Different between Dynamic and static linking

At first what is **linking**?

-Linking is the process of combining object files Including all Libraries and their dependencies Into a final executable file (.exe).

1. Static Linker

a) How it works:

- It copies all required library code into your executable at **compile time.**

b) Pros:

- Faster startup (everything is already in memory).

- No dependency issues at runtime.

c) Cons:

- Larger executable file.

-If the library is updated, program should be recompiled to fetch latest update

2. Dynamic Linker

d) How it works:

- The linker adds references to shared libraries and the actual linking done at **run time.**

e)Pros:

- Smaller executable file.
- Easier to update (just replace the shared library).

f) Cons:

- Larger executable file.
- Slightly slower startup

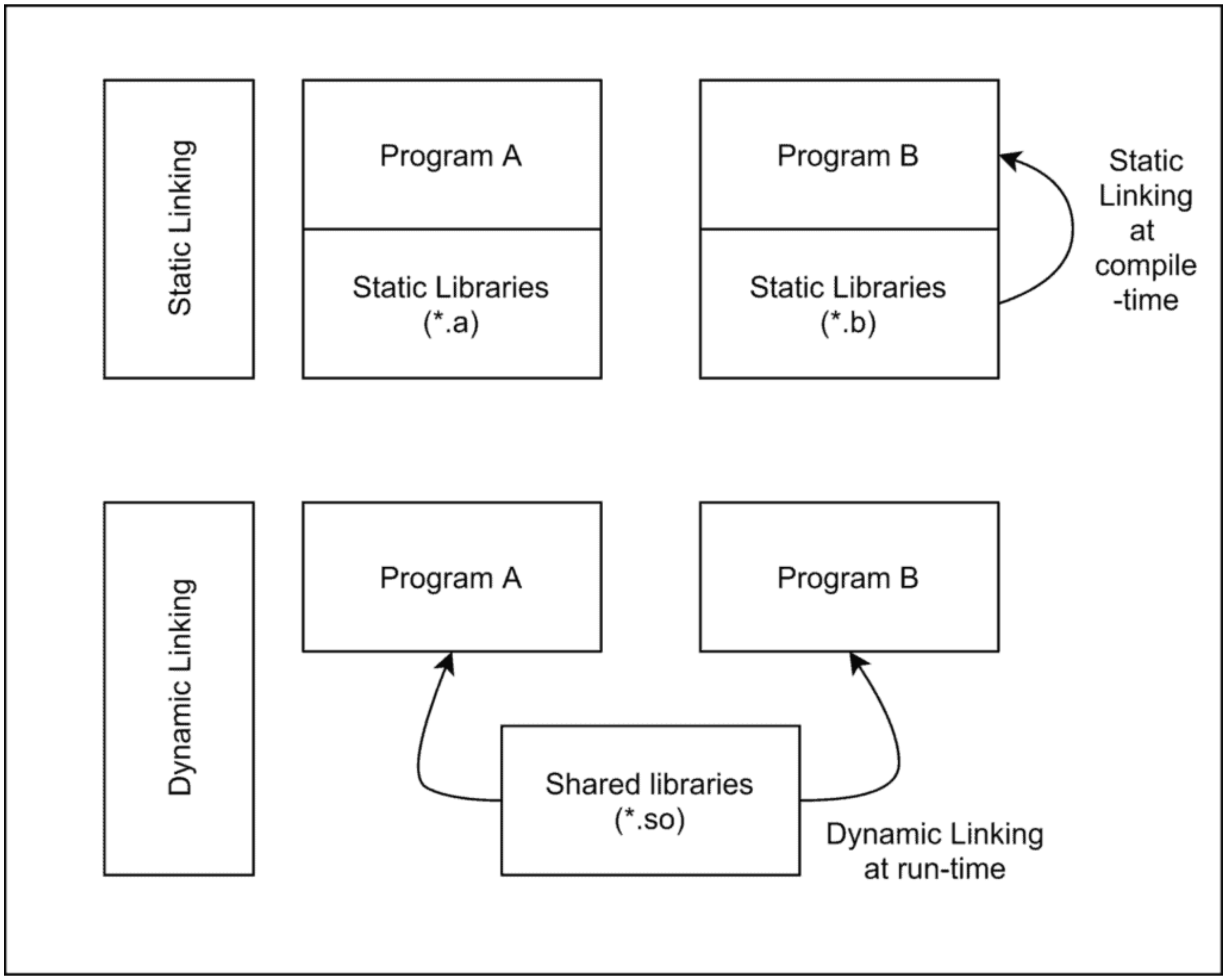
If we use PGO(profile guided optimization)

1.Static Linking

- Allows the compiler to optimize both code and library code together, improving performance on “hot paths”.

2.Dynamic Linking

- Can only optimize your own code not library as it loaded separately at run time.


Static Linking
Program A
Static Libraries (*.a)
Program B
Static Libraries (*.b)
Static Linking at compile-time
Dynamic Linking
Program A
Program B
Shared libraries (*.so)
Dynamic Linking at run-time