Compare different types of loader scheme

3-3 3-20

Explain Static and Dynamic Linking

**Static Linking:**  
When we click the .exe (executable) file of the program and it starts running, all the necessary contents of the binary file have been loaded into the process’s virtual address space. However, most programs also need to run functions from the system libraries, and these library functions also need to be loaded.

In the simplest case, the necessary library functions are embedded directly in the program’s executable binary file. Such a program is statically linked to its libraries, and statically linked executable codes can commence running as soon as they are loaded.

**Dynamic Linking:**  
Every dynamically linked program contains a small, statically linked function that is called when the program starts. This static function only maps the link library into memory and runs the code that the function contains. The link library determines what are all the dynamic libraries which the program requires along with the names of the variables and functions needed from those libraries by reading the information contained in sections of the library.

After which it maps the libraries into the middle of virtual memory and resolves the references to the symbols contained in those libraries. We don’t know where in the memory these shared libraries are actually mapped: They are compiled into position-independent code (PIC), that can run at any address in memory.