**Loading and Executing Programs**

Table of Contents

[Loading 2](#_Toc65538811)

[Executing 3](#_Toc65538812)

## Loading

In computer systems a loader is the part of an operating system that is responsible for loading programs and libraries. It is one of the essential stages in the process of starting a program, as it places programs into memory and prepares them for execution. Loading a program involves reading the contents of the executable file containing the program instructions into memory, and then carrying out other required preparatory tasks to prepare the executable for running. Once loading is complete, the operating system starts the program by passing control to the loaded program code.

* validation (permissions, memory requirements etc.);
* copying the program image from the disk into main memory;
* copying the command-line arguments on the stack;
* initializing registers (e.g., the stack pointer);
* jumping to the program entry point (\_start).
* initialization of structures in the DLL itself (i.e. critical sections, module lists);
* creation of a heap (via the function RtlCreateHeap);
* loading of KERNEL32.DLL to obtain several important functions, for instance BaseThreadInitThunk;
* in debug mode, raising of system breakpoint;

## Executing

Execution in computer and software engineering is the process by which a computer or virtual machine executes the instructions of a computer program. Each instruction of a program is a description of a specific action to be carried out in order for a specific problem to be solved; as instructions of a program and therefore the actions they describe are being carried out by an executing machine, specific effects are produced in accordance to the semantics of the instructions being executed.

Programs for a computer may be executed in a batch process without human interaction or a user may type commands in an interactive session of an interpreter. In this case, the "commands" are simply program instructions, whose execution is chained together.

Prior to execution, a program must first be written. This is generally done in source code, which is then compiled at compile time (and statically linked at link time) to an executable. This executable is then invoked, most often by an operating system, which loads the program into memory (load time), possibly performs dynamic linking, and then begins execution by moving control to the entry point of the program; all these steps depend on the Application Binary Interface of the operating system. At this point execution begins and the program enters run time. The program then runs until it ends, either normal termination or a crash.