БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет информационных технологий

и робототехники

Кафедра «Программное обеспечение вычислительной техники и автоматизированных систем»

О Т Ч Ё Т

о лабораторной работе № 4

Дисциплина

«Системное программирование»

Тема

«Управление памятью в Windows»

Выполнил: студент гр. 10702217 Храмков Д. С.

Проверил: Разорёнов Н. А.

Минск 2019

***Лабораторная работа № 4***

**УПРАВЛЕНИЕ ПАМЯТЬЮ В WINDOWS**

**Цель работы:** изучить основы распределения и управления памятью в Windows.

**Изучаемые вопросы**

1. Виртуальное адресное пространство:
   1. Виртуальное адресное пространство и физическая память.
   2. Состояние страниц.
   3. Атрибуты защиты.
   4. Границы выделения памяти.
   5. Регионы в адресном пространстве.
2. Работа с виртуальной памятью:
   1. Выделение памяти.
   2. Освобождение памяти.
   3. Изменение атрибутов защиты.
   4. Блокировка физической памяти RAM.
   5. Стек потока.
3. Кучи.
4. Стандартные библиотечные функции языка C.

***Постановка задачи***

Разработать приложение, которое предоставляет пользователю возможность управлять памятью в Windows, получать системную информацию о состоянии физической и виртуальной памяти, а также управлять выделением и освобождением виртуальной памяти.

В программе предусмотреть использование различных видов памяти:

1. Статически распределяемой.
2. Стековой памяти.
3. Динамически распределяемой памяти.
4. Регионов виртуальной памяти.
5. **Виртуальное адресное пространство:**
   1. **Виртуальное адресное пространство и физическая память.**

Каждому процессу выделяется собственное виртуальное адресное пространство. Поскольку каждому процессу отводится закрытое адресное пространство, то, когда в процессе выполняется какой-нибудь другой поток, он получает доступ только к той памяти, которая принадлежит его процессу. Общий объём памяти, доступной для всех исполняемых процессов, является суммой физической памяти и свободного места на диске, доступного для увеличения объёма физической памяти.

* 1. **Состояние страниц.**

Свойства страниц определяет её состояние. Страницы виртуального адресного пространства процессо вмогут находиться в одном из трёх состояние:

* Свободное;
* Зарезервированное;
* Фиксированное.
  1. **Атрибуты защиты.**

Атрибуты защиты определяют степень доступа к заданной области памяти. Выделяют следующие атрибуты:

* PAGE\_NOACCESS;
* PAGE\_READONLY;
* PADE\_READWRITE;
* PAGE\_EXECUTE;
* PAGE\_EXECUTE\_READ;
* PAGE\_EXECUTE\_READWRITE;
* PAGE\_GUARD;
* PAGE\_NOCACHE;
* PAGE\_WRITECOPY;
* PAGE\_EXECUTE\_WRITECOPY.
  1. **Границы выделения памяти.**

Границы выделения памяти обуславливаются гранулярностью обращения к памяти через страницы. Так как страница имеет собственную кратность, зависящую процессора, то граница выделения любого региона кратна этому значению. Возможные значения определяются граничными значениями доступных участков.

* 1. **Регионы в адресном пространстве.**

Регионом является любая часть памяти внутри адресного пространства. Резервируя регион в адресном пространстве, система обеспечивает ещё и кратность размера региона размеру страницы. Так называется единица объёма памяти, используемая системой при управлении памятью. Когда зарезерврованный регион адресного пространства становится ненужным, его следует вернуть в общие ресурсы системы.

1. **Работа с виртуальной памятью:**
   1. **Выделение памяти.**

Для выделения памяти в Win32 приложениях используется функция VirtualAlloc, выполняющая:

* Резервацию определённого количества страниц;
* Передачу физической памяти зарезервированным страницам;
* Резервацию и передачу физической памяти страницам.
  1. **Освобождение памяти.**

Для освобождения памяти в Win32 приложениях используется функция VirtualFree, выполняющая:

* Возврат физической памяти;
* Освобождение региона.
  1. **Изменение атрибутов защиты.**

Изменение атрибутов защиты обычно используется для защиты данных от возможных багов, скрытых в прогармме. Так, например, если какой-то блок кода в программе из-за наличия «блуждающего» указателя обратится к данным в связанном списке, вершины которого хранятся в зарезервированном регионе, возникает нарушение доступа. Чтобы получить доступ можно изменять атрибуты защиты переданной памяти при старте на PAGE\_READWRITE, а при завершении – обратно на PAGE\_NOACCESS.

* 1. **Блокировка физической памяти RAM.**

Т.к. виртуальная память работает значительно медленнее, чем физическая, то для ускорения работы приложения можно «перевести» требуемый участок памяти из виртуальной в физическую память, используя функцию VirtualLock. После того, как программа отработала, и кусок памяти больше не нужен, его можно убрать из физической памяти функцией VirtualUnlock.

* 1. **Стек потока.**

Всякий раз, когда в процессе создаётся поток, система резервирует регион адресного пространства для стека потока и передаёт этому региону какой-то объём физической памяти. Зарезервировав регион, система передёт физическую память двум верхним его страницам. Непосредственно перед тем, как приступить к выполнению потока, систем устанавливает регистр указателя стека на конец верхней страницы региона стека. Регион стека и вся физическая память, переданная ему, имеют атрибут защиты PAGE\_READWRITE.

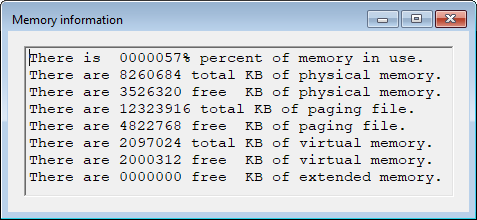
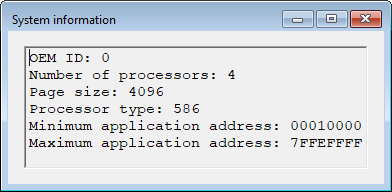
1. **Кучи.**

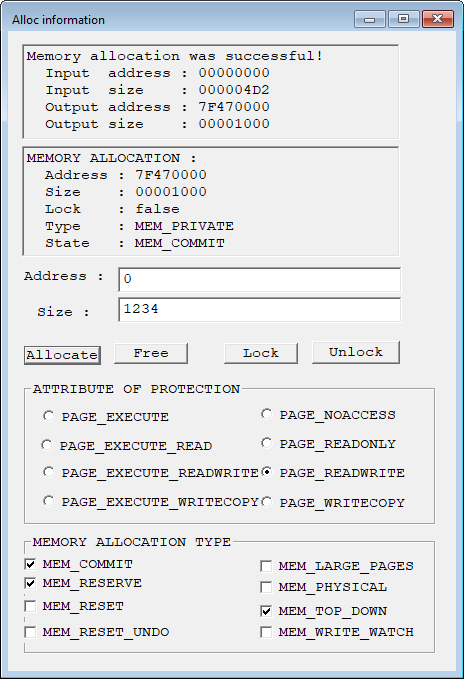
Куча представляет собой часть памяти, зарезервированную для использования в качестве временного запоминающего устройства для структур данных, чей размер не может быть определён, пока программа не запущена. Функции работы с кучей позволяют процессу создавать свою собственную кучу, используя функцию HeapCreate. Память собственной кучи доступна только для создавшего её процесса. По завершению процесса следует уничтожить кучу функцией HeapDestroy.

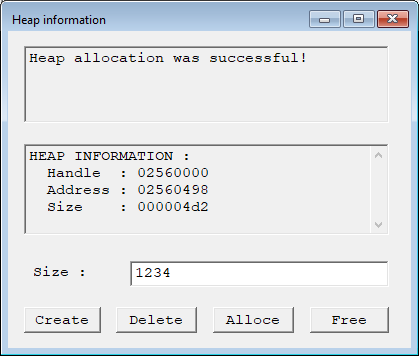
1. **Стандартные библиотечные функции языка C.**

При написании программы по работе с памятью разработчику предоставляется выбор: использовать Win32 API функции, или же реализовывать возможности управления памятью библиотеки программ этапа исполнения языка C и C++. В то время как Win32 API функции предоставляют большую гибкость в использовании и позволяют проводить более тонкую настройку, в то время как библиотечные функции C более устойчивы к ошибкам. Так что в зависимости от задания или цели, которую надо выполнить, можно применять любой из вышеперечисленных методов.

**Результаты выполнения:**







**Вывод:**

* 1. Изучено состояние страниц;
  2. Изучены атрибуты защиты;
  3. Изучены кучи;
  4. Изучены стандартные библиотечные функции языка C.