Минобрнауки России
Федеральное государственное автономное образовательное
Учреждение высшего образования
«Санкт-Петербургский государственный электротехнический
Университет им. В.И. Ульянова (Ленина)»
(СПГЭТУ «ЛЭТИ»)
Факультет компьютерных технологий и информатики

Кафедра вычислительной техники

Отчет по лабораторной работе №2

на тему: «Управление памятью»

по дисциплине «Операционные системы»

Выполнил студент группы 9308: Семенов А.И.

Принял: к.т.н., доцент Тимофеев А.В.

Санкт-Петербург

2021 г.

Содержание

Цель работы

Исследовать механизмы управления виртуальной памятью Win32.

Исследование виртуального адресного пространства процесса

Указания к выполнению

Создайте консольное приложение с меню (каждая выполняемая функция и/или операция должна быть доступна по отдельному пункту меню), которое выполняет:

- получение информации о вычислительной системе (функция Win32 API – **GetSystemInfo**);
- определение статуса виртуальной памяти (функция Win32 API – **GlobalMemoryStatus**);
- определение состояния конкретного участка памяти по заданному с клавиатуры адресу (функция Win32 API – **VirtualQuery**);
- резервирование региона в автоматическом режиме и в режиме ввода адреса начала региона (функция Win32 API – **VirtualAlloc**);
- резервирование региона и передача ему физической памяти в автоматическом режиме и в режиме ввода адреса начала региона (функция Win32 API – **VirtualAlloc**);
- запись данных в ячейки памяти по заданным с клавиатуры адресам;
- установку защиты доступа для заданного (с клавиатуры) региона памяти и ее проверку (функция Win32 API – **VirtualProtect**);
- возврат физической памяти и освобождение региона адресного пространства заданного (с клавиатуры) региона памяти (функция Win32 API –**VirtualFree**).

Примеры работы программы

Программа может запущена с помощью флагов:

- -cb – конвертация байт в другие единицы измерения (максимально возможные)
- -hi – помощь в вводе констант в некоторых пунктах (вывод доступных констант на экран)
- -sa – вывод виртуальных адресов процесса в некоторых пунктах

При запуске программы открывается меню, показанное на рисунке 1.

```
Select a menu item:
1 - Get system info
2 - Get memory status
3 - Get status of a specific part of memory
4 - Reserve virtual memory
5 - Reserve virtual memory and (or) commit physical memory
6 - Input data to memory
7 - Set protection flags for virtual memory
8 - Free virtual memory
9 - Show all addresses of the virtual memory in this process
0 - Exit
```

Рисунок 1. Главное меню

Получение информации о вычислительной системе осуществляется через 1-ый пункт меню. Результат получения информации представлен на рисунке 2.

```
System info:
Processor architecture: x64 (AMD or Intel)
Page size: 4096
OEM ID: 9
Minimum application address: 0x10000
Maximum application address: 0x7fffffefffff
A mask representing the set of processors configured into the system:
1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
The number of logical processors in the current group: 6
The granularity for the starting address at which virtual memory can be allocated: 65536
The architecture-dependent processor level: 6
Processor features:
        The atomic compare and exchange operation (cmpxchg) is available
        The atomic compare and exchange 128-bit operation (cmpxchg16b) is available
        _fastfail() is available
        The MMX instruction set is available
        Data execution prevention is enabled
        The processor is PAE-enabled
        The RDTSC instruction is available
        RDFSBASE, RDGSBASE, WRFSBASE, and WRGSBASE instructions are available
        Second Level Address Translation is supported by the hardware
        The SSE3 instruction set is available
        Virtualization is enabled in the firmware and made available by the operating system
        The SSE instruction set is available
        The SSE2 instruction set is available
        The processor implements the XSAVE and XRSTOR instructions
The architecture-dependent processor revision: 9e0a
Для продолжения нажмите любую клавишу . . .
```

Рисунок 2. Получение информации о вычислительной системе

Определение статуса виртуальной памяти осуществляется через 2-ой пункт меню. Результат получения статуса представлен на рисунке 3.

```
The approximate percentage of physical memory that is in use: 37
The amount of actual physical memory: 15.93 GB
The amount of physical memory currently available: 9.99 GB
The current size of the committed memory limit: 19.93 GB
The maximum amount of memory the current process can commit: 12.16 GB
The size of the user-mode portion of the virtual address space of the calling process: 127.99 TB
The amount of unreserved and uncommitted memory currently in the user-mode portion of
        the virtual address space of the calling process: 127.99 TB
Для продолжения нажмите любую клавишу . . .
```

Рисунок 3. Получение статуса виртуальной памяти

Чтобы определить состояния конкретного участка памяти по указанному адресу, необходимо использовать пункт 3. Результат определения состояния представлен на рисунке 4.

```
No addresses allocated yet
Input the pointer (0x...) of desired part of memory: 0xf10000
A pointer to the base address of the region of pages: 0xf10000
A pointer to the base address of a range of pages allocated by the VirtualAlloc function: 0xee0000
The memory protection option when the region was initially allocated: 4
The size of the region in which all pages have identical attributes: 832.0 KB
Memory state: reserved pages where a range of the process's virtual address space is reserved without any physical storage being allocated
The access protection of the pages in the region: 0x0
The type of pages in the region: the memory pages within the region are private (that is, not shared by other processes)
Для продолжения нажмите любую клавишу . . .
```

Рисунок 4. Определение состояния конкретного участка памяти

Для резервирования региона предназначен пункт 4 меню. Результат резервирования представлен на рисунках 5 и 6.

```
No addresses allocated yet
Input the pointer (0x...) of the beginning of the region to reserve memory (input 0 for auto mode): 0xfe00000

Input the number of the region in bytes: 4096*2
The base address of the allocated region of pages: 0xfe00000
Для продолжения нажмите любую клавишу . . .
```

Рисунок 5. Резервирование региона по указанному адресу

```
Allocated addresses in this process:
Address is 0xfe00000 (not committed memory) Size is 8.0 KB
Input the pointer (0x...) of the beginning of the region to reserve memory (input 0 for auto mode): 0

Input the number of the region in bytes: 4096*4
The base address of the allocated region of pages: 0xe0000
Для продолжения нажмите любую клавишу . . .
```

Рисунок 6. Резервирование региона в автоматическом режиме

Для резервирования и передачи региону физической памяти используется пункт 5 меню. Он работает в двух режимах:

1. Резервирование и передача физической памяти новому региону;
2. Передача физической памяти региону, который уже зарезервирован.

Результат работы данного пункта представлен на рисунках 7 и 8.

```
Allocated addresses in this process:
Address is 0xfe00000 (not committed memory) Size is 8.0 KB
Address is 0xe0000 (not committed memory) Size is 16.0 KB
Input the pointer (0x...) of the beginning of the region to reserve and (or) commit memory (input 0 for auto mode): 0

Input the size of the region in bytes: 4096*1
The base address of the allocated region of pages: 0xf0000
Для продолжения нажмите любую клавишу . . .
```

Рисунок 7. Резервирование и передача физической памяти новому региону

```
Allocated addresses in this process:
Address is 0xfe00000 (not committed memory) Size is 8.0 KB
Address is 0xe0000 (not committed memory) Size is 16.0 KB
Address is 0xf0000 (committed memory) Size is 4.0 KB
Input the pointer (0x...) of the beginning of the region to reserve and (or) commit memory (input 0 for auto mode): 0xfe00000
This address will be commited!
The base address of the allocated region of pages: 0xfe00000
Для продолжения нажмите любую клавишу . . .
```

Рисунок 8. Передача физической памяти зарезервированному региону

Запись данных в ячейки памяти по заданному адресу производится в пункте 6 меню. Результат работы записи представлен на рисунке 9.

```
Allocated addresses in this process:
Address is 0xfe00000 (committed memory) Size is 8.0 KB
Address is 0xe0000 (not committed memory) Size is 16.0 KB
Address is 0xf0000 (committed memory) Size is 4.0 KB
Input the pointer (0x...) of the beginning of the region of pages of some virtual memory: 0xf0000
Input the string to write to memory:
Тут какая-нибудь запись
0xf0000 address filled with this your string:
Тут какая-нибудь запись
Для продолжения нажмите любую клавишу . . .
```

Рисунок 9. Запись данных в ячейки памяти по заданному адресу

Установка защиты доступа для заданного региона памяти производится через пункт 7 меню. Пример установки защиты представлен на рисунке 10.

```
Allocated addresses in this process:
Address is 0xfe00000 (committed memory) Size is 8.0 KB
Address is 0xe0000 (not committed memory) Size is 16.0 KB
Address is 0xf0000 (committed memory) Size is 4.0 KB
Input the pointer (0x...) of the beginning of the region of pages of some virtual memory: 0xf0000
Input the new protection flag (0x...):
Available flags:
        PAGE_EXECUTE - 0x10 | Enables execute access to the COMMITTED region of pages
        PAGE_EXECUTE_READ - 0x20 | Enables execute or read-only access to the COMMITTED region of pages
        PAGE_EXECUTE_READWRITE - 0x40 | Enables execute, read-only, or read/write access to the COMMITTED region of pages
        PAGE_NOACCESS - 0x01 | Disables all access to the COMMITTED region of pages
        PAGE_READONLY - 0x02 | Enables read-only access to the COMMITTED region of pages
        PAGE_READWRITE - 0x04 | Enables read-only or read/write access to the COMMITTED region of pages
Input the new protection flag here: 0x02
You changed protection flag from 0x4 to 0x2
Changed address: 0xf0000
Для продолжения нажмите любую клавишу . . .
```

Рисунок 10. Установка защиты доступа

Освобождение регионов адресного пространства региона памяти производится через пункт 8 меню. Освобождение может происходить в нескольких режимах:

1. Освободить абсолютно все регионы, которые были зарезервированы. Если им передана физическая память, то происходит возврат памяти
2. Освободить конкретный участок, которому передана физическая память:
    a. Возвратить физическую память и оставить участок зарезервированным
    b. Возвратить физическую память и снять резервирование
3. Снять резервирование с конкретного участка регионов, которому не передана физическая память.

Все режимы представлены на рисунках 11, 12, 13 и 14 соответственно.

```
Allocated addresses in this process:
Address is 0x1f0000 (not committed memory) Size is 8.0 KB
Address is 0x620000 (committed memory) Size is 20.0 KB
Address is 0x630000 (committed memory) Size is 32.0 KB
Address is 0x640000 (not committed memory) Size is 8.0 KB
Input the pointer (0x...) of the beginning of the region of pages of some virtual memory (input 0 to free all regions of pages): 0
All regions successfully freed!
Для продолжения нажмите любую клавишу . . .
```

Рисунок 11. Освобождение всех регионов

```
Allocated addresses in this process:
Address is 0xfe00000 (committed memory) Size is 8.0 KB
Address is 0xe0000 (not committed memory) Size is 16.0 KB
Address is 0xf0000 (committed memory) Size is 4.0 KB
Input the pointer (0x...) of the beginning of the region of pages of some virtual memory (input 0 to free all regions of pages): 0xf0000
Do you want to decommit and release this memory? Input 1 for yes, 0 otherwise: 0
Memory at 0xf0000 successfully freed!
Для продолжения нажмите любую клавишу . . .
```

Рисунок 12. Возврат физической памяти без снятия резервирования

```
Allocated addresses in this process:
Address is 0xfe00000 (committed memory) Size is 8.0 KB
Address is 0xe0000 (not committed memory) Size is 16.0 KB
Address is 0xf0000 (not committed memory) Size is 4.0 KB
Input the pointer (0x...) of the beginning of the region of pages of some virtual memory (input 0 to free all regions of pages): 0xfe00000
Do you want to decommit and release this memory? Input 1 for yes, 0 otherwise: 1
Memory at 0xfe00000 successfully freed!
Для продолжения нажмите любую клавишу . . .
```

Рисунок 13. Возврат физической памяти со снятием резервирования

```
Allocated addresses in this process:
Address is 0xe0000 (not committed memory) Size is 16.0 KB
Address is 0xf0000 (not committed memory) Size is 4.0 KB
Input the pointer (0x...) of the beginning of the region of pages of some virtual memory (input 0 to free all regions of pages): 0xf0000
Memory at 0xf0000 successfully freed!
Для продолжения нажмите любую клавишу . . .
```

Рисунок 14. Снятие резервирования с указанного региона

Для того, чтобы узнать, какие участки были зарезервированы и была бы передана им физическая память, используется пункт 9 меню. Пример вывода представлен на рисунке 15.

```
Allocated addresses in this process:
Address is 0x1f0000 (not committed memory) Size is 20.0 KB
Address is 0x620000 (committed memory) Size is 36.0 KB
Address is 0x630000 (committed memory) Size is 28.0 KB
Address is 0x640000 (not committed memory) Size is 36.0 KB
Address is 0x650000 (not committed memory) Size is 324.0 KB
Для продолжения нажмите любую клавишу . . .
```

Рисунок 15. Вывод всех регионов, зарезервированных в процессе работы программы

Вывод по заданию

Были изучены различные функции Win32 API, позволяющие работать с виртуальным адресным пространством. Рассмотренные функции позволяют получить информацию об участках памяти по указанным адресам; зарезервировать необходимый участок и передать ему физическую память, если это нужно; устанавливать защиту доступа к определенному региону памяти; записывать данные в ячейки памяти по указанному адресу.

Использование проецируемых файлов для обмена данными между процессами

Указания к выполнению

Создайте два консольных приложения с меню (каждая выполняемая функция и/или операция должна быть доступна по отдельному пункту меню), которые выполняют:

- приложение-писатель создает проецируемый файл (функции Win32 API – **CreateFile**, **CreateFileMapping**), проецирует фрагмент файла в память (функции Win32 API – **MapViewOfFile**, **UnmapViewOfFile**), осуществляет ввод данных с клавиатуры и их запись в спроецированный файл;
- приложение-читатель открывает проецируемый файл (функция Win32 API – **OpenFileMapping**), проецирует фрагмент файла в память (функции Win32 API – **MapViewOfFile**, **UnmapViewOfFile**), считывает содержимое из спроецированного файла и отображает на экран.

Примеры выполнения программы

Программа-писатель работает по следующему принципу:

1. Вводится имя файла для проецирования.
2. Вводится имя, которое используется для проецирования (оно будет использовано программой-читателем).
3. Вводятся сами данные.
4. Программа-писатель не закрывается, пока читатель не закончит свою работу.

Программа-читатель работает по следующему принципу:

1. Вводится имя файла, которое используется для проецирования (оно было задано программой-писателем).
2. Выводятся сами данные.

Пример работы написания представлен на рисунке 16. Пример работы чтения – на рисунке 17.

```
Input the name of the file to create map view: file
Input the name of the file mapping: fmName
Input the data to write at address 0x1f0000
Any data here
Do not press any key until the reader program stops reading!
When it's done, press any key!
Для продолжения нажмите любую клавишу . . .
```

Рисунок 16. Пример работы программы-писателя

```
Input the name of the file mapping: fmName
Written data at address 0x1b0000
Any data here
Для продолжения нажмите любую клавишу . . .
```

Рисунок 17. Пример работы программы-читателя

Вывод по заданию

Был изучен механизм обмена данными между двумя процессами посредством проецирования файла в память. Наблюдаемое различие при проецировании файла в программе-писателе и программе-читателе в адресах, по которым производится запись/чтение, основывается на работе отображения файла в адресном пространство: используется непосредственно адресное пространство вызывающего отображение процесса.

Вывод

Каждый процесс имеет собственное виртуальное адресное пространство, адреса в котором переводятся в адреса уже физической памяти. Использование виртуальной памяти позволяет предоставить процессу большее значение памяти, чем имеется в физической оперативной памяти, путем использования вторичного хранилища: необходимые данные могут подгружаться в оперативную память со вторичного хранилища на место данных, которые наименее актуальны, т.е. производить обмен между хранилищем и физической оперативной памятью.

Проецирование файлов в память позволяют работать с файлами посредством записи данных в оперативную память или чтения из нее. Данный метод позволяет пользоваться одним файлом нескольким процессам одновременно. Также проецирование позволяет не использовать операции ввода-вывода, а оперировать с данными в памяти непосредственно: изменяя данные в проецируемом файле, изменяется и сам файл. Причем адрес проецируемого файла может отличаться в различных процессах. Это обусловлено тем, что каждый процесс имеет собственное виртуальное адресное пространство.