|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  |  |  | | МИНОБРНАУКИ РОССИИ | | | | Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | | | |
| Институт информационных технологий (ИТ) | |
| Кафедра инструментального и прикладного программного обеспечения (ИиППО) | |

|  |  |
| --- | --- |
| **ОТЧЁТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ №7** | |
| **по дисциплине** | |
| «Системное программное обеспечение»  по теме: «Использование механизма виртуальной памяти в ОС Windows» | |
|  | |
| Выполнил студент группы ИКБО-16-18 | Коротких А.С. |
| Принял преподаватель | Волков М.Ю. |

|  |  |  |
| --- | --- | --- |
| Работа выполнена | «\_\_»\_\_\_\_\_\_\_2021 г. | *(подпись студента)* |
|  |  |  |
| «Зачтено» | «\_\_»\_\_\_\_\_\_\_2021 г. | *(подпись руководителя)* |

Москва 2021

**Цель работы**

Изучение виртуальной памяти в операционной системе Windows.

**Задание**

1. Изучение теоретического материала по виртуальной памяти.

2. Составление алгоритма программы.

3. Программная реализация.

**Ход работы**

Итоговая программа представлена в листинге 1. Результат работы представлен на рисунках 1.

Листинг 1

#include <Windows.h>

#include <iostream>

#include <conio.h>

using namespace std;

SYSTEM\_INFO SysInfo;

void writeSystemInfo() {

GetSystemInfo(&SysInfo);

cout << "Размер страницы: " << SysInfo.dwPageSize << endl;

cout << "Минимальный адрес приложения: " << SysInfo.lpMinimumApplicationAddress << endl;

cout << "Максимальный адрес приложения: " << SysInfo.lpMaximumApplicationAddress << endl;

cout << "Активная маска процессора: " << SysInfo.dwActiveProcessorMask << endl;

cout << "Количество ядер процессора: " << SysInfo.dwNumberOfProcessors << endl;

cout << "Тип процессора: " << SysInfo.dwProcessorType << endl;

cout << "Гранулярность: " << SysInfo.dwAllocationGranularity << endl;

cout << "Уровень процессора: " << SysInfo.wProcessorLevel << endl;

cout << "Пересмотр процессора?: " << SysInfo.wProcessorRevision << endl;

}

void writeMemoryInfo() {

MEMORYSTATUS MemStat;

MemStat.dwLength = { sizeof(MemStat) };

GlobalMemoryStatus(&MemStat);

cout << "Размер страницы: " << MemStat.dwLength << endl;

cout << "Процент использования памяти: " << MemStat.dwMemoryLoad << endl;

cout << "Размер физической памяти в байтах: " << MemStat.dwTotalPhys << endl;

cout << "Доступная память в байтах: " << MemStat.dwAvailPhys << endl;

cout << "Размер файла подкачки: " << MemStat.dwTotalPageFile << endl;

cout << "доступный размер файла подкачки: " << MemStat.dwAvailPageFile << endl;

cout << "Вс вирутальная память: " << MemStat.dwTotalVirtual << endl;

cout << "Доступаня виртуальная память: " << MemStat.dwAvailVirtual << endl;

}

void workWithHeap() {

HANDLE SysHeap = GetProcessHeap(); // максимальное количество выделений памяти

UINT MAX\_ALLOCATIONS = 32; // текущее количество выделений памяти

UINT NumOfAllocations;

for (NumOfAllocations = 0; NumOfAllocations < MAX\_ALLOCATIONS; NumOfAllocations++) // выделение из кучи 1 Кб памяти

if (HeapAlloc(SysHeap, 0, 1024) == NULL) break; // вывод соответствующих сообщений в зависимости от ситуации

if (NumOfAllocations == 0)

cout << "Память из кучи не выделялась.\n";

else

printf("Память из кучи выделялась %d раз.", NumOfAllocations);

cout << endl;

}

void workWithRAM() {

SIZE\_T MEMSIZE = 4096;

LPVOID Mem = VirtualAlloc(NULL, MEMSIZE, MEM\_COMMIT, PAGE\_EXECUTE\_READWRITE);

if (Mem != NULL) {

if (VirtualLock(Mem, MEMSIZE))

printf("Привязка\n");

else

printf("Ошибка привязки\n");

printf("Нажмите любую клавишу, чтобы продолжить...\n");

\_getch();

if (VirtualUnlock(Mem, MEMSIZE))

printf("Привязка снята\n");

else

printf("Ошибка снятия привязки\n");

if (VirtualFree(Mem, 0, MEM\_RELEASE))

printf("Память освобождена\n");

else

printf("Память не освобождена\n");

}

else printf("Память не выделена\n");

}

int main() {

setlocale(LC\_ALL, "Russian");

writeSystemInfo();

cout << endl;

writeMemoryInfo();

cout << endl;

workWithHeap();

cout << endl;

workWithRAM();

cout << endl;

}

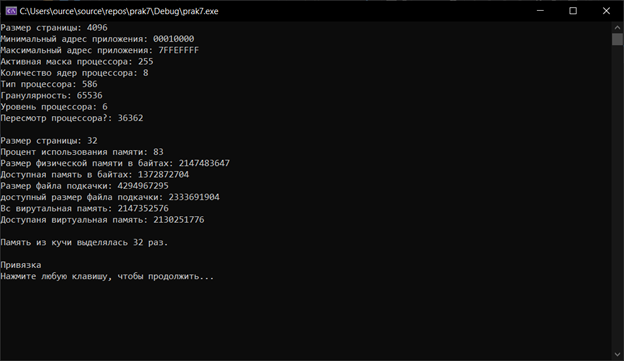


Рис. 1 – Вывод программы.

**Вывод**

В ходе практической работы мы изучили взаимодействие с виртуальной памятью в Windows.

**Список использованных источников**

1. Пошаговое руководство. Создание классического оконного приложение [Электронный ресурс]. URL: https://docs.microsoft.com/ru-ru/cpp/windows/walkthrough-creating-windows-desktop-applications-cpp?view=msvc-160
2. Администрирование в Windows 10. [Электронный ресурс] : [https://docs.microsoft.com/ru-ru/windows/client-management/administrative-tools-in-windows-10 /](https://docs.microsoft.com/ru-ru/windows/client-management/administrative-tools-in-windows-10%20/) Ричард Ворд 2020 — 100 с.
3. «Тайм-менеджмент для системных администраторов». / Лимончелли Т.А. М.: Символ-плюс 2007 — 247 с. ISBN:978-5-93286-090-8