Program Memory Spaces

Joshua Reynolds
NMSU Reverse Engineering
SP 2024

What happens when you “run” a program?

1. A process asks the OS to create a **new process** via a **system call**
2. The OS creates a **virtual address space** for that **process**
3. The OS reads the executable **binary** into that memory space
4. The OS sets up the **stack pointer** and the e**xecution pointer**
5. The OS scheduler schedules the program for **execution** by assigning a CPU core to that address space and loading the **registers**

Process

A running program

The operating system has to keep track of hundreds or thousands of processes

It gives each one a Process ID

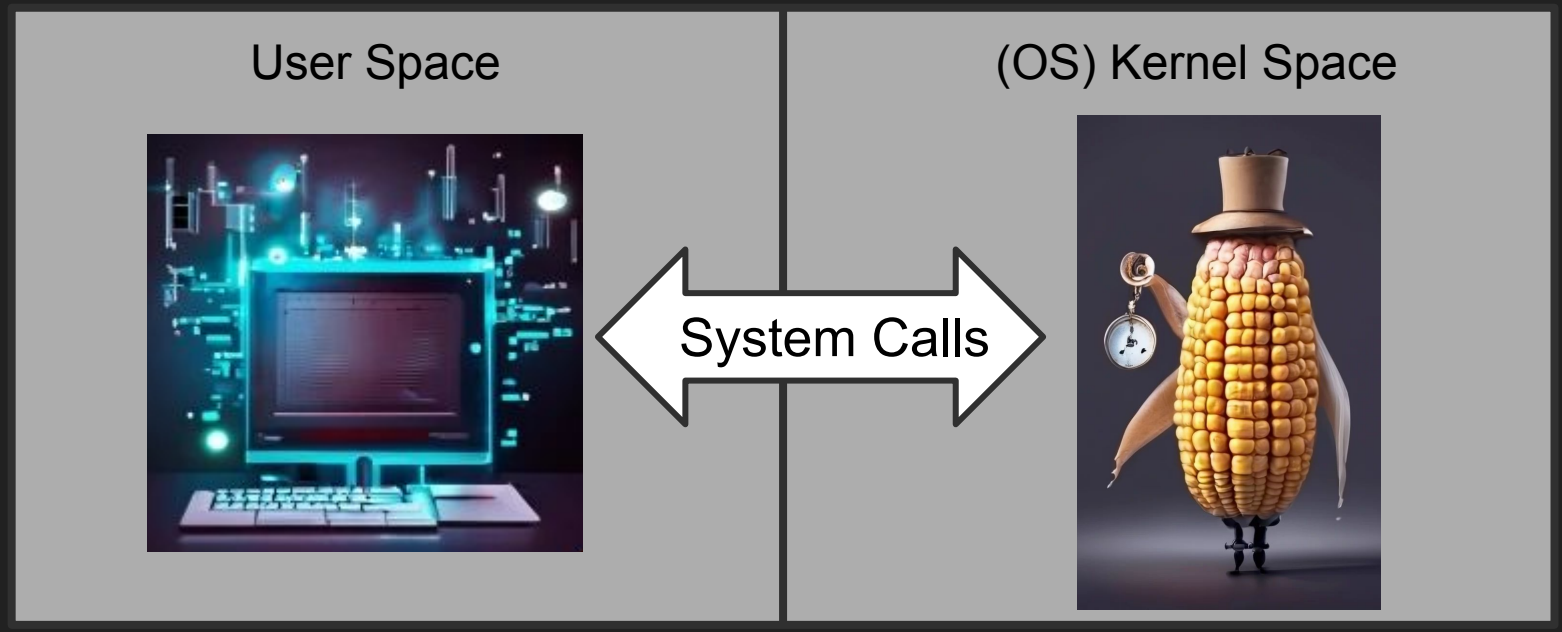
- *NIX: ps -aux,
- Windows: task manager

Each one needs a turn to use the cores on the CPU

Plus the OS needs to run, itself, on the CPU

https://www.tutorialspoint.com/operating_system/os_processes.htm

System calls allow user space programs to ask the OS for services



https://makelinux.github.io/kernel/map/

Executing a program in *NIX and Windows

Windows

- CreateProcess → Create a new process

***NIX:**

- exec → Replace the current process with a new program
- fork/clone → Create a copy of the current process
    - The effect of CreateProcess is created by a fork() then an exec()

Virtual Address Spaces

I have 16GB of RAM

It is addressed as 0x0 through 0x400000000

But every process doesn’t need access to all that memory

Most processes need very little memory

So the OS sits between every process and translates from “virtual” addresses to “real” or “physical” addresses

Every process exists in its own address space, and can imagine it is on a completely blank computer

C Program Memory Layout

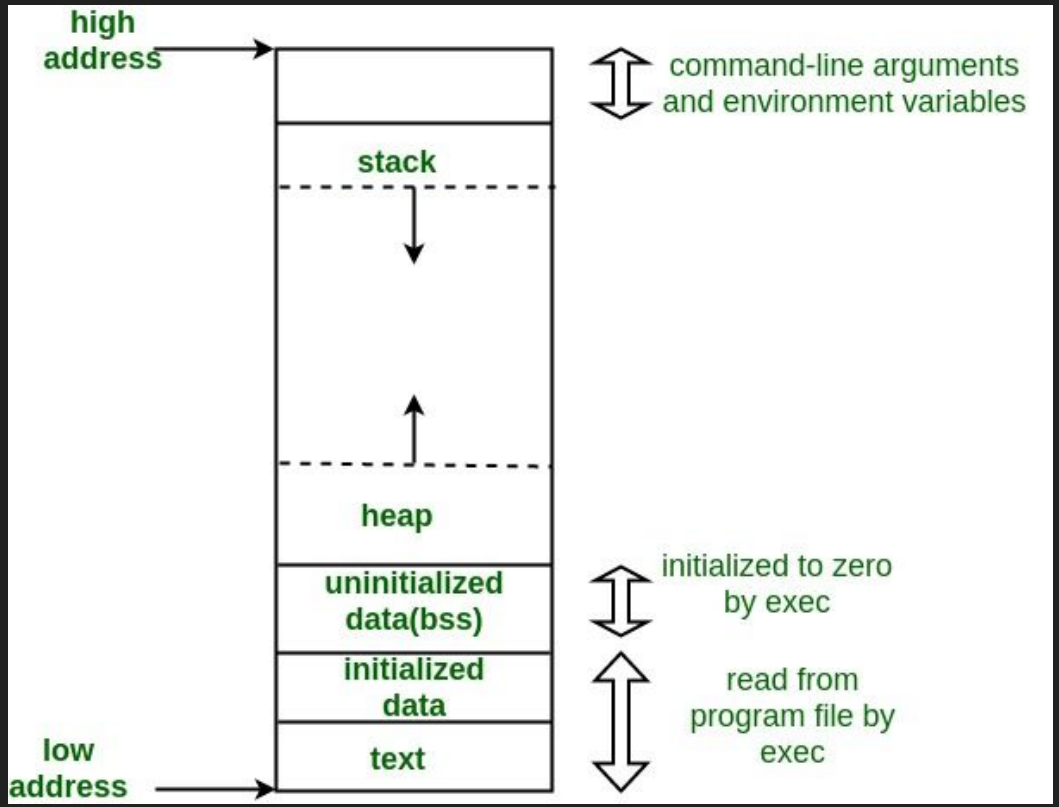


Image source: GeeksForGeeks.org

CPU Registers (more on these next week w/ assembly)

32-bit x86 (introduced with the Intel 386 in 1985)

ESP, EBP, EIP, EAX, EBX, ECX, EDX, EDI, ESI, EFLAGS

64-bit x86

RSP, RBP, RIP, RAX, RAX, RCX, RDX, RDI, RSI, RFLAGS

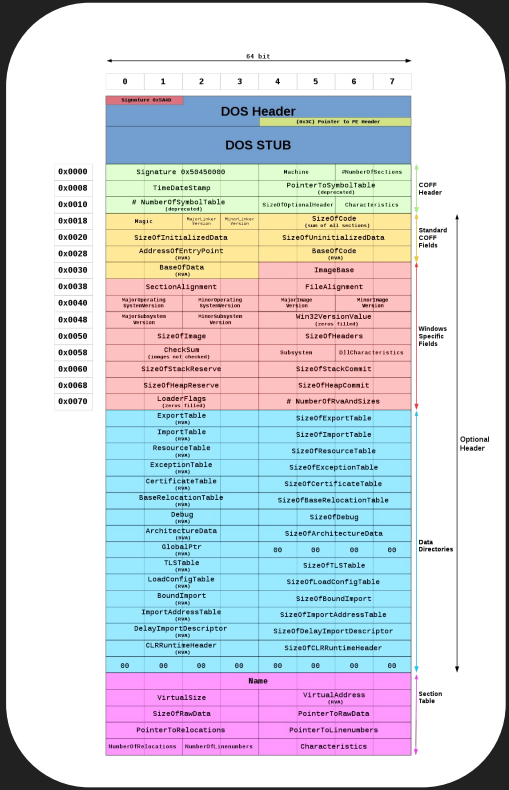
ARM (Similar to RISC)

SP, FP, PC, R0-R5, ZR, CPSR

Windows Portable Executable Format

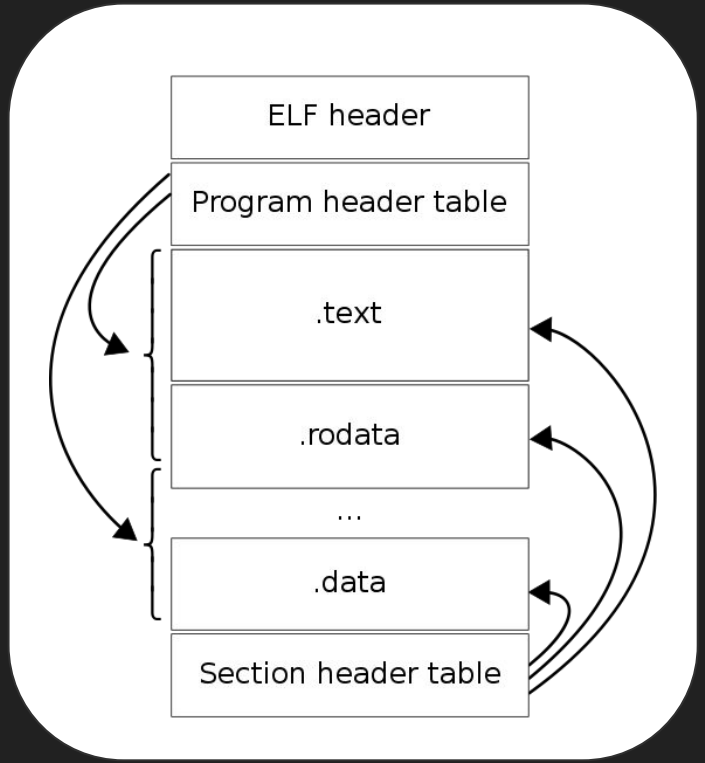
https://upload.wikimedia.org/wikipedia/commons/1/1b/Portable_Executable_32_bit_Structure_in_SVG_fixed.svg

(Zoom Out to View)



Linux ELF Format

https://en.wikipedia.org/wiki/Executable_and_Linkable_Format



Activity

Write a simple program in C.

Compile it

Open the binary in a hex editor and examine

Extra time? Repeat with ELF if you did EXE, or vice versa