Тема 7. Бібліотеки динамічного компонування DLL

План
1. Поняття про DLL.
2. Способи динамічного зв'язування.
3. Створення DLL.
4. Використання DLL
   4.1. Неявне зв'язування.
   4.2. Явне зв'язування.

При створенні програм для Windows широко використовуються бібліотечні модулі.

Існує два способи зв'язування бібліотечних модулів – статичне зв'язування і динамічне зв'язування.

Статичне зв'язування здійснюється на етапі компонування програми у виконуваний модуль. Воно передбачає, що редактор зв'язків копіює код бібліотечної функції у кожний модуль, який викликає її. Але цей підхід марнотратний, тому що кожний із модулів містить ідентичний код для спільних підпрограм.

"Статичне зв'язування" відбувається під час розробки програм, коли об'єктні (.OBJ) модулі, файли бібліотеки (.LIB) часу виконання, і зазвичай скомпільований файл ресурсу (.RES) компонуються для створення .EXE файлу Windows. Якщо змінити одну з підпрограм у бібліотеці, доводиться повторно компонувати усі програми, які використовують змінену підпрограму.

Динамічне зв'язування передбачає, що в кожний модуль включається лише інформація, яка потрібна системі, щоб під час завантаження або виконання програми знайти код експортованої функції в динамічній бібліотеці (DLL) і розмістити його в пам'яті.

Динамічне зв'язування здійснюється під час виконання програми.

Слід чітко розрізняти дві різновидності бібліотечних файлів – об'єктні бібліотеки та бібліотеки імпорту.

Об'єктна бібліотека – це файл із розширенням .LIB, що містить код, який додається компонувальником до .EXE файлу програми в процесі статичного зв'язування. Наприклад, Microsoft Visual C++ має об'єктну C-бібліотеку часу виконання під назвою LIBC.LIB.

Бібліотека імпорту – спеціальна форма файлу об'єктної бібліотеки, яка використовується редактором зв'язків (компонувальником) для організації звернення програми до бібліотечних функцій, які знаходяться в об'єктних бібліотеках. Бібліотеки імпорту також мають розширення .LIB, проте вони не містять ніякого коду, зате містять інформацію про розташування функцій Windows у відповідних DLL, які будуть використовуватися під час виконання програми.

Файли KERNEL32.LIB, USER32.LIB, і GDI32.LIB, що використовуються при компіляції програм – це бібліотеки імпорту для функцій Windows.

1. Поняття про DLL

Динамічно зв'язувані бібліотеки (DLL, динамічні бібліотеки, бібліотечні модулі) – один із найголовніших структурних елементів Microsoft Windows. Більшість дискових файлів, пов'язаних з Windows, – або програмні модулі, або динамічно зв'язувані бібліотечні модулі.

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. Динамічно зв'язана бібліотека починає діяти, лише, коли інший модуль звертається до однієї з функцій бібліотеки.

Прикладом динамічних бібліотек можуть бути системні бібліотеки KERNEL32.DLL, USER32.DLL, і GDI32.DLL; різні файли драйверів, як наприклад KEYBOARD.DRV, SYSTEM.DRV, MOUSE.DRV; і драйвери відео і принтера – всі динамічно зв'язані бібліотеки. Вони є бібліотеками, які можуть використовувати всі програми Windows.

Деякі динамічно зв'язувані бібліотеки (як наприклад файли шрифту) називають "***ресурсними***" (resource-only). Вони містять лише дані (зазвичай у формі ресурсів) і жодного коду. Тому, одним із призначень динамічних бібліотек є забезпечення функції і ресурсів, які можуть використовувати різні програми. У звичайній операційній системі, тільки операційна система безпосередньо містить підпрограми, до яких інші програми можуть звертатися, щоб виконати роботу. У Windows генерується процес одного модуля, що викликає функцію в іншому модулі. Фактично, написанням динамічної бібліотеки розширюється Windows. Також можна вважати, що DLL (у тому числі системні DLL Windows) виступають як розширення до програми.

Хоча модуль динамічно зв'язаної бібліотеки може мати будь-яке розширення (як наприклад .EXE або .FON), стандартне розширення – .DLL. Тільки динамічно зв'язані бібліотеки з розширенням .DLL будуть завантажені автоматично Windows. Якщо файл має інше розширення, програма повинна явно завантажити модуль, використовуючи функцію *LoadLibrary()* або *LoadLibraryEx()*.

Зрозуміло, що динамічні бібліотеки мають найбільший сенс в контексті великої програми. Наприклад, якщо створюється пакет із кількох окремих програм, і кожна із них використовує одні й ті ж підпрограми. Якщо помістити ці загальні підпрограми в динамічно зв'язувану бібліотеку, до якої звертається кожна програма, то одразу вирішуються дві проблеми. По-перше, усі підпрограми зберігаються в одній динамічній бібліотеці, що зменшує місце на диску і в пам'яті. По-друге, зміни, у разі необхідності, вносяться лише в динамічну бібліотеку, а повторне компонування усіх програм не потрібне.

- Переваги DLL:

  Розширення функціональності програми. Код, необхідний для виконання програми, може завантажуватись у пам'ять тоді, коли він потрібний. Можливе використання DLL сторонніх виробників.

  Спрощення керування великим проектом. Якщо DLL використовують кілька програм, то у пам'яті завжди є лише один примірник DLL.

  Використання ресурсів. DLL може містити ресурси (рядки, іконки, курсори, шаблони діалогових вікон та ін.). Ці ресурси можуть використовувати різні програми.

  Спрощення локалізації. Компоненти користувацького інтерфейсу можуть бути оформлені у вигляді динамічної бібліотеки. Створюється по одній DLL для кожної мови. В залежності від вибраної мови підключають ту чи іншу DLL.

  Можливість використання різних мов програмування. Програма може бути написана на одній мові програмування, а DLL на іншій мові.

- Недоліки DLL:

  Збільшення кількості компонентів програми, яка поставляється (exe + dll).

  Збільшення часу завантаження програми.

  Необхідність синхронізації *версій* програми і бібліотек.

2. Способи динамічного зв'язування (завантаження)

Виконуваний код динамічних бібліотек не передбачає самостійного використання. 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.

Відносно новим способом зв'язування є т. зв. ***відкладене зв'язування***. Відкладене зв'язування належить до неявного зв'язування, але відрізняється від нього тим, що DLL завантажується в пам'ять лише тоді, коли програмний модуль звернеться до однієї із функцій DLL.

3. 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 за допомогою модифікатора `__declspec(dllexport)` і забороняють спотворення імен за допомогою `extern "C"`.

```
    #include <windows.h>
// макрос для експорту функцій
    #define EXPORT extern "C" __declspec(dllexport)
// прототипи експортованих функцій
    EXPORT void Print(LPCTSTR text);
```

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

```
#include <stdio.h>
#include "MyLib.h"

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

3. Створюють проект: Win-32 Dynamic-Link Library, додають у нього вищенаведені файли і компілюють. В результаті одержують бібліотечні файли MyLib.dll і MyLib.lib.

4. Використання DLL

Для використання DLL обирають один із способів зв'язування.

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

```
#include <windows.h>
#include "MyLib.h"
int main(int argc, char* argv[])
{
 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 DLLшукаються 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 необхідне повідомлення.

2. 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. Функція повертає значення дескриптора.

```
HINSTANCE LoadLibrary(
  LPCTSTR lpLibFileName       // адреса імені файлу виконуваного модуля
);
```

2. Визначають вказівники на необхідні функції із DLL. Для цього використовують функцію GetProcAddress. Функція повертає вказівник на бібліотечну функцію.

```
FARPROC GetProcAddress(
  HMODULE hModule,            // дескриптор DLL модуля
  LPCSTR lpProcName           // ім'я функції
);
```

3. Викликають необхідні програмі функції через їх вказівники.

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

4. Звільняють динамічну бібліотеку.

```
BOOL FreeLibrary(

  HMODULE hLibModule          // дескриптор завантаженого бібліотечного модуля
);
```