**Лабораторная работа №8**

**Библиотеки**

**Цель работы:** Получение практических навыков в работе со статическими и динамическими библиотеками.

Перед выполнением задания прочитайте его полностью. Схематично представьте себе, что от вас требуется и только после этого выполняйте.

**Постановка задачи для Windows:**

**Приложение Mappings:**

Разработайте статическую библиотеку, содержащую функции для работы с маппингом из лабораторной работы **№7**.

**Приложение Lab-08a:**

Разработайте клиентское приложение на основе консольного меню из лабораторной работы **№7**, но с использованием статической библиотеки **Mappings**.

Проверить факт подключения библиотеки через Process Explorer.

**Приложение Mappingd:**

Разработайте динамическую библиотеку на языке **C**, содержащую функции для работы с маппингом из лабораторной работы **№7**.

Экспорт функций организовать через **.def** файл.

**Приложение Mappingd-2:**

Скопируйте полученную библиотеку и замените экспорт функций с **.def** файла на модификаторы **declspec(dllexport)**.

Просмотреть список экспортируемых функций обеих библиотек через утилиту **dumpbin**.

**Приложение Lab-08b:**

Разработайте клиентское приложение на основе консольного меню из лабораторной работы **№7**, но с использованием динамических библиотек **Mappingd** и **Mappingd-2**.

Подключение библиотеки должно быть явным, с использованием функций WinAPI. Хотя бы одна из функций должна быть импортирована по порядковому номеру.

Просмотреть список импортируемых функций через утилиту **dumpbin**.

Проверить факт подключения библиотеки через Process Explorer.

**Постановка задачи для Linux:**

**Приложение Mappings:**

Разработайте статическую библиотеку, содержащую функции для работы с маппингом из лабораторной работы **№7**.

**Приложение Lab-08c:**

Разработайте клиентское приложение на основе консольного меню из лабораторной работы **№7**, но с использованием статической библиотеки **Mappings**.

**Приложение Mappingd:**

Разработайте динамическую библиотеку на языке **C**, содержащую функции для работы с маппингом из лабораторной работы **№7**.

Просмотреть список экспортируемых функций через утилиту **objdump**.

**Приложение Lab-08d:**

Разработайте клиентское приложение на основе консольного меню из лабораторной работы **№7**, но с использованием динамической библиотеки **Mappingd**.

Подключение библиотеки должно быть явным, с использованием функций POSIX.

Просмотреть список импортируемых функций через утилиту **objdump**.

**Дополнительно (совершенно не обязательно, но огромный плюс):**

Написать примеры внедрения кода с использованием динамических библиотек как на Windows, так и на Linux (примеры из лекций сдавать как свои работы бесполезно ^\_^).

Написать пример демонстрирующий работу с extern С.

**Требования к выполненной работе:**

* Разработанные приложения должны быть скомпилированы и протестированы как на Windows, так и на ОС из семейства Linux.
* Для генерации и сборки проекта использовать средство CMake.
* Для компиляции файлов при сборке проекта использовать компилятор Clang.

**Вопросы для контроля:**

1. Что такое библиотека?

Часто бывает так, что некоторые из исходных файлов можно было бы использовать в нескольких программах. Первым делом, чтобы не заниматься лишней работой, эти файлы можно скомпилировать только один раз и затем уже по необходимости компоновать их с разными исполняемыми файлами. И хотя такой подход уменьшает время компиляции, он все равно не избавляет от необходимости каждый раз указывать все объектные файлы на этапе компоновки. Более того, с увеличением количества таких файлов можно создать неразбериху в каталоге проекта

Чтобы обойти эти проблемы, можно сгруппировать набор объектных файлов в единую сущность – библиотеку объектов (или объектную библиотеку). Библиотеки объектов бывают двух видов: статические и разделяемые (динамические).

1. Что такое статическая библиотека?

Статическая библиотека, по сути, является обычным файлом, содержащим копии всех помещенных в него объектных файлов. В архиве также хранятся различные атрибуты для каждого объектного файла, включая права доступа, числовые идентификаторы пользователя и группы и время последнего изменения.

1. Как подключить статическую библиотеку?

Подключение статической библиотеки включает два основных этапа:

1. На этапе компиляции

Вы должны подключить заголовочный файл (.h), который содержит объявления функций или классов из библиотеки. Это нужно для того, чтобы компилятор знал, как вызывать функции из библиотеки.

2. На этапе линковки

На этом этапе необходимо указать компоновщику файл статической библиотеки (.a в Linux или .lib в Windows). Этот файл содержит машинный код функций, собранных в библиотеке.

1. На каком этапе сборки происходит непосредственно загрузка кода?

В результате такого связывания весь объектный код, содержащийся в библиотеке, внедряется в будущий исполняемый файл на этапе компоновки

1. Что такое динамическая библиотека?

Когда программа компонуется со статической библиотекой (или вовсе без использования библиотек), итоговый исполняемый файл содержит копии всех объектных модулей, скомпонованных с программой.

Таким образом, несколько разных программ могут содержать в себе копии одних и тех же объектных модулей. Подобная избыточность несет в себе несколько недостатков:

* дисковое пространство уходит на хранение нескольких копий одних и тех же объектных модулей. Такие потери могут быть значительными
* если несколько программ, применяющих одни и те же модули, выполняются одновременно, каждая из них будет хранить в виртуальной памяти свою отдельную копию этих модулей, увеличивая тем самым потребление виртуальной памяти в системе
* если объектный модуль статической библиотеки требует каких-либо изменений (возможно, нужно закрыть дыру в безопасности или исправить ошибку), придется заново компоновать все исполняемые файлы, в которых этот модуль используется. Данный недостаток усугубляется тем фактом, что системному администратору необходимо знать, с какими приложениями скомпонована библиотека

Для устранения представленных недочетов были придуманы разделяемые (динамические) библиотеки.

1. Какой механизм лежит в основе работы динамических библиотек?

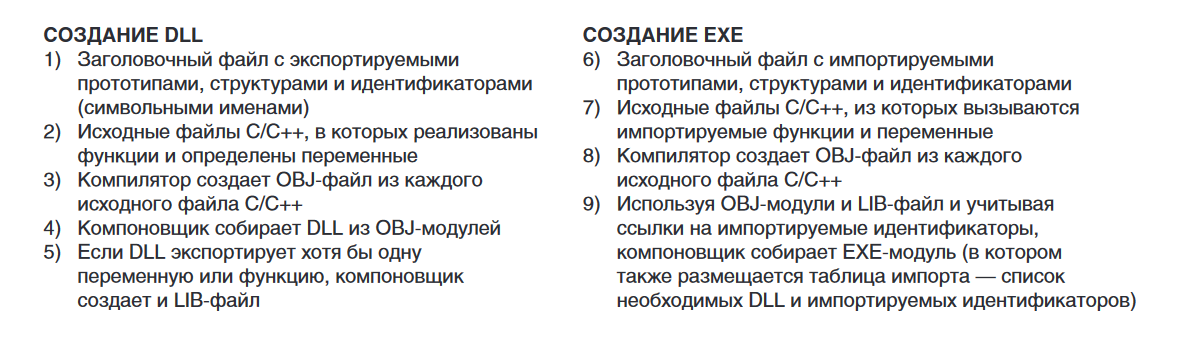
**Разделяемая память.**

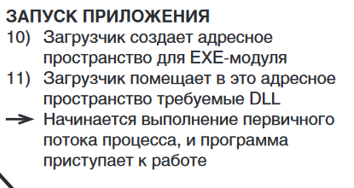
Их ключевая идея состоит в том, что одна копия объектного модуля разделяется между всеми программами, задействующими его. Объектные модули не копируются в компонуемый исполняемый файл; вместо этого единая копия библиотеки загружается в память при запуске первой программы, которой требуются ее объектные модули. Если позже будут запущены другие программы, использующие эту разделяемую библиотеку, они обращаются к копии, уже загруженной в память. Благодаря применению разделяемых библиотек исполняемые файлы требуют меньше места на диске и в виртуальной памяти (при выполнении).

1. Назовите два способа подключения динамической библиотеки? Кратко поясните порядок подключений.

**Общая картина работы с DLL (неявный способ)**

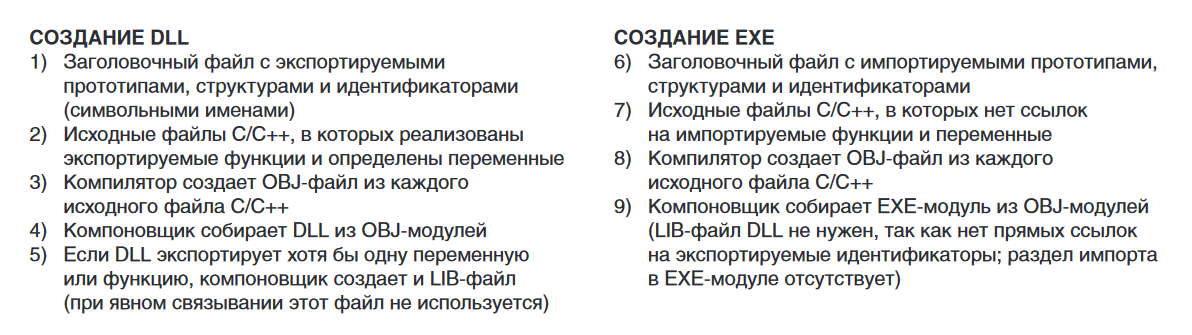
Рассмотренный способ подключения DLL можно подытожить следующим способом:

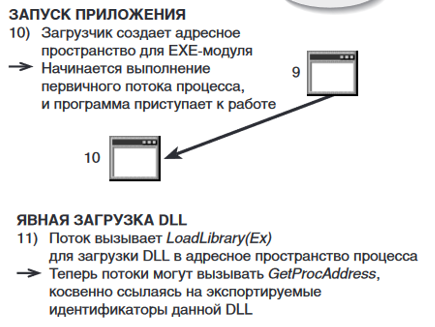




**Общая картина работы с DLL (явный способ)**

Рассмотренный способ явного подключения DLL можно подытожить следующим способом:





В конце - FreeLibrary

1. Что такое библиотека импорта?

Если он указан перед переменной, прототипом функции или C++-классом, компилятор Microsoft С/С++ встраивает в конечный OBJ-файл дополнительную информацию. Она понадобится компоновщику при сборке DLL из OBJ-файлов.

Обнаружив такую информацию, компоновщик создает LIB-файл со списком идентификаторов, экспортируемых из DLL. Этот LIB-файл нужен при сборке любого EXE-модуля, ссылающегося на такие идентификаторы, и называется библиотекой импорта

1. Что такое DLL-injection?

DLL injection – это техника используемая для запуска кода в адресном пространстве другого процесса путем принуждения загрузки в него DLL

1. Для чего нужен extern “C”?

Альтернативным способом сохранения имени функции будет модификатор extern “C”.

Импортируя идентификатор, необязательно прибегать к \_declspec(dllimport) – можно использовать стандартное ключевое слово extern языка C. Но компилятор создаст чуть более эффективный код, если ему будет заранее известно, что идентификатор, на который мы ссылаемся, импортируется из LIB-файла DLL-модуля