Static & Dynamic Libraries — Simple Step-by-Step Notes

1️⃣What is a library?

A library is a collection of ready-made functions that we can reuse in programs.

There are two types:

- **Static Library (.a)** — added inside the program while compiling
- **Dynamic / Shared Library (.so)** — loaded when the program runs

---

2️⃣Static Library (Very Simple Understanding)

👉The library code is **copied into the final program**. 👉The program works even if the library file is removed.

☸️Steps to create a static library

🤩Compile source files to object files

```
gcc -c file1.c file2.c
```

🤪Create library

```
ar rcs libmylib.a file1.o file2.o
```

🫢Use the library while compiling program

```
gcc main.c -L. -lmylib -o app
```

♀️When static library is useful

- When we want a **single portable executable**
- Embedded systems / devices
- No dependency problems

#️ Limitations

- Program size becomes bigger
- Need to re-compile if library changes

3️⃣Dynamic (Shared) Library — Simple Understanding

👉The library code is **NOT copied into the program**. 👉Program loads the library from the system when it runs.

☸️Steps to create a shared library

🤩Compile with position-independent option

```
gcc -fPIC -c file1.c file2.c
```

🤪Create shared library

```
gcc -shared -o libmylib.so file1.o file2.o
```

🫢Link the program with library

```
gcc main.c -L. -lmylib -o app
```

✅Two Ways to Make Program Find the Shared Library

🟢Way 1 — Using `LD_LIBRARY_PATH` (Temporary)

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:.
```

👉Works only for the current terminal session.

🟢Way 2 — Copy library to `/usr/lib` (Permanent & System-wide)

🤩Copy library

```
sudo cp libmylib.so /usr/lib/
```

🤪Update linker cache

```
sudo ldconfig
```

👉Now any program can use the library without setting paths.

♀️ When shared library is useful

- Many programs use the **same library**
- Saves memory
- Can update library **without recompiling program**

Limitations

- Program will not run if `.so` is missing
- Slight delay when loading

4️⃣Quick Difference (Very Easy Table)

Topic	Static (.a)	Dynamic (.so)
Linked	During compile	During run time
Code inside program	Yes	No
Program size	Bigger	Smaller
Update library	Re-compile needed	Just replace .so

5️⃣Useful Commands

👉🏿Check which shared libraries a program uses

```
ldd app
```

👉🏿See symbols in a library

```
nm libmylib.a
nm libmylib.so
```

6️⃣One-line Remember Points (Interview Friendly)

- Static library → copied inside program
- Dynamic library → loaded at run time
- `ar rcs` → creates static library
- `-fPIC` → needed for shared library code
- `ldd` → shows shared dependencies

7️⃣Example — Simple Program Using add / sub / mul Functions

👉You have three files placed separately:

☣️File 1 — add.c

```
#include<stdio.h>
int add(int a, int b) {
    return a + b;
}
```

☣️File 2 — sub.c

```
#include<stdio.h>
int sub(int a, int b) {
    return a - b;
}
```

☣️File 3 — mul.c

```
#include<stdio.h>
int mul(int a, int b) {
    return a * b;
}
```

☣️Header File — math.h (Function Declarations)

```
#ifndef MATH_H
#define MATH_H

int add(int a, int b);
int sub(int a, int b);
int mul(int a, int b);

#endif
```

🟡main.c (User inputs values & selects operation)

```
#include <stdio.h>
#include "math.h"

int main() {
    int a, b, choice;

    printf("Enter value of a: ");
    scanf("%d", &a);

    printf("Enter value of b: ");
    scanf("%d", &b);

    printf("
Choose operation
");
    printf("0 --> Add
");
    printf("1 --> Subtract
");
    printf("2 --> Multiply
");
    printf("Enter choice: ");
    scanf("%d", &choice);

    switch (choice) {
        case 0:
            printf("Result = %d
", add(a, b));
            break;
        case 1:
            printf("Result = %d
", sub(a, b));
            break;
        case 2:
            printf("Result = %d
", mul(a, b));
            break;
        default:
            printf("Invalid option
");
    }

    return 0;
}
```

🟢Compile object files

```
gcc -c add.c sub.c mul.c
```

🟢Create static library

```
ar rcs libcalc.a add.o sub.o mul.o
```

🟢Compile with static library

```
gcc main.c -L. -lcalc -o app_static
```

🟣Create shared library

```
gcc -fPIC -c add.c sub.c mul.c

gcc -shared -o libcalc.so add.o sub.o mul.o
```

🟣Run program (Way-1: temporary path)

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:.

gcc main.c -L. -lcalc -o app_shared
./app_shared
```

🟣Run program (Way-2: copy to /usr/lib)

```
sudo cp libcalc.so /usr/lib/
sudo ldconfig

gcc main.c -lcalc -o app_shared
./app_shared
```

👉This example clearly shows how the same functions work with both **static** and **dynamic** libraries.