Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Операционные системы

Студент: Песецкий И.А.

ФИТ 3 курс 5 группа

Преподаватель: Савельева М.Г.

Минск 2023

**Лабораторная работа №8**

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

|  |  |
| --- | --- |
| wmic |  |

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

|  |  |
| --- | --- |
| powershell |  |

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

|  |  |
| --- | --- |
| systeminfo |  |

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

|  |  |
| --- | --- |
| performance monitor |  |

Задание 2. Разработайте консольное приложение OS06\_02, выполняющее длинный цикл.

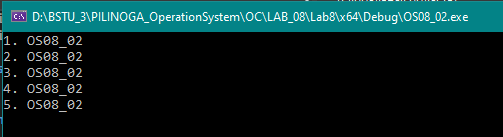
Продемонстрируйте с помощью отладчика адреса расположения модулей приложения OS08\_02.

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

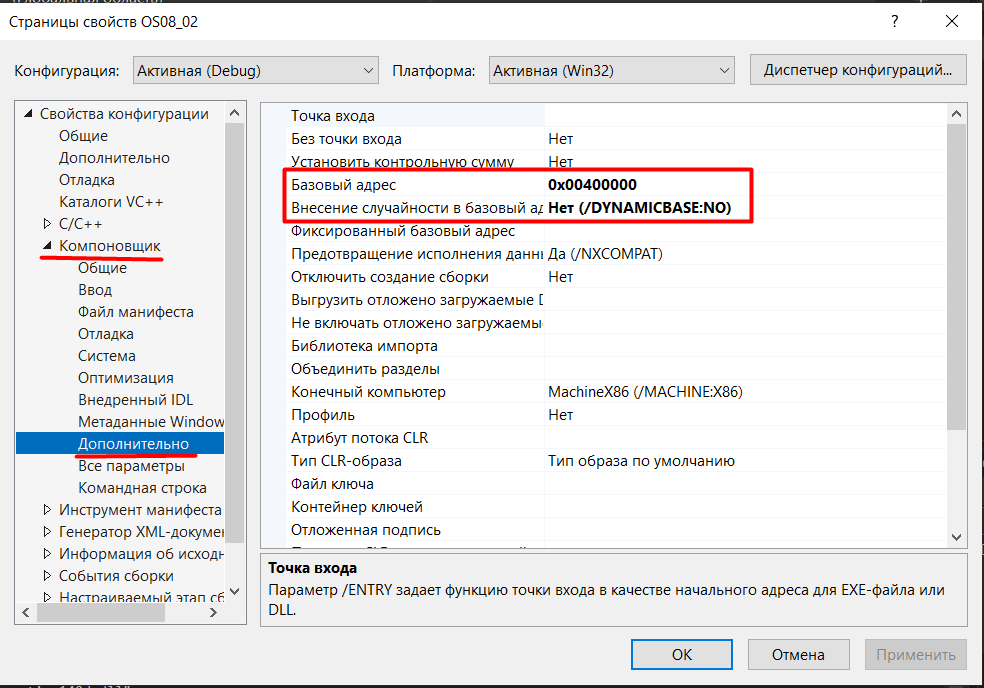
Продемонстрируйте с помощью отладчика стандартный адрес расположения модулей приложения OS08\_02

|  |  |
| --- | --- |
| Код программы OS08\_02 | #include <iostream>  #include <Windows.h>  using namespace std;  int main()  {  for (int i = 1; i <= 9999999999; ++i)  {  cout << i << ". OS08\_02" << "\n";  Sleep(2000);  }  }}  } |

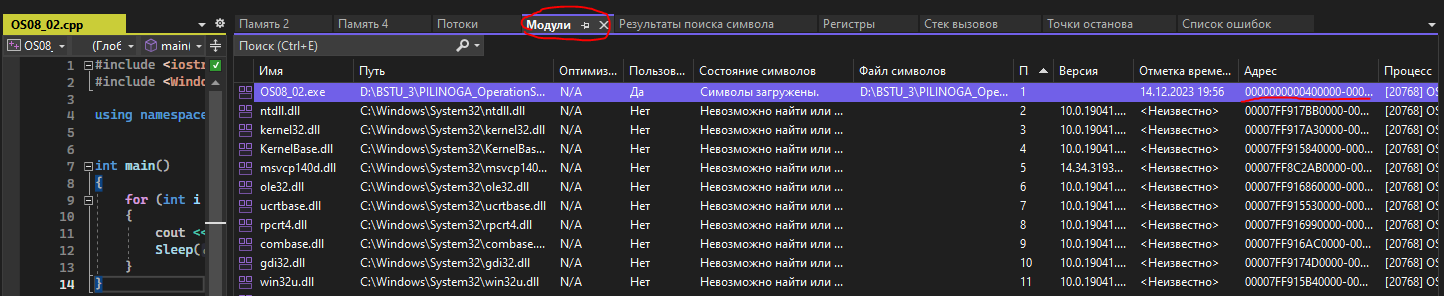
Скриншот запуска



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



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



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



**Задание 3.** Разработайте консольное приложение OS06\_03, выполняющее получение 256 страниц оперативной памяти.

Разместите в этой памяти массив типа int, полностью занимающее все 256 страниц.

Заполните этот массив нарастающей последовательностью чисел с шагом 1.

Запишите 3 первых буквы своей фамилии в 16-ричными числами в кодировке Windows-1251.

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

Код программы OS08\_03

|  |  |  |
| --- | --- | --- |
|  | #include <iostream>  #include <Windows.h>  using namespace std;  #define KB (1024)  #define MB (1024 \* KB)  #define PG (4 \* KB)  void saymem()  {  MEMORYSTATUS ms;  GlobalMemoryStatus(&ms);  cout << "Объём физической памяти: " << ms.dwTotalPhys / KB << " KB\n";  cout << "Доступно физической памяти: " << ms.dwAvailPhys / KB << " KB\n";  cout << "Объем виртуальной памяти: " << ms.dwTotalVirtual / KB << " KB\n";  cout << "Доступно виртуальной памяти: " << ms.dwAvailVirtual / KB << " KB\n\n";  }  //Запишите 3 первых буквы своей фамилии в 16-ричными числами в кодировке Windows-1251.  /\*  "П" = 207 = 0xCF  "е" = 229 = 0xE5  "с" = 241 = 0xF1  Страница CF = 207  207 \* 4096 = 847872(10) = 0xCF000 - добавить для перехода на страницу  E5F = 3679  Искомое значение: начало массива + 0xCF000 + 0xE5F  \*/  int main()  {  setlocale(LC\_ALL, "ru");  //9. Разработайте консольное приложение OS06\_03, выполняющее получение 256 страниц оперативной памяти.  int pages = 256;  int countItems = pages \* PG / sizeof(int);  SYSTEM\_INFO system\_info;  GetSystemInfo(&system\_info);  cout << "\t Изначально в системе\n";  saymem();  // выделено 1024 KB виртуальной памяти  LPVOID xmemaddr = VirtualAlloc(NULL, pages \* PG, MEM\_COMMIT, PAGE\_READWRITE);    cout << "\tВыделено " << pages \* PG / 1024 << " KB вирт. памяти\n";  saymem();  int\* arr = (int\*)xmemaddr;  for (int i = 0; i < countItems; i++)  arr[i] = i;  int\* byte = arr + 207 \* 1024 + 3679;  cout << "------ Значение памяти в байте: " << \*byte << " ------\n";  VirtualFree(xmemaddr, NULL, MEM\_RELEASE) ?  cout << "\tВиртуальная память освобождена\n"  :  cout << "\tВиртуальная память не освобождена\n";  saymem();  } | |
| Скриншот запуска |  |
| Запишите 3 первых буквы своей фамилии в 16-ричными числами в кодировке Windows-1251. | "п" (п в нижнем регистре) : 0xDF  "е" (е в нижнем регистре) : 0xE5  "с" (с в нижнем регистре) : 0xF1 | |
| 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. |  | |

Задание 4. Разработайте консольное приложение OS08\_04, которое включает функцию sh, принимающую 1 параметр: дескриптор (HANDLE) heap.

Функция sh выводит на консоль, общий размер heap, размеры распределенной и нераспределенных областей памяти heap.

Приложение OS06\_04 размещает в стандартной heap процесса int-массив размерности 300000.

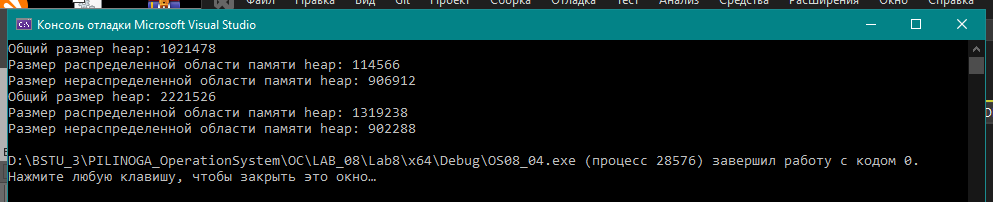
Выведите с помощью функции sh информацию до размещения массива и после.

Объясните результат.

Код программы OS08\_04

|  |
| --- |
| #include <windows.h>  #include <iostream>  //14. Разработайте консольное приложение OS06\_04,  //которое включает функцию sh, принимающую 1 параметр: дескриптор (HANDLE) heap.  void sh(HANDLE heap) {  SetConsoleOutputCP(1251);  PROCESS\_HEAP\_ENTRY entry;  entry.lpData = NULL;  SIZE\_T totalSize = 0, allocatedSize = 0, unallocatedSize = 0;  while (HeapWalk(heap, &entry) != FALSE) {  totalSize += entry.cbData;  if (entry.wFlags & PROCESS\_HEAP\_ENTRY\_BUSY) {  allocatedSize += entry.cbData;  }  else {  unallocatedSize += entry.cbData;  }  }  //15. Функция sh выводит на консоль, общий размер heap,  //размеры распределенной и нераспределенных областей памяти heap.  std::cout << "Общий размер heap: " << totalSize << "\n";  std::cout << "Размер распределенной области памяти heap: " << allocatedSize << "\n";  std::cout << "Размер нераспределенной области памяти heap: " << unallocatedSize << "\n";  }  int main() {  HANDLE heap = GetProcessHeap();  sh(heap);  //16. Приложение OS06\_04 размещает в стандартной heap процесса int-массив размерности 300000.  const int ARRAYSIZE = 300000;  int\* array = new int[ARRAYSIZE];  //17. Выведите с помощью функции sh информацию до размещения массива и после.  sh(heap);  delete[] array;  return 0;  } |

Скриншот запуска



При создании системнйо кучи в неё помимо даннх будут заноситься и данные от системы и они с каждым разом при добавлении элементов в кучу будут перезаписываться и размер будет уменьшаться

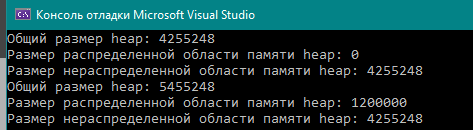
А к пользовательской системные данные в кучу не заносятся и размер до и после одинаковый

Задание 5. Разработайте консольное приложение OS08\_05 аналогичное приложению OS08\_04, но использующее пользовательскую heap, которая имеет первоначальный размер 4MB.

Объясните результат.

|  |
| --- |
| #include <windows.h>  #include <iostream>  //14. Разработайте консольное приложение OS06\_04,  //которое включает функцию sh, принимающую 1 параметр: дескриптор (HANDLE) heap.  void sh(HANDLE heap) {  PROCESS\_HEAP\_ENTRY entry;  entry.lpData = NULL;  SIZE\_T totalSize = 0, allocatedSize = 0, unallocatedSize = 0;  while (HeapWalk(heap, &entry) != FALSE) {  totalSize += entry.cbData;  if (entry.wFlags & PROCESS\_HEAP\_ENTRY\_BUSY) {  allocatedSize += entry.cbData;  }  else {  unallocatedSize += entry.cbData;  }  }  //15. Функция sh выводит на консоль, общий размер heap,  //размеры распределенной и нераспределенных областей памяти heap.  std::cout << "Общий размер heap: " << totalSize << "\n";  std::cout << "Размер распределенной области памяти heap: " << allocatedSize << "\n";  std::cout << "Размер нераспределенной области памяти heap: " << unallocatedSize << "\n";  }  // Параметры пользовательской кучи:  // 1. доступ не синхронизирован  // 2. куча заполняется нулями  // 3. начальный размер 4 мб  // 4. конечный размер ограничен размером виртуальной памяти  int main() {  SetConsoleOutputCP(1251);  //19. Разработайте консольное приложение OS06\_05 аналогичное приложению OS06\_05,  //но использующее пользовательскую heap, которая имеет первоначальный размер 4MB.  const SIZE\_T HEAPSIZE = 4 \* 1024 \* 1024;  HANDLE heap = HeapCreate(0, HEAPSIZE, 0);  sh(heap);  //16. Приложение OS06\_04 размещает в стандартной heap процесса int-массив размерности 300000.  const int ARRAYSIZE = 300000;  int\* array = (int\*)HeapAlloc(heap, 0, ARRAYSIZE \* sizeof(int));  //17. Выведите с помощью функции sh информацию до размещения массива и после.  sh(heap);  HeapFree(heap, 0, array);  HeapDestroy(heap);  return 0;  } |

Скриншот запуска



Задание 6. Разработайте консольное приложение OS08\_06, выполняющее длинный цикл.

Продемонстрируйте с помощью файловой системы /proc структуру адресного пространства.

Продемонстрируйте с помощью pmap структуру адресного пространства.

Определите с помощью утилиты objdump адрес загрузки main-модуля, секций с кодом, данными, неинициализированными глобальными переменными.

|  |  |
| --- | --- |
| Код программы OS08\_06 | #define \_GNU\_SOURCE  #include <stdio.h>  #include <stdlib.h>  #include <unistd.h>  #include <sys/types.h>  #include <sched.h>  /\*  sudo cat /proc/pid/maps  pmap -X pid  objdump -f OS08\_06  size -Ax OS08\_06  .text - секция с кодом (+константы)  .data - инициализированные данные  .bss - неинициализированные данные  \*/  int main()  {  pid\_t pid = getpid();  nice(0);  for (int i = 0; i < 10000000; i++)  {  printf("[OS08\_06] PID = %d\n", pid);  sleep(1);  }  exit(0);  } |
| Скриншот запуска | Sudo cat /proc/pid/map    Pmap –x pid    Odjdump –f OS08\_06    Odjdump –text OS08\_06    Objdump –data OS08\_06    Size –Ax Os08\_06 |
|  | **sudo cat /proc/pid/maps:** Эта команда выводит информацию о карте памяти процесса с идентификатором pid. Карта памяти содержит различные сегменты памяти, используемые процессом.  **pmap -x pid:** Эта команда также предоставляет информацию о памяти процесса, включая использование, резидентность и т. д. Опция -x включает дополнительные сведения. (структуру адресного пространства)  **objdump -f OS08\_06:** Эта команда выводит общую информацию об исполняемом файле, такую как архитектура, тип файла, флаги и другие характеристики.  **objdump -t OS08\_06:** Эта команда предоставляет раздел текста (code) исполняемого файла. Вы увидите вывод ассемблерного кода и связанные с ним сведения.  **objdump -d OS08\_06:** Эта команда также выводит раздел текста, но с более подробным представлением ассемблерного кода.  **objdump -s OS08\_06:** Эта команда предоставляет информацию о разделе данных исполняемого файла.  **size -Ax OS08\_06:** Эта команда показывает размер секций в исполняемом файле. Опция -A включает вывод в формате, который легче читать для программистов.   1. Определите с помощью утилиты objdump адрес загрузки main-модуля, секций с кодом, данными, неинициализированными глобальными переменными. |

Задание 7. Разработайте консольное приложение OS08\_07, которое динамически выделяет 256 МБ памяти.

В выделенной памяти разместите int-массив максимальной размерности. Проинициализируйте массив последовательными значениями с шагом 1.

Выведите на консоль адрес выделенной памяти.

После инициализации приложение должно приостановить свое выполнение на длительный интервал времени.

С помощью утилиты pmap определите область памяти в которой выделена память.

|  |  |
| --- | --- |
| Код программы OS08\_07 | #define \_GNU\_SOURCE  #include <stdio.h>  #include <stdlib.h>  #include <unistd.h>  #include <sys/types.h>  #include <sched.h>  int main()  {  pid\_t pid = getpid();  printf("[OS08\_07] PID = %d\n", pid);  sleep(10);  void\* memr = malloc(1024 \* 1024 \* 256);  printf("memr(1024 \* 1024) = %p\n", memr);  int\* arr = (int\*)memr;  for (int i = 0; i < 256; i++)  arr[i] = i;  sleep(20);  exit(0);  } |
| Скриншот запуска |  |