**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра ВТ**

отчет

**По Лабораторной работе №4**

**По дисциплине «Операционные системы»**

Тема: Управление памятью

|  |  |  |
| --- | --- | --- |
| Студент гр. 7307 |  | Торопов В.А. |
| Преподаватель |  | Тимофеев А.В. |

Санкт-Петербург

2019

**Цель работы**

Исследовать механизмы управления виртуальной памятью Win32.

**Основные теоретические положения**

LiveKD – оболочка для запуска отладчика ядра Microsoft KD, входящего в Windows Driver Kit (WDK). LiveKD позволяет выполнять просмотр моментального снимка работающей локальной системы, не перезагружая систему в режим отладки.

**Обработка экспериментальных данных**

**Задание 4.1.** Исследовать процесс трасляции виртуальных адресов в 64-разрядной ОС Windows.

С помощью команды !memusage 0x8, отобразим информацию о списках страниц виртуальной памяти.

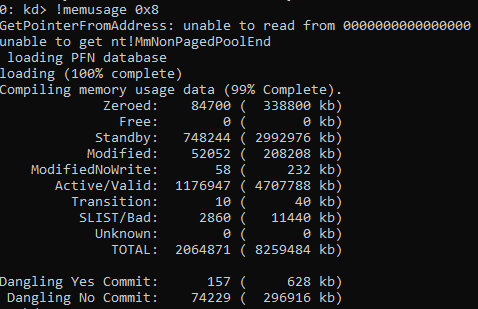


Рисунок 1. Результат команды !memusage

Получим базовые сведения об управлении памятью, доступные через соответствующие счетчики производительности, с помощью команды !vm.

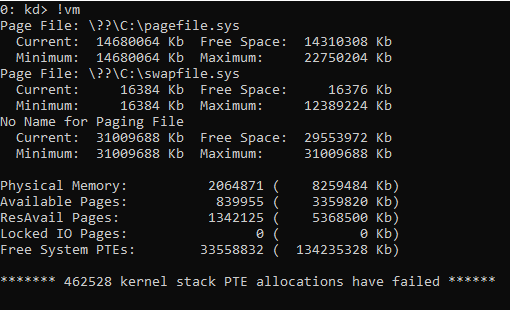


Рисунок 2. Вывод команды !vm (1 часть)

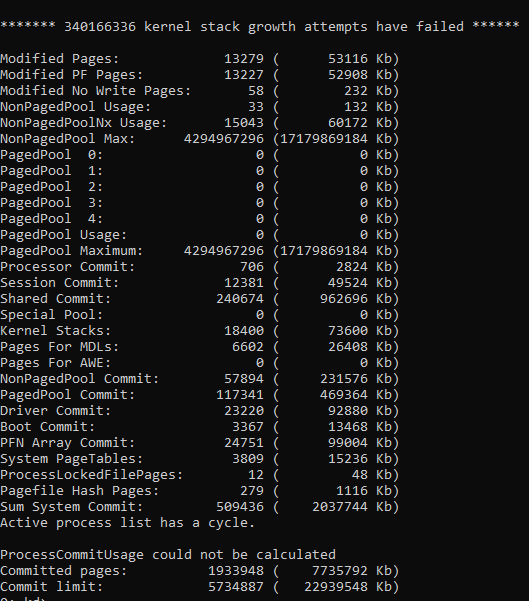


Рисунок 3. Вывод команды !vm (2 часть)

Для дальнейшей работы работы был выбран процесс notepad.exe с PID = 0x3798.

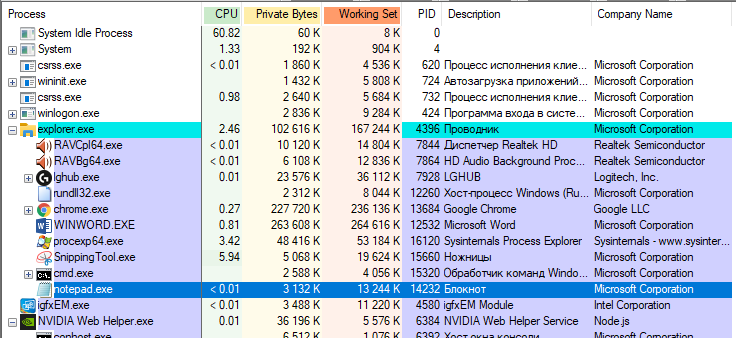


Рисунок 4. Выбранный процесс

Выведем информацию о процессе с помощью команды !process, т.к. нужно краткая информация используется флаг 1.

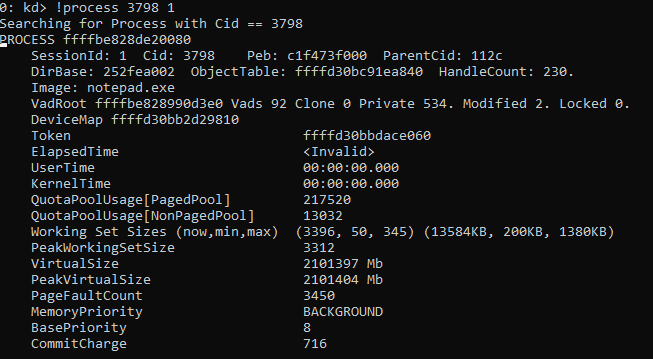


Рисунок 5. Результат выполнения команды !process 3798 1

В выбранном процессе нам понадобятся значения DirBase = 0x252fea002 и VadRoot = 0xffffbe828990d3e0. Затем получим с помощью команды !vad ffffbe828990d3e0 краткую информацию о регионах виртуального адресного пространства выбранного процесса.

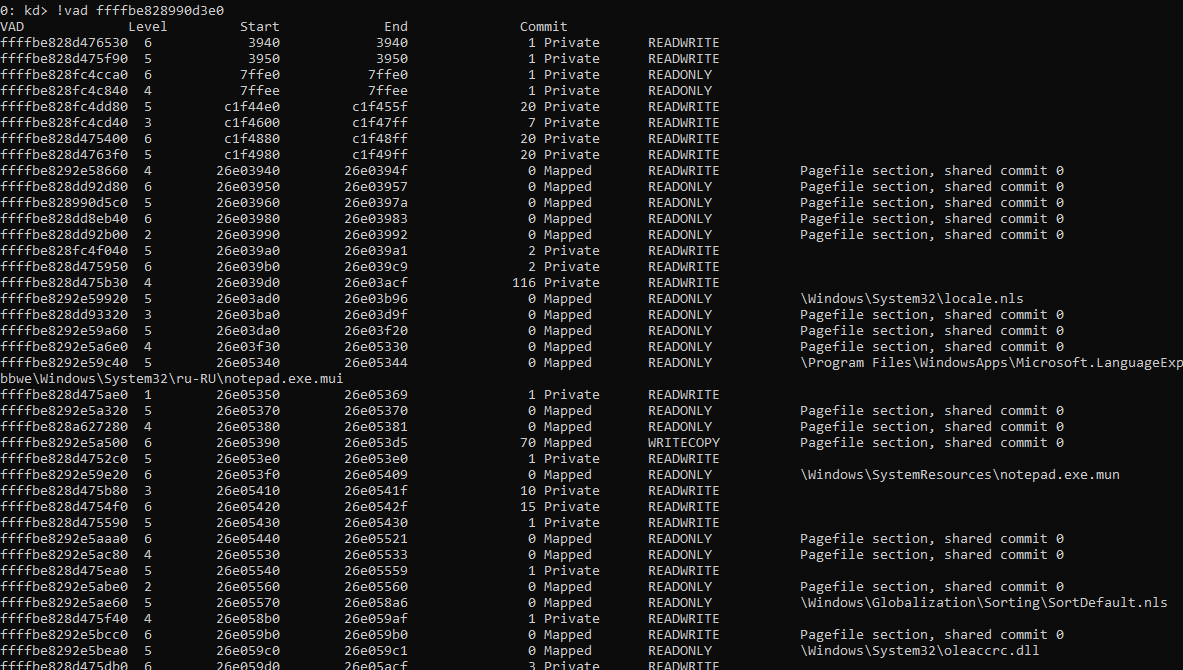


Рисунок 6. Результат команды !vad ffffbe828990d3e0

Найдем адресное пространства notepad.exe. Его виртуальный адрес 0xffffbe828990bae0.



Рисунок 7. Результат !vad ffffbe828990d3e0 с файлом notepad.exe

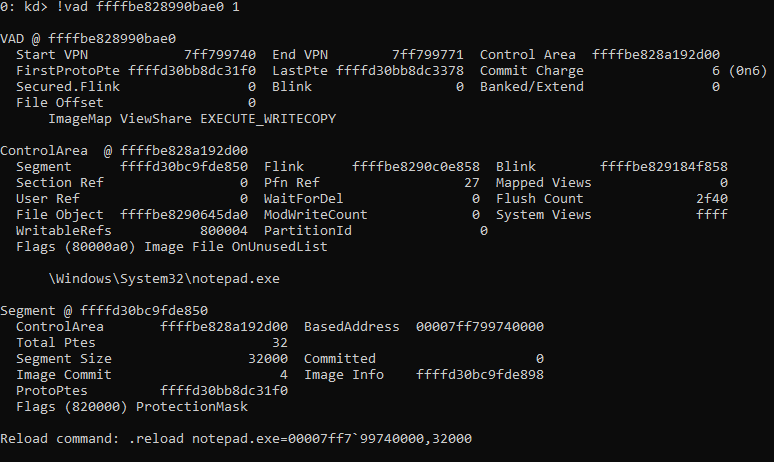


Рисунок 8. Результат команды !vad ffffbe828990bae0 1.

Воспользуемся командой перезагрузки контекста процессора в режиме отладчика .context.

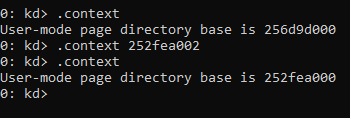


Рисунок 9. Смена контекста

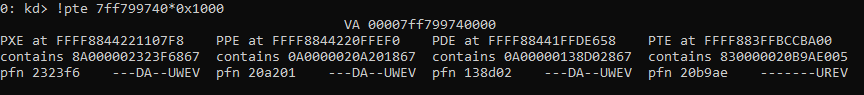


Рисунок 10. Результат выполнения команды !pte

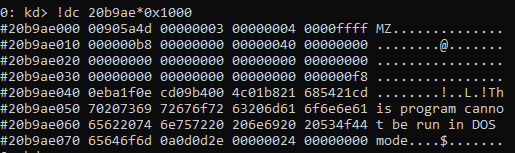


Рисунок 11. Результат выполнения команды !dc

Вывод: С помощью средств livekd можно получить доступ к просмотру виртуальной памяти, выделенной для процесса, а также физической памяти, которая выделена для виртуальной.

**Задание 4.2.**

По заданию была написана программа. Далее будут представлены результаты работы этой программы по отдельным пукнктам задания:

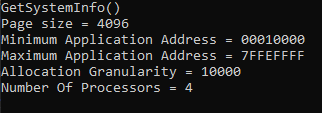


Рисунок 11. Получение информации о вычислительной системе

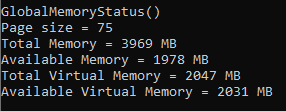


Рисунок 12. Определение статуса виртуальной памяти

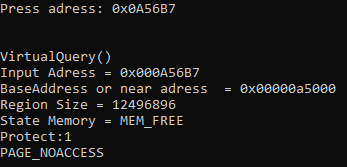


Рисунок 13. Определение состояния конкретного участка памяти

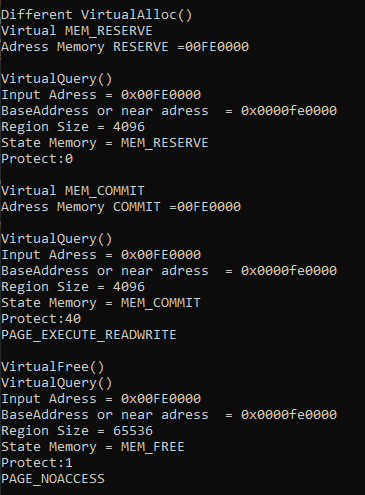


Рисунок 14. Раздельное резервирование региона и передачу ему физической памяти

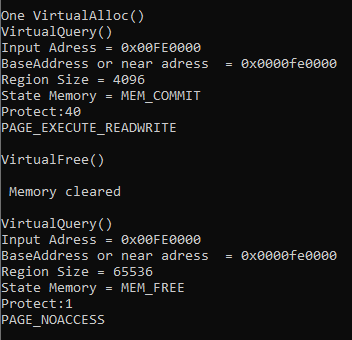


Рисунок 15. Одновременное резервирование региона и передача ему физической памяти

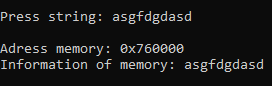


Рисунок 16. Запись данных в ячейки памяти по заданным с клавиатуры адресам

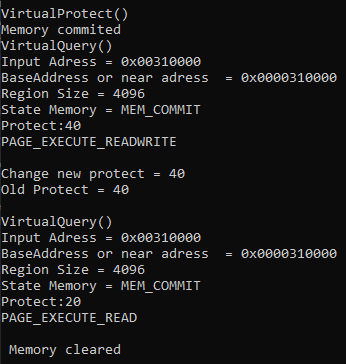


Рисунок 17. Установка защиты доступа для заданного региона памяти

Затем перезапустим приложение и проведем анализ с помощью livekd: Pid = 8396 = 0x20cc.

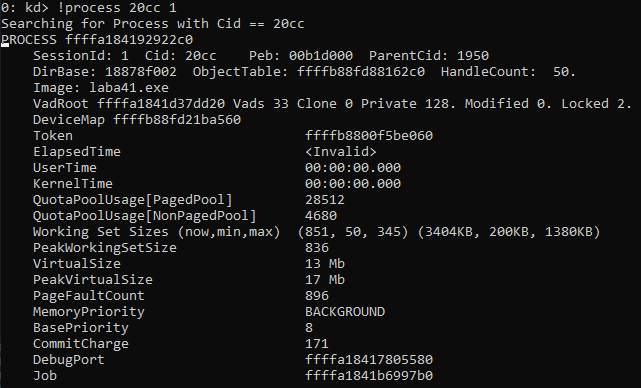


Рисунок 18. Снимок информации процесса во время выбора в меню приложения

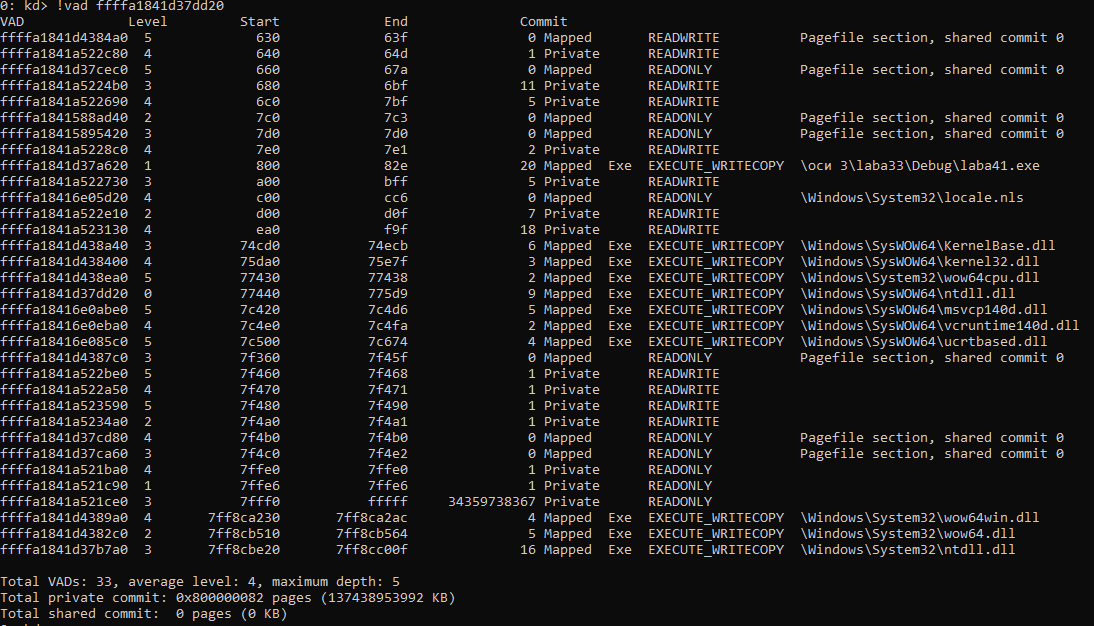


Рисунок 19. Виртуальное адресное пространство во время выбора в меню приложения

Рассмотрим 4 пункт с раздельными резервированием и передачей региона памяти.

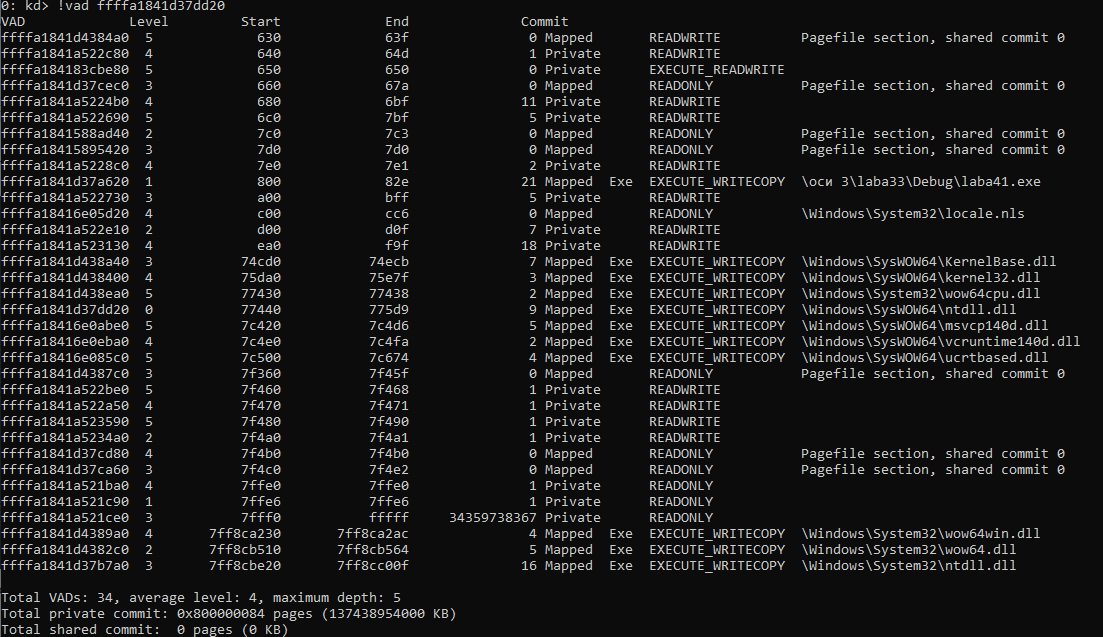


Рисунок 20. Виртуальное адресное пространство раздельного после резервирования региона:

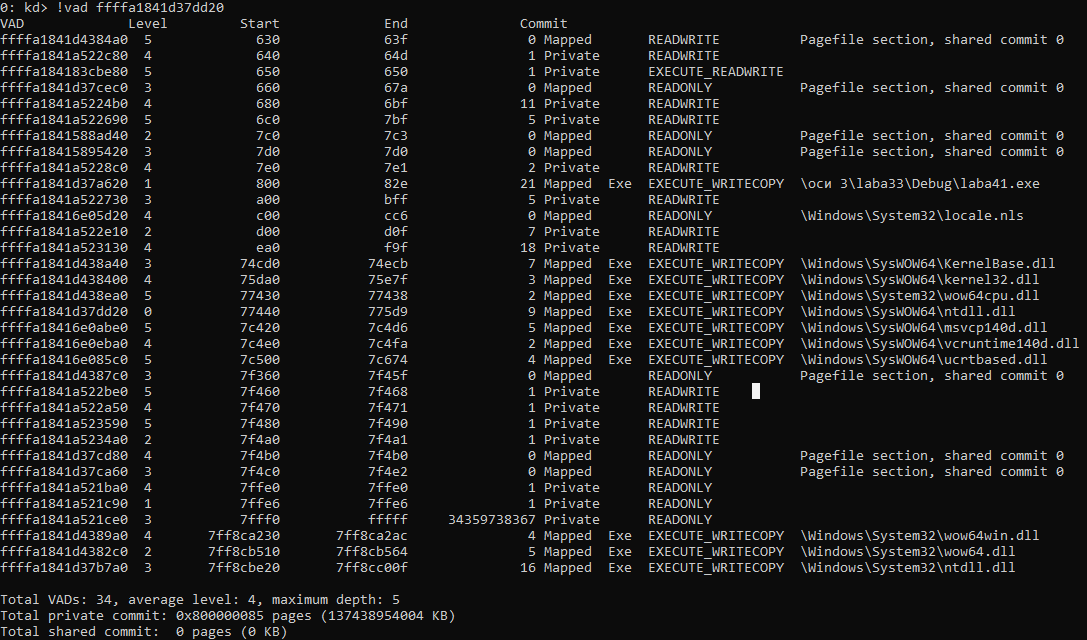


Рисунок 21. Виртуальное адресное пространство раздельного после после передачи зарезервированного региона

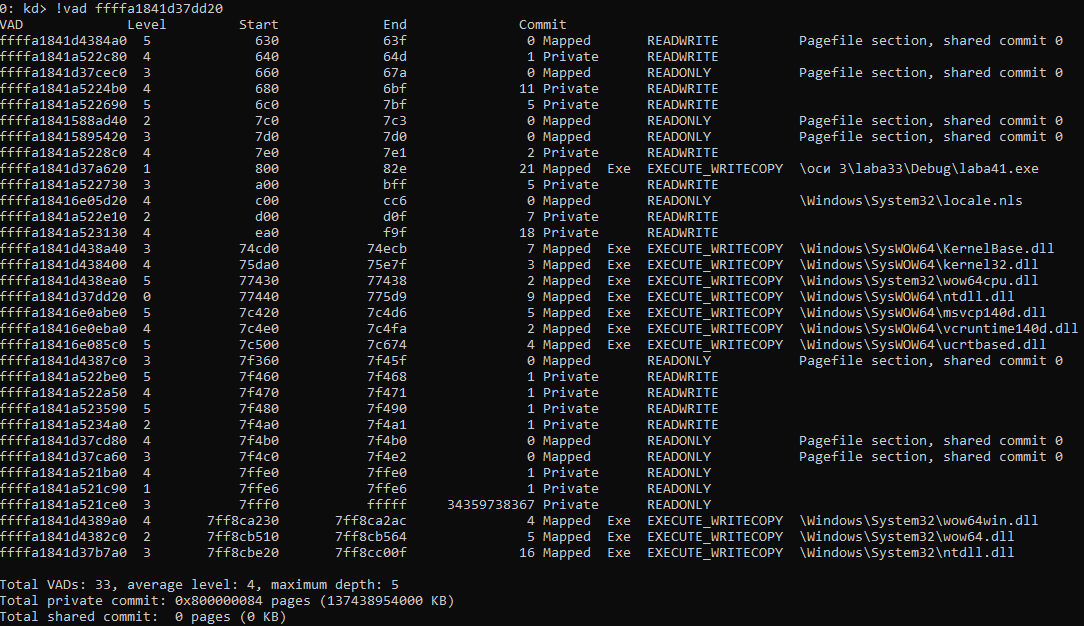


Рисунок 22. Виртуальное адресное после освобождения региона

Как видно на рисунках при резервировании и передаче количество адресного пространства увеличилось на 1. Появился и затем изменялся участок с началом 650 и концом 650. На рисунке 20 видно что память лишь выделена т.к. столбец Commit = 0, а после передачи памяти на рисунке 21 Commit = 1. После освобождения памяти, количество виртуальных адресов возвращается к первоначальному количеству(33).

Рассмотрим 5 пункт с одновременным резервированием и передачей региона памяти.

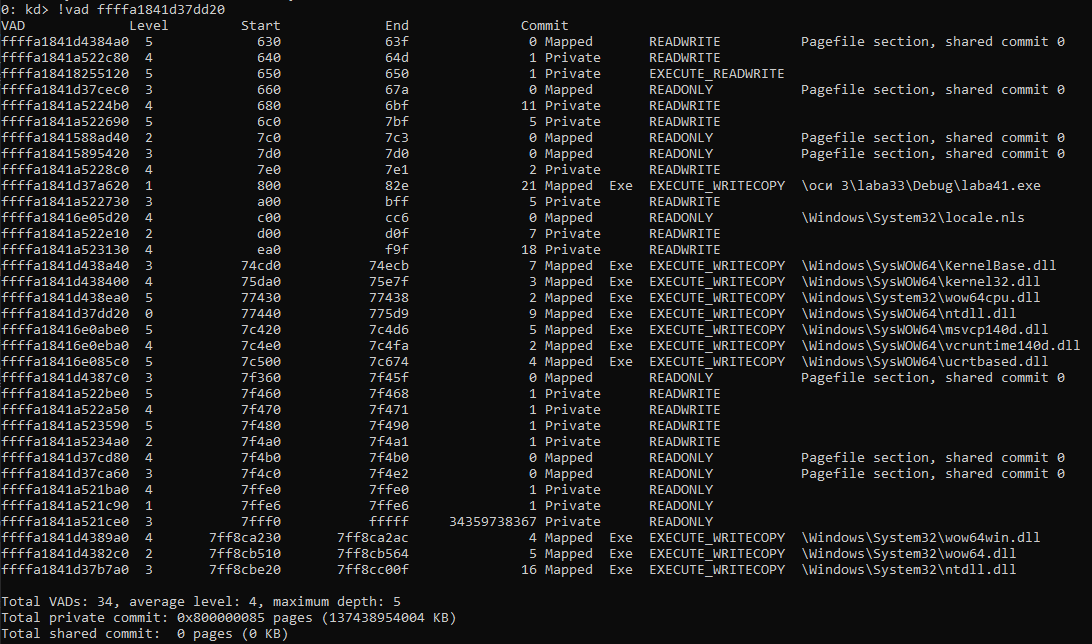


Рисунок 23. Виртуальное адресное пространство одновременного резервирования и передачи региона памяти.

Как видно на рисунках при одновременном резервировании и передаче количество адресного пространства увеличилось на 1. Появился и затем изменялся участок с началом 650 и концом 650. На рисунке 20 видно что память выделена и передана т.к. Commit = 1.

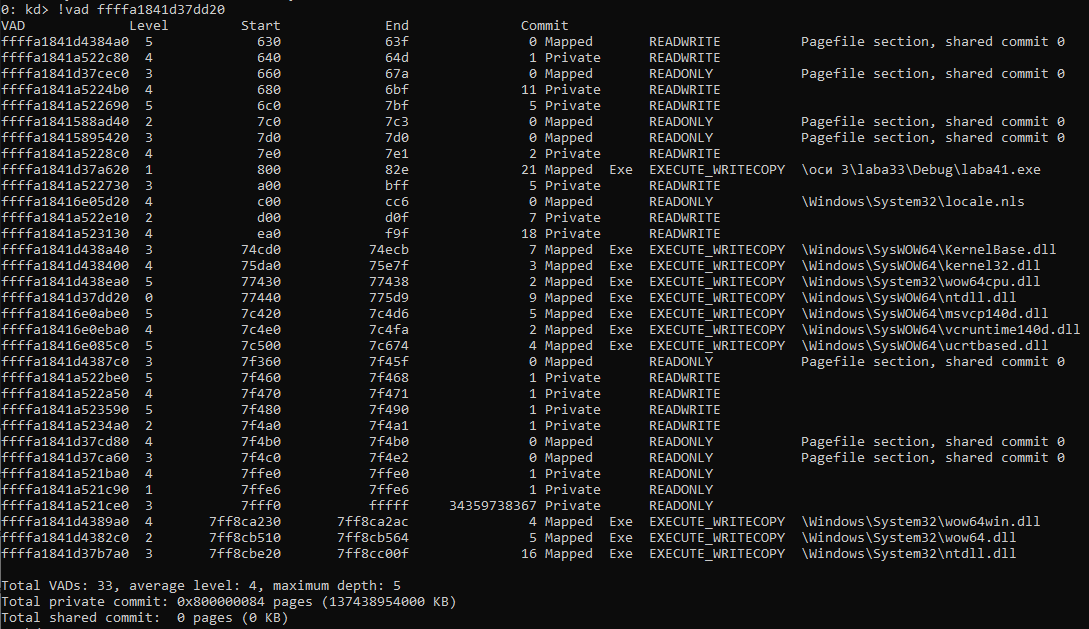


Рисунок 24. Завершение выполнения команды с ошибкой

После освобождения памяти, количество виртуальных адресов возвращается к первоначальному количеству(33).

Давайте запишем данные в выделенное пространство. Это 6 пункт задания, но у меня это было вынесено в 7 пункт меню – это никак не влияет на работу программы. На следующих слайдах мы запишем строку которую запишем в адресное пространство и увидим адрес выделенный под данные. Затем мы последовательно просмотрим через livekd содержимое нашей памяти.

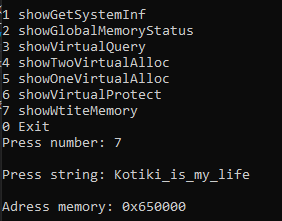


Рисунок 25. Выбор пункта и ввод строки

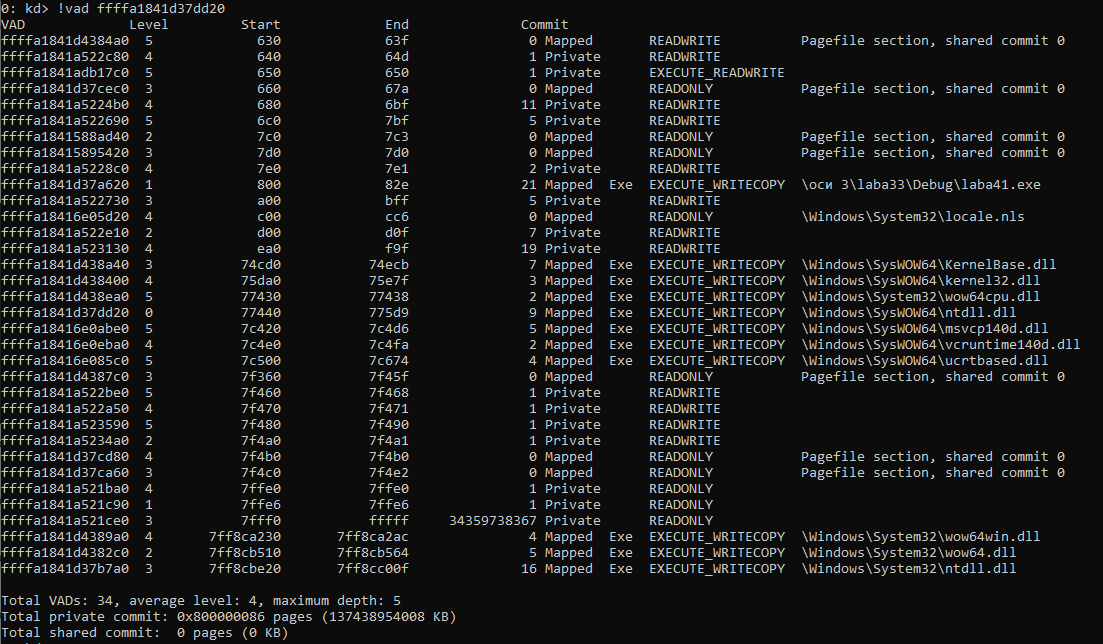


Рисунок 26. Виртуальное адресное пространство после выделения памяти и записи информации.

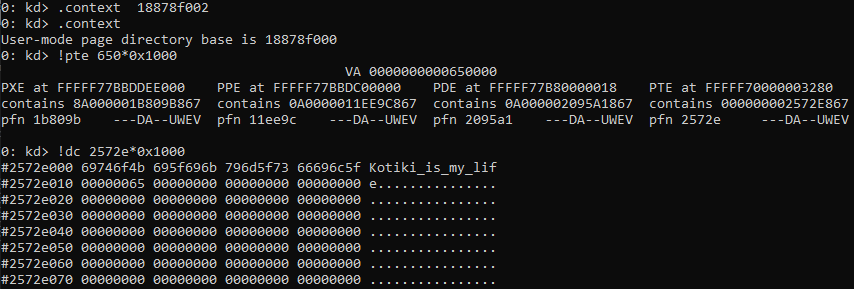


Рисунок 27. Смена контекста, и результаты работы команд !pte и !dc

Как видно информация записана.

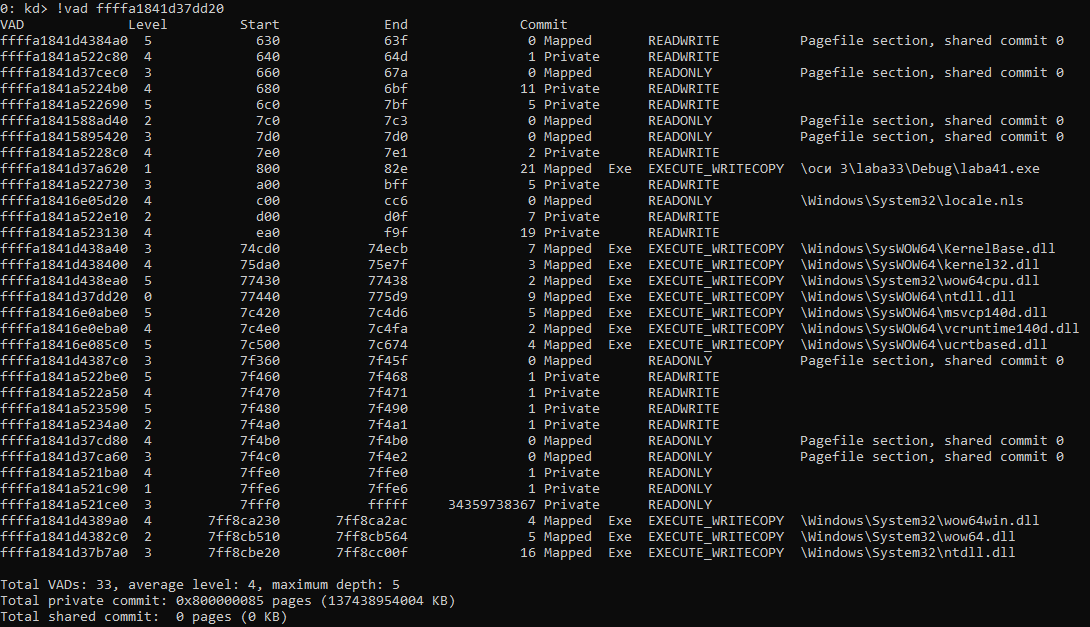


Рисунок 28. Очистка памяти после записи

Количество участков вернулось в прежнее количество (33).

Установка защиты доступа для заданного региона памяти.

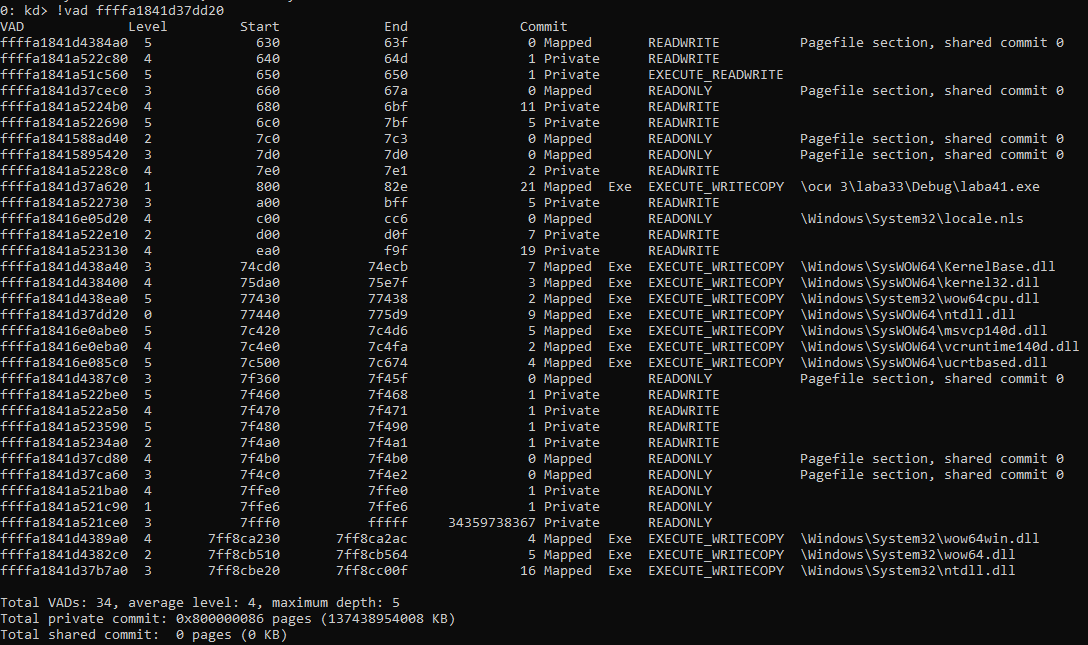


Рисунок 29. Завершение выполнения команды с ошибкой

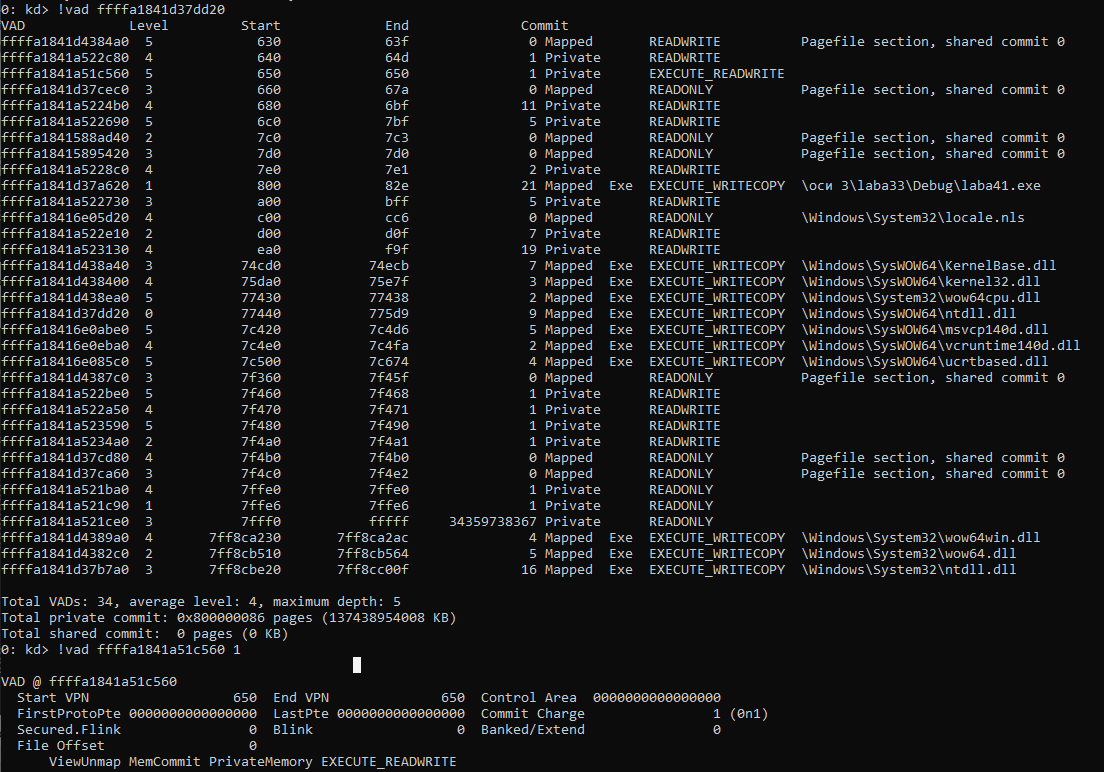
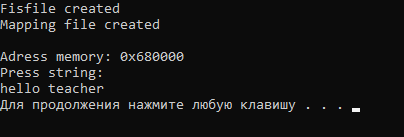


Рисунок 30. Завершение выполнения команды с ошибкой

Вывод: Функции winapi предоставляют полный интерфейс для работы с памятью системы. С помощью утилиты livekd возможно проверить работу программы с памятью.

**Задание 4.3.**

Были написаны 2 приложения. Первое открывает файл в общее виртуальное пространство и записывает в него информацию. Второе открывает файл в общем пространстве и считывает с него информацию. Далее представлены работы двух приложений. Первое после записи информации ожидает пока сработает 2 приложение.



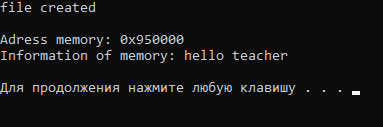


Рисунок 31а и 31б. Работа первого и второго приложения соответственно

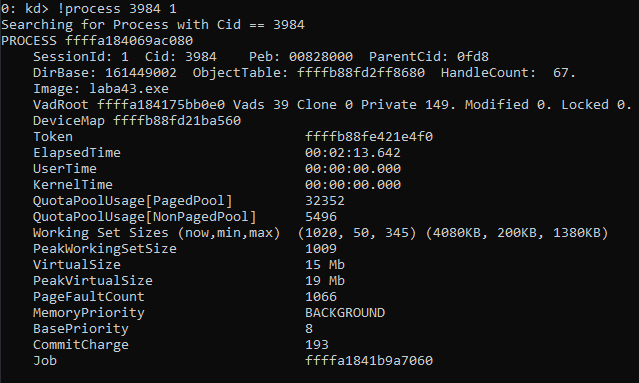


Рисунок 32. Информация о процессе 1 приложения

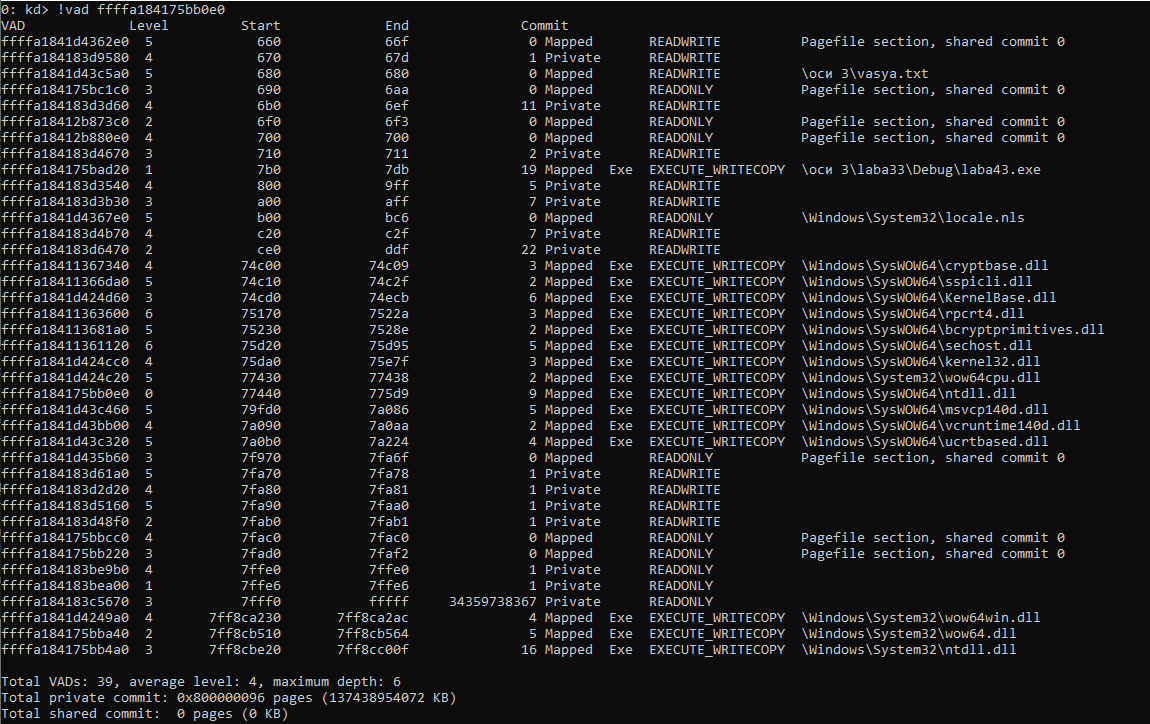


Рисунок 33. Виртуальное адресное пространство 1 приложения.

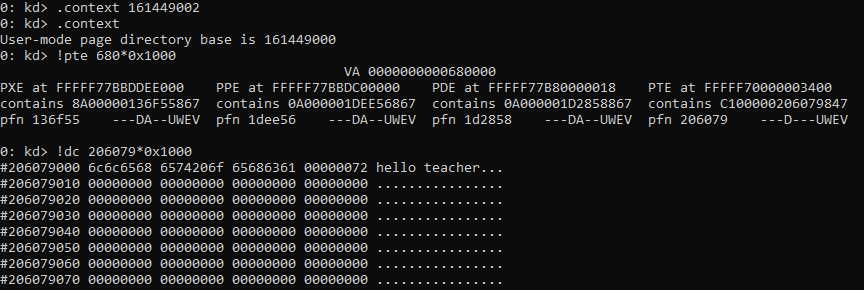


Рисунок 34. Результат поиска записанной информации в память 1 приложением

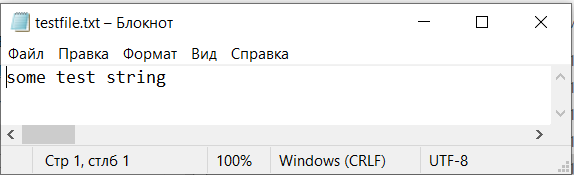


Рисунок 35. Информация в файле после работы приложений.

Вывод: С помощью механизма “проецирования” возможно отображать частично или полностью файлы в оперативной памяти, что позволяет работать с ними нескольким приложениям в системе реального времени.

**Выводы**

С помощью механизма “проецирования” возможно отображать частично или

полностью файлы в оперативной памяти, что позволяет работать с ними

быстрее и удобнее.

Приложение 1

#include <windows.h>  
#include <iostream>  
#include <string>  
#include <iomanip>  
using namespace std;  
  
void show(char\* a) {  
 cout << "|";  
 for (int i = 0; i < 10; i++)  
 cout << (char)a[i];  
  
 cout << "|\n ";  
 for (int i = 0; i < 10; i++)  
 cout << i;  
 cout << endl;  
}  
  
  
void showGetSystemInf() {  
 int n;  
 char dd[4];  
  
 SYSTEM\_INFO info;  
  
 cout << "GetSystemInfo()" << endl;  
  
 GetSystemInfo(&info);  
  
  
  
 cout << "Page size = " << (unsigned long)info.dwPageSize << endl;  
 cout << "Minimum Application Address = " << hex << (unsigned long\*)info.lpMinimumApplicationAddress << endl;  
 cout << "Maximum Application Address = " << hex << info.lpMaximumApplicationAddress << endl;  
  
 cout << "Allocation Granularity = " << hex << info.dwAllocationGranularity << dec << endl;  
 cout << "Number Of Processors = " << (unsigned long)info.dwNumberOfProcessors << endl;  
  
 /\*cout << "wProcessorLevel = " << info.wProcessorLevel << endl;  
 cout << "dwActiveProcessorMask = " << (unsigned long) info.dwActiveProcessorMask << endl;  
 cout << "wProcessorArchitecture = " << (unsigned long) info.wProcessorArchitecture << endl;  
 \*/  
  
  
}  
  
void showGlobalMemoryStatus() {  
 int n;  
 char dd[4];  
 MEMORYSTATUSEX mem;  
 int x = 1 << 25;  
 cout << "GlobalMemoryStatus()" << endl;  
 mem.dwLength = sizeof(mem);  
 GlobalMemoryStatusEx(&mem);  
 //printf("ERROR %d \n", GetLastError());  
 cout << "Page size = " << (unsigned long)mem.dwMemoryLoad << endl;  
 cout << "Total Memory = " << (unsigned long)mem.ullTotalPhys / 1024 / 1024 << " MB" << endl;  
 cout << "Available Memory = " << (unsigned long)mem.ullAvailPhys / 1024 / 1024 << " MB" << endl;  
 cout << "Total Virtual Memory = " << (unsigned long)mem.ullTotalVirtual / 1024 / 1024 << " MB" << endl;  
 cout << "Available Virtual Memory = " << (unsigned long)mem.ullAvailVirtual / 1024 / 1024 << " MB" << endl;  
  
}  
  
  
void showVirtualQuery(void\* adress) {  
 int n;  
 char dd[4];  
 MEMORY\_BASIC\_INFORMATION mem;  
 void\* adressMemory = adress;  
 cout << "VirtualQuery()" << endl;  
  
 cout << "Input Adress = 0x" << hex << adressMemory << endl;  
 VirtualQuery(adressMemory, &mem, sizeof(MEMORY\_BASIC\_INFORMATION));  
  
  
 cout << "BaseAddress or near adress = " << "0x" << setw(10) << setfill('0') << fixed << (unsigned long long)mem.BaseAddress << endl;  
 cout << dec << "Region Size = " << (unsigned long)mem.RegionSize << endl;  
 cout << "State Memory = ";  
 switch (mem.State) {  
 case 0x1000:  
 cout << "MEM\_COMMIT" << endl;  
 break;  
 case 0x10000:  
 cout << "MEM\_FREE" << endl;  
 break;  
 case 0x2000:  
 cout << "MEM\_RESERVE" << endl;  
 break;  
  
 }  
 cout << "Protect:" << hex << (unsigned long)mem.Protect << dec << endl;  
 switch (mem.Protect) {  
 case PAGE\_READONLY:  
 cout << "PAGE\_READONLY" << endl;  
 break;  
 case PAGE\_READWRITE:  
 cout << "PAGE\_READWRITE" << endl;  
 break;  
 case PAGE\_NOACCESS:  
 cout << "PAGE\_NOACCESS" << endl;  
 break;  
 case PAGE\_WRITECOPY:  
 cout << "PAGE\_WRITECOPY" << endl;  
 break;  
 case PAGE\_EXECUTE:  
 cout << "PAGE\_EXECUTE" << endl;  
 break;  
 case PAGE\_EXECUTE\_READ:  
 cout << "PAGE\_EXECUTE\_READ" << endl;  
 break;  
 case PAGE\_EXECUTE\_READWRITE:  
 cout << "PAGE\_EXECUTE\_READWRITE" << endl;  
 break;  
 case PAGE\_EXECUTE\_WRITECOPY:  
 cout << "PAGE\_EXECUTE\_WRITECOPY" << endl;  
 break;  
 case PAGE\_GUARD:  
 cout << "PAGE\_GUARD" << endl;  
 break;  
 case PAGE\_NOCACHE:  
 cout << "PAGE\_NOCACHE" << endl;  
 break;  
  
 }  
 cout << endl;  
 //cout << "Type = " << (unsigned long)mem.Type << endl<<endl;  
  
 //printf("ERROR %d \n\n", GetLastError());  
  
}  
  
  
void showTwoVirtualAlloc(void\* adress) {  
 int n;  
 char dd[4];  
 int\* a = NULL;  
 MEMORYSTATUS mem;  
 MEMORY\_BASIC\_INFORMATION memo;  
 int x = 1 << 25;  
 cout << "Different VirtualAlloc()" << endl;  
 cout << "Virtual MEM\_RESERVE" << endl;  
  
 a = (int\*)VirtualAlloc(adress, 10 \* sizeof(int), MEM\_RESERVE, PAGE\_EXECUTE\_READWRITE);  
 if (a == NULL) {  
 cout << "Error virtual alloc";  
 return;  
 }  
 cout << "Adress Memory RESERVE =" << hex << a << dec << endl << endl;  
  
  
 showVirtualQuery((void\*)a);  
 cout << "Virtual MEM\_COMMIT" << endl;  
  
 a = (int\*)VirtualAlloc(a, 10 \* sizeof(int), MEM\_COMMIT, PAGE\_EXECUTE\_READWRITE);  
 cout << "Adress Memory COMMIT =" << hex << a << dec << endl << endl;  
  
 showVirtualQuery(a);  
 //printf("ERROR %d \n", GetLastError());  
  
 //cout << "VirtualFree()" << endl;  
 //VirtualFree(a, 0, MEM\_RELEASE);  
 //showVirtualQuery(a);  
 //cout << " Memory cleared" << endl;  
}  
  
void showOneVirtualAlloc(void\* adress) {  
 int n;  
 char dd[4];  
 int\* a = NULL;  
 MEMORYSTATUS mem;  
 int x = 1 << 25;  
 cout << "One VirtualAlloc()" << endl;  
  
 a = (int\*)VirtualAlloc(adress, 20 \* sizeof(int), MEM\_COMMIT | MEM\_RESERVE, PAGE\_EXECUTE\_READWRITE);  
 if (a == NULL) {  
 cout << "Error virtual alloc";  
 return;  
 }  
 showVirtualQuery(a);  
  
 //printf("ERROR %d \n", GetLastError());  
  
 /\*cout << "VirtualFree()" << endl<<endl;  
 VirtualFree(a, 0, MEM\_RELEASE);  
  
 cout << " Memory cleared" << endl;  
 cout << endl;  
 showVirtualQuery(a);\*/  
}  
  
  
  
void showVirtualProtect(void \*adress) {  
 int n;  
 unsigned long fdwNewProtect, fdwOldProtect;  
 string s;  
 MEMORYSTATUS mem;  
 int x = 1 << 25;  
 void\* a = adress;  
 boolean f;  
 //char\* a = (char\*)VirtualAlloc(a, 20 \* sizeof(int), MEM\_COMMIT | MEM\_RESERVE, PAGE\_EXECUTE\_READWRITE);  
 if (a == NULL) {  
 cout << "Error virtual alloc";  
 }  
 else {  
  
 cout << "VirtualProtect()" << endl;  
 cout << "Memory commited" << endl;  
 showVirtualQuery(a);  
 cout << "Press new protect: ";  
 cin >> fdwNewProtect;  
 cout << endl;  
 printf("\nNew Protect = %x\n\n", fdwNewProtect);  
 f = VirtualProtect(a, 20 \* sizeof(int), (DWORD)fdwNewProtect, &fdwOldProtect);  
 cout << "Operation is " << (f?"Success" : "Wrong");  
 if (f) {  
 //printf("ERROR %d \n", GetLastError());  
 printf("\nOld Protect = %x\n\n", fdwOldProtect);  
  
  
 showVirtualQuery(a);  
 }  
 /\*  
 VirtualFree(a, 0, MEM\_RELEASE);  
  
  
 //printf("ERROR %d \n", GetLastError());  
  
 cout << " Memory cleared" << endl;\*/  
 }  
  
  
  
}  
  
  
  
  
  
  
int main()  
{  
 int x = -1;  
 int z = -2;  
 void\* adr;  
 char y = 'a';  
 string s;  
 while (x != 0) {  
 cout << "1 showGetSystemInf" << endl;  
 cout << "2 showGlobalMemoryStatus " << endl;  
 cout << "3 showVirtualQuery" << endl;  
 cout << "4 showTwoVirtualAlloc" << endl;  
 cout << "5 showOneVirtualAlloc" << endl;  
 cout << "6 showVirtualProtect" << endl;  
 cout << "7 showWtiteMemory" << endl;  
 cout << "8 VirtualFree" << endl;  
 cout << "0 Exit" << endl;  
 cout << "Press number: ";  
 cin >> x;  
  
 cout << endl;  
 switch (x) {  
 case 1:  
 showGetSystemInf();  
  
 break;  
 case 2:  
 showGlobalMemoryStatus();  
 break;  
  
 case 3:  
 cout << "Press adress: ";  
 cin >> adr;  
 cout << "\n\n";  
 showVirtualQuery(adr);  
 break;  
 case 4:  
 cout << "Press adress: ";  
 cin >> adr;  
 cout << "\n\n";  
 showTwoVirtualAlloc(adr);  
 break;  
 case 5:  
 cout << "Press adress: ";  
 cin >> adr;  
 cout << "\n\n";  
 showOneVirtualAlloc(adr);  
 break;  
 case 6:  
 cout << "Press adress: ";  
 cin >> adr;  
 cout << "\n\n";  
 showVirtualProtect(adr);  
 break;  
  
 case 7: {  
 cout << "Press adress: ";  
 cin >> adr;  
 cout << "\n\n";  
// char\* a = (char\*)VirtualAlloc(adr, 20 \* sizeof(char), MEM\_COMMIT | MEM\_RESERVE, PAGE\_EXECUTE\_READWRITE);  
 char\* a = (char\*)adr;  
 cout << "Press string: ";  
 cin >> s;  
 cout << "\n" << "Adress memory: 0x" << hex << (unsigned long)a << dec << endl;  
  
 CopyMemory(a, s.c\_str(), sizeof(char) \* s.length());  
  
 cout << "Information of memory: ";  
 for (int i = 0; i < 20; i++)  
 cout << a[i] ;  
 cout << "\n\n";  
  
 //VirtualFree(a, 0, MEM\_RELEASE);  
 break;  
 }  
 case 8:  
 cout << "Press adress: ";  
 cin >> adr;  
 VirtualFree(adr, 4096, MEM\_RELEASE);  
 }  
  
 cout << endl << endl;  
  
 }  
}

Приложение 2

#include <windows.h>  
#include <iostream>  
#include <string>  
  
using namespace std;  
  
int main()  
{  
 SYSTEM\_INFO info;  
 HANDLE handleFisFile, handleMappingFile;  
 handleFisFile = CreateFileA("D://NIR\_dir/os/os/4\_3/testfile.txt", GENERIC\_READ | GENERIC\_WRITE, 0, NULL, OPEN\_EXISTING, FILE\_ATTRIBUTE\_NORMAL, NULL);  
 if (handleFisFile == INVALID\_HANDLE\_VALUE)  
 printf("handleFisFile ERROR !!%x \n", GetLastError());  
 else  
 cout << "Fisfile created" << endl;  
  
  
 handleMappingFile = CreateFileMappingA(handleFisFile, NULL, PAGE\_READWRITE,0,512,"Local\\TEST" );  
 if (handleMappingFile == INVALID\_HANDLE\_VALUE)  
 printf("handleMappingFile ERROR !! %x \n", GetLastError());  
 else  
 cout << "Mapping file created" << endl;  
 GetSystemInfo(&info);  
 char\* a =(char\*) MapViewOfFile(handleMappingFile, FILE\_MAP\_ALL\_ACCESS,0, 0, 512);  
 cout << "\n" << "Adress memory: 0x" << hex << (unsigned long)a << dec << endl;  
 string str;  
  
 cout << "Press string: " << endl;  
 getline(cin, str);  
 CopyMemory(a, str.c\_str(), str.length());  
  
 system("pause");  
 UnmapViewOfFile(a);  
}

Приложение 3

#include <windows.h>  
#include <iostream>  
#include <string>  
  
using namespace std;  
  
int main()  
{  
 SYSTEM\_INFO info;  
 HANDLE handleMappingFile;  
  
 handleMappingFile = OpenFileMappingA(FILE\_MAP\_ALL\_ACCESS, false, "Local\\TEST");  
 if (handleMappingFile == INVALID\_HANDLE\_VALUE)  
 printf("handleMappingFile ERROR !! %x \n", GetLastError());  
 else  
 cout << "file created" << endl;  
 GetSystemInfo(&info);  
 char\* a = (char\*)MapViewOfFile(handleMappingFile, FILE\_MAP\_ALL\_ACCESS, 0, 0, 512);  
 cout << "\n" << "Adress memory: 0x" << hex << (unsigned long)a << dec << endl;  
  
 cout << "Information of memory: ";  
 for (int i = 0; i < 50; i++)  
 cout << a[i];  
 cout << "\n\n";  
  
 system("pause");  
 UnmapViewOfFile(a);  
}