Лабораторная работа 08

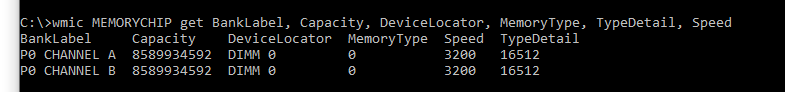
Управление памятью

OC, ПОИТ-3

**Задание 01. Windows**

1. Получите с помощью утилиты **wmic** информации об физической оперативной памяти компьютера, поясните эту информацию.

wmic MEMORYCHIP get BankLabel, Capacity, DeviceLocator, MemoryType, TypeDetail, Speed



 **BankLabel**: Обозначение банка памяти.

 **Capacity**: Размер памяти в байтах.

 **DeviceLocator**: Местоположение модуля памяти на материнской плате.

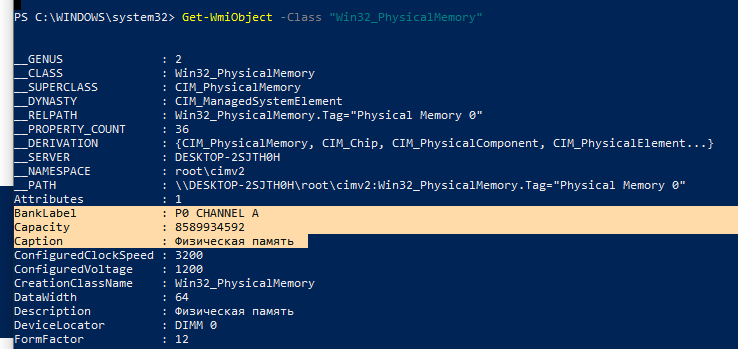
 **MemoryType**: Тип памяти (например, DDR, DDR2).

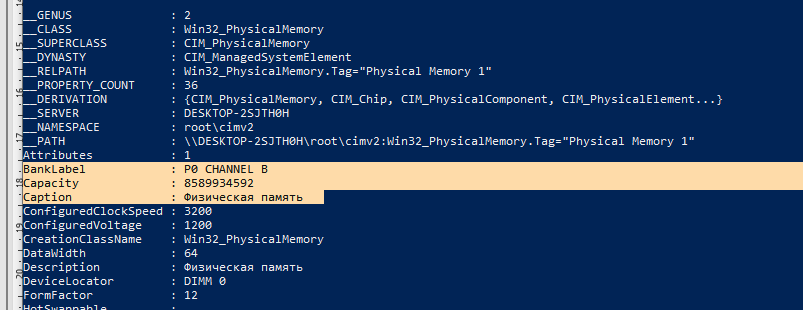
 **TypeDetail**: Дополнительные характеристики памяти.

 **Speed**: Скорость памяти в МГц.

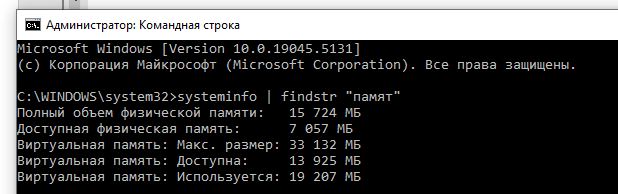
1. Получите с помощью утилиты **powershell** информации об физической оперативной памяти компьютера, поясните эту информацию.

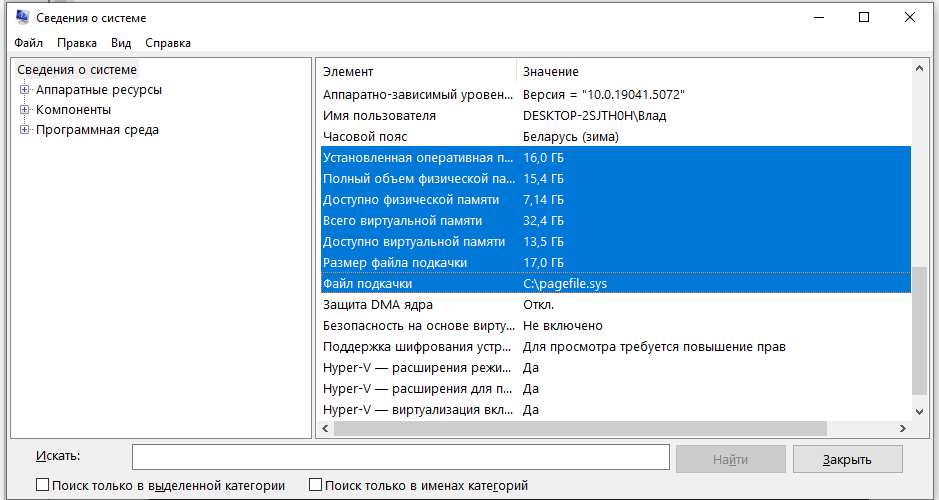
Get-WmiObject -Class "Win32\_PhysicalMemory"



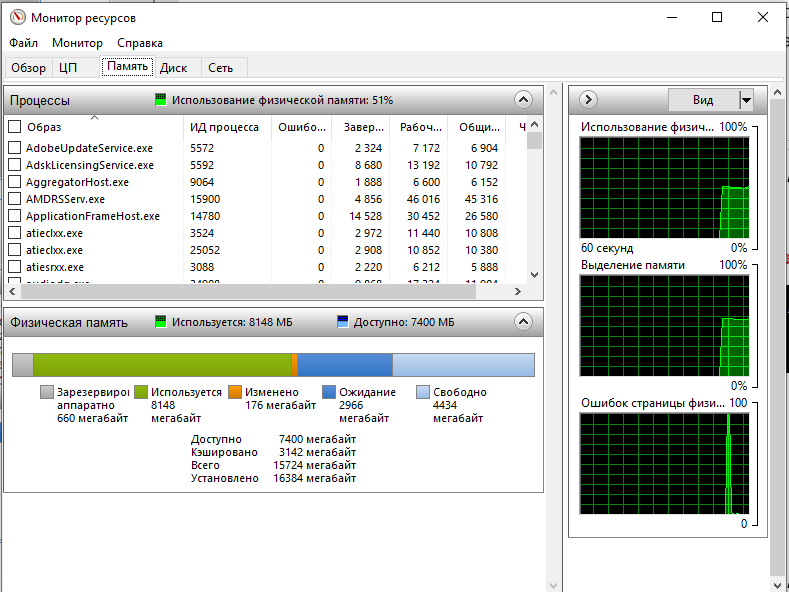


1. Получите с помощью утилиты **systeminfo** информации об оперативной памяти компьютера, поясните эту информацию.



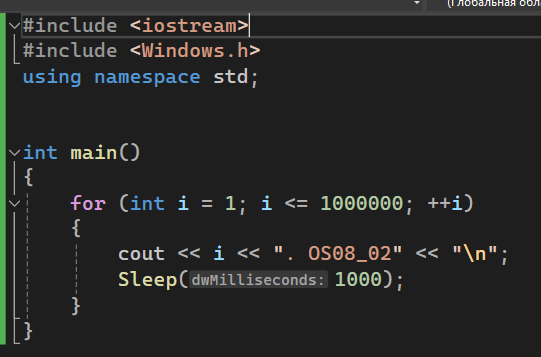


1. Получите с помощью утилиты **performance monitor** информации об оперативной памяти компьютера, поясните эту информацию.

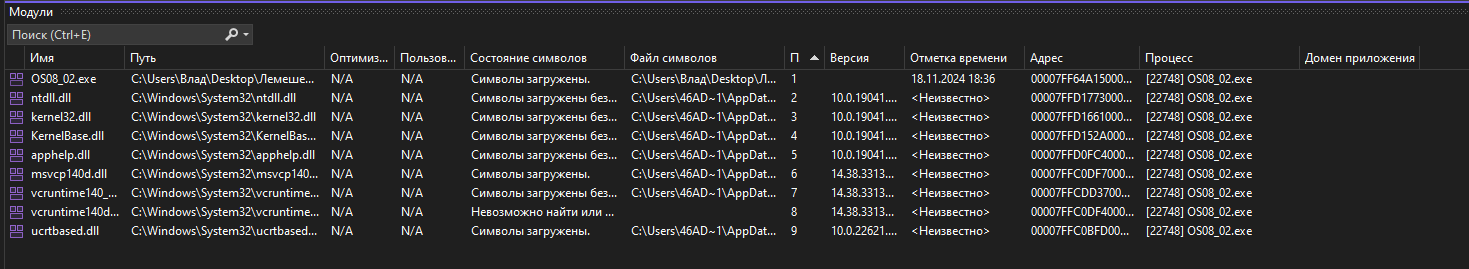
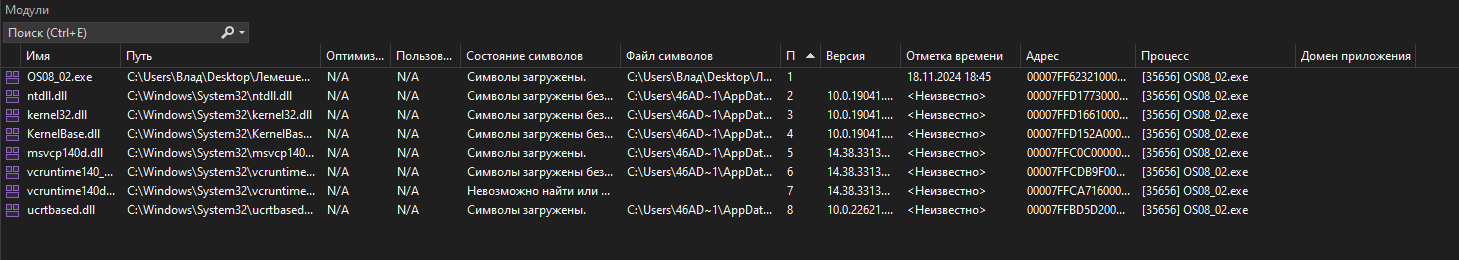
****

**Задание 02.Windows**

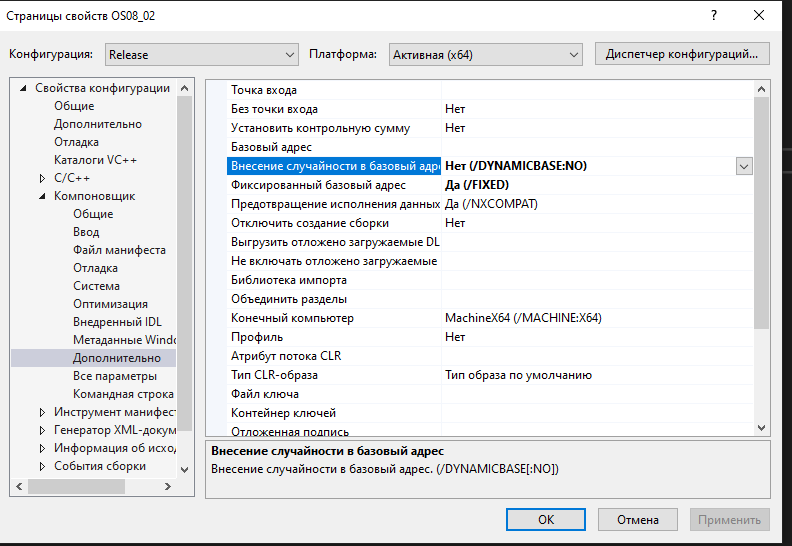
1. Разработайте консольное приложение **OS08\_02**, выполняющее длинный цикл.



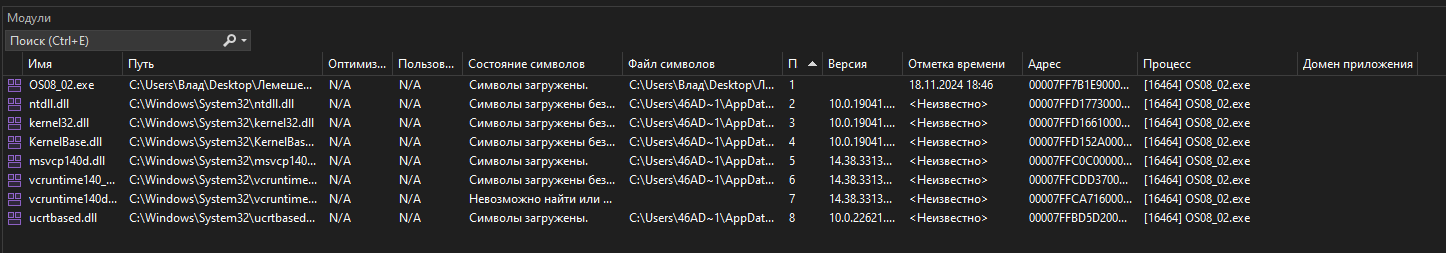
1. Продемонстрируйте с помощью отладчика адреса расположения модулей приложения **OS08\_02**.

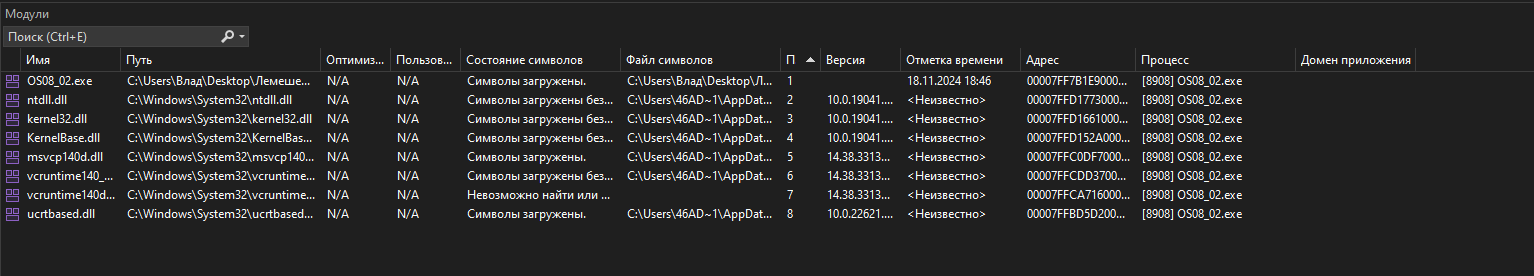


1. Установите для приложения **OS08\_02** стандартный адрес загрузки в память.



1. Продемонстрируйте с помощью отладчика стандартный адрес расположения модулей приложения **OS08\_02**





**Задание 03.Windows**

1. Разработайте консольное приложение **OS08\_03**, выполняющее получение 256 страниц оперативной памяти.

|  |
| --- |
| 1. #include <iostream> 2. #include <Windows.h> 3. using namespace std; 4. #define KB (1024) 5. #define MB (1024 \* KB) 6. #define PG (4 \* KB) 7. void GetMemory() 8. { 9. MEMORYSTATUS ms; 10. GlobalMemoryStatus(&ms); // сохраняет сведения о текущей доступности памяти в этой структуре 11. cout << "Объём физической памяти: " << ms.dwTotalPhys / KB << " KB\n"; 12. cout << "Доступно физической памяти: " << ms.dwAvailPhys / KB << " KB\n"; 13. cout << "Объем виртуальной памяти: " << ms.dwTotalVirtual / KB << " KB\n"; 14. cout << "Доступно виртуальной памяти: " << ms.dwAvailVirtual / KB << " KB\n\n"; 15. } 16. /\* 17. Л - 203(10) - CB(16) 18. Е - 197(10) - C5(16) 19. М - 204(10) - CC(16) 20. Страница CB = 203 21. Страница = Номер\_страницы \* Размер\_страницы 22. 203 \* 4096 - добавить для перехода на страницу 23. Смещение = 0xCC0 (в 16-ричной системе) = 3264 (в десятичной системе). 24. Смещение CC0 = 3264(10) = 0x00000CC0(16) 25. Искомое значение: начало массива + 203\*4096 + 0x00000CC0 26. \*/ 27. int main() 28. { 29. setlocale(LC\_ALL, "ru"); 30. int pages = 256; 31. int countItems = pages \* PG / sizeof(int); 32. SYSTEM\_INFO system\_info; 33. GetSystemInfo(&system\_info); // Извлекает сведения о текущей системе 34. cout << "\tИзначально в системе\n"; 35. GetMemory(); 36. LPVOID xmemaddr = VirtualAlloc(NULL, pages \* PG, MEM\_COMMIT, PAGE\_READWRITE); // выделено 1024 KB виртуальной памяти 37. cout << "\tВыделено " << pages \* PG / 1024 << " KB вирт. памяти " << " VirtualAlloc = " << hex << xmemaddr << dec << endl; 38. GetMemory(); 39. int\* arr = (int\*)xmemaddr; 40. for (int i = 0; i < countItems; i++) 41. arr[i] = i; 42. int\* byte = arr + (203 \* 1024) + 3264 ; 43. cout << "[Bytes]: " << \*byte << "\n"; 44. cout << "[HEX]: " << hex << byte << dec <<"\n"; 46. // Освобождение выделенной виртуальной памяти 47. VirtualFree(xmemaddr, NULL, MEM\_RELEASE) ? cout << "\tВиртуальная память освобождена" << " VirtualFree = " << hex << xmemaddr << dec << endl : cout << "\tВиртуальная память не освобождена\n"; 48. GetMemory(); 49. system("pause"); 50. } |

1. Разместите в этой памяти массив типа **int,** полностью занимающее все 256 страниц.
2. Заполните этот массив нарастающей последовательностью чисел с шагом 1.
3. Запишите 3 первых буквы своей фамилии в 16-ричными числами в кодировке Windows-1251.
4. Найдите в полученной области памяти с помощью отладчика значение в байте, имеющем адрес вычисленный по следующему принципу: номер страницы = число в нулевом байте, смещение в странице = число 12 бит в 1ом и втором байтах.

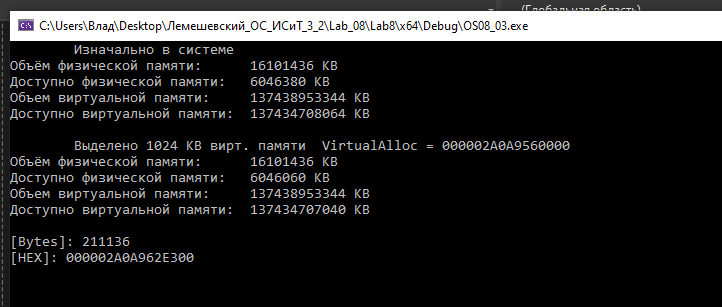
*Пример: Иванов*

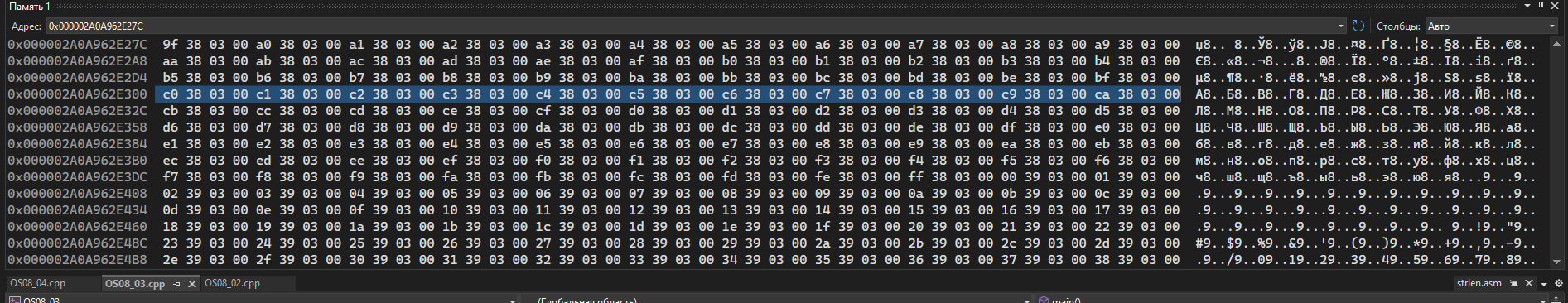
*И = C8*

*в = E2*

*а = E0*

*Страница C8 = 200, смещение E2E = 3630*

**

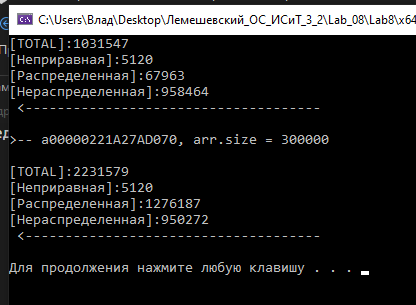


**Задание 04.Windows**

1. Разработайте консольное приложение **OS08\_04,** которое включает функцию **sh**, принимающую 1 параметр: дескриптор (HANDLE) heap.

|  |
| --- |
| 1. #include <Windows.h> 2. #include <iostream> 3. #include <iomanip> 4. #include <locale> 5. using namespace std; 6. void sh(HANDLE heap) { 7. SetConsoleOutputCP(1251); 8. PROCESS\_HEAP\_ENTRY phe; 9. SIZE\_T total = 0, unstopped = 0 , alloc = 0, unalloc = 0; 10. phe.lpData = NULL; 11. while (HeapWalk(heap,&phe)) { 12. total += phe.cbData; 13. if (phe.wFlags & PROCESS\_HEAP\_REGION) { 14. unstopped += phe.cbData; 15. } 16. else if (phe.wFlags & PROCESS\_HEAP\_UNCOMMITTED\_RANGE) { 17. unalloc += phe.cbData; 18. } 19. else { 20. alloc += phe.cbData; 21. } 22. } 23. cout << "[TOTAL]:" << total << endl; 24. cout << "[Неприравная]:" << unstopped << endl; 25. cout << "[Распределенная]:" << alloc << endl; 26. cout << "[Нераспределенная]:" << unalloc << endl; 27. cout << " <------------------------------------- \n\n"; 28. } 29. int main() { 30. HANDLE heap = GetProcessHeap(); 31. sh(heap); 32. int size = 300000; 33. int\* a = new int[size]; 34. cout << ">-- a" << hex << a << ", arr.size = " << dec << size << "\n\n"; 35. sh(heap); 36. system("pause"); 37. return 0; 38. } |

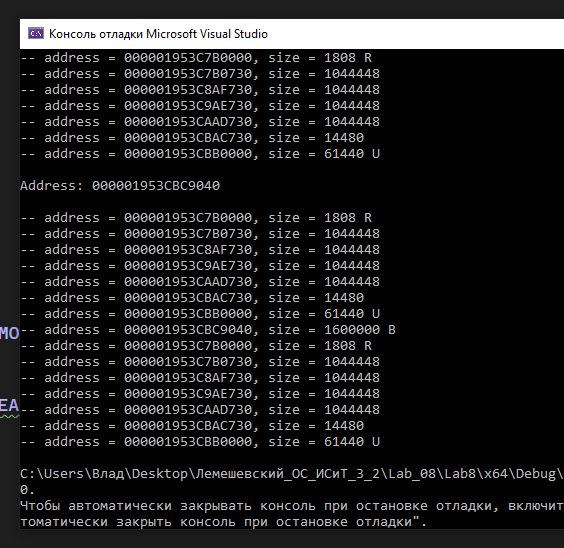
1. Функция **sh** выводит на консоль, общий размер heap, размеры распределенной и нераспределенных областей памяти heap.
2. Приложение **OS08\_04** размещает в стандартной heap процесса int-массив размерности 300000.
3. Выведите с помощью функции **sh** информацию до размещения массива и после.
4. Объясните результат.



**Задание 05.Windows**

1. Разработайте консольное приложение **OS08\_05** аналогичное приложению **OS06\_05,** но использующее пользовательскую heap, которая имеет первоначальный размер 4MB.
2. Объясните результат.

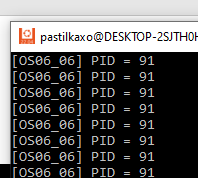
|  |
| --- |
| #include <iostream>  #include <Windows.h>  using namespace std;  #define KB (1024)  void sh(HANDLE pheap)  {  PROCESS\_HEAP\_ENTRY phe;  phe.lpData = NULL;  while (HeapWalk(pheap, &phe))  {  // R - начало непрерывной области  // U - нераспределенная область  // B - распределенная область  cout << "-- address = " << showbase << hex << phe.lpData <<  ", size = " << dec << phe.cbData <<  ((phe.wFlags & PROCESS\_HEAP\_REGION) ? " R" : "") <<  ((phe.wFlags & PROCESS\_HEAP\_UNCOMMITTED\_RANGE) ? " U" : "") <<  ((phe.wFlags & PROCESS\_HEAP\_ENTRY\_BUSY) ? " B" : "") << "\n";  }  }  int main()  {  // Параметры пользовательской кучи:  // 1. доступ не синхронизирован  // 2. куча заполняется нулями  // 3. начальный размер 4 мб  // 4. конечный размер ограничен размером виртуальной памяти  HANDLE heap = HeapCreate(HEAP\_NO\_SERIALIZE | HEAP\_ZERO\_MEMORY, 4096 \* 1024, 0);  sh(heap);  int\* x1 = (int\*)HeapAlloc(heap, HEAP\_NO\_SERIALIZE | HEAP\_ZERO\_MEMORY, 400000 \* sizeof(int));  cout << "\nAddress: " << x1 << "\n\n";  sh(heap);  HeapFree(heap, HEAP\_NO\_SERIALIZE, x1);  sh(heap);  HeapDestroy(heap);   1. } |



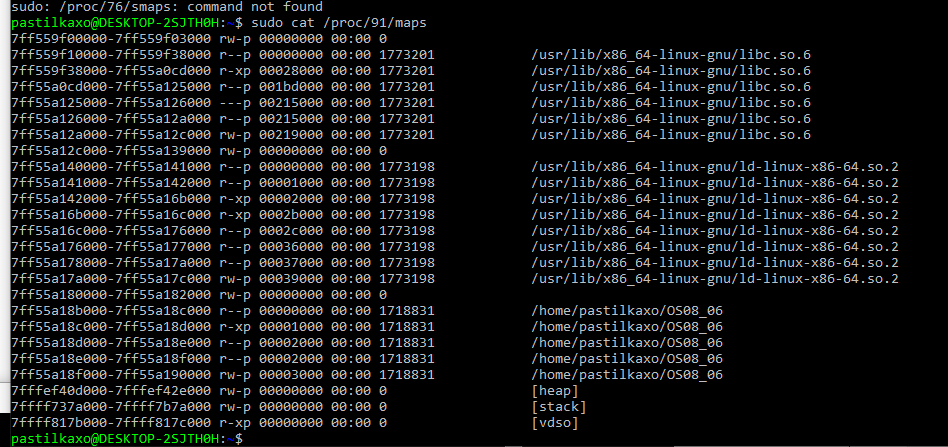
**Задание 06.Linux**

1. Разработайте консольное приложение **OS08\_06**, выполняющее длинный цикл.

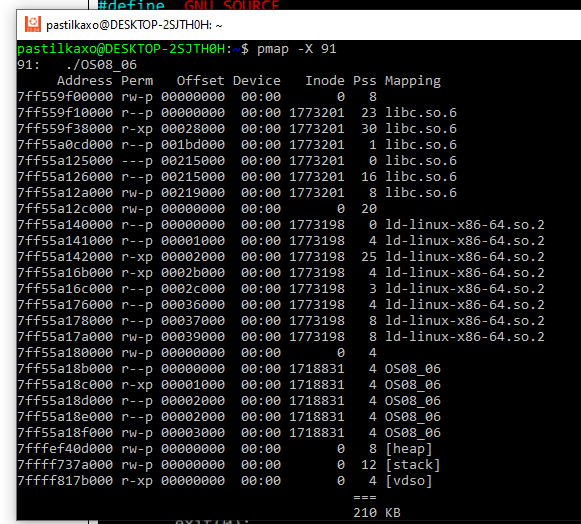
|  |
| --- |
|  |

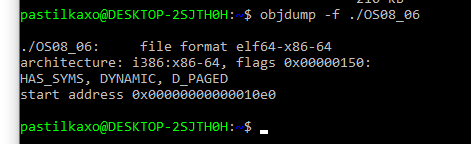


1. Продемонстрируйте с помощью файловой системы **/proc** структуру адресного пространства.



1. Продемонстрируйте с помощью **pmap**  структуру адресного пространства.

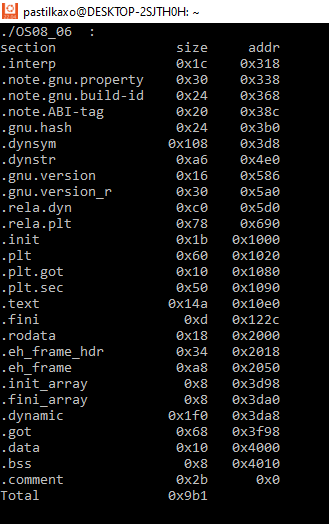


1. Определите с помощью утилиты objdump адрес загрузки main-модуля  
   

, секций с кодом

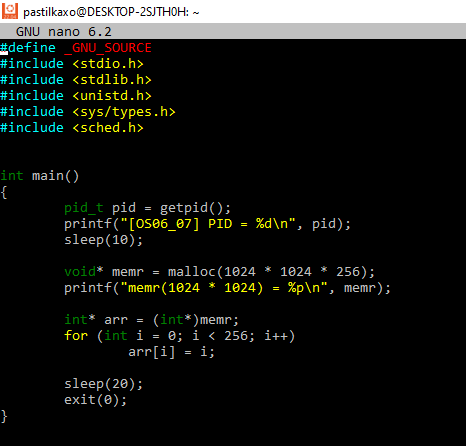
, данными,

неинициализированными глобальными переменными.



**Задание 07.Linux**

1. Разработайте консольное приложение **OS08\_07**, которое динамически выделяет 256 МБ памяти.



1. В выделенной памяти разместите int-массив максимальной размерности. Проинициализируйте массив последовательными значениями с шагом 1.
2. Выведите на консоль адрес выделенной памяти.
3. После инициализации приложение должно приостановить свое выполнение на длительный интервал времени.
4. С помощью утилиты pmap определите область памяти в которой выделена память.

**Задание 08** Ответьте на следующие вопросы

1. Поясните понятие «виртуальная память».
2. Поясните понятие «свопинг».
3. Поясните понятие «страничная память».
4. Поясните понятие MMU.
5. Поясните понятие TLB.
6. Какая информация содержится в строке таблицы страниц
7. Поясните принцип применения хэш-таблиц.
8. Поясните применение «инвертированной таблицы физических» страничной памяти.
9. Поясните понятие «рабочий набор страниц».
10. Поясните принцип работы алгоритма LRU.
11. Windows: поясните назначение сервиса SysMain.
12. Windows: поясните назначение файла hiberfil.sys.
13. Windows: поясните назначение файла pagefile.sys.
14. Windows: поясните назначение файла swapfile.sys.
15. Windows: перечислите области адресного пространства (от младших к старшим адресам) и поясните их назначения.
16. Windows: какой стандартный начальный размер области heap?
17. Windows: каким образом можно изменить начальный размер области памяти heap приложения?
18. Windows: какой стандартный размер области памяти stack?
19. Windows: каким образом можно изменить размер области памяти stack приложения?
20. Windows: поясните назначение функции Windows API: GlobalMemoryStatus.
21. Windows: поясните назначение функции Windows API: VirtualQuery; перечислите значения атрибутов Protect, State и Type.
22. Windows: что такое «рабочее множество»? поясните принцип управления рабочим множеством с помощью OS API.
23. Windows: что означает «страница заблокирована»? с помощью каких функций OS API можно установить блокировку страниц и снять блокировку? Какое максимальное количество страниц можно заблокировать?
24. Windows: что такое «heap»? Что такое «heap процесса»? Что такое «пользовательская heap»? Поясните принцип устройства heap.
25. Linux: перечислите области адресного пространства (от младших к старшим адресам) и поясните их назначения.
26. Linux: в какой части адресного пространства выделяется памяти с помощью функций malloc, calloc?