Министерство образования и науки РФ

Севастопольский государственный университет

Отчёт

по лабораторной №5

**«Исследование возможностей управления памятью и обмена данными между процессами в ОС Windows»**

Выполнил:

ст.гр.ИСб-22д

Воронин И.Ю.

Проверил:

Забаштанский А.К.

Севастополь

2015

1.Цель работы

Изучить возможности программного интерфейса приложений (API) операционных систем Windows по управления памятью и обмена данными между процессами. Приобрести практические навыки использования Win API для управления памятью и обмена данными между процессами.

2.Вариант задания

Написать программу MemShare, выполняющую следующие действия:

Создание(запуск) процесса MemSort(действия, выполняемые этим процессом описаны в п. 3.2), используя вызов CreateProcess;

Выделение с помощью вызова VirtualAlloc заданного по варианту задания количества страниц памяти (страница 4096 байт);

Заполнение выделенной памяти случайными числами в диапазоне от 0 до MAX (значение MAX выбирается исходя из типа данных, заданному по варианту задания);

Перевод режима доступа выделенной памяти в ReadOnly (с помощью вызова VirtualProtect);

Используя механизм отобра­жения файлов на память (функции CreateFileMapping и MapViewOfFile), создание памяти общего доступа, с размером соответствующим размеру выделенной с помощью VirtualAlloc памяти (задать имя создаваемого объекта отображения "memshare");

Копирование данных из памяти, выделенной с помощью VirtualAlloc, в память общего доступа (например, с помощью вызова CopyMemory);

Ожидание процесса MemSort (используя заданный по варианту задания объект синхронизации необходимо ожидать пока процесс MemSort не подготовит данные);

Перевод режима доступа выделенной памяти в ReadWrite(с помощью вызова VirtualProtect);

Копирование данных из памяти общего доступа в память, выделенную с помощью VirtualAlloc (например, с помощью вызова CopyMemory);

Вывод на экран данных из памяти, выделенной с помощью VirtualAlloc;

Освобождение выделенной памяти (с помощью вызова VirtualFree);

Освободить память общего доступа (используя вызовы UnmapViewOfFile и СloseHandle);

Написать программу MemSort выполняющую следующие действия:

* Ожидание пока процесс MemShare не подготовит данные в памяти общего доступа (использовать заданный по варианту задания тип объекта синхронизации);
* Открытие памяти общего доступа "memshare" (используя вызовы OpenFileMapping и MapViewOfFile);
* Сортировка данных в памяти общего доступа, методом указанным в варианте задания.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № варианта | Количество страниц памяти | Тип данных для заполнения памяти | Тип сортировки | Объект синхронизации |
| 4 | 4 | int | Быстрая | Мютекс |

3.Текст программы

**Main:**

#include <iostream>

#include <windows.h>

void mass\_print(int \*arr, int n){

std::cout << std::endl;

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

std::cout << arr[i] << ' ';

}

std::cout << std::endl;

}

const int MAX = 10;

using namespace std;

int main(){

STARTUPINFO info;

memset(&info, 0, sizeof(info));

info.cb = sizeof(info);

PROCESS\_INFORMATION pinfo;

HANDLE mtx = CreateMutex(NULL,FALSE,"mtx");

//WaitForSingleObject(mtx,INFINITE);

CreateProcess(NULL, "OS5.exe", NULL, NULL, FALSE,

CREATE\_NEW\_CONSOLE, NULL, NULL, &info, &pinfo);

int \*arr = (int\*)VirtualAlloc(NULL, sizeof(int)\*MAX, MEM\_COMMIT, PAGE\_READWRITE);

//заполнение массива

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

arr[i] = rand();

}

//печать массива

mass\_print(arr,MAX);

DWORD oldprot;

VirtualProtect(arr,sizeof(int)\*MAX,PAGE\_READONLY,&oldprot);

HANDLE hFileMap = CreateFileMapping(INVALID\_HANDLE\_VALUE, NULL, PAGE\_READWRITE, 0, 4, ("share"));

int \*Map =(int\*) MapViewOfFile(hFileMap, FILE\_MAP\_READ | FILE\_MAP\_WRITE, 0, 0, sizeof(int)\*MAX);

CopyMemory(Map,arr,sizeof(int)\*MAX);

ReleaseMutex(mtx);

WaitForSingleObject(mtx,INFINITE);

VirtualProtect(arr,sizeof(int)\*MAX,PAGE\_READWRITE,&oldprot);

//копирование отсортированного массива

CopyMemory(arr,Map,sizeof(int)\*MAX);

mass\_print(arr,MAX);

ReleaseMutex(mtx);

return 0;

}

**Sort:**

#include <iostream>

#include <windows.h>

#include <algorithm>

const int MAX = 10;

void qs(int\* s\_arr, int first, int last);

void mass\_print(int \*arr, int n);

int main(){

HANDLE mtx = OpenMutex(SYNCHRONIZE, FALSE, "mtx");

WaitForSingleObject(mtx,INFINITE);

int \*arr = (int\*)VirtualAlloc(NULL, MAX\*(sizeof(int)), MEM\_COMMIT, PAGE\_READWRITE);;

HANDLE hFileMap2 = OpenFileMapping(FILE\_MAP\_ALL\_ACCESS, FALSE, "share");

WaitForSingleObject(hFileMap2, INFINITE);

int \* pViewFile2 = (int\*)MapViewOfFile(hFileMap2, FILE\_MAP\_READ | FILE\_MAP\_WRITE, 0, 0, MAX\*(sizeof(int)) );

CopyMemory(arr, pViewFile2, MAX\*sizeof(int));

qs(arr,0,MAX);

CopyMemory(pViewFile2, arr, MAX\*sizeof(int));

UnmapViewOfFile(pViewFile2);

ReleaseMutex(mtx);

return 0;

}

void qs(int\* s\_arr, int first, int last){

int i = first, j = last, x = s\_arr[(first + last) / 2];

do {

while (s\_arr[i] > x) i++;

while (s\_arr[j] < x) j--;

if (i <= j) {

if (s\_arr[i] < s\_arr[j]) std::swap(s\_arr[i], s\_arr[j]);

i++;

j--;

}

} while (i <= j);

if (i < last)

qs(s\_arr, i, last);

if (first < j)

qs(s\_arr, first, j);

}

4.Тест программы

При запуске, основная программа выделяет 40 байт под 10 элементов int. Создаётся процесс сортировки, в приостановленном состоянии. После массив перегружается в общую часть памяти, где его видит процесс сортировки. После этого массив сортируется и снова помещается в общую память, но уже изменённым. На этом второй процесс завершает свою работу. После этого исходный процесс считывает массив из общей памяти и выводит его на экран. (рис.4.1)

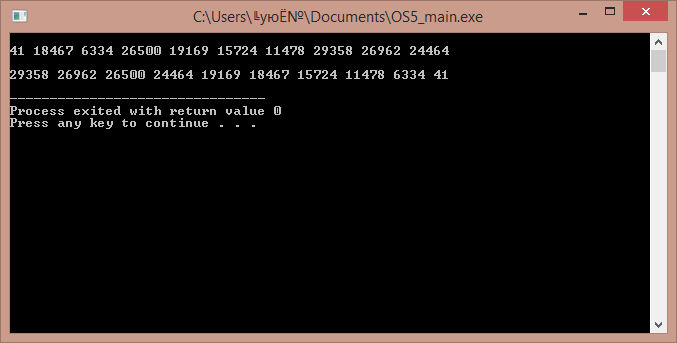
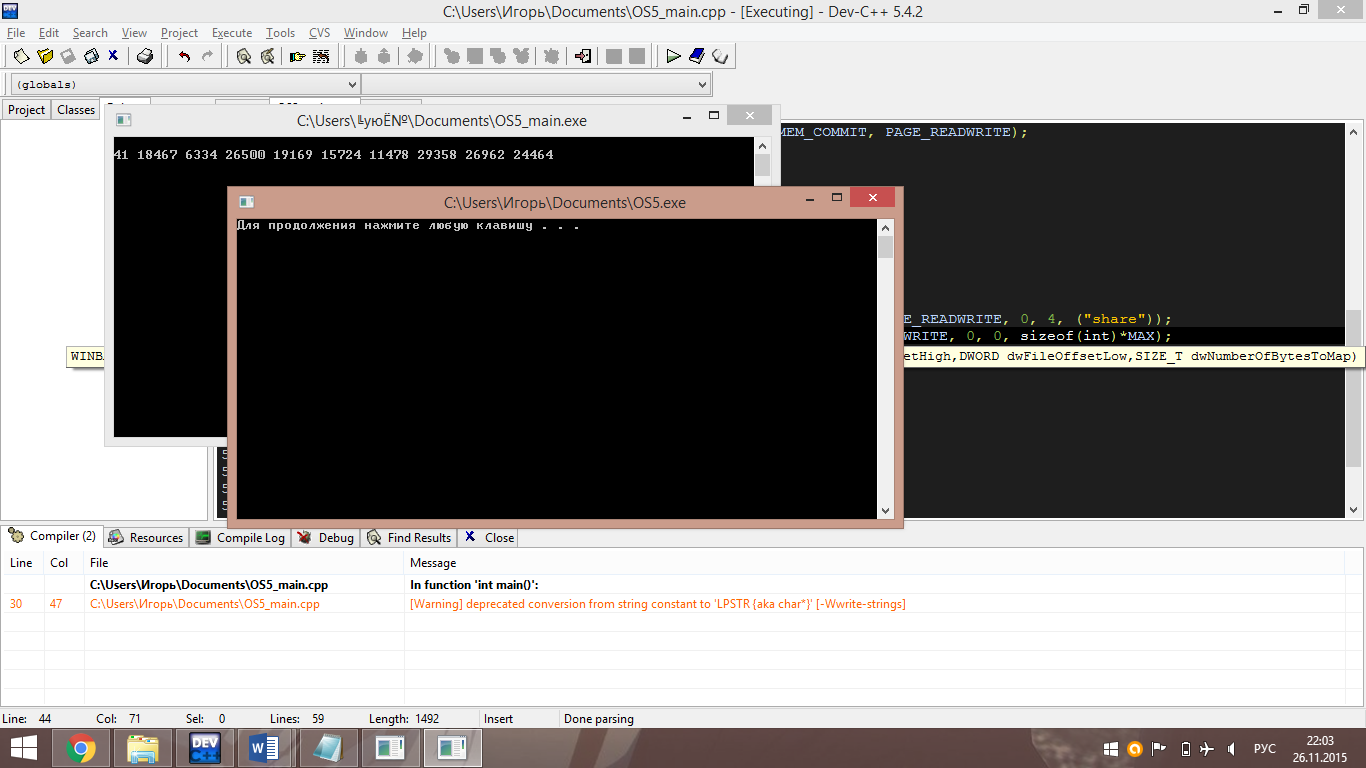


Рисунок 4.1-Тест программы.

ВЫВОДЫ

В данной лабораторной работе были изучены методы разделения общей памяти и её выделение между процессами. Программа написана при помощи такого объекта синхронизации, как мьютекс.