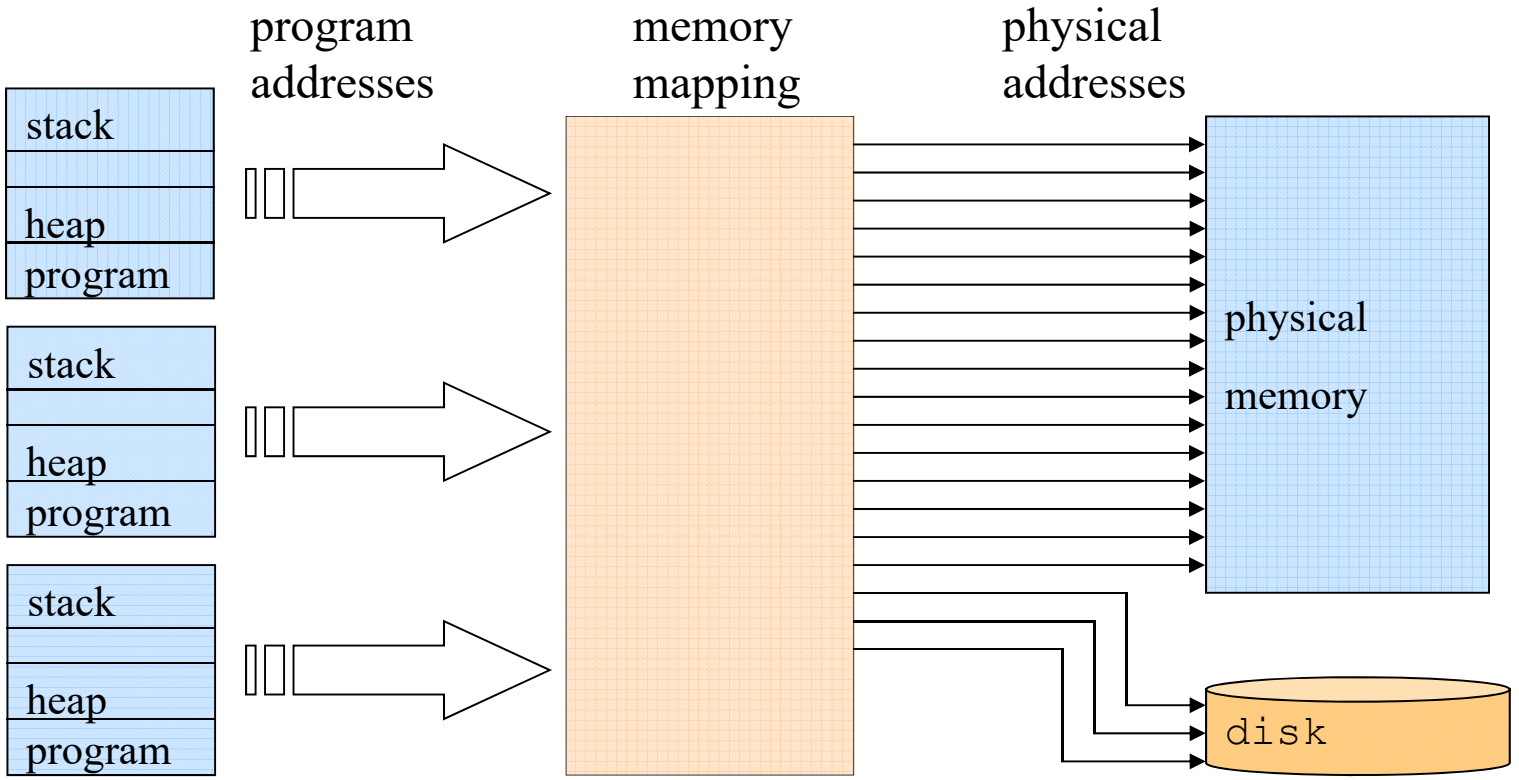
Memory management unit (MMU)

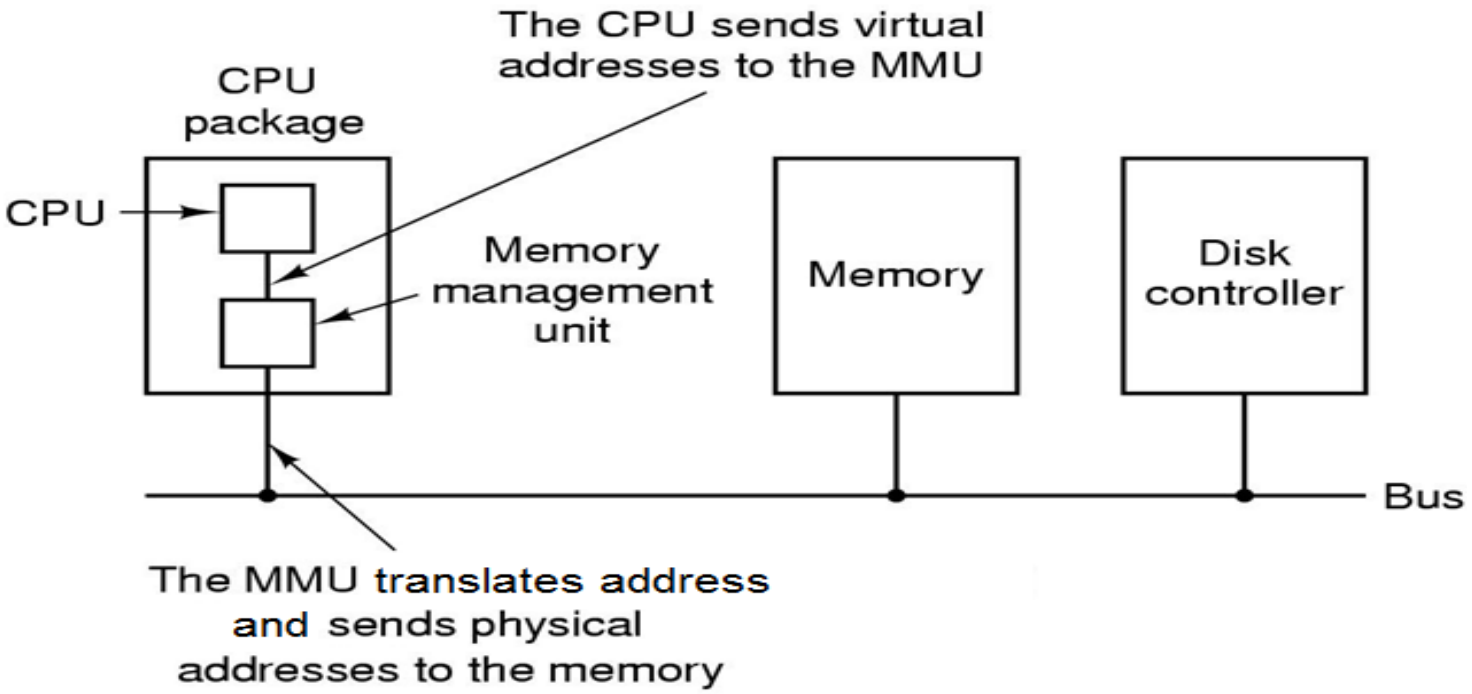
- The input to MMU (its memory mapping hardware) is the program's logical address and output is the corresponding physical address.



MMU: is a hardware device that maps **virtual** to **physical** address

MMU cont…

- The memory protection and memory access is provided by several different ways while the common way is using MMU hardware.

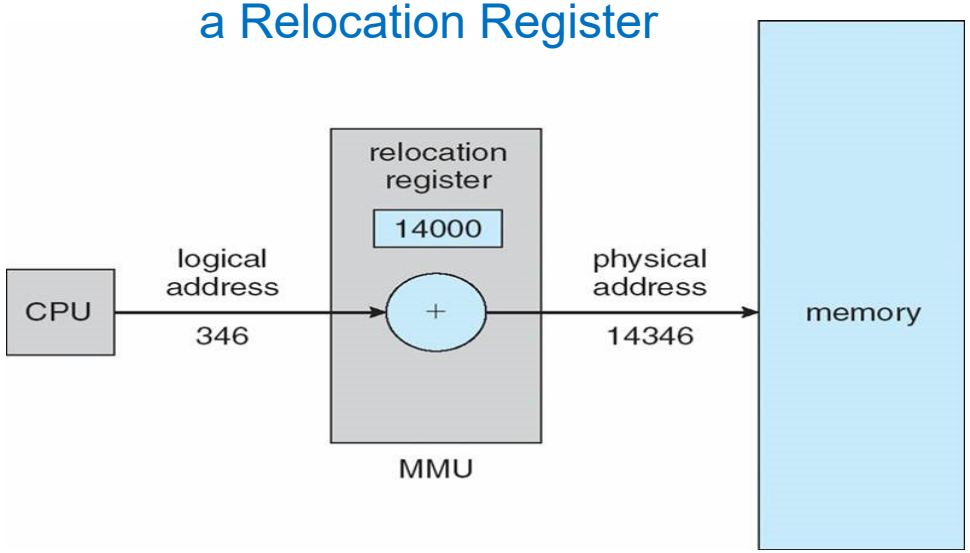


The position and function of the MMU

Address binding at execution time cont…

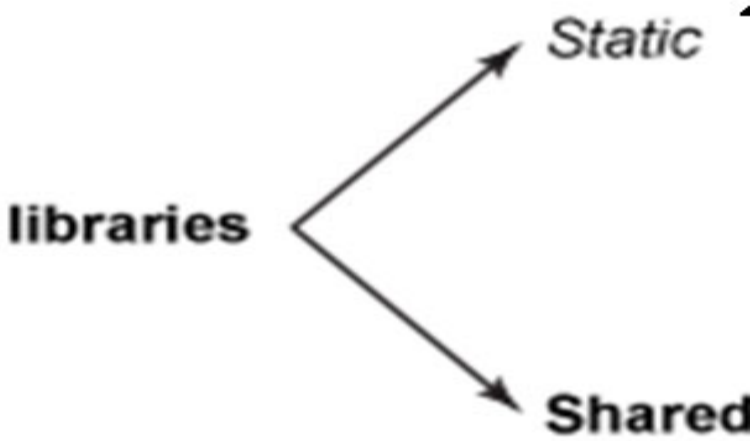
- The value in the relocation register is added to every logical address:
  - CPU will generate logical address for eg: 346
  - MMU will generate relocation register for eg:14000
  - In Memory physical address is located eg:(346+14000= 14346)

Dynamic Relocation Using a Relocation Register



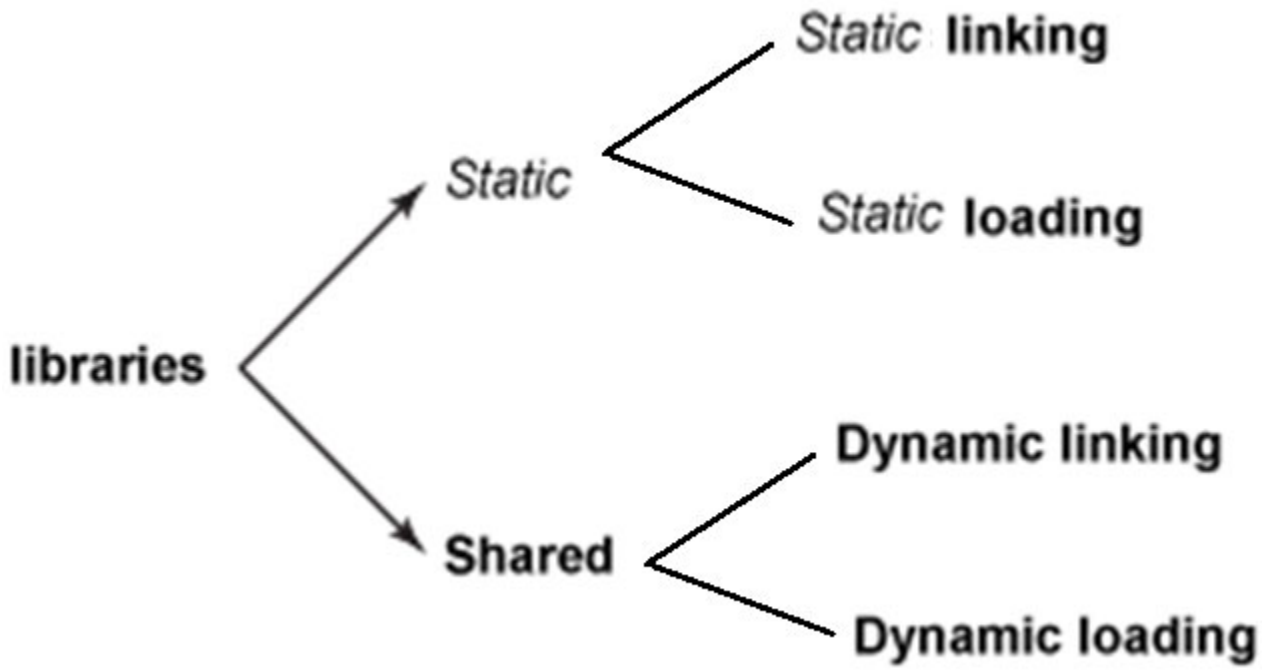
Types of Libraries

- Library functions are those which are common to many applications.
- There are two types of libraries:



Types of Libraries cont…

- Based on two types of Libraries, there are two ways of using these libraries when building an application:

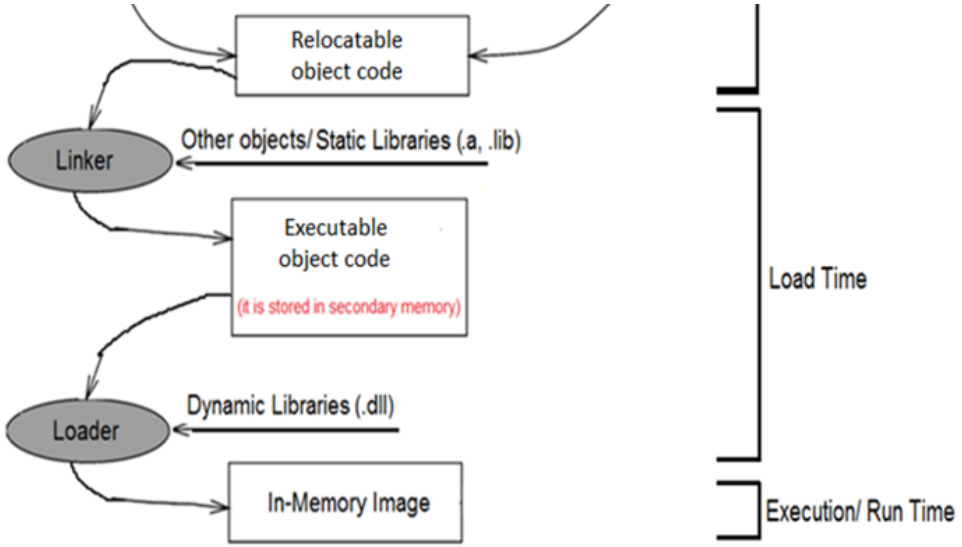


Static Linking (of Libraries)

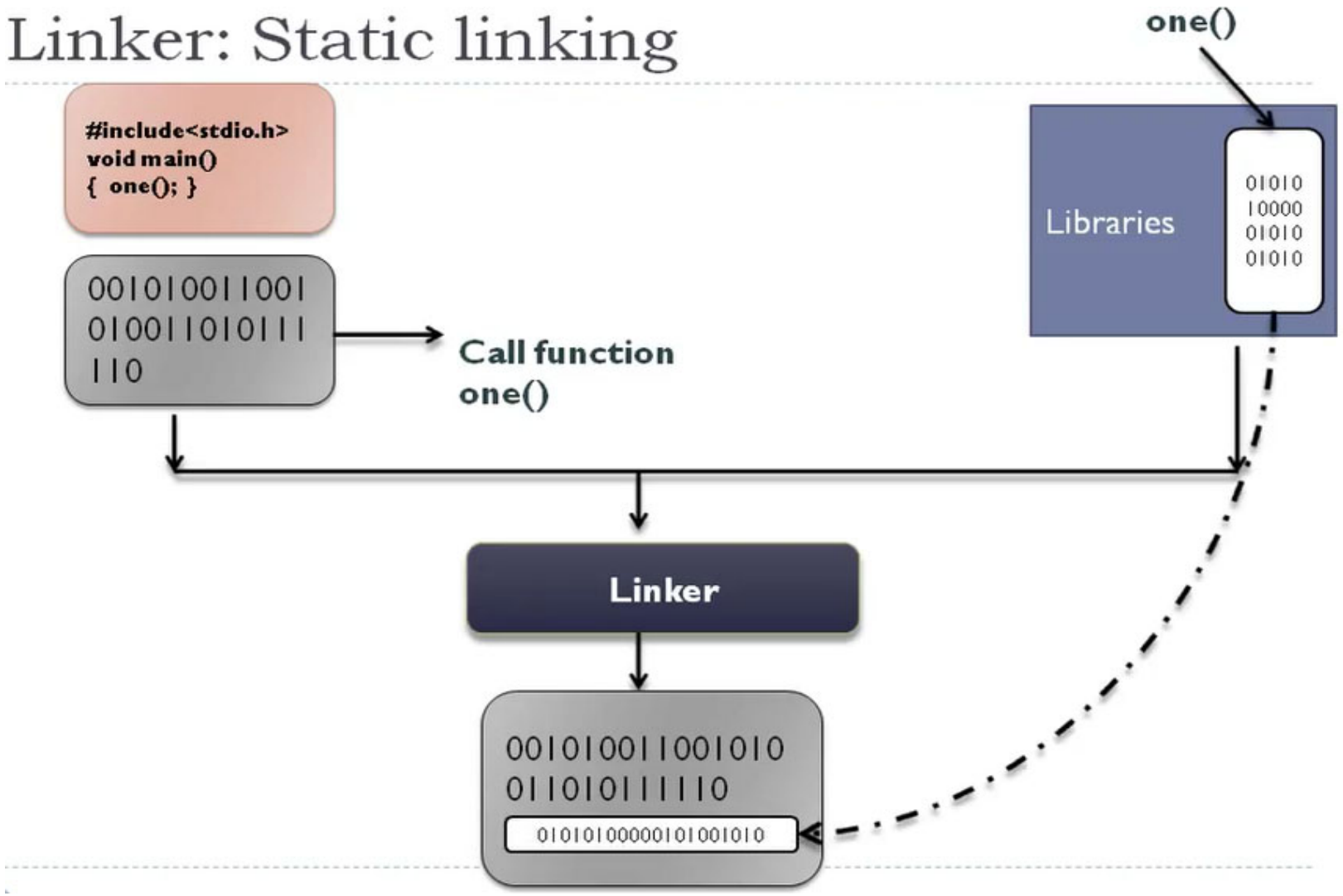
- At **link time**, a copy of the entire library is copied into the final application and there is no need to have the library into the system where the executable file will be run.

- The two common dynamic libraries are:
  - Archive (.a) is a static library for Linux
  - library (.lib) is a static library for Windows

- It is called an archive since it is just a package of compiled object files. These libraries are in directories such as /lib, /usr/lib or /usr/local/lib.



Static Linking cont…



Static linking cont…

- Advantage:
  - they allow the entire application to be self-contained.

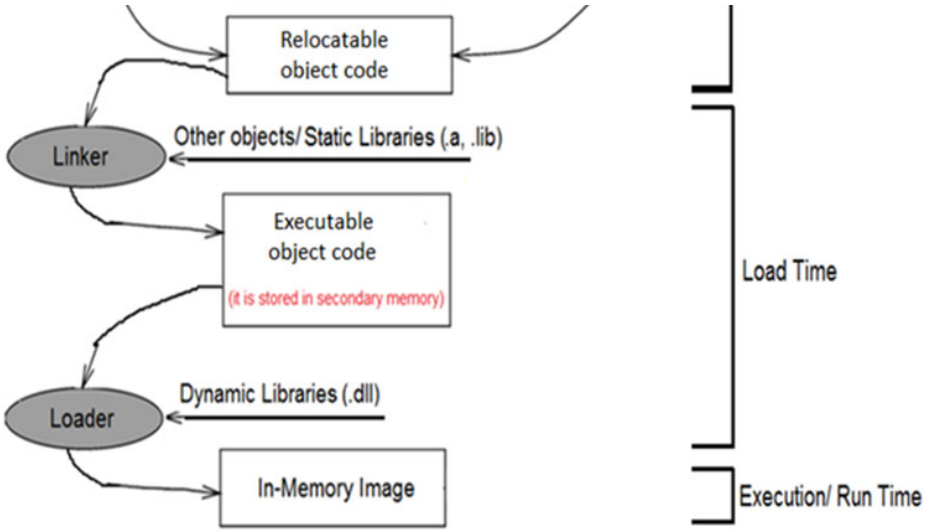
- Disadvantage:
  - they contain portions of each library to which it is linked. For large, popular libraries, such as the C library, the amount of duplicate content can be very high. This means that the executables require more disk storage, memory space, and network bandwidth than if the duplicate content were eliminated.
  - Every time a library changes, the program need to re-compile and re-link again else the changes won't reflect in existing executable file.

Dynamic Linking (of Libraries)

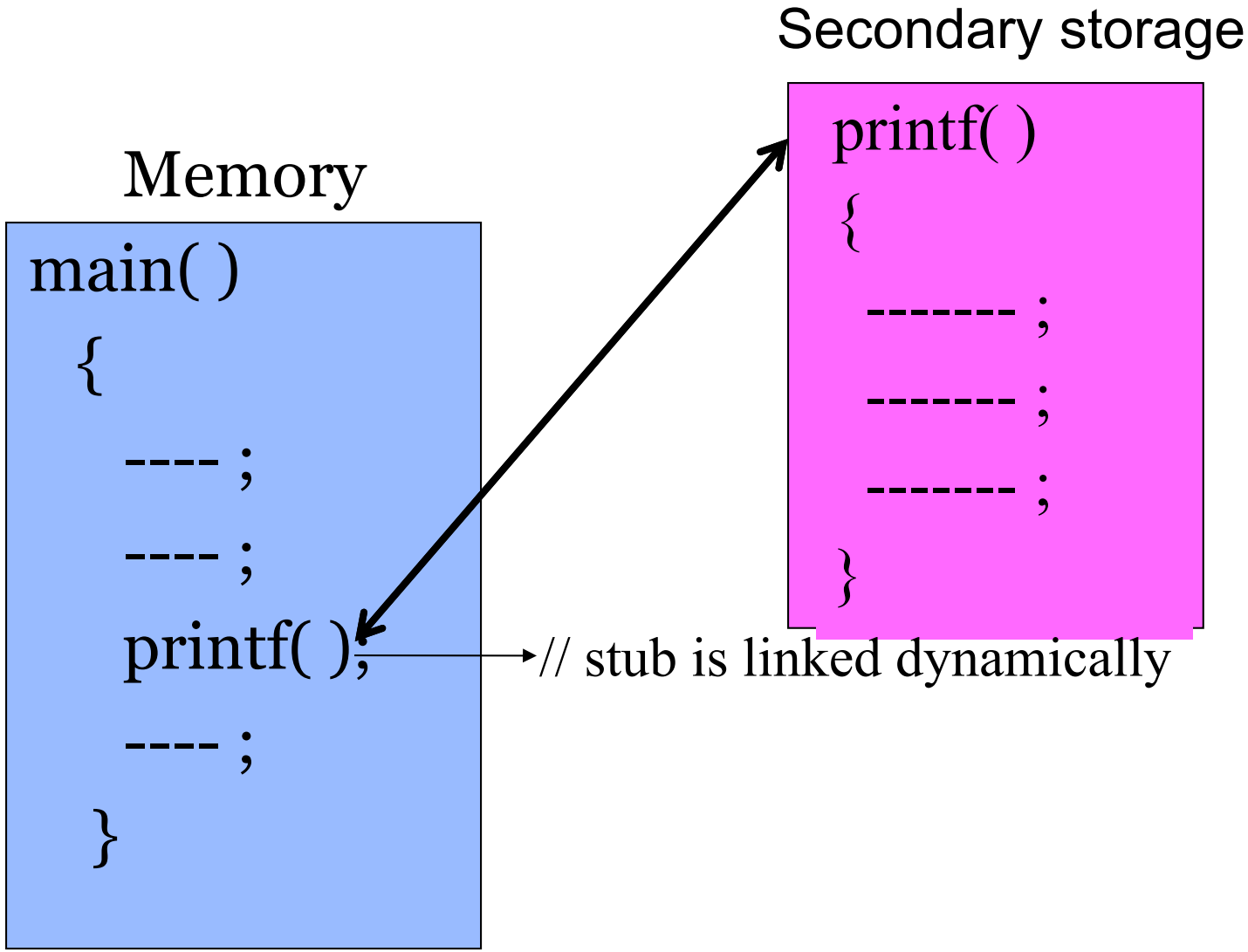
- Dynamic/Shared library: is a library that is linked **at run time** to the executable file and does not embedded in the final executable. The library isn't actually used until run time, where it is needed.

- At link-time all the linker does is store the names of the necessary libraries in the executable. A **stub** is included in the code for each library routine reference. A stub is a **small piece of code** that indicates **how to locate and load the library** routine. At execution-time, when a routine is called, stub replaces itself with the address of the routine, and executes the routine.

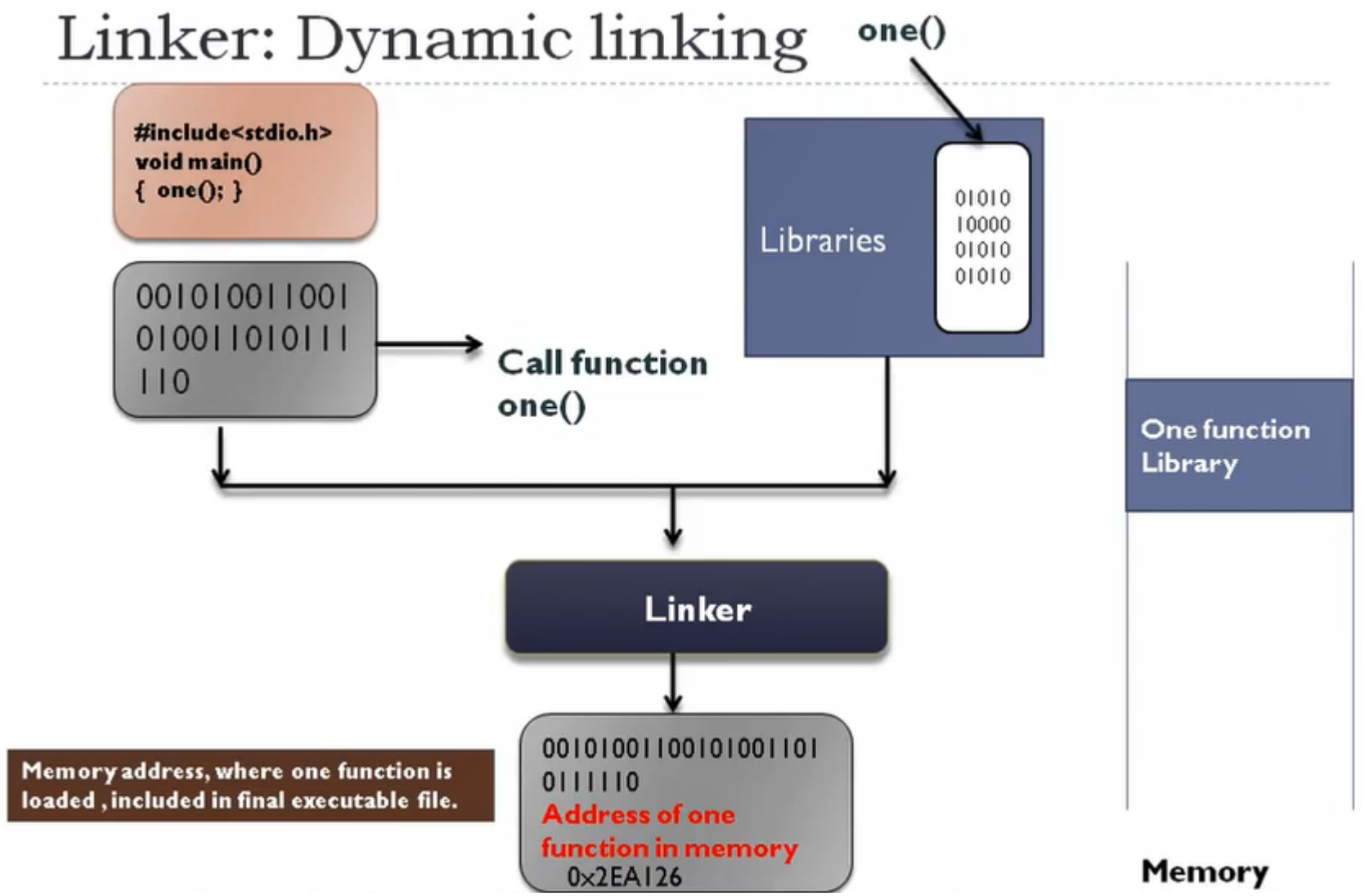
- The two common dynamic libraries are:
  - Shared object (.so) is a dynamic library for Linux
  - dynamic link library (.dll) is a dynamic library for Windows



Stub


Secondary storage
printf( )
{
------ ;
------ ;
------ ;
}
Memory
main( )
{
---- ;
---- ;
printf( );
---- ;
}
// stub is linked dynamically

Dynamic Linking cont…



Dynamic linking cont…

- Advantage:
  - Since the linker only includes the address of the library (not the entire code of it like what is did in static linking), the growing size issue is solved.
  - There is no need to recompile the entire program (like what is done in static linking) whenever the library changes. The library goes to a new address which will be included by linker.
  - only one copy of shared library is kept in memory. This significantly reduces the size of executable programs, thereby saving memory and disk space.

- Disadvantage:
  - application is not self-contained
  - bugs in dynamically linked library infect all applications that share that library.

Static Loading

- Static Loading: the entire program and its corresponding routines is loaded into memory during load time. In this case, the size of a process is limited to the size of physical memory.

Dynamic Loading

- To obtain better memory-space utilization, we use dynamic loading. With dynamic loading a routine is not loaded into memory until it is called. Thereby, loading is postponed till runtime and unused routines are never loaded (better memory-space utilization).

- All routines are kept on disk in relocatable format. The main program is loaded into memory and is executed. When a routine needs to call another routine, the calling routine first checks to see whether the other routine has been loaded.
- If the routine is not in memory, loader loads the desired routine into memory and update the program's address tables to reflect this change. Then control is passed to the newly loaded routine.

Memory management techniques

- Various memory management techniques exist such as partitioning memory, dynamic memory allocation, overlays, swapping. But the two most important and widely used strategies are paging and segmentation.