Статические и динамические библиотеки. Преимущества и недостатки. Способы создания. Примеры.

Библиотека - это набор готовых функций и процедур, которые можно использовать в своих программах программист.
Две основные категории:

- Статические библиотеки (.a – Linux, .lib - Windows)
- Динамические библиотеки (.so – Linux, .dll – Windows, .dylib – MacOS)

Особенности:
Экономия времени разработки:

- Использование готового кода вместо написания с нуля
  Упрощение поддержки:
- Изменения требуются только в библиотеке, а не во всех местах использования
  Оптимизация ресурсов:
- Статические библиотеки уменьшают объем дублируемого кода
- Динамические библиотеки экономят оперативную память
  Модульность:
- Программа становится более структурированной
- Упрощается тестирование и отладка

Плюсы и минусы каждой библиотеки:

Аспект	Статическая библиотека	Динамическая библиотека
Размер исполняемого файла	Больше (все функции включены)	Меньше (функции загружаются по необходимости)
Производительность	Выше (нет дополнительной загрузки)	Немного ниже (время загрузки)
Изменения	Требует компиляции программы	Можно обновить без компиляции
Память	Каждая программа имеет свою копию	Одна копия используется всеми программами

Статическая библиотека:

- Когда важна производительность.

- Когда нужно создать автономное приложение.
- Когда обновления библиотеки не планируются.

Динамическая библиотека:

- Когда важно экономить место.
- Когда нужно обновлять библиотеку без компиляции программы.
- Когда несколько программ используют одну и ту же библиотеку.

Статическая библиотека — это архив объектных файлов (.o), объединённых в один файл.

- Предкомпилированные функции и данные.
- Символы (имена функций и переменных), доступные для линковки.

Создание статической библиотеки:

1. Разработка функций и их реализация в **.c** файлах.
2. Перевод исходного кода в машинный код.
3. Создание архива с помощью утилиты **ar**.

```
gcc -c math.c -o math.o # Компиляция в объектный файл
ar rcs libmath.a math.o # Создание архива
```

Линковка приложения со статической библиотекой:

```
gcc main.c -L. -lmath -o program
    -L.: Указывает путь к библиотеке.
    -lmath: Ищет libmath.a или libmath.so.
```

Пример:

```
// math.c
int add(int a, int b) { return a + b; }

// main.c
#include <stdio.h>
extern int add(int, int);  // Объявление функции из библиотеки
int main() { printf("%d\n", add(2, 3)); }
```

Динамическая библиотека (.so, .dll) загружается в память при запуске приложения.

- Позиционно-независимый код (PIC/PIE).
- Символы (функции и переменные) доступны для всех программ, использующих её.
- Управление версиями через символические ссылки (например, libmath.so.1.0).
  Создание динамической библиотеки:

- Написание исходного кода
- Компиляция с флагом PIC

```
gcc -fPIC -c math.c -o math.o
```

- Создание .so файла (в Linux)

```
gcc -shared -o libmath.so math.o
```

Исходный код (math.c) —> Объектный файл (math.o с PIC) —> Динамическая библиотека (libmath.so)
Линковка приложения с динамической библиотекой:

```
gcc main.c -L. -lmath -o program
```

Загрузка библиотеки во время выполнения :

- Операционная система (через ld-linux.so или loader) ищет libmath.so.
- Пути поиска задаются через LD_LIBRARY_PATH.