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 № 5

Дисциплина

«Системное программирование»

Тема

«Создание и использование DLL»

Выполнил: студент гр. 10702217 Храмков Д. С.

Проверил: Разорёнов Н. А.

Минск 2019

***Лабораторная работа № 4***

**СОЗДАНИЕ И ИСПОЛЬЗОВАНИЕ DLL**

**Цель работы:** рассмотреть ряд аспектов создания и использованя динамических библиотек DLL в операционной среде Win32.

**Изучаемые вопросы**

1. Функция DllMain. Последовательность вызовов в мноргопоточном приложении. Назначение и классификация диалоговых окон.
2. Экспорт/импорт функций.
3. Экспорт/импорт ресурсов.
4. Соглавсование интерфейсов.
5. Загрузка неявно подключаемой DLL.
6. Динамическая загрузка и выгрузка DLL.
7. Вызов функции по номеру.
8. Список динамических библиотек процесса.

***Постановка задачи***

Разработать многопоточное Win32-приложение и две DLL-библиотеки, одна из которых загружается явно, а вторая неявно. Первая динамическая библиотека содержит код обработки информации (например, код доступа к системной информации файловых систем FAT или NTFS), вторая – ресурсы типа диалог. Индивидуальное задание получить у преподавателя.

1. **Функция DllMain. Последовательность вызовов в многопоточном приложении. Назначение и классификация диалоговых окон.**

Функция DllMain определяет порядок действий при подключении, отключении библиотеки к процессам/потокам функции. Каждый раз, как новый поток начинает использовать библиотеку функция DllMain запускается с параметром ul\_reason\_for\_call равным DLL\_THREAD\_ATTACH. При прекращении работы этого потока параметр ul\_reason\_for\_call равен DLL\_THREAD\_DETACH.

switch (ul\_reason\_for\_call)

{

case DLL\_PROCESS\_ATTACH: {

MessageBox(NULL, L"PROCESS\_1\_ATTACH", L"msg", MB\_OK);

break;

}

case DLL\_THREAD\_ATTACH: {

MessageBox(NULL, L"THREAD\_1\_ATTACH", L"msg", MB\_OK);

break;

}

case DLL\_THREAD\_DETACH: {

MessageBox(NULL, L"THREAD\_1\_DETACH", L"msg", MB\_OK);

break;

}

case DLL\_PROCESS\_DETACH: {

MessageBox(NULL, L"PROCESS\_1\_DETACH", L"msg", MB\_OK);

break;

}

}

1. **Экспорт/импорт функций.**

Чтобы экспортировать функцию нужно определить её таковой, используя модификатор extern “C” или квалификатор \_declspec(dllexport). Для последующего подключения этих библиотек в исходном коде также необходимо выполнить некотороые команды. При использовании статитечской компоновки, необходимо включать в фкйл проекта приложения соответствующий lib-файл, содержащий нужную библиотеку объектных модулей, содержащую исполняемый код. Если же используется динамическя компановка, в загрузочный файл приложения записывается не исполняемый код функции, а ссылка на соответствующую DLL-библиотеку и функцию внутри неё.

\_\_declspec(dllexport) string GetSystemInformation();

\_\_declspec(dllexport) string GetMemoryInformation();

1. **Экспорт/импорт ресурсов.**

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.

HMODULE secondDll = LoadLibrary(L"C:\\Users\\user\\Desktop\\5\\SecondDll\\Debug\\SecondDll.dll");

1. **Согласование интерфейсов.**

Подключая библиотеки к проекту стоит помнить о согласованности функции с её прототипом. Например, name mangling позволяет C++ библиотекам имплементировать пергрузку, в то время как C библиотеки не поддерживают данный функционал. Для подключения C библиотеки в C++ код необходимо объявлять эти библиотеки как внешние, используя extern “C” блок для применения команды extern “C” для каждой библиотеки из подключаемого C пакета.

extern "C" {

#include "SomeLib.h"

}

1. **Загрузка неявно подключаемой DLL.**

При неявном подключении библиотек в папку с приложением необходимо вставить lib файл, содержащий список указателей на используемые объекты библиотеки.

#include "C:\\Users\\user\\Desktop\\5\FirstDll\\FirstDll\\FirstDll.h"

#pragma comment(lib, "C:\\Users\\user\\Desktop\\5\\FirstDll\\Debug\\FirstDll.lib")

1. **Динамическая загрузка и выгрузка DLL.**

Динамические библиотеки позволяют экономить ресурсы и увеличивать скорость выполнения программы. Для подключения используется функция LoadLibrary с именем загружаемого модуля в качестве аргумента. После завершения работы с динамической библиотекой ресурсы освобождаются командой FreeLibrary.

HMODULE firstDll = LoadLibrary(L"C:\\Users\\user\\Desktop\\5\\FirstDll\\Debug\\FirstDll.dll");

FreeLibrary(firstDll);

1. **Вызов функции по номеру.**

Для упрощения вызова функции её вызов может осуществляться по номеру. Для этого в строке экспорта функции можно явно задать её порядковый номер, поставив перед ним символ @. В дальнейшем этот номер может использоваться для обращения к функции через функцию GetProcAddress.

LIBRARY "SomeLibrary"

EXPORTS

ProcessInfo @1

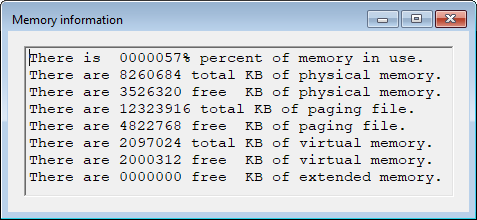
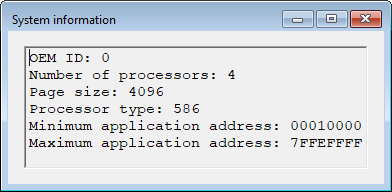
ModuleInfo @2

1. **Список динамических библиотек процесса.**

Существует ряд способов, используемых для получения списка подключённых динамических библиотек процесса. Одним из них является программа dumpbin, включённая в стандартную систему разработки Microsoft Visual C++.

dumpbin C:\Users\user\Desktop\5\Win32Project1\Debug\Win32Project1.exe

**Результаты выполнения:**



**Вывод:**

* 1. Изучена структура функции DllMain;
  2. Освоен процесс экспорта и импорта библиотек;
  3. Приобретены навыки загрузки неявно подключаемой DLL;
  4. Ознакомились с вызовом функции по номеру.