Правительство Санкт-Петербурга

Комитет по науке и высшей школе

Санкт-Петербургское государственное бюджетное

профессиональное образовательное учреждение

«Политехнический колледж городского хозяйства»

**Отчет**

**МДК 01.04 Системное программирование**

**Тема: Отображение файлов в память**

**Специальность: 09.02.07 Информационные системы и программирование**

Группа ИП-22-3

Студент Фролов Дмитрий Сергеевич

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_г.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /Калиниченко Т. М./

Санкт-Петербург

2024 г

**Листинг 30.1 - Работа с видом файла**

#include <windows.h>

#include <fstream>

#include <iostream>

using namespace std;

int main()

{

int a[10] = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 };

wchar\_t file\_name[] = L"Demo.bin";

HANDLE hFile, hMapping; // дескрипторы файла и объекта отображения

int\* ptr; // для указателя на массив

// открываем файл для вывода

ofstream out(file\_name, ios::out | ios::binary);

if (!out)

{

cerr << "File constructor failed." << endl;

return 0;

}

// выводим исходный массив в файл и на консоль

cout << "Initial array: ";

for (int i = 0; i < 10; ++i)

{

out.write((char\*)&a[i], sizeof(int));

cout << a[i] << ' ';

}

cout << endl;

// закрываем выходной файл

out.close();

// ----------------------------------------------------------------------

// открываем файл для отображения в память

hFile = CreateFile(file\_name, GENERIC\_READ | GENERIC\_WRITE,

0, NULL, OPEN\_EXISTING, FILE\_ATTRIBUTE\_NORMAL, NULL);

if (hFile == INVALID\_HANDLE\_VALUE)

{

cerr << "Create file failed." << endl;

return GetLastError();

}

// открываем объект, отображающий файл в память

hMapping = CreateFileMapping(

hFile, // дескриптор открытого файла

NULL, // атрибуты защиты по умолчанию

PAGE\_READWRITE, // режим доступа

0, 0, // размер объекта отображения равен размеру файла

NULL); // имя не используем

if (!hMapping)

{

cerr << "Create file mapping failed." << endl;

return GetLastError();

}

// создаем вид файла

ptr = (int\*)MapViewOfFile(

hMapping, // дескриптор объекта отображения

FILE\_MAP\_WRITE, // режим доступа к виду

0, 0, // отображаем файл с начала

0); // отображаем весь файл

// изменяем значения элементов массива

for (int i = 0; i < 10; ++i)

ptr[i] += 10;

// отменяем отображение файла в память

if (!UnmapViewOfFile(ptr))

{

cerr << "Unmap view of file failed." << endl;

return GetLastError();

}

// закрываем объект отображения файла в память

if (!CloseHandle(hMapping))

{

cerr << "Close file failed." << endl;

return GetLastError();

}

// закрываем файл

if (!CloseHandle(hFile))

{

cerr << "Close file failed." << endl;

return GetLastError();

}

// ----------------------------------------------------------------------

// открываем файл для ввода

ifstream in(file\_name, ios::in | ios::binary);

if (!in)

{

cerr << "File constructor failed." << endl;

return 0;

}

// вводим финальный массив из файла и выводим на консоль

cout << "Final array: ";

for (int i = 0; i < 10; ++i)

{

in.read((char\*)&a[i], sizeof(int));

cout << a[i] << ' ';

}

cout << endl;

// закрываем входной файл

in.close();

return 0;

}

**Результат**

Изображение выглядит как текст, программное обеспечение, Шрифт, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

**Вывод**

В листинге 30.1 приведена программа, которая отображает файл в память. Предварительно создадим файл, который представляет собой массив целых чисел. После этого увеличим значения элементов этого массива на десять, используя вид файла. Затем выведем содержимое измененного файла на экран.

**Листинги 30.2 и 30.3 - Передача данных через отображаемый в память файл**

**30.2**

#include <windows.h>

#include <fstream>

#include <iostream>

using namespace std;

int main()

{

wchar\_t MappingName[] = L"MappingName";

HANDLE hMapping; // дескриптор объекта, отображающего файл

int\* ptr; // для указателя на массив

const int n = 10; // размерность массива

cout << "This is a parent process." << endl;

// открываем объект отображения файла в память

hMapping = CreateFileMapping(

INVALID\_HANDLE\_VALUE, // файл подкачки страниц

NULL, // атрибуты защиты по умолчанию

PAGE\_READWRITE, // режим доступа: чтение и запись

0, // старшее слово = 0

n \* sizeof(int), // младшее слово = длине массива

MappingName); // имя объекта отображения

if (!hMapping)

{

cerr << "Create file mapping failed." << endl;

return GetLastError();

}

// создаем вид файла

ptr = (int\*)MapViewOfFile(

hMapping, // дескриптор объекта отображения

FILE\_MAP\_WRITE, // режим доступа к виду

0, 0, // отображаем файл с начала

0); // отображаем весь файл

// инициализируем массив и выводим его на консоль

cout << "Array: ";

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

{

ptr[i] = i;

cout << ptr[i] << ' ';

}

cout << endl;

//----------------------------------------------------------------------

// создаем процесс, который будет читать данные из отображаемого

// в память файла

wchar\_t lpszAppName[] = L"D:\\listing30\_3\\x64\\Debug\\listing30\_3.exe";

STARTUPINFO si;

PROCESS\_INFORMATION piApp;

ZeroMemory(&si, sizeof(STARTUPINFO));

si.cb = sizeof(STARTUPINFO);

// создаем новый консольный процесс

if (!CreateProcess(lpszAppName, NULL, NULL, NULL, FALSE,

CREATE\_NEW\_CONSOLE, NULL, NULL, &si, &piApp))

{

cerr << "Create process failed." << endl;

return GetLastError();

}

// ждем завершения созданного прцесса

WaitForSingleObject(piApp.hProcess, INFINITE);

// закрываем дескрипторы этого процесса в текущем процессе

CloseHandle(piApp.hThread);

CloseHandle(piApp.hProcess);

//----------------------------------------------------------------------

// отменяем отображение файла в память

if (!UnmapViewOfFile(ptr))

{

cerr << "Unmap view of file failed." << endl;

return GetLastError();

}

// закрываем объект отображения файла в память

if (!CloseHandle(hMapping))

{

cerr << "Close file failed." << endl;

return GetLastError();

}

// ждем команду на завершение процесса

char c;

cout << "Input any char to exit: ";

cin >> c;

return 0;

}

**30.3**

#include <windows.h>

#include <fstream>

#include <iostream>

using namespace std;

int main()

{

wchar\_t MappingName[] = L"MappingName";

HANDLE hMapping; // дескриптор объекта, отображающего файл

int\* ptr; // для указателя на массив

const int n = 10; // размерность массива

cout << "This is a child process." << endl;

// открываем объект отображения файла в память

hMapping = CreateFileMapping(

INVALID\_HANDLE\_VALUE, // файл подкачки страниц

NULL, // атрибуты защиты по умолчанию

PAGE\_READWRITE, // режим доступа: чтение и запись

0, // старшее слово = 0

n \* sizeof(int), // младшее слово = длине массива

MappingName); // имя объекта отображения

if (!hMapping)

{

cerr << "Create file mapping failed." << endl;

return GetLastError();

}

// создаем вид файла

ptr = (int\*)MapViewOfFile(

hMapping, // дескриптор объекта отображения

FILE\_MAP\_WRITE, // режим доступа к виду

0, 0, // отображаем файл с начала

0); // отображаем весь файл

// выводим массив из вида на консоль

cout << "Array: ";

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

cout << ptr[i] << ' ';

cout << endl;

// отменяем отображение файла в память

if (!UnmapViewOfFile(ptr))

{

cerr << "Unmap view of file failed." << endl;

return GetLastError();

}

// закрываем объект отображения файла в память

if (!CloseHandle(hMapping))

{

cerr << "Close file failed." << endl;

return GetLastError();

}

// ждем команду на завершение процесса

char c;

cout << "Input any char to exit: ";

cin >> c;

return 0;

}

**Результат**

**Изображение выглядит как текст, Шрифт, линия, число

Контент, сгенерированный ИИ, может содержать ошибки.**

**Вывод**

В программах из листингов 30.2, 30.3 показывается, как передать данные другому процессу через файл подкачки страниц. Первая программа создает объект, отображающий файл подкачки, а затем записывает в вид файла подкачки массив целых чисел. После этого она запускает вторую программу, которая будет читать созданный массив, через отображаемый в память файл, используя тот же объект, отображающий файл в память. Отметим, что для обращения к одному и тому же объекту, отображающему файл, используется имя этого объекта.

**Листинги 30.4 и 30.5 - Сброс вида в файл на диске**

**30.4**

#include <windows.h>

#include <fstream>

#include <iostream>

using namespace std;

int main()

{

int a[10] = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 };

wchar\_t file\_name[] = L"D:\\Demo.bin";

wchar\_t mapping\_name[] = L"MappingName";

HANDLE hFile, hMapping; // дескрипторы файла и объекта отображения

int\* ptr; // для указателя на массив

// открываем файл для вывода

ofstream out(file\_name, ios::out | ios::binary);

if (!out)

{

cerr << "File constructor failed." << endl;

return 0;

}

// выводим исходный массив в файл и на консоль

cout << "Initial array: ";

for (int i = 0; i < 10; ++i)

{

out.write((char\*)&a[i], sizeof(int));

cout << a[i] << ' ';

}

cout << endl;

// закрываем выходной файл

out.close();

//----------------------------------------------------------------------

// открываем файл для отображения в память

hFile = CreateFile(file\_name, GENERIC\_READ | GENERIC\_WRITE,

FILE\_SHARE\_READ | FILE\_SHARE\_WRITE,

NULL, OPEN\_EXISTING, FILE\_ATTRIBUTE\_NORMAL, NULL);

if (hFile == INVALID\_HANDLE\_VALUE)

{

cerr << "Create file failed." << endl;

return GetLastError();

}

// открываем объект, отображающий файл в память

hMapping = CreateFileMapping(

hFile, // дескриптор открытого файла

NULL, // атрибуты защиты по умолчанию

PAGE\_READWRITE, // режим доступа

0, 0, // размер объекта отображения равен размеру файла

mapping\_name); // имя объекта отображения

if (!hMapping)

{

cerr << "Create file mapping failed." << endl;

return GetLastError();

}

// создаем вид файла

ptr = (int\*)MapViewOfFile(

hMapping, // дескриптор объекта отображения

FILE\_MAP\_WRITE, // режим доступа к виду

0, 0, // отображаем файл с начала

0); // отображаем весь файл

// изменяем значения элементов массива

for (int i = 0; i < 10; ++i)

ptr[i] += 10;

// сбрасываем весь вид на диск

if (!FlushViewOfFile(ptr, 0))

{

cerr << "Flush view of file failed." << endl;

return GetLastError();

}

//----------------------------------------------------------------------

// создаем процесс, который будет читать данные из отображаемого

// в память файла

wchar\_t lpszAppName[] = L"D:\\listing30\_5\\x64\\Debug\\listing30\_5.exe";

STARTUPINFO si;

PROCESS\_INFORMATION piApp;

ZeroMemory(&si, sizeof(STARTUPINFO));

si.cb = sizeof(STARTUPINFO);

// создаем новый консольный процесс

if (!CreateProcess(lpszAppName, NULL, NULL, NULL, FALSE,

CREATE\_NEW\_CONSOLE, NULL, NULL, &si, &piApp))

{

cerr << "Create process failed." << endl;

return GetLastError();

}

// ждем завершения созданного прцесса

WaitForSingleObject(piApp.hProcess, INFINITE);

// закрываем дескрипторы этого процесса в текущем процессе

CloseHandle(piApp.hThread);

CloseHandle(piApp.hProcess);

//----------------------------------------------------------------------

// отменяем отображение файла в память

if (!UnmapViewOfFile(ptr))

{

cerr << "Unmap view of file failed." << endl;

return GetLastError();

}

// закрываем объкт отображения файла в память

if (!CloseHandle(hMapping))

{

cerr << "Close file failed." << endl;

return GetLastError();

}

// закрываем файл

if (!CloseHandle(hFile))

{

cerr << "Close file failed." << endl;

return GetLastError();

}

// ждем команду на завершение процесса

char c;

cout << "Input any char to exit: ";

cin >> c;

return 0;

}

**30.5**

#include <windows.h>

#include <fstream>

#include <iostream>

using namespace std;

int main()

{

char file\_name[] = "D:\\Demo.bin";

int a[10];

// открываем файл для ввода

ifstream in(file\_name, ios::in | ios::binary);

if (!in)

{

cerr << "File constructor failed." << endl;

return 0;

}

// вводим финальный массив из файла и выводим на консоль

cout << "Final array: ";

for (int i = 0; i < 10; ++i)

{

in.read((char\*)&a[i], sizeof(int));

cout << a[i] << ' ';

}

cout << endl;

// закрываем входной файл

in.close();

// ждем команду на завершение процесса

char c;

cout << "Input any char to exit: ";

cin >> c;

return 0;

}

**Результат**

**Изображение выглядит как текст, Шрифт, программное обеспечение, линия

Контент, сгенерированный ИИ, может содержать ошибки.**

**Созданный файл**

**Изображение выглядит как снимок экрана, текст, программное обеспечение, линия

Контент, сгенерированный ИИ, может содержать ошибки.**

**Вывод**

В листингах 30.4, 30.5, приведены две программы, которые иллюстрируют использование функции FlushViewOfFile. Первая программа создает файл, затем отображает его в память и изменяет его содержимое. После этого запускается вторая программа, которая читает содержимое отображенного в память файла. Чтобы вторая программа прочитала измененный файл, вид перед ее запуском сбрасывается на диск при помощи вызова функции FlushViewOfFile.