How Software Runs

3 main components we need to know of: Hard Drive, RAM, CPU

The .exe file is stored on the hard drive(saved as a static file)

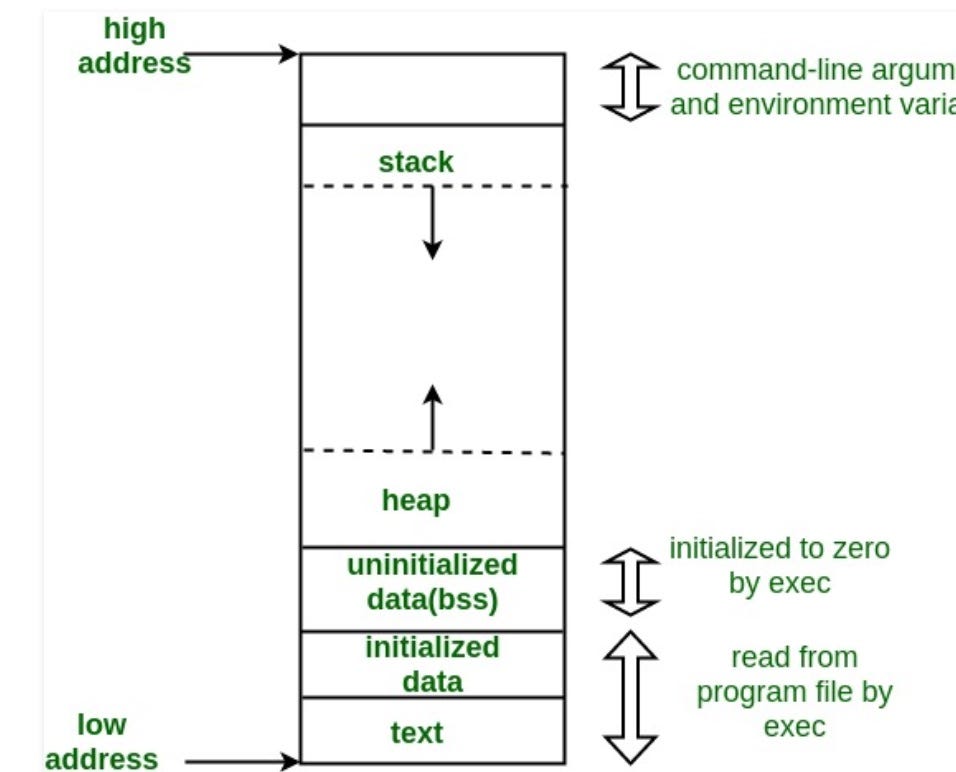
-C/C++ and Java code use OS system calls from the kernel level

When opened it is loaded into RAM

When you click a button, type something, any altercation or event. Causes the CPU to Fetch the instructions.

* During the fetch-decode-execute cycle, the CPU retrieves the address stored in the PC/IP(Program Counter/Instruction Pointer).
* It then fetches the instruction from that memory address.
* Typically, the PC/IP is incremented to point to the next instruction in sequence, though its value can be modified by control flow instructions like jumps, calls, or returns.

That instruction is in memory, Text Segment at the bottom(text).



That instruction is going to alter memory in the stack or heap (where all our values are held).

**Hard drive and C/C++ Deep Dive**

File Pointers are indexes to the file they fetch. They DO NOT point to the hard drive. The operating system has that mapped(every file).

3. When you call CreateFile() to create or open a file, Windows looks up its internal representation by calling into its object manager component (which then calls the filesystem driver that finds the file on disk).

If everything is successful, you get back a reference to the file object. The reference type is called HANDLE. It's (usually) only valid in your process and doesn't point to anything even though it's defined as a void\*. It's just an arbitrary value that you pass into other \*File API functions where it's mapped to the file object it's pointing to and then the actual operation is done by the FS driver again, as long as the permissions on the handle allow it.

Why do I need SysCalls?

1. Incase of any error, the syscall handles it at a kernel level. Otherwise your program would crash your entire system.
2. Providing any address of the hard to read/write would be a major vulnerability, because it would lead to a regular user reading/writing root/system files. The process of opening files also heavily relies on the type of hard disk and the controller. *For example in a virtual machine that you are dealing with a PCI IDE with ATA drives, you can read by providing a CHS or LBA address(which is represented by a hex number). Explained*[*here*](https://wiki.osdev.org/PCI_IDE_Controller#Read.2FWrite_From_ATA_Drive)*.*

This may seem a bit confusing, so I will wrap up with Your OS has a file system, the hard drive has a schema. IEEE standards make them compatible with one another. YOUR ENTIRE OS IS LOADED INTO RAM AT START UP. When a syscall is made, those instructions already exist inside of memory(RAM).

The BLUE SCREEN OF DEATH was caused by other programs needing so much memory it would overwrite that of the operating system. So knowing that your entire OS is loaded into memory helps a lot to understand.