Министерство науки и образования РФ
Федеральное государственное автономное образовательное
учреждение высшего профессионального образования
«Санкт-Петербургский государственный электротехнический
университет «ЛЭТИ» им. В. И. Ульянова (Ленина)»
(СПбГЭТУ «ЛЭТИ»)

Кафедра вычислительной техники

Отчёт
по лабораторной работе № 4
на тему:
“Управлению в Windows”
по дисциплине “Операционные системы”

Выполнил студент гр. 4306:
Табаков А.В.
Принял: Тимофеев А.В.

Санкт-Петербург
2016

Цель работы: исследовать механизмы управления виртуальной памятью Windows.

Задание 4.1. Исследовать процесс трансляции виртуальных адресов в 64-разрядной операционной системе Windows.

Размеры списков страниц виртуальной памяти:

0: kd> !memusage 0x8

```
 loading PFN database
loading (100% complete)
Compiling memory usage data (99% Complete).
           Zeroed: 202141 (808564 kb)
             Free:      0 (     0 kb)
          Standby: 599555 (2398220 kb)
         Modified:  19678 ( 78712 kb)
   ModifiedNoWrite:      0 (     0 kb)
      Active/Valid: 875904 (3503616 kb)
       Transition:     10 (    40 kb)
        SLIST/Bad:    341 (  1364 kb)
          Unknown:      0 (     0 kb)
            TOTAL: 1697629 (6790516 kb)
```

Список процессов с их идентификаторами и объемом занимаемой памяти:

0: kd> !vm

```
*** Virtual Memory Usage ***
      Physical Memory:     1697629 (   6790516 Kb)
      Page File: \??\C:\pagefile.sys
        Current:   4194304 Kb  Free Space:   39863
        Minimum:   4194304 Kb  Maximum:      83886
      Available Pages:      792405 (   3169620 Kb)
      ResAvail Pages:      1550219 (   6200876 Kb)
      Locked IO Pages:           0 (         0 Kb)
      Free System PTEs:   33492568 ( 133970272 Kb)
      Modified Pages:        18879 (     75516 Kb)
      Modified PF Pages:     18114 (     72456 Kb)
      NonPagedPool Usage:    19644 (     78576 Kb)
      NonPagedPool Max:    1260031 (   5040124 Kb)
      PagedPool 0 Usage:     83461 (    333844 Kb)
      PagedPool 1 Usage:     10066 (     40264 Kb)
      PagedPool 2 Usage:      2592 (     10368 Kb)
      PagedPool 3 Usage:      2471 (      9884 Kb)
      PagedPool 4 Usage:      2556 (     10224 Kb)
      PagedPool Usage:      101146 (    404584 Kb)
      PagedPool Maximum:  33554432 ( 134217728 Kb)
      Session Commit:        15323 (     61292 Kb)
      Shared Commit:        143852 (    575408 Kb)
      Special Pool:              0 (         0 Kb)
      Shared Process:        13584 (     54336 Kb)
      Pages For MDLs:         2574 (     10296 Kb)
      PagedPool Commit:      101350 (    405400 Kb)
```

```
        Driver Commit:         14430 (     57720 Kb)
        Committed pages:     1088658 (   4354632 Kb)
        Commit limit:        2745741 (  10982964 Kb)

        Total Private:        702117 (   2808468 Kb)
         0c7c chrome.exe       83667 (    334668 Kb)
         19f4 chrome.exe       75713 (    302852 Kb)
……
         0aa8 Telegram.exe     13056 (     52224 Kb) – процесс для исследования
……
         12e0 livekd.exe         249 (       996 Kb)
         0124 smss.exe           137 (       548 Kb)
         0004 System              42 (       168 Kb)
         1bb0 FoxitReaderUpd       0 (         0 Kb)

0: kd> !process 0 0
**** NT ACTIVE PROCESS DUMP ****
PROCESS fffffa80060f8430
    SessionId: none  Cid: 0004    Peb: 00000000  ParentCid: 0000
    DirBase: 00187000  ObjectTable: fffff8a0000017d0  HandleCount: 1783.
    Image: System
PROCESS fffffa8006914b10
    SessionId: none  Cid: 0124    Peb: 7ffffffdb000  ParentCid: 0004
    DirBase: 1ad89f000  ObjectTable: fffff8a0002e4b10  HandleCount:  32.
    Image: smss.exe
…..
PROCESS fffffa8008e95b10
    SessionId: 1  Cid: 0aa8    Peb: 7efdf000  ParentCid: 0690
    DirBase: 15b838000  ObjectTable: fffff8a004255a30  HandleCount: 353.
    Image: Telegram.exe
…..
PROCESS fffffa800a1f2410
    SessionId: 1  Cid: 0628    Peb: 7fffffdf000  ParentCid: 0714
    DirBase: 69acf000  ObjectTable: fffff8a0070e71a0  HandleCount: 119.
    Image: kd.exe
PROCESS fffffa8006bf1390
    SessionId: 0  Cid: 096c    Peb: 7fffffd6000  ParentCid: 0434
    DirBase: 1d70ea000  ObjectTable: fffff8a005d2ea50  HandleCount: 769.
    Image: taskeng.exe
```

Выбранный процесс для исследования: Telegram.exe

```
0: kd> !process 0aa8 1
Searching for Process with Cid == aa8
PROCESS fffffa8008e95b10
    SessionId: 1  Cid: 0aa8    Peb: 7efdf000  ParentCid: 0690
    DirBase: 15b838000  ObjectTable: fffff8a004255a30  HandleCount: 353.
    Image: Telegram.exe
    VadRoot fffffa80092c03f0 Vads 248 Clone 0 Private 10661. Modified 19425. Loc
ked 0.
    DeviceMap fffff8a00137e9a0
    Token                         fffff8a00420b060
    ElapsedTime                   01:01:05.958
```

```
    UserTime                    00:00:00.280
    KernelTime                  00:00:00.156
    QuotaPoolUsage[PagedPool]       297544
    QuotaPoolUsage[NonPagedPool]    37472
    Working Set Sizes (now,min,max)  (16777, 50, 345) (67108KB, 200KB, 1380KB)
    PeakWorkingSetSize          16822
    VirtualSize                 218 Mb
    PeakVirtualSize             238 Mb
    PageFaultCount              91760
    MemoryPriority              BACKGROUND
    BasePriority                8
    CommitCharge                13056
```

Регионы виртуального адресного пространства выбранного процесса

```
0: kd> !vad fffffa80092c03f0
VAD           level      start      end    commit
fffffa800849b730 ( 6)         10        1f         0 Mapped       READWRITE
     Pagefile-backed section
fffffa8007b22e10 ( 5)         20        20         1 Private      READWRITE
fffffa8008ea5210 ( 6)         30        30         1 Private      READWRITE
fffffa800842fc10 ( 4)         40        40         0 Mapped  Exe  EXECUTE_WRITE
COPY  \Windows\System32\apisetschema.dll
fffffa80084041f0 ( 6)         50        53         0 Mapped       READONLY
     Pagefile-backed section
fffffa80083ed3f0 ( 5)         60        60         0 Mapped       READONLY
     Pagefile-backed section
fffffa8008ea69c0 ( 7)         70        70         1 Private      READWRITE
…..
fffffa8008e30250 ( 2)         2a0      24f3       854 Mapped  Exe  EXECUTE_WRITE
COPY  \Users\Komdosh\AppData\Roaming\Telegram Desktop\Telegram.exe
…..
fffffa8008e47620 ( 6)        68b0      68ef         7 Private      READWRITE
fffffa8009154d30 ( 4)        68f0      69ef         3 Private      READWRITE
fffffa8009540430 ( 6)       60001 fffffa8009540450      156501072 Private LargeP
agSec EXECUTE_WRITECOPY WRITECOMBINE


Total VADs: 89, average level: 6, maximum depth: 8
```

Подробно рассмотрим один регион виртуального адресного пространства:

```
0: kd> !vad fffffa8009154d30 1
VAD @ fffffa8009154d30
  Start VPN               68f0  End VPN           69ef  Control Area  000000
0000000000
  FirstProtoPte 0000000000000000  LastPte 0000000000000000  Commit Charge
        3 (3.)
  Secured.Flink             0  Blink               0  Banked/Extend
        0
  File Offset               0
     ViewUnmap PrivateMemory READWRITE
```

Переключение контекста процессора

```
0: kd> !process 0aa8 1
```

```
Searching for Process with Cid == aa8
PROCESS fffffa8008e95b10
   SessionId: 1  Cid: 0aa8    Peb: 7efdf000  ParentCid: 0690
   DirBase: 15b838000  ObjectTable: fffff8a004255a30  HandleCount: 353.
…..

0: kd> .context 15b838000
0: kd> .context
User-mode page directory base is 15b838000
0: kd> !pte 2a0*0x1000
                                           VA 00000000002a0000
PXE at FFFFF6FB7DBED000    PPE at FFFFF6FB7DA00000    PDE at FFFFF6FB40000008
PTE at FFFFF68000001500
contains 1AC000015F33D867  contains 035000015F761867  contains 036000015F2E2867
contains 8370000153FA0025
pfn 15f33d    ---DA--UWEV  pfn 15f761    ---DA--UWEV  pfn 15f2e2    ---DA--UWEV
pfn 153fa0    ----A--UR-V

0: kd> !dc 153fa0*0x1000
#153fa0000 00905a4d 00000003 00000004 0000ffff MZ..............
#153fa0010 000000b8 00000000 00000040 00000000 ........@.......
#153fa0020 00000000 00000000 00000000 00000000 ................
#153fa0030 00000000 00000000 00000000 00000168 ............h...
#153fa0040 0eba1f0e cd09b400 4c01b821 685421cd ........!..L.!Th
#153fa0050 70207369 72676f72 63206d61 6f6e6e61 is program canno
#153fa0060 65622074 6e757220 206e6920 20534f44 t be run in DOS
#153fa0070 65646f6d 0a0d0d2e 00000024 00000000 mode....$.......
```

Вывод: менеджер виртуальной памяти в операционной системе скрывает реальные адреса ячеек памяти и предоставляет непрерывное виртуальное адресное пространство.

Задание 4.2. Исследовать виртуальное адресное пространство процесса.

Меню программы

Выберите пункт меню

1 - Информация о вычислительной системе

2 - Статус памяти

3 - Состояние участка виртуальной памяти

4 - Раздельное резервирование региона и передача физ. памяти

5 - Одновременное резервирование региона и передача физ. памяти

6 - Запись данных

7 - Защита доступа региона памяти

8 – Очистить память

0 - Выход

1. Информация о вычислительной системе
   Архитектура процессора: x64 (AMD or Intel)
   Количество ядер: 4

Тип процессора: AMD x8664
Уровень процессора (CPU vendor): 21
Размер страницы: 4096
Минимальный адрес для приложений: 0000000000010000
Максимальный адрес для приложений: 000007FFFFFEFFFF
Активные ядра: 00001111

2. Статус памяти
   77% памяти используется
   6790516 всего Kb памяти
   1541292 доступно Kb памяти
   10982964 всего Kb страничного файла
   4655832 доступно Kb страничного файла
   8589934464 всего Kb виртуальной памяти
   8589922036 доступно Kb виртуальной памяти
   0 доступно Kb расширенной памяти
3. Состояние участка виртуальной памяти
   Адреса округляются до начального адреса страницы, в моём случае до $4096_{10}=1000_{16}$

   Введите адрес в диапазоне от 0x0000000000010000 до 0x000007FFFFFEFFFF): 0x00000458234
   Базовый адрес: 0x0000000000458000
   Базовый адрес выделенной памяти: 0x0000000000440000
   Размер региона: 2846720
   Режим доступа: 0x2
   PAGE_READONLY
   Состояние страницы: 0x1000
   MEM_COMMIT
   Тип страницы: 0x40000
   MEM_MAPPED

4. Раздельное резервирование региона и передача физ. Памяти
   Данная функция показывает стандартный жизненный цикл страницы памяти

   Базовый адрес (0 для автоматического задания адреса системой): 0x0
   Размер (в байтах): 16384

   Памяти зарезервирована:
   Базовый адрес: 0x00000000000D0000
   Базовый адрес выделенной памяти: 0x00000000000D0000
   Размер региона: 16384
   Режим доступа: 0x0
   Состояние страницы: 0x2000
   MEM_RESERVE
   Тип страницы: 0x20000
   MEM_PRIVATE

Память использована:
Базовый адрес: 0x00000000000D0000
Базовый адрес выделенной памяти: 0x00000000000D0000
Размер региона: 16384
Режим доступа: 0x4
 PAGE_READWRITE
Состояние страницы: 0x1000
 MEM_COMMIT
Тип страницы: 0x20000
 MEM_PRIVATE

5. Одновременное резервирование региона и передача физ. Памяти

   Базовый адрес (0 для автоматического задания адреса системой): 0x0
   Размер (в байтах): 8000

   Память выделена:
   Базовый адрес: 0x0000000000060000
   Базовый адрес выделенной памяти: 0x0000000000060000
   Размер региона: 8192
   Режим доступа: 0x4
    PAGE_READWRITE
   Состояние страницы: 0x1000
    MEM_COMMIT
   Тип страницы: 0x20000
    MEM_PRIVATE

   Память очищена:
   Базовый адрес: 0x0000000000060000
   Базовый адрес выделенной памяти: 0x0000000000000000
   Размер региона: 65536
   Режим доступа: 0x1
    PAGE_NOACCESS
   Состояние страницы: 0x10000
    MEM_FREE
   Тип страницы: 0x0

6. Запись данных
   Базовый адрес (0 для автоматического задания адреса системой): 0xD0000
   Данные: 76523
   Чтение памяти (0x00000000000 D0000): 76523

7. Защита доступа региона памяти
   Запрещает доступ к странице памяти

Базовый адрес (0 для автоматического задания адреса системой): 0xD0000

Память защищена:
Базовый адрес: 0x00000000000D0000
Базовый адрес выделенной памяти: 0x00000000000D0000
Размер региона: 16384
Режим доступа: 0x1
 PAGE_NOACCESS
Состояние страницы: 0x1000
 MEM_COMMIT
Тип страницы: 0x20000
 MEM_PRIVATE

Проверка 6 функции записи в память

Определим процесс

```
0: kd> !process 1CBC 1
Searching for Process with Cid == 1cbc
PROCESS fffffa8009087920
   SessionId: 1  Cid: 1cbc    Peb: 7fffffdc000  ParentCid: 13e4
   DirBase: 1d0b58000  ObjectTable: fffff8a00a27e3a0  HandleCount:  10.
   Image: Ae?ooaeuiay iaiyou.exe
   VadRoot fffffa800a925130 Vads 30 Clone 0 Private 153. Modified 0. Locked 0.
   DeviceMap fffff8a00137e9a0
   Token                             fffff8a018e6d060
   ElapsedTime                       00:00:04.597
   UserTime                          00:00:00.000
   KernelTime                        00:00:00.000
   QuotaPoolUsage[PagedPool]         27000
   QuotaPoolUsage[NonPagedPool]      3480
   Working Set Sizes (now,min,max)  (813, 50, 345) (3252KB, 200KB, 1380KB)
   PeakWorkingSetSize                813
   VirtualSize                       12 Mb
   PeakVirtualSize                   12 Mb
   PageFaultCount                    811
   MemoryPriority                    BACKGROUND
   BasePriority                      8
   CommitCharge                      187
   DebugPort                         fffffa8009218e00
```

Адресное пространство до записи

```
0: kd> !vad fffffa800a925130
VAD           level      start      end    commit
fffffa8009d368e0 ( 4)         10        1f         0 Mapped       READWRITE
     Pagefile-backed section
fffffa80094e4010 ( 3)         20        2f         0 Mapped       READWRITE
     Pagefile-backed section
fffffa80089783f0 ( 4)         30        33         0 Mapped       READONLY
     Pagefile-backed section
fffffa80083f94f0 ( 2)         40        40         0 Mapped       READONLY
```

Pagefile-backed section
ffffa8009d99090 (3) 50 50 1 Private READWRITE
…..
ffffa8008516eb0 (3) 7fffffdc 7fffffdc 1 Private READWRITE
ffffa8008bfceb0 (4) 7fffffde 7fffffdf 2 Private READWRITE

Total VADs: 30, average level: 4, maximum depth: 5Total VADs: 30, average level: 4, maximum depth: 5

После записи добавился VAD:

ffffa80087ce6d0 (4) d0 d0 1 Private READWRITE
Информация
0: kd> !vad ffffa80087ce6d0 1
VAD @ ffffa80087ce6d0
 Start VPN d0 End VPN d0 Control Area 000000
0000000000
 FirstProtoPte 0000000000000000 LastPte 0000000000000000 Commit Charge
  1 (1.)
 Secured.Flink 0 Blink 0 Banked/Extend
  0
 File Offset 0
  ViewUnmap MemCommit PrivateMemory READWRITE
Переключим контекст

0: kd> .context 1d0b58000
Преобразуем виртуальный адрес в физический

0: kd> !pte d0*0x1000
  VA 00000000000d0000
PXE at FFFFF6FB7DBED000 PPE at FFFFF6FB7DA00000 PDE at FFFFF6FB40000000
 PTE at FFFFF68000000680
contains 0310000070119867 contains 01200001C3A5C867 contains 01600000463FD867
 contains B49000000A53B867
pfn 70119 ---DA--UWEV pfn 1c3a5c ---DA--UWEV pfn 463fd ---DA--UWEV
 pfn a53b ---DA--UW-V

Чтение физической памяти и поиск числа ($108_{10}=6C_{16}$)
0: kd> !dc a53b*0x1000
a53b000 0000006c 00000000 00000000 00000000 l.....
a53b010 00000000 00000000 00000000 00000000 .....
a53b020 00000000 00000000 00000000 00000000 .....

Вывод: процесс хранит данные в своем виртуальном адресном пространстве

Задание 4.3. Использование проецируемых файлов для обмена данными между процессами.

Писатель.exe:

Данные для передачи: ETU_RULES!
Файл успешно создан
Маппинг объекта создан
Адрес проекции: 0x00000000000D0000
Закрыть проекцию? (y/n)

Читатель.exe:
Проекция открыта
Данные по адресу (0x00000000000D0000): ETU_RULES!
Закрыть проекцию? (y/n)

Информация о приложении Писатель.exe
0: kd> !process 1dd0 1

Searching for Process with Cid == 1dd0
PROCESS fffffa8006db0520
SessionId: 1 Cid: 1dd0 Peb: 7fffffd5000 ParentCid: 1af4
DirBase: 17d223000 ObjectTable: fffff8a005fbd0b0 HandleCount: 7.
Image: Ienaoaeu (4.3).exe
VadRoot fffffa8009271d60 Vads 28 Clone 0 Private 147. Modified 0. Locked 0.
DeviceMap fffff8a00137e9a0
Token fffff8a01a89a060
ElapsedTime 00:00:11.034
UserTime 00:00:00.000
KernelTime 00:00:00.000
QuotaPoolUsage[PagedPool] 20488
QuotaPoolUsage[NonPagedPool] 3240
Working Set Sizes (now,min,max) (718, 50, 345) (2872KB, 200KB, 1380KB)
PeakWorkingSetSize 718
VirtualSize 8 Mb
PeakVirtualSize 8 Mb
PageFaultCount 716
MemoryPriority BACKGROUND
BasePriority 8
CommitCharge 178
DebugPort fffffa8009f0a270

Данные:

0: kd> !vad fffffa8009271d60
VAD level start end commit
fffffa8009a21730 (4) 10 1f 0 Mapped READWRITE
Pagefile-backed section
fffffa80099ba170 (5) 20 2f 0 Mapped READWRITE
Pagefile-backed section
fffffa800ab5e860 (3) 30 33 0 Mapped READONLY
Pagefile-backed section

```
fffffa800a4472c0 ( 4)        40       40        0 Mapped      READONLY
    Pagefile-backed section
fffffa8009874dd0 ( 2)        50       50        1 Private     READWRITE

fffffa800a357980 ( 4)        60       c6        0 Mapped      READONLY
    \Windows\System32\locale.nls
fffffa800a370520 ( 3)       100      1ff       81 Private     READWRITE

fffffa8009a8a360 ( 1)       210      30f        6 Private     READWRITE

fffffa8006dc6230 ( 4)     77590     776ae        4 Mapped  Exe  EXECUTE_WRITE
COPY  \Windows\System32\kernel32.dll
fffffa8007830420 ( 3)     777b0     77959       14 Mapped  Exe  EXECUTE_WRITE
COPY  \Windows\System32\ntdll.dll
fffffa8009997a60 ( 5)     7efe0     7f0df        0 Mapped      READONLY
    Pagefile-backed section
fffffa8006df3f70 ( 4)     7f0e0     7ffdf        0 Private     READONLY

fffffa8006b69580 ( 2)     7ffe0     7ffef       -1 Private     READONLY

fffffa8009f88bc0 ( 3)    13f920    13f94b       20 Mapped  Exe  EXECUTE_WRITE
COPY  \Users\Komdosh\Documents\Visual Studio 2015\Projects\╧шёрЄхы№\x64\Debug\╧ш
ёрЄхы№ (4.3).exe
fffffa8009a69ec0 ( 4)   7fed01f0  7fed03aa        5 Mapped  Exe  EXECUTE_WRITE
COPY  \Windows\System32\ucrtbased.dll
fffffa8009271d60 ( 0)   7fed03b0  7fed04a5        7 Mapped  Exe  EXECUTE_WRITE
COPY  \Windows\System32\msvcp140d.dll
fffffa8006a886e0 ( 4)   7fedf160  7fedf181        2 Mapped  Exe  EXECUTE_WRITE
COPY  \Windows\System32\vcruntime140d.dll
fffffa8009116510 ( 3)   7fefa690  7fefa692        0 Mapped  Exe  EXECUTE_WRITE
COPY  \Windows\System32\api-ms-win-core-synch-l1-2-0.dll
fffffa800a323010 ( 5)   7fefb160  7fefb162        0 Mapped  Exe  EXECUTE_WRITE
COPY  \Windows\System32\api-ms-win-core-file-l1-2-0.dll
fffffa800a4b0ec0 ( 4)   7fefb170  7fefb172        0 Mapped  Exe  EXECUTE_WRITE
COPY  \Windows\System32\api-ms-win-core-processthreads-l1-1-1.dll
fffffa800a40a190 ( 2)   7fefb180  7fefb182        0 Mapped  Exe  EXECUTE_WRITE
COPY  \Windows\System32\api-ms-win-core-localization-l1-2-0.dll
fffffa8008fc4c40 ( 3)   7fefb3b0  7fefb3b2        0 Mapped  Exe  EXECUTE_WRITE
COPY  \Windows\System32\api-ms-win-core-file-l2-1-0.dll
fffffa8009cad4b0 ( 4)   7fefb3c0  7fefb3c2        0 Mapped  Exe  EXECUTE_WRITE
COPY  \Windows\System32\api-ms-win-core-timezone-l1-1-0.dll
fffffa800a7f1210 ( 1)   7fefd700  7fefd769        3 Mapped  Exe  EXECUTE_WRITE
COPY  \Windows\System32\KernelBase.dll
fffffa8006d378d0 ( 3)   7feffad0  7feffad0        0 Mapped  Exe  EXECUTE_WRITE
COPY  \Windows\System32\apisetschema.dll
fffffa8008c73610 ( 2)   7fffffb0  7fffffd2        0 Mapped      READONLY
    Pagefile-backed section
fffffa80089012a0 ( 3)   7fffffd5  7fffffd5        1 Private     READWRITE

fffffa80091193d0 ( 4)   7fffffde  7fffffdf        2 Private     READWRITE


Total VADs: 28, average level: 4, maximum depth: 5
```

После записи добавилась новая запись:

```
fffffa80099f4c80 ( 3)         d0        d0         0 Mapped       READWRITE
     \Users\Komdosh\Documents\Visual Studio 2015\Projects\╧шёрЄхы№\╧шёрЄхы№\tem
pTextFile.txt
```

Это наш файл:

```
0: kd> !vad fffffa80099f4c80 1

VAD @ fffffa80099f4c80
  Start VPN                 d0  End VPN              d0  Control Area  fffffa
8008ff4d50
  FirstProtoPte fffff8a00131e410  LastPte fffff8a00131e410  Commit Charge
        0 (0.)
  Secured.Flink               0  Blink                0  Banked/Extend
        0
  File Offset                 0
     ViewShare READWRITE

ControlArea  @ fffffa8008ff4d50
  Segment      fffff8a00b4ef220  Flink      0000000000000000  Blink        fffff
a80084966f8
  Section Ref                 2  Pfn Ref                 1  Mapped Views
         1
  User Ref                    2  WaitForDel              0  Flush Count
         0
  File Object  fffffa800a74add0  ModWriteCount            0  System Views
         0
  WritableRefs                2
  Flags (8080) File WasPurged

     \Users\Komdosh\Documents\Visual Studio 2015\Projects\╧шёрЄхы№\╧шёрЄхы№\tem
pTextFile.txt

Segment @ fffff8a00b4ef220
  ControlArea     fffffa8008ff4d50  ExtendInfo    0000000000000000
  Total Ptes                  100
  Segment Size             100000  Committed                0
  Flags (c0000) ProtectionMask
```

Запускаем читателя и смотрим на информацию о нём:

```
0: kd> !process 26a0 1

Searching for Process with Cid == 26a0
PROCESS fffffa800a968760
    SessionId: 1  Cid: 26a0    Peb: 7fffffd5000  ParentCid: 2040
    DirBase: 1d2516000  ObjectTable: fffff8a01949fc00  HandleCount:   9.
    Image: ?eoaoaeu (4.3).exe
    VadRoot fffffa800a127f70 Vads 29 Clone 0 Private 147. Modified 0. Locked 0.
    DeviceMap fffff8a00137e9a0
```

```
    Token                             fffff8a00aa8d060
    ElapsedTime                       00:00:29.753
    UserTime                          00:00:00.000
    KernelTime                        00:00:00.000
    QuotaPoolUsage[PagedPool]         20464
    QuotaPoolUsage[NonPagedPool]      3360
    Working Set Sizes (now,min,max)  (742, 50, 345) (2968KB, 200KB, 1380KB)
    PeakWorkingSetSize                742
    VirtualSize                       8 Mb
    PeakVirtualSize                   8 Mb
    PageFaultCount                    769
    MemoryPriority                    BACKGROUND
    BasePriority                      8
    CommitCharge                      178
    DebugPort                         fffffa800a2f7f90
```

У читателя есть файл, созданный писателем:

```
fffffa8009db9c30 ( 4)         d0        d0        0 Mapped       READWRITE
     \Users\Komdosh\Documents\Visual Studio 2015\Projects\╧шёрЄхы№\╧шёрЄхы№\tem
pTextFile.txt

0: kd> .context 1d2516000

0: kd> !pte d0*0x1000
                                           VA 00000000000d0000
PXE at FFFFF6FB7DBED000    PPE at FFFFF6FB7DA00000    PDE at FFFFF6FB40000000
 PTE at FFFFF68000000680
contains 053000012FD52867  contains 2200000065BF8867  contains 0140000092799867
 contains AD500001D5891825
pfn 12fd52    ---DA--UWEV  pfn 65bf8     ---DA--UWEV  pfn 92799     ---DA--UWEV
 pfn 1d5891    ----A--UR-V

0: kd> !dc 1d5891*0x1000
#1d5891000 5f555445 454c5552 00002153 00000000 ETU_RULES!.....
#1d5891010 00000000 00000000 00000000 00000000 ................
```

Мы можем наблюдать, что читатель видит тот же файл, что создал писатель, т.к. текст совпадает. А также, что обе программы имеют доступ к данному файлу.

Вывод: Файловые проекции, в ОС Windows, позволяют передавать информацию между процессов. Данный механизм обычно используется для предоставления доступа приложений к системным библиотекам.

Тексты программ

Виртуальная память (4.2)
Main.cpp

```
#include <iostream>
#include "VirtualMemoryAPI.h"

using namespace std;

int menu();

int main() {
	setlocale(0, ".1251");
	int notExit;

	do {
		switch (notExit = menu())
		{
		case 1:
			systemInfo();
			break;
		case 2:
			virtualMemoryStatus();
			break;
		case 3:
			virtualPageStatus(0);
			break;
		case 4:
			separateReserveCommit();
			break;
		case 5:
			simultaneousReserveCommit();
			break;
		case 6:
			writeData();
			break;
		case 7:
			protectVirtualPage();
			break;
		case 8:
			freeVirtualPage();
			break;

		case 0:
			break;
		default:
			if (notExit)
```

```
                cout << "Такого варианта нет, повторите ввод" << endl;
            }
            if (notExit)
                system("pause");
        } while (notExit);
        cin.get();
        return 0;
}

int menu()
{
        system("cls");
        int point;
        do {
            cin.clear();
            cin.sync();

            cout << "Выберите пункт меню" << endl;
            cout << "1 - Информация о вычислительной системе" << endl;
            cout << "2 - Статус виртуальной памяти" << endl;
            cout << "3 - Состояние участка виртуальной памяти" << endl;
            cout << "4 - Раздельное резервирование региона и передача физ. памяти" <<
endl;
            cout << "5 - Одновременное резервирование региона и передача физ.
памяти" << endl;
            cout << "6 - Запись данных" << endl;
            cout << "7 - Защита доступа региона памяти" << endl;
            cout << "0 - Выход" << endl;
            cout << "> ";
            cin >> point;
            if (cin.fail())
                cout << "Что-то пошло не так, выберите пункт меню повторно" <<
endl;
        } while (cin.fail());
        system("cls");
        return point;
}
```

VirtualMemoryAPI.h

```
#pragma once


#define WINVER 0x0500


#include <iostream>

#include <windows.h>

#include <winbase.h>
```

```
#include <io.h>
#include <stdio.h>
#include <tchar.h>
#include <string>
#include <bitset>

void systemInfo();
void virtualMemoryStatus();
void virtualPageStatus(DWORD address);
void separateReserveCommit();
void simultaneousReserveCommit();
void writeData();
void protectVirtualPage();
```

VirtualMemoryAPI.cpp

```
#include "VirtualMemoryAPI.h"

using namespace std;

void systemInfo() {
	SYSTEM_INFO systemInfo;
	GetSystemInfo(&systemInfo);
	cout << "Информация о вычислительной системе:" << endl;
	cout << "Архитектура процессора: ";
	switch (systemInfo.wProcessorArchitecture)
	{
	case 0:
		cout << "Intel x86" << endl;
		break;
	case 5:
		cout << "ARM" << endl;
		break;
	case 6:
		cout << "Intel Itanium-based" << endl;
		break;
	case 9:
		cout << "x64 (AMD or Intel)" << endl;
		break;
	default:
		cout << "Неизвестная архитектура" << endl;
		break;
	}
	cout << "Количество ядер: " << systemInfo.dwNumberOfProcessors << endl;
```

```
	cout << "Тип процессора: ";
	switch (systemInfo.dwProcessorType)
	{
	case 386:
		cout << "Intel 386" << endl;
		break;
	case 486:
		cout << "Intel 486" << endl;
		break;
	case 586:
		cout << "Intel Pentium" << endl;
		break;
	case 2200:
		cout << "Intel IA64" << endl;
		break;
	case 8664:
		cout << "AMD x8664" << endl;
		break;
	default:
		cout << "ARM" << endl;
		break;
	}
	cout << "Уровень процессора (CPU vendor): " << systemInfo.wProcessorLevel <<
endl;
	cout << "Размер страницы: " << systemInfo.dwPageSize << endl;
	cout << "Минимальный адрес для приложений: " <<
systemInfo.lpMinimumApplicationAddress << endl;
	cout << "Максимальный адрес для приложений: " <<
systemInfo.lpMaximumApplicationAddress << endl;
	bitset<8> x(systemInfo.dwActiveProcessorMask);
	cout << "Активные ядра: " << x << endl;
}
void virtualMemoryStatus() {
	const int divider = 1024;
	const int width = 12;
	MEMORYSTATUSEX memoryStatusEx;
	memoryStatusEx.dwLength = sizeof(memoryStatusEx);
	GlobalMemoryStatusEx(&memoryStatusEx);
	cout.width(width-1);
	cout << memoryStatusEx.dwMemoryLoad << "% памяти используется\n";
	cout.width(width);
	cout << memoryStatusEx.ullTotalPhys / divider << " всего Kb памяти\n";
	cout.width(width);
	cout << memoryStatusEx.ullAvailPhys / divider << " доступно Kb памяти\n";
	cout.width(width);
	cout << memoryStatusEx.ullTotalPageFile / divider << " всего Kb страничного
файла\n";
	cout.width(width);
```

```
	cout << memoryStatusEx.ullAvailPageFile / divider << " доступно Kb страничного
файла\n";
	cout.width(width);
	cout << memoryStatusEx.ullTotalVirtual / divider << " всего Kb виртуальной
памяти\n";
	cout.width(width);
	cout << memoryStatusEx.ullAvailVirtual / divider << " доступно Kb виртуальной
памяти\n";
	cout.width(width);
	cout << memoryStatusEx.ullAvailExtendedVirtual / divider << " доступно Kb
расширенной памяти\n";
}
void virtualPageStatus(DWORD address) {
	MEMORY_BASIC_INFORMATION memoryInfo;
	SYSTEM_INFO systemInfo;
	GetSystemInfo(&systemInfo);

	if (!address) {
		cout << "Введите адрес в диапазоне от 0x" <<
systemInfo.lpMinimumApplicationAddress
			<< " до 0x" << systemInfo.lpMaximumApplicationAddress << "): 0x";
		cin >> hex >> address;
	}

	VirtualQuery((LPCVOID)address, &memoryInfo, sizeof(memoryInfo));
	cout << "Базовый адрес: 0x" << memoryInfo.BaseAddress << endl;
	cout << "Базовый адрес выделенной памяти: 0x"  << memoryInfo.AllocationBase <<
endl;

	cout << "Размер региона: " << memoryInfo.RegionSize << endl;

	cout << "Режим доступа: 0x" << hex << memoryInfo.Protect << endl;
	if (memoryInfo.Protect & PAGE_NOACCESS) cout << "  PAGE_NOACCESS" <<
endl;
	if (memoryInfo.Protect & PAGE_READONLY) cout << "  PAGE_READONLY" <<
endl;
	if (memoryInfo.Protect & PAGE_READWRITE) cout << "  PAGE_READWRITE" <<
endl;
	if (memoryInfo.Protect & PAGE_EXECUTE_WRITECOPY) cout << "
PAGE_EXECUTE_WRITECOPY" << endl;
	if (memoryInfo.Protect & PAGE_EXECUTE) cout << "  PAGE_EXECUTE" << endl;
	if (memoryInfo.Protect & PAGE_EXECUTE_READ) cout << "
PAGE_EXECUTE_READ" << endl;
	if (memoryInfo.Protect & PAGE_EXECUTE_READ) cout << "
PAGE_EXECUTE_READ" << endl;
	if (memoryInfo.Protect & PAGE_EXECUTE_READWRITE) cout << "
PAGE_EXECUTE_READWRITE" << endl;
```

```
	if (memoryInfo.Protect & PAGE_EXECUTE_WRITECOPY) cout << "
PAGE_EXECUTE_WRITECOPY" << endl;
	if (memoryInfo.Protect & PAGE_GUARD) cout << "  PAGE_GUARD" << endl;
	if (memoryInfo.Protect & PAGE_NOCACHE) cout << "  PAGE_NOCACHE" << endl;
	if (memoryInfo.Protect & PAGE_WRITECOMBINE) cout << "
PAGE_WRITECOMBINE" << endl;
	cout << "Состояние страницы: 0x" << hex << memoryInfo.State << endl;
	if (memoryInfo.State & MEM_COMMIT) cout << "  MEM_COMMIT" << endl;
	if (memoryInfo.State & MEM_FREE) cout << "  MEM_FREE" << endl;
	if (memoryInfo.State & MEM_RESERVE) cout << "  MEM_RESERVE" << endl;
	cout << "Тип страницы: 0x" << hex << memoryInfo.Type << endl;
	if (memoryInfo.Type & MEM_IMAGE) cout << "  MEM_IMAGE" << endl;
	if (memoryInfo.Type & MEM_MAPPED) cout << "  MEM_MAPPED" << endl;
	if (memoryInfo.Type & MEM_PRIVATE) cout << "  MEM_PRIVATE" << endl;
	cout << dec;
}
void separateReserveCommit() {
	PVOID pMemory = 0;
	PVOID pBaseAddress = 0;
	DWORD dwSize = 0;
	cout << "Базовый адрес (0 для автоматического задания адреса системой): 0x";
	cin >> hex >> pBaseAddress;

	cout << "Размер (в байтах): ";
	cin >> dec >> dwSize;

	pMemory = VirtualAlloc(pBaseAddress, dwSize, MEM_RESERVE,
PAGE_READWRITE);
	if (pMemory != NULL) {
		cout << "\n\nПамяти зарезервирована: \n";
		virtualPageStatus((DWORD)pMemory);
	}
	else {
		cout << "Не удалось выделить память!" << endl;
		return;
	}

	pMemory = VirtualAlloc(pMemory, dwSize, MEM_COMMIT, PAGE_READWRITE);
	if (pMemory != NULL) {
		cout << "\n\nПамять использована:\n";
		virtualPageStatus((DWORD)pMemory);
	}
	else {
		cout << "Не удалось выделить память!" << endl;
		return;
	}
	//VirtualFree(pMemory, dwSize, MEM_DECOMMIT);
	//cout << "\n\nПамять вышла из использования: \n";
```

```
	//virtualPageStatus((DWORD)pMemory);

	//VirtualFree(pMemory, 0, MEM_RELEASE);
	//cout << "\n\nПамять очищена: \n";
	//virtualPageStatus((DWORD)pMemory);
}
void simultaneousReserveCommit() {
	PVOID pMemory = 0;
	PVOID pBaseAddress = 0;
	DWORD dwSize = 0;
	cout << "Базовый адрес (0 для автоматического задания адреса системой): 0x";
	cin >> hex >> pBaseAddress;

	cout << "Размер (в байтах): ";
	cin >> dec >> dwSize;

	pMemory = VirtualAlloc(pBaseAddress, dwSize, MEM_RESERVE | MEM_COMMIT,
PAGE_READWRITE);
	if (pMemory != NULL) {
		cout << "\n\nПамять выделена: \n";
		virtualPageStatus((DWORD)pMemory);
	}
	else {
		cout << "Не удалось выделить память!" << endl;
		return;
	}

	//VirtualFree(pMemory, 0, MEM_RELEASE);
	//cout << "\n\nПамять очищена: \n";
	//virtualPageStatus((DWORD)pMemory);
}
void writeData() {
	PVOID pMemory = 0;
	PVOID pBaseAddress = 0;
	DWORD dwSize = 0;
	int data = 0;
	cout << "Базовый адрес (0 для автоматического задания адреса системой): 0x";
	cin >> hex >> pBaseAddress;
	MEMORY_BASIC_INFORMATION memoryInfo;
	VirtualQuery((LPCVOID)pBaseAddress, &memoryInfo, sizeof(memoryInfo));
	if (memoryInfo.Protect & PAGE_NOACCESS) {
		cout << "Память защищена" << endl;
		return;
	}

	cout << "Данные: ";
	cin >> dec >> data;
```

```
	if (pBaseAddress) {
		memcpy(pBaseAddress, &data, sizeof(int));
		cout << "Чтение памяти (0x" << hex << pBaseAddress << dec << "): " <<
(*(PDWORD)pBaseAddress) << endl;
	}
	else {
		pMemory = VirtualAlloc(pBaseAddress, sizeof(int), MEM_RESERVE |
MEM_COMMIT, PAGE_READWRITE);
		memcpy(pMemory, &data, sizeof(int));
		cout << "Чтение памяти (0x" << hex << pMemory << dec << "): " <<
(*(PDWORD)pMemory) << endl;
		//VirtualFree(pMemory, 0, MEM_RELEASE);
	}
}
void protectVirtualPage() {
	PVOID pMemory = 0;
	PVOID pBaseAddress = 0;
	DWORD dwSize = 0;
	cout << "Базовый адрес (0 для автоматического задания адреса системой): 0x";
	cin >> hex >> pBaseAddress;

	if (pBaseAddress) {
		DWORD flOldProtect = 0;
		VirtualProtect(pBaseAddress, sizeof(int), PAGE_NOACCESS, &flOldProtect);
		cout << "\n\nПамять защищена: \n";
		virtualPageStatus((DWORD)pBaseAddress);
	}
	else {
		pMemory = VirtualAlloc(pBaseAddress, sizeof(int), MEM_RESERVE |
MEM_COMMIT, PAGE_READWRITE);
		if (pMemory != NULL) {
			cout << "\n\nПамять выделена: \n";
			virtualPageStatus((DWORD)pMemory);
		}

		DWORD flOldProtect = 0;
		VirtualProtect(pMemory, sizeof(int), PAGE_NOACCESS, &flOldProtect);
		cout << "\n\nПамять защищена: \n";
		virtualPageStatus((DWORD)pMemory);

		VirtualFree(pMemory, 0, MEM_RELEASE);
	}
}

void freeVirtualPage(DWORD address) {
	PVOID pMemory = 0;

	if (!address) {
```

```
		SYSTEM_INFO systemInfo;
		GetSystemInfo(&systemInfo);
		cout << "Введите адрес в диапазоне от 0x" <<
systemInfo.lpMinimumApplicationAddress
			<< " до 0x" << systemInfo.lpMaximumApplicationAddress << "): 0x";
		cin >> hex >> address;
	}

	VirtualFree((LPVOID)address, 0, MEM_RELEASE);
	cout << "\n\nПамять очищена: \n";
	virtualPageStatus((DWORD)address);
}
```

Читатель (4.3)

```
#include <iostream>
#include <windows.h>

using namespace std;

int main()
{
	setlocale(0, ".1251");
	char answer;
	HANDLE hMapFile;
	PVOID lpMapAddress;
	char data[1024];

	hMapFile = OpenFileMappingA(FILE_MAP_ALL_ACCESS, FALSE,
"myMappedFile");
	if (hMapFile != INVALID_HANDLE_VALUE)
		cout << "Проекция открыта\n";

	lpMapAddress = MapViewOfFile(hMapFile, FILE_MAP_ALL_ACCESS, 0, 0, 0);
	if (lpMapAddress == 0){
		cerr << "Не удаётся открыть проекцию файла\n";
		system("pause");
		return 1;
	}
	memcpy(data, (char*)lpMapAddress, 1024);
	cout << "Данные по адресу (0x"<< lpMapAddress <<"): " << data << "\n";

	cout << "Закрыть проекцию? (y/n) ";
	cin >> answer;
	if (answer == 'y') {
		UnmapViewOfFile(lpMapAddress);
	}
	CloseHandle(hMapFile);
	return 0;
}
```

Писатель(4.3)

```
#include <iostream>
#include <windows.h>
#include <stdio.h>

using namespace std;

int main()
{
	setlocale(0, ".1251");
	char answer = 0;
	HANDLE hFile;
	HANDLE hMapFile;
	LPVOID lpMapAddress;
	wchar_t filename[250] = L"tempTextFile.txt";
	char data[1024];
	cout << "Данные для передачи: ";
	cin >> data;

	hFile = CreateFile(filename, GENERIC_ALL, FILE_SHARE_READ |
FILE_SHARE_WRITE, NULL, CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL,
NULL);
	if (hFile != INVALID_HANDLE_VALUE)
		cout << "Файл успешно создан" << endl;

	hMapFile = CreateFileMappingA(hFile, NULL, PAGE_READWRITE, 0, 1024,
"myMappedFile");

	if (hMapFile != INVALID_HANDLE_VALUE)
		cout << "Маппинг объекта создан" << endl;

	lpMapAddress = MapViewOfFile(hMapFile, FILE_MAP_ALL_ACCESS, 0, 0, 0);

	memcpy(lpMapAddress, data, strlen(data));
	cout << "Адрес проекции: 0x" << lpMapAddress <<endl;

	cout << "Закрыть проекцию? (y/n) ";
	cin >> answer;
	if (answer == 'y') {
		UnmapViewOfFile(lpMapAddress);
	}
	CloseHandle(hMapFile);
	CloseHandle(hFile);
	return 0;
}
```