Лабораторна робота № 4

(4 години)

Тема: Робота з динамічними бібліотеками у програмах для Windows.

Мета: Ознайомитися із засобами створення та використання динамічних бібліотек у програмах для Windows

Короткі теоретичні відомості

Динамічно зв'язувані бібліотеки (dynamic link library, DLL, динамічні бібліотеки, бібліотечні модулі) – один із найголовніших структурних елементів Microsoft Windows.

Динамічно зв'язані бібліотеки загалом не є безпосередньо виконуваними програмами, і вони не отримують повідомлень. ***DLL*** – це окремі файли, що містять функції, до яких можуть звертатися програми і інші DLL, щоб виконувати певні завдання. Динамічно зв'язана бібліотека починає діяти, лише, коли інший модуль звертається до однієї з функцій бібліотеки.

Динамічні бібліотеки можуть містити набір функцій та/або ресурси.

Є два способи виклику функцій DLL із програмного модуля:

Динамічне зв'язування часу завантаження (*Неявне завантаження*). Динамічні бібліотеки завантажуються в пам'ять при завантаженні програми. Програмний модуль здійснює явні виклики експортованих функцій DLL. Це вимагає, щоб модуль був пов'язаний з бібліотекою імпорту для DLL. Бібліотека імпорту забезпечує операційну систему інформацією, необхідною для завантаження DLL і визначення розташування експортованих функцій при завантаженні програми.

Динамічне зв'язування часу виконання (*Явне завантаження*). Програмний модуль завантажує необхідні DLL по мірі необхідності під час виконання. Для завантаження бібліотеки викликається функція *LoadLibrary* або *LoadLibraryEx*. Після того, як DLL завантажена в пам'ять, слід викликати функцію GetProcAddress, щоб одержати адресу експортованої функції DLL. Модуль викликає експортовану функцію, використовуючи вказівник на функцію, повернений GetProcAddress. При цьому відпадає потреба в бібліотеці імпорту.

Створення DLL

Створення DLL навіть простіше, ніж програми, оскільки вона, як і будь-яка бібліотека, складається лише із набору функцій.

Для створення DLL необхідно:

1. Створити заголовковий файл (наприклад MyLib.h). Необхідні функції відкривають для зовнішнього використання (експортують) за допомогою модифікатора `__declspec(dllexport)`. У випадку використання мови C++ забороняють спотворення імен за допомогою `extern "C"`.

```
/* ------------------------------------------------
 MyLib.h – заголовковий файл динамічної бібліотеки
 ------------------------------------------------*/

// макрос для експорту функцій без спотворення імен
#ifdef __cplusplus
#define EXPORT extern "C" __declspec (dllexport)
#else
#define EXPORT __declspec (dllexport)
#endif

// прототипи експортованих функцій
EXPORT void Print(LPCTSTR text);
```

2. Створити файл DLL-модуля (наприклад MyLib.c)

```
/* ------------------------------------------
  MyLib.c – файл динамічної бібліотеки
  ------------------------------------------*/
#include <windows.h>
#include "MyLib.h"

EXPORT void Print(LPCTSTR text)
{
     MessageBox(NULL, text, "Title: DLL", MB_OK);
}
```

3. Створити проект: Win-32 Dynamic-Link Library, додати у нього вищенаведені файли і скомпілювати. В результаті одержують бібліотечні файли MyLib.dll і MyLib.lib.

Використання DLL

Для використання DLL обирають один із способів зв'язування (завантаження).

1. Неявне завантаження

1. Створюють проект необхідної програми, додаючи в нього заголовковий файл та виклики бібліотечних функцій, а також повідомляють компонувальнику приєднати файл MyLib.lib (для MS Visual Studio: *Project* → *Settings*, закладка *Link*: в список файлів *Object/Library modules*: додати свій файл бібліотеки імпорту). Файл MyLib.lib додають в робочий каталог проекту.

```
/* ------------------------------------------------------------
  MyProg.c – файл програми із використанням MyLib.dll
  ------------------------------------------------------------*/
#include <windows.h>
#include "MyLib.h"
int main()
{
     Print("Hello!");
     Print("The DLL is working!");
     Print("By!");

     return 0;
}
```

При компіляції програми одержують виконуваний модуль із розширенням .EXE. Оскільки програмний модуль не містить усіх необхідних для його роботи функцій, то в ньому створюється розділ імпорту, в якому перераховуються імена усіх DLL-модулів та функцій, які використовуються. Ця інформація використовується завантажувальником ОС при запуску програми.

При запуску програми із неявним зв'язуванням завантажувальник ОС створює віртуальний адресний простір для процесу і проектує в нього виконуваний модуль. Потім аналізує розділ імпорту, визначає усі DLL і проектує їх на адресний простір. Якщо DLL викликає інші DLL, то процедура повторюється для кожної із таких DLL.

Необхідні для програми DLL шукаються на диску в такій послідовності:

1. Каталог програми.
2. Поточний каталог процесу.
3. Системний каталог Windows.
4. Основний каталог Windows.
5. Каталоги, визначені змінною оточення PATH.

Якщо необхідна DLL не знайдена, видається відповідне повідомлення.

Явне завантаження

При явному завантаженні бібліотеки імпорту не потрібні, а операції по завантаженню необхідної DLL в пам'ять виконуються програмістом. Усі операції проводяться в три етапи:

1. Завантажують необхідні динамічні бібліотеки функцією LoadLibrary(). При успішному завершенні функція повертає значення дескриптора бібліотечного модуля. Якщо функція завершується не успішно, вона повертає NULL.

```
HMODULE hLib;
hLib = LoadLibrary("MyLib.dll");
```

2. Визначають вказівники на необхідні функції із DLL. Для цього використовують функцію GetProcAddress. Функція повертає вказівник на бібліотечну функцію. Для приймому вказівника організовують змінну типу "вказівник на функцію" (див. додаток 1). Оскільки функція дає результат типу **FARPROC**, використовують приведення типів.

```
lpPrint = (MYPROC)(GetProcAddress (hLib,"_Print"));
```

Замість імені функції можна використати її порядковий номер в DLL через макрос MAKEINTRESOURCE(N), де N – номер функції.

3. Викликають необхідні програмі функції через їх вказівники.

```
lpPrint ("Hello!");
```

4. Звільняють динамічну бібліотеку.

```
BOOL FreeLibrary(hLib);
```

Функція входу/виходу DLL

DLL може мати окрему функцію, яка викликається операційною системою в таких випадках:

- коли процес завантажує DLL;
- коли процес, що використовує DLL, викликає новий потік;
- коли завершається потік, який належить процесу, що використовує DLL;
- коли процес звільняє DLL.

Функція входу/виходу має назву DllMain або DLLEntryPoint:

```
BOOL WINAPI DllMain(
   HINSTANCE hinstDLL,          // дескриптор DLL-модуля
   DWORD fdwReason,             // причина виклику DLL
   LPVOID lpvReserved           // додаткова інформація
);
```

Другий параметр **fdwReason** може приймати чотири значення:

DLL_PROCESS_ATTACH	DLL завантажена в адресний простір процесу
DLL_PROCESS_DETACH	DLL відключається від адресного простору процесу
DLL_THREAD_ATTACH	створено новий потік
DLL_THREAD_DETACH	потік завершується

Функція DllMain містить обробники значень fdwReason, які передаються їй через відповідний параметр.

Завдання для виконання

1. Створити власну DLL, що містить кілька власних функцій.
2. Скомпілювати проект і одержати файли бібліотеки імпорту і динамічної бібліотеки.
3. Створити програму, яка викликає бібліотечні функції із DLL, використовуючи неявне зв'язування (при завантаженні).
4. Створити програму, яка викликає бібліотечні функції із DLL, використовуючи явне зв'язування (під час виконання).
5. Створити DLL ConsoleIO.dll, яка призначена для роботи з консоллю і містить функції PrintStrings, PrintMsg, ConsolePrompt. Текст функцій міститься у файлі Printing.c. Написати консольну програму для перевірки DLL.

Контрольні питання

1. Що таке статична і динамічна бібліотеки? Що в них спільного і що відмінного?
2. Які переваги надає використання динамічних бібліотек?
3. Яким чином динамічна бібліотека зв’язується з основною програмою?
4. Як створити динамічну бібліотеку?
5. Що таке бібліотека імпорту і як її одержати?
6. Яким чином програма знаходить необхідну їй динамічну бібліотеку?
7. Що таке явне і неявне зв’язування динамічних бібліотек?
8. Як створити програму яка неявно завантажує динамічну бібліотеку?
9. Як створити програму яка явно завантажує динамічну бібліотеку?
10. Як знайти в динамічній бібліотеці необхідну функцію і як викликати її?

Додаток 1

Оголошення вказівника на функцію

Вказівник на функцію це змінна, яка вказує на функцію певного типу із заданим типом аргументів. Отже повинна існувати функція, на яку буде посилатися вказівник. Нехай треба зробити вказівник на функцію: `int Calc (float x, char s)`

1. Для оголошення вказівника на функцію використовують конструкцію виду:

```
return_type (*pointer_name)(list_of_parameter_types);
```

Наприклад:

```
int (*ptrToFunc) (float, char);
ptrToFunc = Calc;
```

2. Для оголошення вказівника на функцію вводять тип: вказівник на функцію, описують змінну введеного типу і присвоюють їй значення – адресу функції

```
typedef int (*MYPTR) (float, char);
MYPTR ptrToFunc;
ptrToFunc = Calc;
```

3. Оголошення вказівника суміщають із присвоєнням йому значення:

```
int (*ptrToFunc) (float, char) = Calc;
```

Примітка. Середовище DEV-C++ використовує компілятор GCC, який зазвичай створює бібліотеку імпорту з розширенням ".a". Цей файл потрібно додати до проекту основної програми.