**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.
* Задания для ОС Windows должны быть выполнены с использованием функций исключительно из WinAPI, а для ОС Linux из POSIX API.

**Вопросы для контроля:**

1. Что такое библиотека?
2. Что такое статическая библиотека?
3. Как подключить статическую библиотеку?
4. На каком этапе сборки происходит непосредственно загрузка кода?
5. Что такое динамическая библиотека?
6. Какой механизм лежит в основе работы динамических библиотек?
7. Назовите два способа подключения динамической библиотеки? Кратко поясните порядок подключений.
8. Что такое библиотека импорта?
9. Что такое DLL-injection?
10. Для чего нужен extern “C”?

Вот ответы на ваши вопросы, основанные на содержании лекции:

### 1. Что такое библиотека?

Библиотека — это набор объектных файлов, сгруппированных в единую сущность, которые могут использоваться при сборке приложений. Существует два типа библиотек: статические и динамические.

### 2. Что такое статическая библиотека?

Статическая библиотека представляет собой файл, содержащий копии всех объектных файлов, которые в нее помещены. При компоновке исполняемого файла весь объектный код из библиотеки внедряется в исполняемый файл на этапе компоновки.

### 3. Как подключить статическую библиотеку?

Чтобы подключить статическую библиотеку, нужно указать компилятору путь к библиотеке и её название.

Чтобы подключить статическую библиотеку, в командной строке компоновщика нужно указать имя библиотеки вместо длинного списка объектных файлов. Компоновщик сам найдет и извлечет необходимые объекты.

### 4. На каком этапе сборки происходит непосредственно загрузка кода?

Загрузка кода происходит во время компоновки (link-time) для статических библиотек и во время исполнения (run-time) для динамических библиотек.

### 5. Что такое динамическая библиотека?

Динамическая библиотека — это библиотека, которая загружается в память при запуске программы. Вместо того чтобы копировать объектные модули в исполняемый файл, программы обращаются к единой копии библиотеки в памяти.

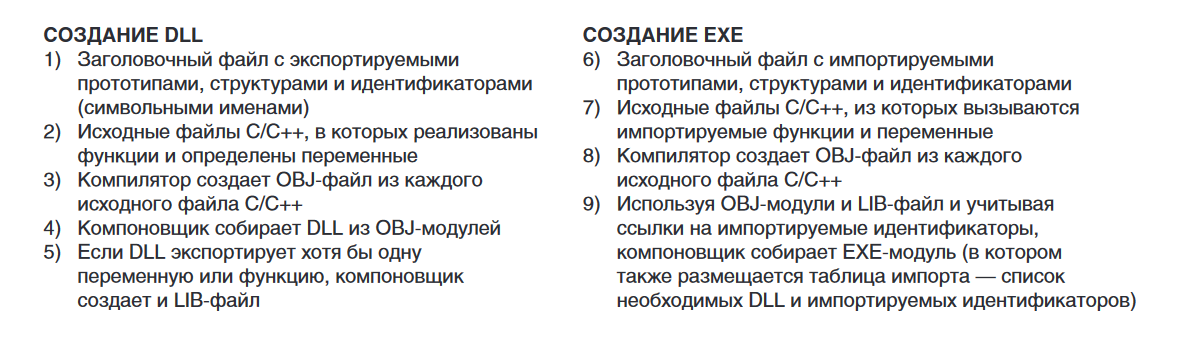
### 6. Какой механизм лежит в основе работы динамических библиотек?

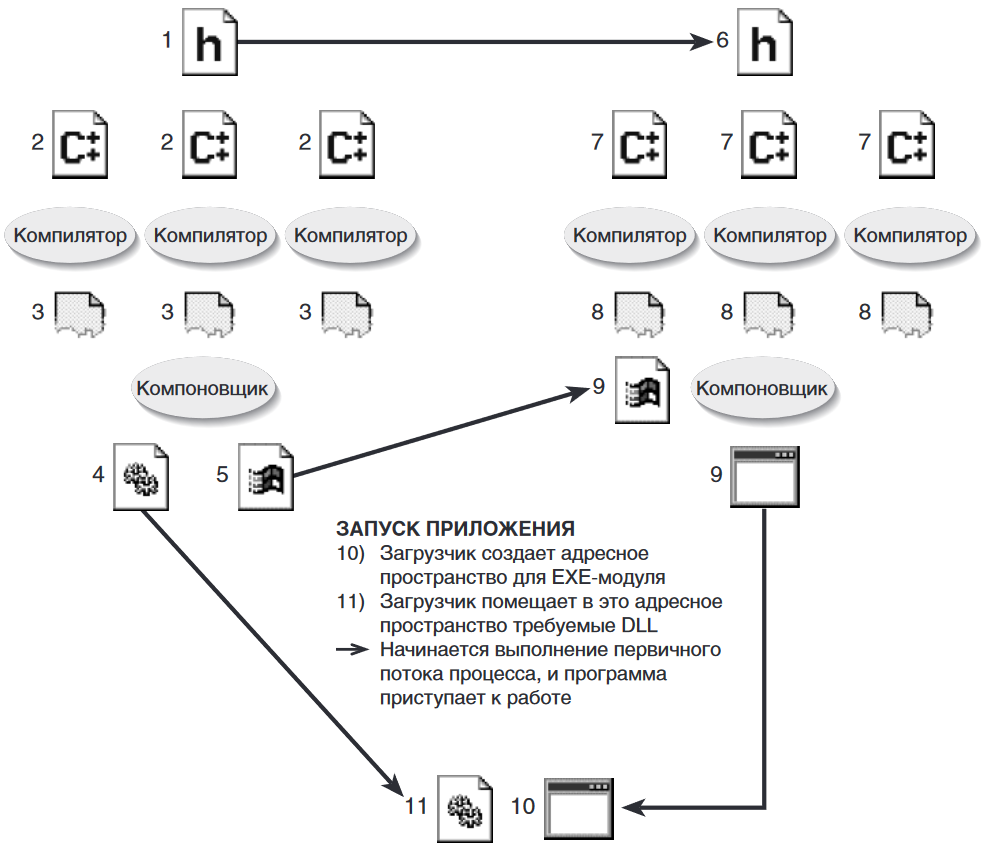
Чтобы приложение (или другая DLL) могло вызывать функции, содержащиеся в DLL, образ ее файла нужно сначала спроецировать на адресное пространство вызывающего процесса. Это достигается либо за счет неявного связывания при загрузке, либо за счет явного – в период выполнения

Как только DLL спроецирована на адресное пространство вызывающего процесса, ее функции доступны всем потокам этого процесса. Фактически библиотеки при этом теряют почти всю индивидуальность: для потоков код и данные DLL – просто дополнительные код и данные, оказавшиеся в адресном пространстве процесса. Когда поток вызывает из DLL какую-то функцию, та считывает свои параметры из стека потока и размещает в этом стеке собственные локальные переменные. Кроме того любые созданные кодом DLL объекты принадлежат вызывающему потоку или процессу – DLL ничем не владеет.

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

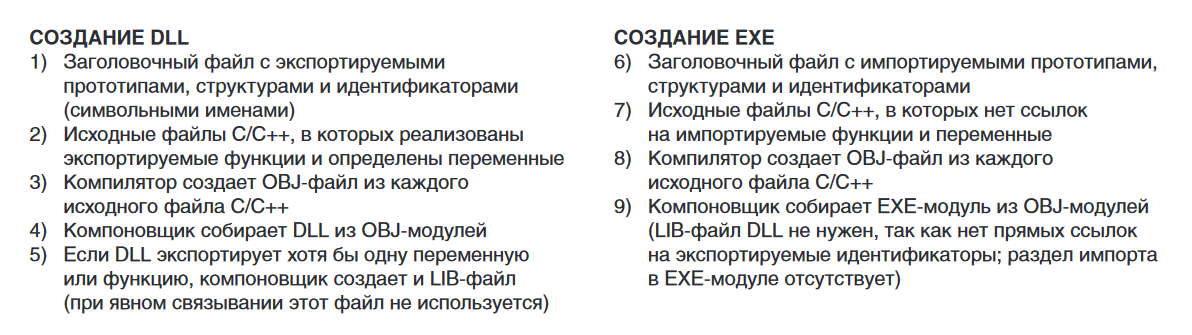
1. \*\*Неявное связывание (отложенное декларативное связывание)\*\*: Библиотека загружается автоматически при запуске программы. Компоновщик создает библиотеку импорта, которая содержит ссылки на необходимые функции.
2. **Общая картина работы с DLL (неявный способ)**

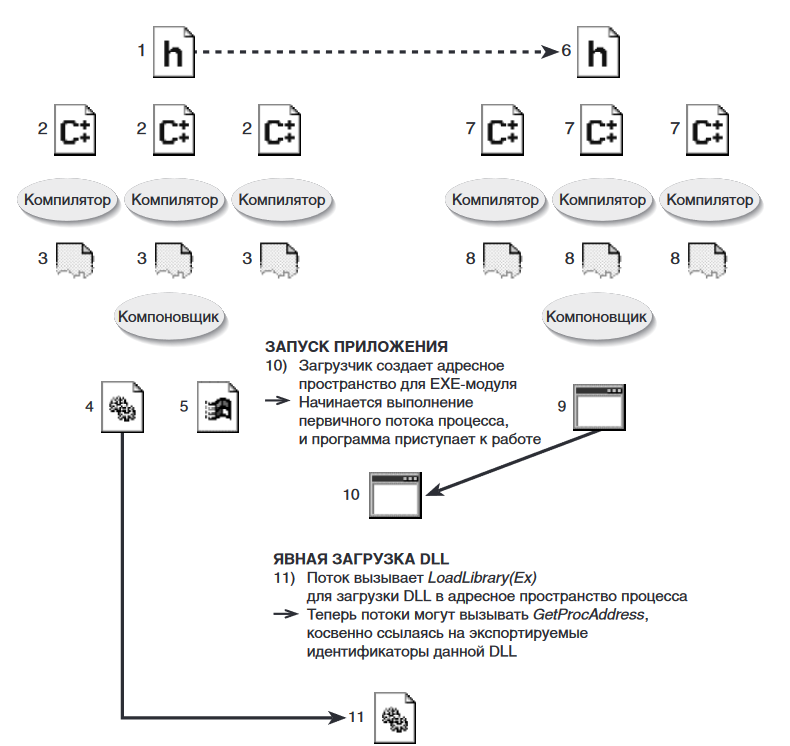




1. \*\*Явное связывание (отложенное императивное связывание)\*\*: Программа загружает библиотеку и функции в процессе выполнения с помощью функций API, таких как `LoadLibrary` и `GetProcAddress`.

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





### 8. Что такое библиотека импорта?

**\_\_declspec(dllexport)**

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**

Компоновщик также вставляет в конечный DLL-файл **таблицу** экспортируемых идентификаторов – **[раздел экспорта](https://learn.microsoft.com/en-us/windows/win32/debug/pe-format)**, в котором содержится список (в алфавитном порядке) идентификаторов экспортируемых функций, переменных и классов. Туда же помещается **[относительный виртуальный адрес](https://learn.microsoft.com/en-us/windows/win32/debug/pe-format)** (**relative virtual address, RVA**) каждого идентификатора внутри DLL-модуля.

Библиотека импорта — это специальный файл, который содержит ссылки на все идентификаторы, экспортируемые из динамической библиотеки. Она необходима для связывания программ с динамическим кодом.

**\_\_declspec(dllimport)**

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

Разрешая ссылки на импортируемые идентификаторы, компоновщик создает в конечном EXE-модуле **[раздел или таблицу импорта](https://learn.microsoft.com/en-us/windows/win32/debug/pe-format)** (**imports section**). В нем перечисляются DLL, необходимые этому модулю, и идентификаторы, на которые есть ссылки из всех используемых DLL

### 9. Что такое DLL-injection?

DLL-injection — это техника, используемая для запуска кода в адресном пространстве другого процесса путем принуждения загрузки в него DLL. Это позволяет выполнять код в контексте другого процесса.

**DLL-injection** — это метод, с помощью которого внешний процесс может "вставить" свою динамическую библиотеку (DLL) в процесс другого приложения.

### 10. Для чего нужен extern “C”?

`extern "C"` используется для предотвращения изменения имен функций (name mangling) компилятором C++. Это позволяет сохранить имена функций в том виде, в каком они объявлены, что упрощает их использование в динамических библиотеках и при связывании с другими языками.