Программирование на C++

Минцифры России

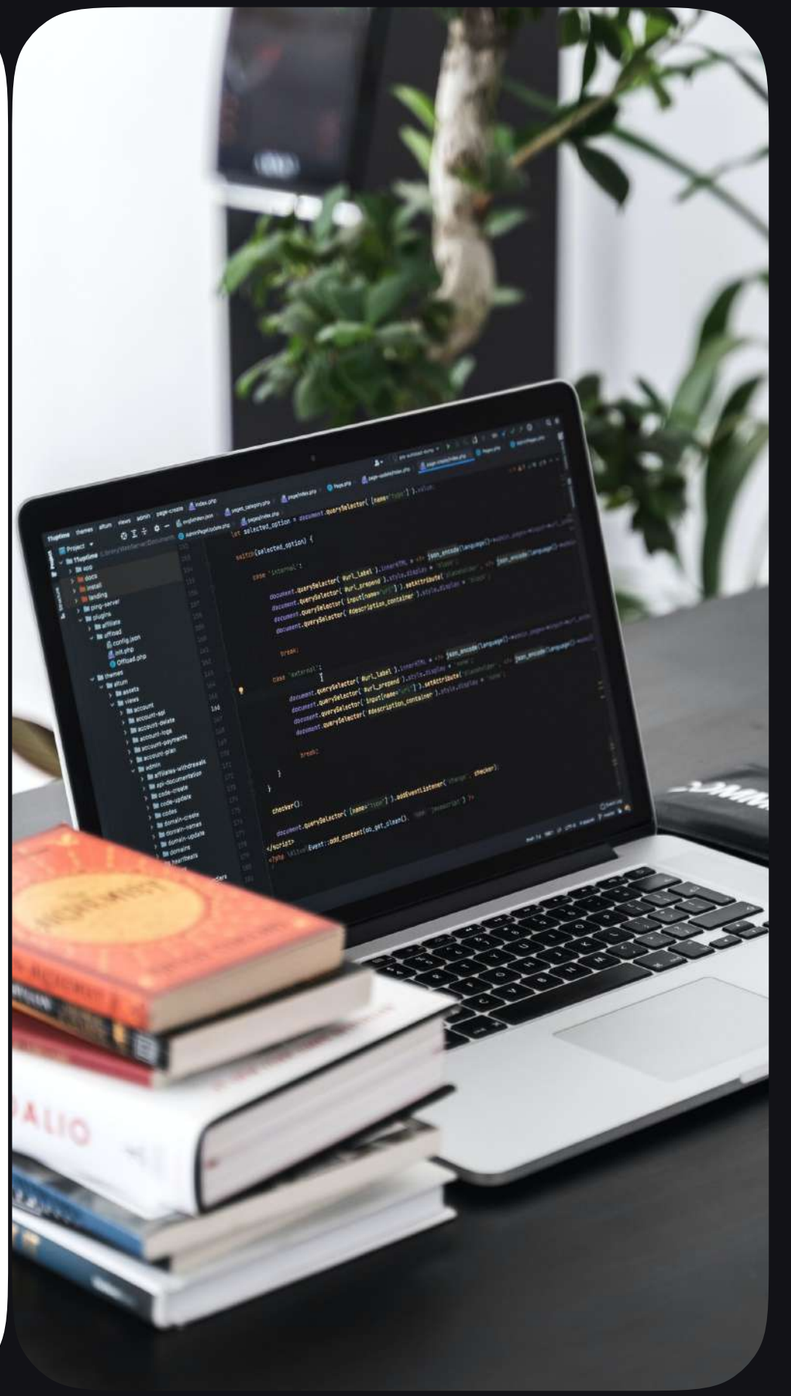
UCHi DOMA

20.35 УНИВЕРСИТЕТ

Урок 2 Модуль 4

Динамическая библиотека. Связывание

Полезные материалы



Цели урока

- изучить, как устроены динамические библиотеки
- отработать на практике написание алгоритмов с использованием динамических библиотек на C++

Недостатки статических библиотек

Библиотека — это пакет кода, который предназначен для повторного использования многими программами.

- Каждый раз, когда вы вносите изменения в свою библиотеку, вы должны перекомпилировать свой код.
- Каждый файл, использующий статическую библиотеку, содержит копию в своем исполняемом файле.
- Размер исполняемого файла становится больше, поскольку код библиотеки хранится внутри исполняемого файла.

Библиотека

Динамическая библиотека (также называемая **shared library**, «общая библиотека») состоит из подпрограмм, которые загружаются в приложение во время выполнения.

Когда компилируется программа, использующая динамическую библиотеку, библиотека не становится частью исполняемого файла — она остается отдельной единицей. В Windows динамические библиотеки обычно имеют расширение .dll (dynamic link library, библиотека динамической компоновки), а в Linux — расширение .so (shared object, общий объект).

Преимущества динамических библиотек

- ✓ Многие программы могут совместно использовать одну копию библиотеки, что экономит место.
- ✓ Динамическую библиотеку можно обновить до более новой версии без замены всех исполняемых файлов, которые ее используют.

Связывание исполняемого файла с библиотекой DLL

Исполняемый файл можно связать с библиотекой DLL (загрузить ее) одним из двух способов:

- Неявное связывание
- Явное связывание

Неявное связывание

- Операционная система загружает библиотеку DLL в тот момент, когда она используется исполняемым файлом.
- Исполняемый файл клиента вызывает экспортированные функции библиотеки DLL так же, как статически скомпонованные и включенные в состав самого исполняемого файла функции.
- Процесс неявного связывания также иногда называют статической загрузкой или динамической компоновкой времени загрузки.

Явное связывание

Операционная система загружает библиотеку DLL по запросу во время выполнения. Исполняемый файл, который использует библиотеку DLL, должен явно загружать и выгружать ее.

Кроме того, в нем должен быть настроен указатель функции для доступа к каждой используемой функции из библиотеки DLL. В отличие от вызовов функций в статически скомпонованной или неявно связанной библиотеке DLL, при работе с явно связанной DLL исполняемый файл клиента должен вызывать экспортированные функции с помощью указателей функций.

Процесс явного связывания также иногда называют динамической загрузкой или динамической компоновкой времени выполнения.

Связывание

Для связывания с одной и той же библиотекой DLL исполняемый файл может использовать любой из этих способов. Кроме того, они не являются взаимоисключающими, то есть два разных исполняемых файла могут связываться с одной библиотекой DLL разными способами.

Библиотека импорта

Поскольку динамические библиотеки не связаны с разрабатываемой программой, программы, использующие динамические библиотеки, должны явно загружать и взаимодействовать с динамической библиотекой. Для упрощения использования динамических библиотек используют библиотеку импорта.

Библиотека импорта — это библиотека, которая автоматизирует процесс загрузки и использования динамической библиотеки.

В Windows это обычно делается с помощью небольшой статической библиотеки (.lib) с тем же именем, что и динамическая библиотека (.dll). Статическая библиотека подключается к программе во время компиляции, и затем функциональные возможности динамической библиотеки можно эффективно использовать, как если бы это была статическая библиотека.

В Linux файл общих объектов (.so) выполняет функции динамической библиотеки и библиотеки импорта.

Большинство компоновщиков при создании динамической библиотеки могут создать библиотеку импорта для этой динамической библиотеки.

Неявное связывание

Неявное связывание происходит в момент, когда из кода приложения вызывается экспортированная функция библиотеки DLL. При компиляции или сборке исходного кода вызывающего исполняемого файла для вызова функции DLL в коде объекта создается ссылка на внешнюю функцию.

Для разрешения этой внешней ссылки приложение должно связаться с библиотекой импорта (LIB-файл), которая предоставляется разработчиком библиотеки DLL.

Неявное связывание

- Библиотека импорта содержит только код для загрузки библиотеки DLL и реализации вызовов ее функций. При поиске внешней функции в библиотеке импорта компоновщик определяет, что код этой функции находится в библиотеке DLL.
- Для разрешения внешних ссылок на библиотеки DLL компоновщик просто добавляет в исполняемый файл сведения, на основании которых система определяет, где следует искать код DLL при запуске процесса.

Неявное связывание

- При запуске программы, которая содержит динамически связываемые ссылки, система использует сведения из исполняемого файла для поиска необходимых библиотек DLL.
- Если найти библиотеку DLL не удается, система завершает процесс и отображает диалоговое окно с сообщением об ошибке. В противном случае система сопоставляет модули DLL в адресном пространстве процесса.

Неявное связывание

- Если любая из библиотек DLL содержит функцию точки входа для инициализации и завершения кода, например **DllMain**, операционная система вызывает эту функцию. С помощью одного из параметров, передаваемого в функцию точки входа, задается код, который указывает на связывание библиотеки DLL с процессом.
- Если функция точки входа не возвращает значение TRUE, система завершает процесс и возвращает ошибку.

Использование неявного связывания

Чтобы использовать библиотеку DLL посредством неявного связывания, исполняемые файлы клиента должны получить от ее поставщика следующие файлы:

- Один или несколько файлов заголовка (с расширением h), в которых содержатся объявления экспортированных данных, функций и классов C++ в библиотеке DLL. Все классы, функции и данные, экспортируемые из библиотеки DLL, должны быть отмечены в файле заголовка как __declspec(dllimport).
- Библиотека импорта, которая связывается с исполняемым файлом. Библиотека импорта создается компоновщиком при построении DLL.
- DLL-файл библиотеки.

Использование неявного связывания

Чтобы использовать данные, функции и классы в библиотеке DLL посредством неявного связывания, исходный файл клиента должен содержать файлы заголовков, в которых они объявляются. С точки зрения кода вызовы экспортированных функций выполняются так же, как и в случае с любыми другими функциями.

Явное связывание применяется, когда:

- Имя библиотеки DLL, которую необходимо загружать, становится известно приложению только во время выполнения. Например, приложение может получать имя библиотеки DLL и экспортированные функции из файла конфигурации во время запуска.
- Если при запуске не удается найти нужную библиотеку DLL, процесс, в котором используется неявное связывание, завершается операционной системой. Процесс с явным связыванием в таких ситуациях не завершается и может попытаться восстановиться после ошибки. Например, процесс может уведомить пользователя об ошибке и запросить указать другой путь к библиотеке DLL.
- Процесс с неявным связыванием также завершается, если в любой из связанных библиотек DLL функция DllMain завершается сбоем. Процесс с явным связыванием в таких ситуациях не завершается.
- Приложение, в котором применяется неявное связывание с множеством библиотек DLL, может долго запускаться, поскольку операционная система Windows при загрузке приложения загружает все библиотеки DLL. Чтобы ускорить процесс запуска приложения, можно выполнять неявное связывание только с теми библиотеками DLL, которые необходимы непосредственно после загрузки. Другие библиотеки DLL могут загружаться позднее по мере необходимости посредством явного связывания.
- При явном связывании приложению не требуется библиотека импорта.

Явное связывание с библиотекой DLL

Чтобы использовать библиотеку DLL посредством явного связывания, необходимо вызвать функцию для явной загрузки библиотеки DLL во время выполнения. Для явного связывания с библиотекой DLL приложение должно выполнить следующие действия:

- Вызвать LoadLibraryEx или аналогичную функцию для загрузки библиотеки DLL и получения дескриптора модуля.
- Вызвать GetProcAddress, чтобы получить указатель для каждой экспортированной функции, которую вызывает приложение. Поскольку приложения вызывают функции DLL с помощью указателя, компилятор не создает внешние ссылки и нет необходимости связываться с библиотекой импорта. Тем не менее необходимо использовать инструкцию **typedef** или **using**, определяющую сигнатуру вызова для вызываемых экспортированных функций.
- По завершении работы с библиотекой DLL вызовите FreeLibrary.

Пример

Вызывается **LoadLibrary** для загрузки библиотеки MyDLL, затем вызывается **GetProcAddress** для получения указателя на функцию DLLFunc1, далее вызывается эта функция и сохраняется результат, после чего вызывается **FreeLibrary** для выгрузки библиотеки DLL.

В большинстве случаев **LoadLibrary** и **FreeLibrary** следует вызывать в приложении только один раз для нужной библиотеки DLL.

```
#include "windows.h"

typedef HRESULT (CALLBACK* LPFNDLLFUNC1)(DWORD,UINT*);

HRESULT LoadAndCallSomeFunction(DWORD dwParam1, UINT * puParam2)
{
    HINSTANCE hDLL;               // Handle to DLL
    LPFNDLLFUNC1 lpfnDllFunc1;    // Function pointer
    HRESULT hrReturnVal;

    hDLL = LoadLibrary("MyDLL");
    if (NULL != hDLL)
    {
        lpfnDllFunc1   (LPFNDLLFUNC1)GetProcAddress(hDLL, "DLLFunc1");
        if (NULL != lpfnDllFunc1)
        {
            // call the function
            hrReturnVal = lpfnDllFunc1(dwParam1, puParam2);
        }
        else
        {
            // report the error
            hrReturnVal = ERROR_DELAY_LOAD_FAILED;
        }
        FreeLibrary(hDLL);
    }
    else
    {
        hrReturnVal = ERROR_DELAY_LOAD_FAILED;
    }
    return hrReturnVal;
}
```