**Лабораторная работа №3**

**Библиотеки**

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

**Функционал для библиотек:**

Любая библиотека из данной лабораторной работы должна содержать следующий функционал:

* Итеративную функцию бинарного поиска;
* Рекурсивную функцию бинарного поиска;
* Массив целых чисел размером 1024 элемента.

Реализацию функций можно взять [отсюда](https://rosettacode.org/wiki/Binary_search#C), а массив сгенерировать [тут](https://onlinetools.com/integer/create-integer-array).

**Постановка задачи для Windows:**

**Проект <Ваши Инициалы>Lib:**

Должен содержать код для создания статической библиотеки с заданным функционалом (см. выше). Код должен быть кроссплатформенным, так как будет применяться и для Linux/macOS (Для тех кто не берёт код из указанного выше источника).

**Проект Lab-03a:**

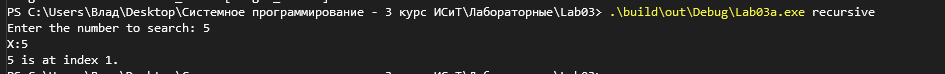
Разработать клиентское приложение которое будет использовать статическую библиотеку. Код также должен быть кроссплатформенным, так как и в этом случае, код будет применяться для Linux/macOS. Приложение должно быть консольным и может быть вызывано в виде:

***Lab-03a <Имя вызываемой функции> [Искомое число]***

******

******

******

******

******

Если искомое число не задано, то оно запрашивается для ввода с консоли. В результате выполнения этого приложения должен быть выведен один из следующих результатов:

* Не указана вызываемая функция!
* <Имя вызванной функции>: Заданное число не найдено!
* <Имя вызванной функции>: Число <Х> найдено на позиции <Y>!

Проверить факт подключения библиотеки через Process Explorer.

***Вопрос:*** *Является ли это возможным? Почему?*

**Проект <Ваши Инициалы>DLib:**

Разработать динамическую библиотеку на языке **C** с заданным функционалом (см. выше). Также должна быть реализована функция **DllMain** c логированием событий в некоторый файл (путь к файлу должен быть абсолютным).

Экспорт функций и переменных организовать через **.def** файл так, чтобы:

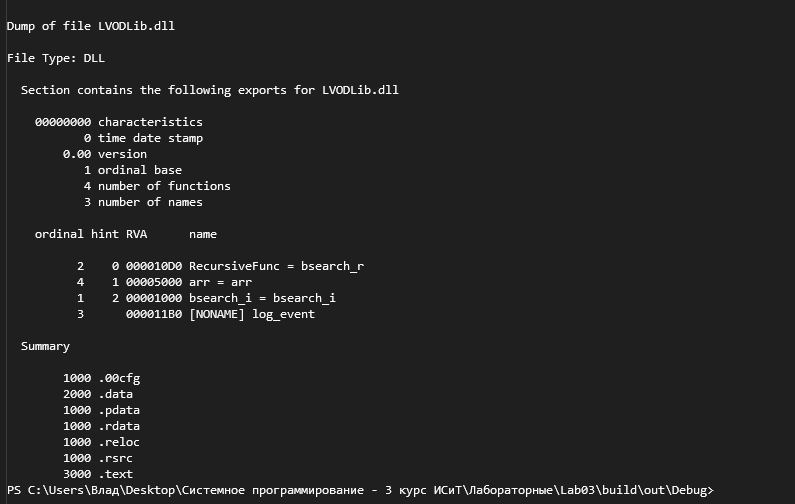
* Одна из функций экспортировалась без имени;
* Одна из функций экспортировалась с именем отличающимся от оригинального.

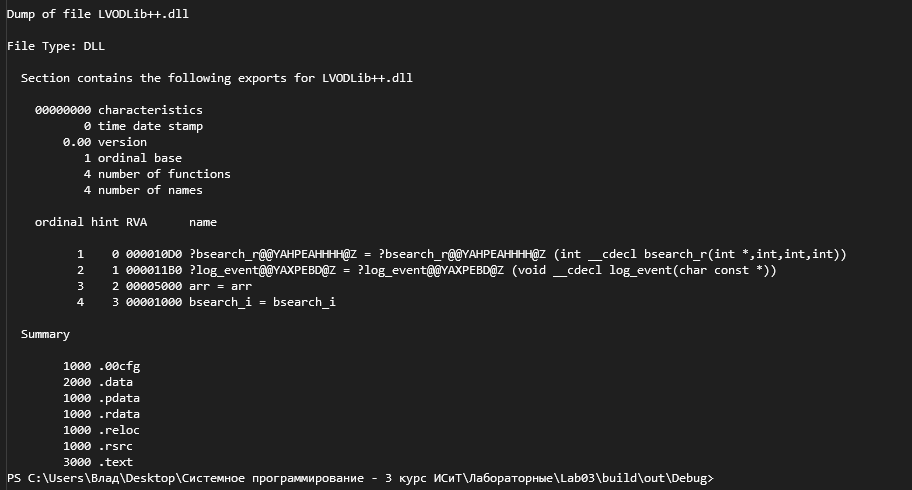
**Проект <Ваши Инициалы>DLib++:**

Разработать динамическую библиотеку на языке **C++** с заданным функционалом (см. выше). Также должна быть реализована функция **DllMain** c логированием событий в некоторый файл (путь к файлу должен быть абсолютным).

Экспорт функций и переменных организовать через **\_\_declspec(dllexport)**. Одной из функций указать модификатор **extern “C”**.

Изучить таблицы экспорта обеих библиотек с помощью утилиты **dumpbin** (данная утилита доступна из Developer версии терминала Visual Studio)и предоставить скриншоты с их содержимым! Имя функции преобразованное компилятором C++ сохраните для дальнейших вызовов этой функции!





**Проект Lab-03b-Explicit:**

Разработать клиентское приложение которое сможет **явно** подключить любую из двух динамических библиотек. Приложение должно быть консольным и может быть вызывано в виде:

***Lab-03b-ex <Подключаемая библиотека> <Имя или номер вызываемой функции> [Искомое число]***

**Информация!** Параметры «Подключаемая библиотека» и «Имя вызываемой функции» должны быть непосредственно значениями передаваемыми в функции **LoadLibrary** и **GetProcAddress**.

Если искомое число не задано, то оно запрашивается для ввода с консоли (для наглядности работы динамической библиотеки поместите этот код непосредственно перед вызовом функции поиска). В результате выполнения этого приложения должен быть выведен один из следующих результатов:

* Не указана загружаемая библиотека!



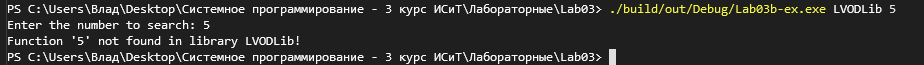
* Не указана вызываемая функция!



* Загружаемая библиотека не найдена!



* Искомая функция <Имя вызываемой функции> не найдена в библиотеке <Имя загруженной библиотеки>!

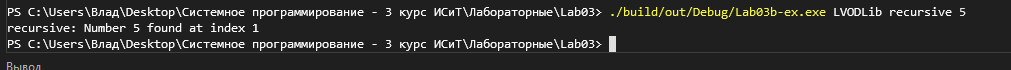


* <Имя вызванной функции>: Заданное число не найдено!



* <Имя вызванной функции>: Число <Х> найдено на позиции <Y>!







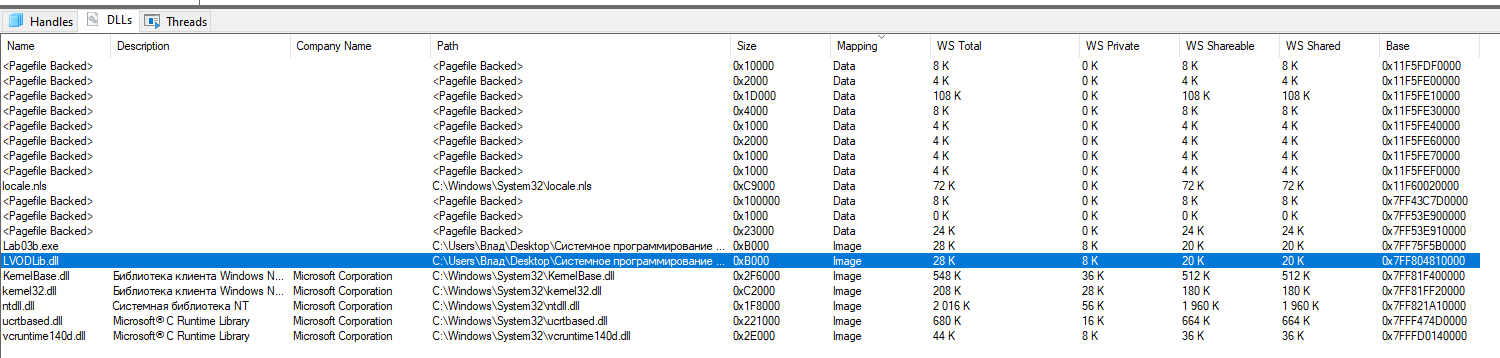


Изучить таблицу импорта приложения с помощью утилиты **dumpbin** и предоставить скриншот с её содержимым!



***Вопрос:*** *Что оказалось в таблице импорта? Почему?*

Проверить факт подключения библиотеки через Process Explorer. (Для этого не указывайте число в аргументах при запуске приложения и дайте приложению войти в режим ожидания ввода с консоли).

****

**Проект Lab-03b-Implicit:**

Разработать клиентское приложение которое неявно подключает одну любую из двух динамических библиотек. Приложение должно быть консольным и может быть вызывано в виде:

***Lab-03b-im <Имя вызываемой функции> [Искомое число]***

Если искомое число не задано, то оно запрашивается для ввода с консоли. В результате выполнения этого приложения должен быть выведен один из следующих результатов:

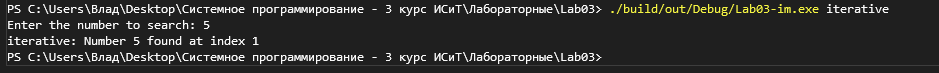
* Не указана вызываемая функция!

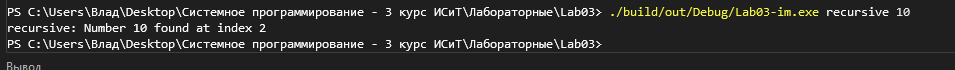


* <Имя вызванной функции>: Заданное число не найдено!

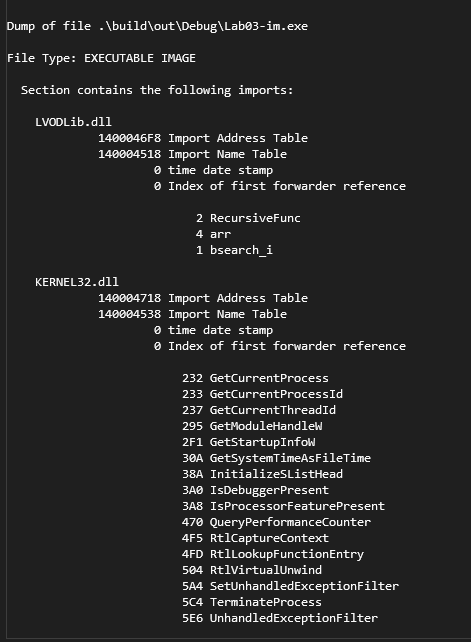


* <Имя вызванной функции>: Число <Х> найдено на позиции <Y>!



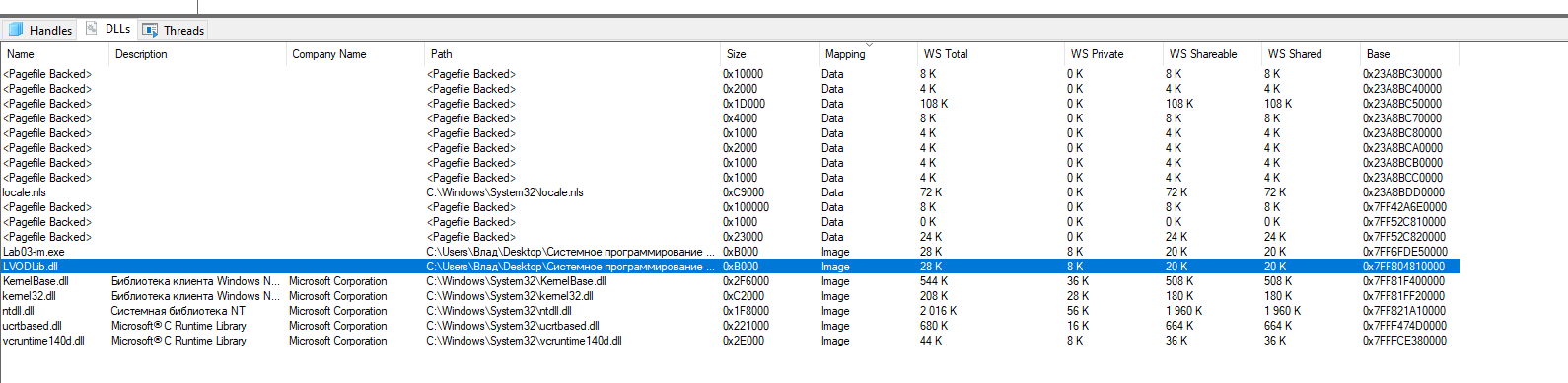


Изучить таблицу импорта приложения с помощью утилиты **dumpbin** и предоставить скриншот с её содержимым!



***Вопрос:*** *Что оказалось в таблице импорта? Почему?*

Проверить факт подключения библиотеки через Process Explorer. (Для этого не указывайте число в аргументах при запуске приложения и дайте приложению войти в режим ожидания ввода с консоли).



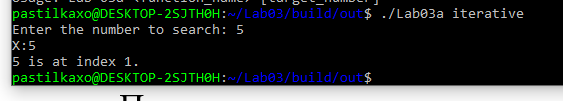
**Постановка задачи для Linux:**

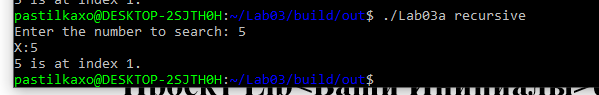
**Проект <Ваши Инициалы>Lib:**

Собрать статическую библиотеку из проекта с аналогичным названием на Windows.

**Проект Lab-03a:**

Собрать приложение из проекта с аналогичным названием на Windows. Проверить работоспособность приложения.

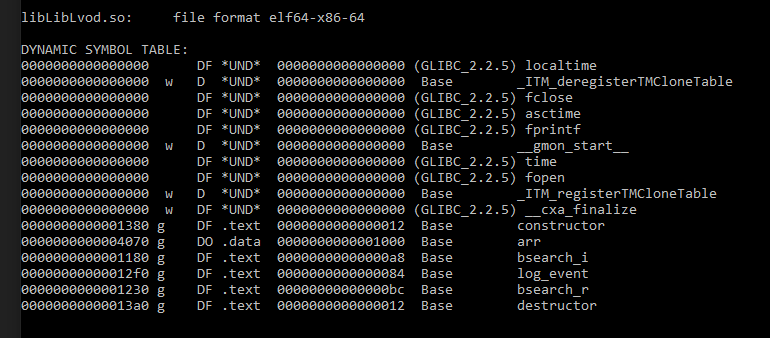




**Проект Lib<Ваши Инициалы>d:**

Разработать разделяемую библиотеку на языке **C** с заданным функционалом (см. выше). Также должна быть реализованы функции **constructor** и **destructor** с логированием событий в некоторый файл (путь к файлу должен быть абсолютным).

Просмотреть список экспортируемых символов через утилиту **objdump**/**readelf** и предоставить скриншот с его содержимым!



**Проект Lab-03c:**

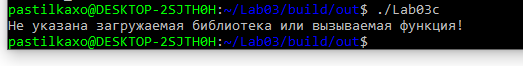
Разработать клиентское приложение которое сможет **явно** подключать разделяемую библиотеку. Приложение должно быть консольным и может быть вызывано в виде:

***Lab-03с <Подключаемая библиотека> <Имя или номер вызываемой функции> [Искомое число]***

**Информация!** Параметры «Подключаемая библиотека» и «Имя вызываемой функции» должны быть непосредственно значениями передаваемыми в функции **dlopen** и **dlsym**.

Если искомое число не задано, то оно запрашивается для ввода с консоли (для наглядности работы динамической библиотеки поместите этот код непосредственно перед вызовом функции поиска). В результате выполнения этого приложения должен быть выведен один из следующих результатов:

* Не указана загружаемая библиотека!



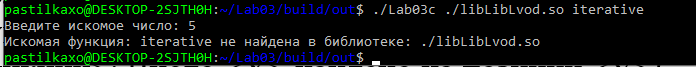
* Не указана вызываемая функция!



* Загружаемая библиотека не найдена!



* Искомая функция <Имя вызываемой функции> не найдена в библиотеке <Имя загруженной библиотеки>!

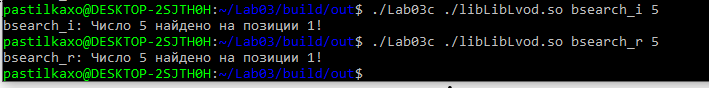


* <Имя вызванной функции>: Заданное число не найдено!

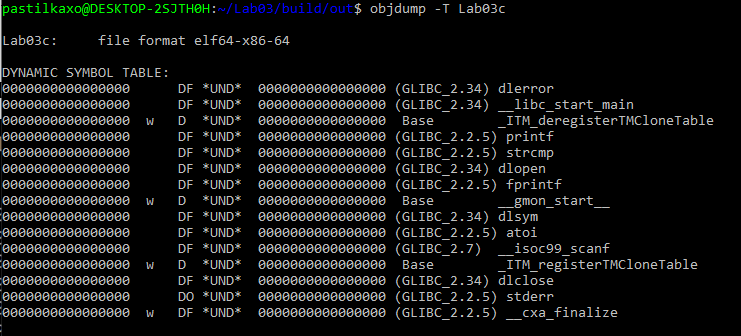


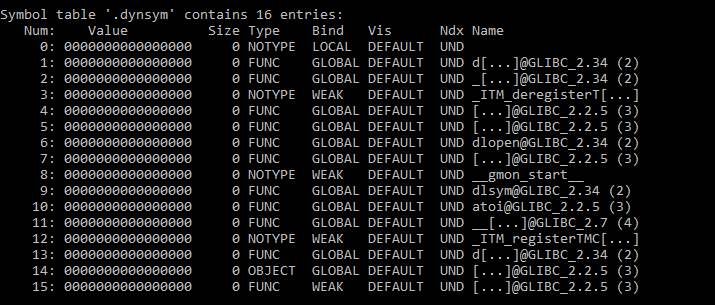


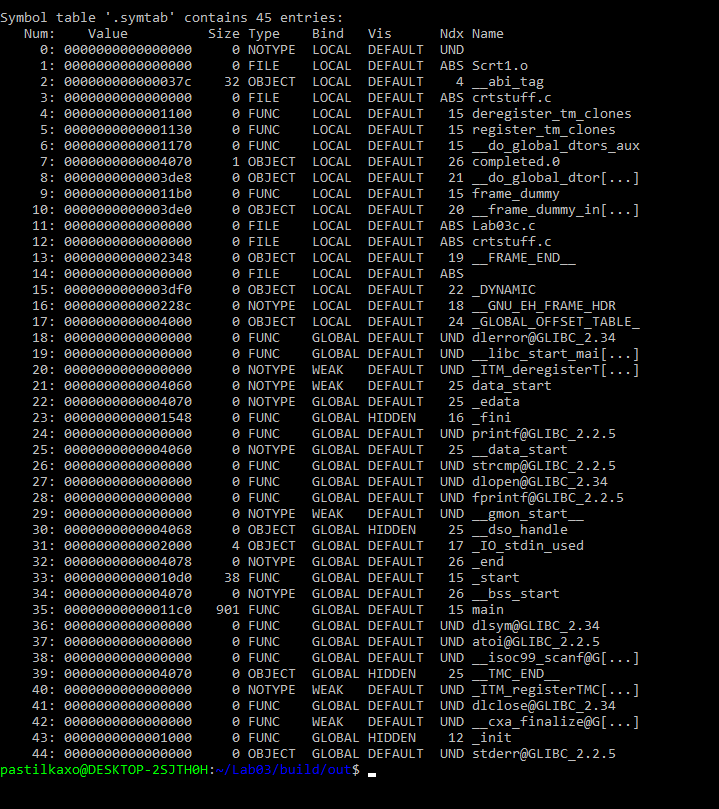
* <Имя вызванной функции>: Число <Х> найдено на позиции <Y>!



Просмотреть список импортируемых символов через утилиту **objdump**/**readelf** и предоставить скриншот с его содержимым!







**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 ^\_^).

**Требования к выполненной работе:**

* Для генерации и сборки проекта использовать как средство CMake, так и clang/clang++ напрямую;
* Для обоих случаев выучить команды (консольные или в CMakeLists.txt) которые используются для сборки библиотек или приложений использующих библиотеки!
* Для обоих случаев подготовить серию скриншотов со сборкой библиотек и приложений!

При использовании CMake должно быть выполнено следующее:

* Сборка должна быть «out-of-source»;
* Для Windows использовать генератор «Visual Studio»/«Ninja» для Linux – «Unix Makefiles», для macOS – «XCode»;
* При сборке проектов всегда должен использоваться clang;
* Все проекты для обеих ОС являются частью одного большого проекта. Добавить в конфигурационные файлы условия сборки отдельных проектов для разных ОС. Конечные файлы генерируются в общий каталог!

**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. Для чего нужен extern “C”?
10. Функции жиненного цикла динамических библиотек в Windows и Linux?

Библиотека объектных файлов – это файл, содержащий несколько объектных файлов, которые будут использоваться вместе на стадии сборки (связывания, линковки) программы. Бывают статические и динамические.

Нормальная библиотека содержит символьный индекс, состоящий из названий функций, переменных и т. д., которые содержатся в библиотеке. Это позволяет ускорить процесс сборки программы.

Связывание - компоновка программы из многих obj модулей.

Когда программа компилируется, каждый её obj модуль создаётся отдельно, и на этом этапе компилятор не может знать, где находятся данные или функции из других модулей. Он лишь ставит заглушки — адреса или ссылки, которые будут заменены в процессе связывания.

Виды связывания - раннее(во время трансляции или во время сборки) и позднее(при загрузке или отложенное[итеративное, декларативное]).

При статической библиотеке - раннее.

На этапе компоновки или трансляции obj-код внедряется в исполняемый файл.

У нас использовалось на этапе трансляции. Есть еще во время сборки - используется pragma.

Для ДЛЛ - неявное связывание при загрузке или явное в период выполнения.

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.

Статическая библиотека - обычный файл, который содержит копии всех помещенных в него obj файлов. Там короче еще атрибуты есть: права доступа, id пользователя и группы, время ласт изменения. Итоговый исполняемый файл содержит копии всех obj модулей.

+ набор часто используемых файлов можно впихнуть в единую библиотеку, при этом не нужно будет перекомпилировать ориг код.

+ не перечисляем список obj файлов, а указываем только имя библиотеки.

может быть несколько копий одних и тех же obj модулей, поэтому минусы:

- дисковое пространство хранит одни и те же копии

- виртуальная память потребляется дохуя, потому что отдельная копия каждого obj файла

- если вносим изменение, то перекомпилируем каждый исполняемый файл, где используется модуль

Динамическая(разделяемая) библиотека - одна единая копия модуля разделяется между программами.

Запускается первая программа, модуль загружается в память, а дальше уже все обращаются к этой копии.

DLL(Dynamic Link Library) - win - файл в формате pe(portable executable)

Shared Objects(разделяемые объекты) - linux - файл в формате elf(executable linkable format)

+можно загружать в адресное пространство динамически, что позволяет приложению подгружать нужный код.

+ разные ЯП

+ проще управлять проектом

+ экономия памяти

+ разделение ресурсов

Последовательность поиска DLL:

-каталог с exe

-текущий каталог процесса

-системный виндовс

-основной виндовс

-каталоги path

в конечный DLL:

Библиотека импорта - LIB-файл, нужен для любого exe-модуля, содержит идентификаторы.

Раздел экспорта - таблица экспортируемых идентификаторов.

Относительный виртуальный адрес каждого идентификатора.

def-файл

+ настраивать экспорт по порядковому номеру

+ указать конечное имя для функции

+ область видимости экспортируемых функций

+ можно и без имен, чисто по порядковым номерам