::endl;

// Release the semaphore

ReleaseSemaphore(hSemaphore, 1, NULL);

Sleep(1000);

}

return 0;

}

int main()

{

// Create a file for the shared buffer

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

// Create a file mapping object for the shared buffer

HANDLE hFileMapping = CreateFileMapping(

hFile,

NULL,

PAGE\_READWRITE,

0,

BUFFER\_SIZE \* sizeof(int),

NULL);

if (hFileMapping == NULL)

{

std::cout << "CreateFileMapping failed, error code: " << GetLastError() << std::endl;

return 1;

}

// Map the shared buffer into the current process's virtual address space

lpMapAddress = MapViewOfFile(

hFileMapping,

FILE\_MAP\_ALL\_ACCESS,

0,

0,

BUFFER\_SIZE \* sizeof(int));

if (lpMapAddress == NULL)

{

std::cout << "MapViewOfFile failed, error code: " << GetLastError() << std::endl;

CloseHandle(hFileMapping);

CloseHandle(hFile);

return 1;

}

// Create a semaphore to synchronize access to the shared buffer

hSemaphore = CreateSemaphore(

NULL,

1, // Initial count

1, // Maximum count

NULL);

if (hSemaphore == NULL)

{

std::cout << "CreateSemaphore failed, error code: " << GetLastError() << std::endl;

UnmapViewOfFile(lpMapAddress);

CloseHandle(hFileMapping);

CloseHandle(hFile);

return 1;

}

// Create a producer and a consumer thread

HANDLE hProducer = CreateThread(NULL, 0, Producer, NULL, 0, NULL);

HANDLE hConsumer = CreateThread(NULL, 0, Consumer, NULL, 0, NULL);

// Wait for the threads to finish

WaitForSingleObject(hProducer, INFINITE);

WaitForSingleObject(hConsumer, INFINITE);

// Clean up

CloseHandle(hProducer);

CloseHandle(hConsumer);

CloseHandle(hSemaphore);

UnmapViewOfFile(lpMapAddress);

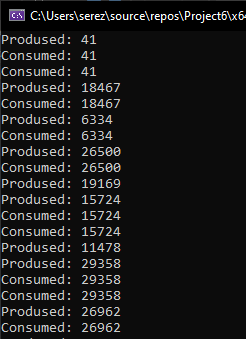
CloseHandle(hFileMapping);

CloseHandle(hFile);

return 0;

}

**Результат роботи програми**

****

**Висновок**

Відображення файлів і файловий ввід-вивід із відображенням пам’яті — це методи, які використовуються в C++ для відображення файлу або частини файлу у віртуальний адресний простір процесу. Це дозволяє процесу отримувати доступ до вмісту файлу так, ніби він є пам’яттю, забезпечуючи ефективніші та зручніші операції введення/виведення файлів.

Підсумовуючи, відображення файлів і файловий ввід-вивід із відображенням пам’яті є потужними методами, які можна використовувати для підвищення продуктивності та ефективності операцій файлового введення-виведення в C++. Функції CreateFileMapping, MapViewOfFile і UnmapViewOfFile забезпечують засоби відображення файлу або частини файлу у віртуальний адресний простір процесу, дозволяючи процесу отримувати доступ до вмісту файлу, як до пам’яті. Семафори можна використовувати для синхронізації доступу до спільного буфера, гарантуючи, що тільки один потік може отримати доступ до буфера одночасно.