**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?
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”?
11. Что такое библиотека?

Библиотека — это совокупность функций, классов, структур и других элементов кода, которые можно переиспользовать в различных программах.

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

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.

На каком этапе сборки происходит непосредственно загрузка кода?

Как понятно из названия при использовании статической библиотеки речь идёт о раннем (статическом) связывании

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

Что такое динамическая библиотека?

**Динамическая библиотека** — это файл, содержащий компилированный код (функции, классы, переменные), который может быть загружен и использован программой во время её выполнения.

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

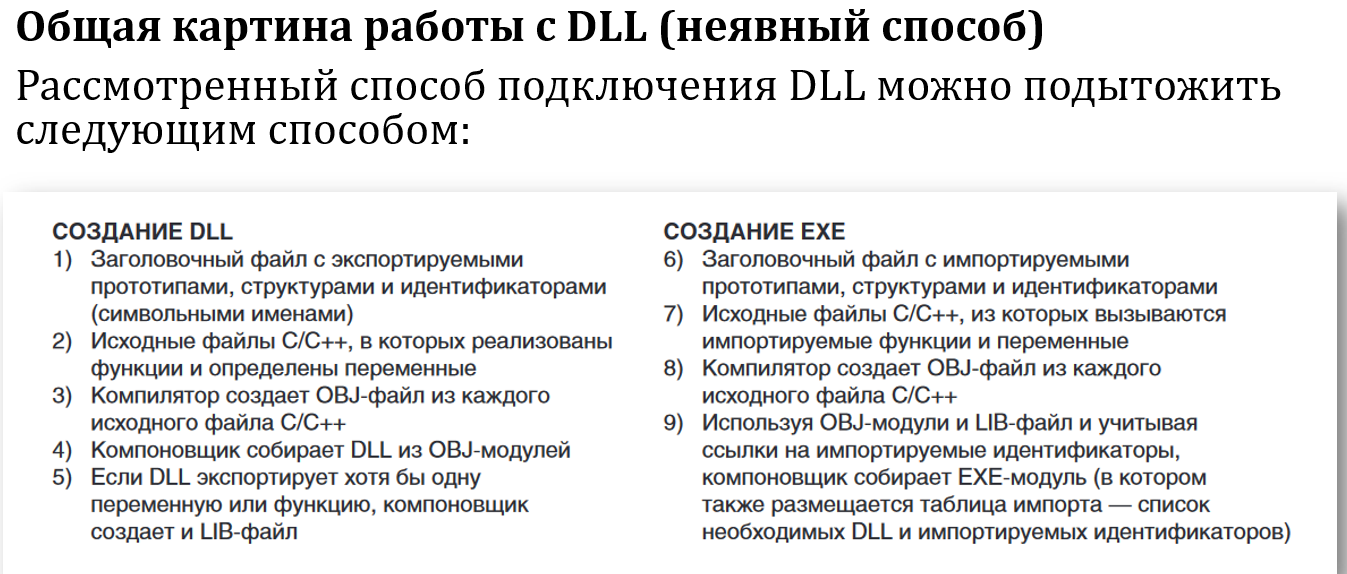
Их ключевая идея состоит в том, что одна копия объектного модуля разделяется между всеми программами, задействующими его. Объектные модули **не копируются** в компонуемый исполняемый файл; вместо этого единая копия библиотеки загружается в память при запуске первой программы, которой требуются ее объектные 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)

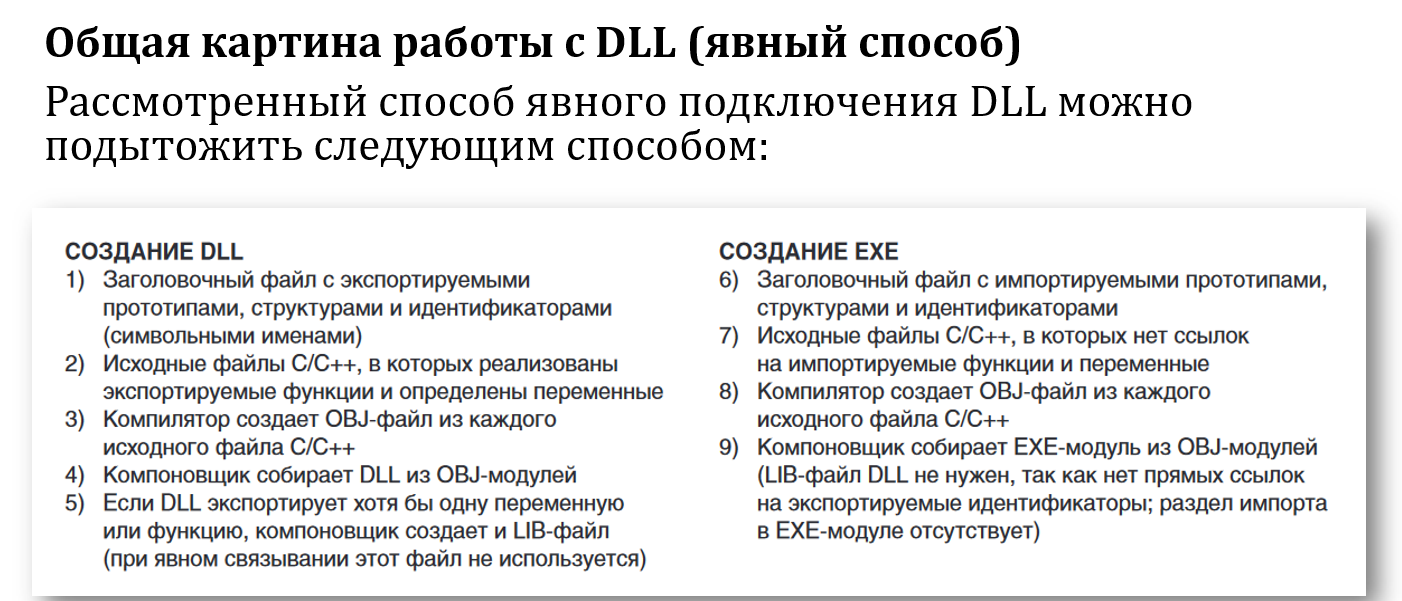
Какой механизм лежит в основе работы динамических библиотек?

### **Механизм работы динамических библиотек**

В основе работы динамических библиотек лежит **динамическое связывание** (dynamic linking) и **динамическая загрузка** (dynamic loading). Эти процессы управляются операционной системой и позволяют программе использовать код из библиотеки в момент выполнения, а не во время компиляции или линковки.

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





Что такое библиотека импорта?

**Библиотека импорта** (import library) — это специальный файл, используемый в процессе сборки программ, которые будут работать с динамическими библиотеками (DLL). Этот файл служит посредником между программой и динамической библиотекой, предоставляя информацию о функциях и данных, экспортируемых из DLL.

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**

**Таблица импорта** — **это** структура данных, содержащая информацию о функциях, импортированных из других библиотек **DLL**

Что такое DLL-injection?

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

Для чего нужен extern “C”?

Если мы говорим о создании библиотек на языках отличных от С, то скорее всего при создании символов для таблицы экспорта/импорта компилятор преобразует имена из кода в понятные для себя имена. Данный процесс называется **name mangling** (компилятор C такого не делает, он сохраняет имя функций)

.def-файл позволяет оставить имена функций в том виде, который понятен человеку

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

### **Name mangling**

В C++ компилятор изменяет (модифицирует) имена функций, чтобы включить в них информацию о:

* Типах аргументов.
* Пространстве имён.
* Классе, к которому функция принадлежит.

Пример (без extern "C"):

cpp

Copy code

void myFunction(int x) { }

Имя функции может быть преобразовано в нечто вроде \_Z10myFunctioni (зависит от компилятора).

### **Как работает** extern "C"

#### ****1. Для функций****

С помощью extern "C" мы отключаем name mangling:

cpp

Copy code

extern "C" void myFunction(int x);

Теперь имя функции будет таким же, как в C: myFunction.

**extern "C"** — это спецификатор, используемый в языке C++, чтобы сообщить компилятору, что функции или переменные имеют соглашение о вызовах (calling convention) языка C.