Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное

образовательное учреждение высшего образования

«Комсомольский-на-Амуре государственный университет»

Факультет компьютерных технологий

Кафедра «МОП ЭВМ»

ЛАБОРАТОРНАЯ РАБОТА 3

по дисциплине «Операционные системы и оболочки»

## Управление памятью в операционной системе Windows

Студент группы 6ИСб-1 И. Нозимзода

Преподаватель В.Я Столяров

2019

**Цель работы**: Изучить теоретические вопросы управления памятью в ОС Windows и освоить приемы практической реализации этого управления с использованием системных функций.

**Задания:**

Необходимо выделить участки памяти размером 32 Кбт по адресам: 900 000, 1 000 000 и 3 000 000. Заполнить эти участки единицами. Защитить от записи и попробовать очистить. При возникновении ошибки – перехватить ее и выдать об этом сообщение.

**Выполнение:**

В наше время в основе многих компьютерных архитектур располагается специ­альный механизм управления памятью, который называют MMU (Memory Management Unit ‑ устройство управления памятью).

Управлять работой MMU можно при помощи вызова VirtualAlloc. В качестве аргументов функция принимает адрес области памяти, ее размер, режим выделения и уровень защиты. Если в качестве первого аргумента функ­ции передать значение NULL, операционная система самостоятельно определит подходящий адрес выделяемой области. Вместо этого вы можете попросить ее выделить память с конкретным начальным адресом (этот адрес должен соответствовать границе страницы). Второй аргумент ‑ количество байт, которое необходимо выделить. Windows автоматически округлит это количество таким образом, чтобы оно равнялось числу, кратному размеру страницы. Если вы хотите просто выделить память, в качестве третьего аргумента необходимо указать значение МЕМ\_СОММ1Т. Последний аргумент ‑ уровень защиты, который будет соответствовать выделяемой памяти. Очевидно, что использовать некоторые из перечисленных здесь атрибутов (например, PAGE\_READONLY) не имеет смысла, так как память не инициализирована начальными значениями.

Программа, код которой приведен в листинге 1, получает в индивидуальное пользование относительно большой участок памяти, при этом используются три разных метода. По умолчанию программа работает с блоком памяти объемом 75 Мбайт. Это значение приемлемо для тестирования механизмов выделения и освобождения памяти.

Листинг 1 – Текст файла prog1.cpp

|  |
| --- |
| #include <windows.h>  #include <locale.h>  #include <iostream>  using namespace std;  int filter(unsigned int code, struct \_EXCEPTION\_POINTERS \*ep) {  if (code == EXCEPTION\_ACCESS\_VIOLATION) {  return EXCEPTION\_EXECUTE\_HANDLER;  }  else {  return EXCEPTION\_CONTINUE\_SEARCH;  };  }  int main()  {  setlocale(LC\_ALL, "rus");  system("color 0A ");  // Блоки памяти по 32 Кб = 32000 Байт = 8000 \* 4 byte = 8000 \* sizeof(int)  const size\_t blockByteSize = 8000 \* sizeof(int);  const size\_t blockElementSize = blockByteSize / sizeof(int);  // Массив указателей на три блока выделяемой памяти  int\* arrayPointer[3];  // по адресам: 900 000, 1 000 000 и 3 000 000...  cout << "Выделение участков памяти размером 32Кб ";  // MEM\_COMMIT - Выполняется выделение страниц памяти для непосредственной работы с ними. Выделенные страницы заполняются нулями.  // PAGE\_READWRITE - Режим на чтение и запись.  arrayPointer[0] = (int\*)VirtualAlloc(NULL, blockByteSize, MEM\_COMMIT, PAGE\_READWRITE);  arrayPointer[1] = (int\*)VirtualAlloc(NULL, blockByteSize, MEM\_COMMIT, PAGE\_READWRITE);  arrayPointer[2] = (int\*)VirtualAlloc(NULL, blockByteSize, MEM\_COMMIT, PAGE\_READWRITE);  if (arrayPointer[0] == NULL || arrayPointer[1] == NULL || arrayPointer[2] == NULL) {  MessageBox(0, "Память не выделена", "ERROR", MB\_ICONSTOP);  return 0;  }  else  {  cout << "Память выделена." << endl;  }  cout << "Заполнение участков памяти единицами... ";  for (int i = 0; i < 3; ++i)  for (int j = 0; j < blockElementSize; ++j)  {  arrayPointer[i][j] = 1;  }  cout << "Память заполнена." << endl;  cout << "Установка защиты для памяти матрици... ";  // Последний аргумент — это указатель на DWORD. Эта переменная будет содержать уровень безопасности, которым обла¬дал участок до обращения к VirtualProtect.  DWORD param;  VirtualProtect(arrayPointer[0], sizeof(int), PAGE\_READONLY, &param);  VirtualProtect(arrayPointer[1], sizeof(int), PAGE\_READONLY, &param);  VirtualProtect(arrayPointer[2], sizeof(int), PAGE\_READONLY, &param);  cout << "Защита установлена." << endl;  cout << "Обнуление участков памяти... ";  \_\_try  {  for (int i = 0; i < 3; ++i)  for (int j = 0; j < blockElementSize; ++j)  {  arrayPointer[i][j] = 0;  }  cout << "" << endl;  }  \_\_except (filter(GetExceptionCode(), GetExceptionInformation()))  {  cout << "ERROR! Msg: EXCEPTION\_ACCESS\_VIOLATION" << endl;  }  system("pause");  return 0;  } |

Программа выделает три блока памяти по 32 Кб, по указанным адресам, заполняет их единицами, устанавливает на эту память модификатор READONLY и обрабатывает исключение при попытке изменить данные из этой памяти.

Результат работы программы показан на рисунке 1.

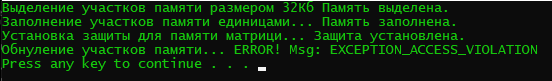


Рисунок 1 – Программа по индивидуальному заданию

**Список использованных источников**

1. Саймон, Р. Windows 2000 API Энцеклопедия программиста / Р. Саймон. - М: DiaSoft, 2002