5 -Лабораторная работа №5

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

5.1. 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.

5.2. Постановка задачи

Приобрести навыки работы с механизмами управления памяти с помощью средств WinAPI.

5.3. Ход выполнения работы

5.3.1. Тексты программ

Программа написана на языке программирования C, компилятор – gcc 6.3.0. Тексты программ представлены в Листингах 1 и 2.

Листинг 1 – программа Master5

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include <windows.h>

#define MEMPAGE_COUNT 3
#define MEMPAGE_SIZE 4096
#define TOTAL_MEMSIZE ((MEMPAGE_COUNT) * (MEMPAGE_SIZE))

#define MAX 256

#define MUTEX_NAME L"mutex_lab5"
#define MAP_NAME L"memshare_lab5"

typedef char arr_elem_t;

/* Заполнить массив случайными данными */
void fill_array (arr_elem_t * arrv, size_t arrc)
{
```

```
    for (size_t i = 0; i < arrc; i++)
        arrv[i] = rand() % MAX;
}

int main ()
{
    /* Создание процесса и Мьютекса */
    HANDLE mutex = CreateMutex(NULL, TRUE, MUTEX_NAME);
    STARTUPINFO si; PROCESS_INFORMATION pi;
    memset(&si, 0, sizeof(STARTUPINFO)); memset(&pi, 0,
sizeof(PROCESS_INFORMATION));
    si.cb = sizeof(STARTUPINFO); si.lpTitle = "MemSorter 3000";
    CreateProcess(NULL, "MemSort.exe", 0, NULL, FALSE, CREATE_NEW_CONSOLE,
NULL, NULL, &si, &pi);

    /* Выделение участка виртуальной памяти */
    DWORD old_sec_attrs;
    LPVOID base_address = VirtualAlloc(NULL, TOTAL_MEMSIZE, MEM_COMMIT,
PAGE_READWRITE);
    arr_elem_t *arr = (arr_elem_t *) base_address;
    /* Заполнение памяти */
    fill_array (arr, TOTAL_MEMSIZE / sizeof(arr_elem_t));

    /* Создание общей памяти */
    HANDLE file_map = CreateFileMapping(INVALID_HANDLE_VALUE, NULL,
PAGE_READWRITE, 0, TOTAL_MEMSIZE, "memshare_lab5");
    /* Отображение файла в адресное пространство программы */
    LPVOID view = MapViewOfFile(file_map, FILE_MAP_ALL_ACCESS, 0, 0,
TOTAL_MEMSIZE);
    /* Копирование в файл из Виртуальной памяти */
    CopyMemory(view, base_address, TOTAL_MEMSIZE);

    /* Изменение прав доступа к памяти */
    VirtualProtect(base_address, TOTAL_MEMSIZE, PAGE_READONLY,
&old_sec_attrs);

    /* Освобождение мьютекса и ожидание завершения процесса */
    ReleaseMutex(mutex);
    WaitForSingleObject(pi.hProcess, INFINITE);

    /* Изменение прав доступа к памяти */
```

```
    VirtualProtect(base_address, TOTAL_MEMSIZE, PAGE_READWRITE,
&old_sec_attrs);

    /* Копирование из общей памяти в виртуальную память */
    CopyMemory(base_address, view, TOTAL_MEMSIZE);

    /* Печать данных на экран */
    for (size_t i = 0; i < TOTAL_MEMSIZE / sizeof(arr_elem_t); i++)
        printf("%d ", (int) arr[i]);
    printf("\r\n");

    /* Освобождение памяти и Выход */
    UnmapViewOfFile(view);
    CloseHandle(file_map);
    VirtualFree(base_address, TOTAL_MEMSIZE, MEM_RELEASE);
    return 0;
}
```

Листинг 2 – программа MemSort

```
#include <stdio.h>
#include <stdlib.h>

#include <windows.h>

#define MEMPAGE_COUNT 3
#define MEMPAGE_SIZE 4096
#define TOTAL_MEMSIZE ((MEMPAGE_COUNT) * (MEMPAGE_SIZE))

#define MAX 256

#define MUTEX_NAME L"mutex_lab5"
#define MAP_NAME "memshare_lab5"

typedef char arr_elem_t;

void sort(arr_elem_t *arrv, size_t arrc)
{
    arr_elem_t tmp = 0;
    for (int i = arrc - 1; i >= 0; i--)
        for (int j = 0; j < i; j++)
```

```
        {
            if (arrv[j] > arrv[j + 1])
            {
                tmp = arrv[j];
                arrv[j] = arrv[j + 1];
                arrv[j + 1] = tmp;
            }
        }
}

int main()
{
    /* Открытие мьютекса */
    HANDLE mutex = OpenMutex(MUTEX_ALL_ACCESS, FALSE, MUTEX_NAME);
    WaitForSingleObject(mutex, INFINITE);
    /* Открытие общей памяти и отображение её в программе */
    HANDLE shared_mem = OpenFileMapping(FILE_MAP_ALL_ACCESS, FALSE,
MAP_NAME);
    arr_elem_t *arr = (arr_elem_t *) MapViewOfFile(shared_mem,
FILE_MAP_ALL_ACCESS, 0, 0, TOTAL_MEMSIZE);

    sort(arr, TOTAL_MEMSIZE / sizeof(arr_elem_t));

    ReleaseMutex(mutex);
    UnmapViewOfFile((LPVOID) arr);
    CloseHandle(shared_mem);
    return 0;
}
```

5.3.2. Тестовые примеры

Рисунок 5.1 демонстрирует вывод основной программы – отсортированные по возрастанию данные.

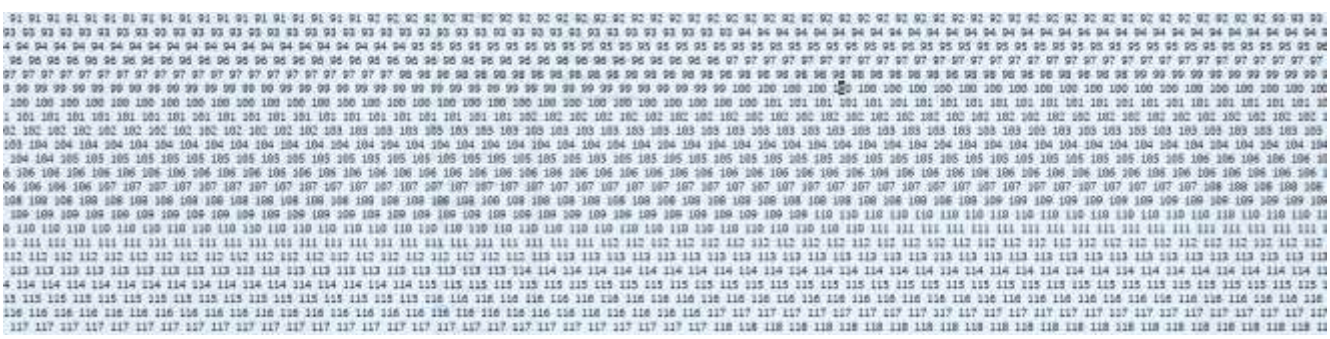

Рисунок 5.1 – Тестовый пример №1

Вывод

При выполнении данной лабораторной работы были получены навыки создания и работы с общей памятью, выделения памяти с помощью средств WinAPI. Также были повторно закреплены навыки работы с мьютексами.