Тема 6. Программные ресурсы ЭВМ

Динамически подключаемая библиотека (DLL, Dynamic Link Library) − программный модуль, загружаемый в виртуальную память процесса как статически (во время создания процесса), так и динамически (во время выполнения процесса). Динамически подключаемые библиотеки предназначены для хранения библиотек функций, которые могут использоваться разными приложениями.

Программный модуль хранится в файле с расширением dll. Для загрузки библиотеки в память используется механизм отображения файлов в память. Процессы совместно используют один экземпляр кода из динамически подключаемой библиотеки, а набор переменных каждый процесс использует свой.

Библиотеки позволяют избежать недостатков, присущих непосредственной сборке файлов в один проект:

• библиотечные функции не связываются во время компоновки. Вместо этого их связывание осуществляется во время загрузки программы (неявное связывание) или во время ее выполнения(явное связывание). Это позволяет существенно уменьшить размер модуля программы, поскольку библиотечные функции в него не включаются;

- возможно создание общих библиотек (shared libraries). Одну и ту же библиотеку могут совместно использовать несколько одновременно выполняющихся программ, но в память будет загружена только одна ее копия. Все программы отображают код DLL на адресные пространства своих процессов, хотя каждый поток будет иметь собственный экземпляр неразделяемого хранилища в стеке;
- новые версии программ могут поддерживаться путем простого предоставления новой версии DLL, а все программы, использующие эту библиотеку, могут выполняться как новая версия без внесения каких бы то ни было дополнительных изменений;
- в случае явного связывания решение о выборе версии библиотеки программа может принимать во время выполнения. Библиотека выполняется в том же процессе и том же потоке, что и вызывающая программа.

Библиотеки используются практически в любой ОС. В UNIX библиотеки фигурируют под названием "разделяемых библиотек" (shared libraries). В Windows библиотеки используются, помимо прочего, для создания интерфейсов ОС. Весь Windows API поддерживается одной DLL, которая для предоставления дополнительных услуг вызывает ядро Windows.

Один код DLL может совместно использоваться несколькими процессами Windows, но после его вызова он выполняется как часть вызывающего процесса или потока, Поэтому библиотека может использовать ресурсы вызывающего процесса, например дескрипторы файлов, и стек потока. Следовательно, DLL

должны создаваться таким образом, чтобы обеспечивалась безопасная многопоточная поддержка (thread safety). Кроме того, DLL могут экспортировать переменные, а также точки входа функций.

Неявное связывание, или связывание во время загрузки (load-time linking) является простейшей из двух методик связывания. Порядок действий в случае использования Microsoft C++ следующий:

1. После того как собраны все необходимые для новой DLL функции, осуществляется сборка DLL.

2. В процессе сборки создается библиотечный .LIB-файл. Этот файл должен помещаться в каталог библиотек общего пользования, указанный в проекте.

3. В процессе сборки создается также DLL-файл, содержащий исполняемый модуль. В типичных случаях этот файл размещается в том же каталоге, что и приложение, которое будет его использовать, и приложение загружает DLL в процессе своей инициализации. Вторым возможным местом расположения DLL является рабочий каталог, а далее ОС будет осуществлять поиск .DLL-файла в системном каталоге, каталоге Windows, а также в путях доступа, указанных в переменной окружения PATH.

4. В исходном коде DLL следует предусмотреть экспортирование интерфейсов функций.

При добавлении функции следует объявить ее экспортируемой. В процессе сборки создаются файлы с расширениями lib (библиотека-заглушка, которая должна быть скомпонована с вызывающей программой для разрешения внешних ссылок и создания актуальных связей с dll-файлом во время загрузки) и dll (библиотека).

Явное связывание или связывание во время выполнения (run-time linking), требует, чтобы в программе содержались конкретные указания относительно того, когда именно необходимо загрузить или освободить библиотеку DLL. Далее программа получает адрес запрошенной точки входа и использует этот адрес в качестве указателя при вызове функции. В вызывающей программе функция не объявляется; вместо этого в качестве указателя на функцию объявляется переменная. Поэтому во время компоновки программы присутствие библиотеки не является обязательным.

Поскольку библиотеки DLL являются совместно используемым ресурсом, системой поддерживается счетчик ссылок на каждую DLL (который увеличивается на единицу при каждом вызове любой из указанных выше функций загрузки).

При использовании библиотек DLL могут возникнунь проблемы, связанные с совместимостью различных версий:

• в результате добавления новых функций в случае неявного связывания могут стать недействительными смещения, определенные для приложений во время компоновки с .lib-файлами. От этой проблемы можно избавиться, применив явное связывание;

• поведение новых версий функций может быть иным, в результате чего существующие приложения могут испытывать проблемы, если не будут своевременно обновлены;

• для приложений, использующих обновленную функциональность DLL, возможны случая связывания с прежними версиями DLL.