CS202 - The Last Lecture

- Announcement

Final Exam: Dec 21 (Thursday)

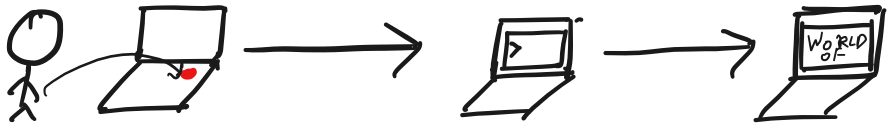
12:00 - 1:50 PM Here

Cummulative, Includes Everything Covered This Sem

Review Next Class. Bring Questions

On To Fun Stuff

Goal



Going To Cheat In Some Way

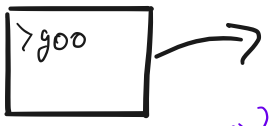
(Did Not Cover Some Background)

Assumption: Only statically linked binaries

```
cc -static -ohello hello.c
```

Rare but stali/oasis/...

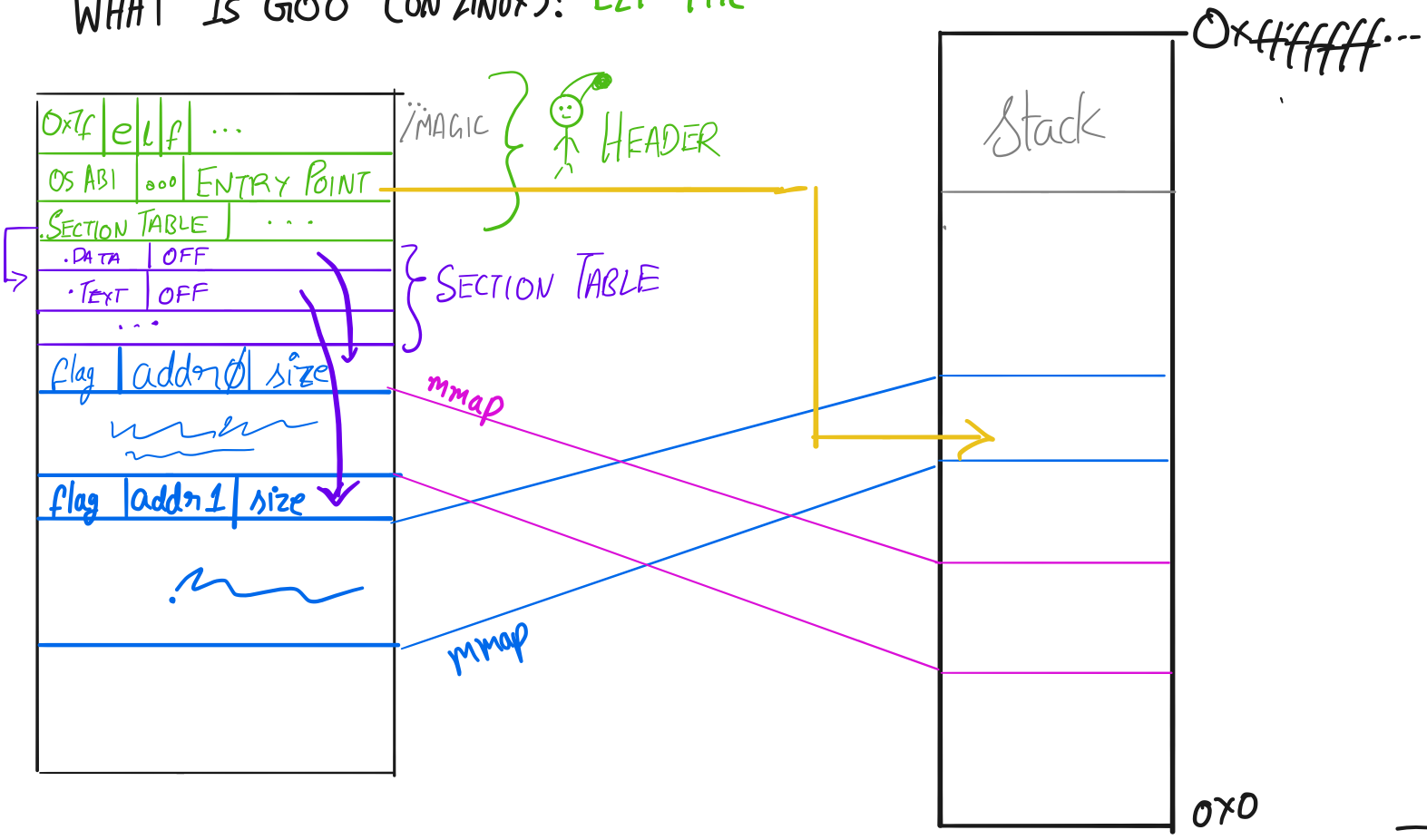
Step 1: Understanding Binary Loading




```
if (fork() == 0) {
    execve("goo", argv, envp);
}
```

if (fork()==0) { if(rand()%1000==0) execve ... }

What Is Goo (on Linux)? ELF File



execve

1. Check file permissions [where? for what?]
2. Open file, read header. Check format
3. Use munmap to remove old mappings
4. Use mmap to map sections to correct place.
5. jmp to entry point.

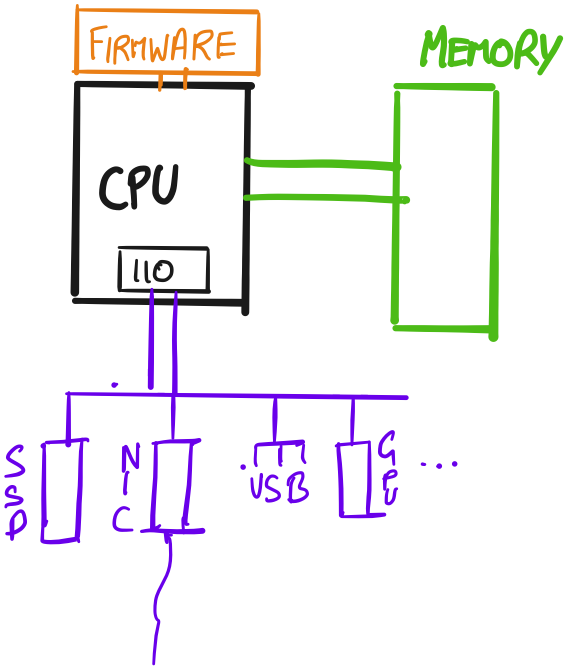
Windows is similar except file format [PE/Portable Exec]

has different layout & different magic number [MZ]

Q. In what process is this executed?

Q. In what mode (user/supervisor)?

```
int main(...) {
    printf("Hello\n");
    execve("goo", ...);
}
```



POWER UP

① CPU sets registers to known values

- User accessible: $0 (%rax, ...)
- Control Registers (%cr1-4, gdt, ldt, tss, ...)

[INTEL/AMD: Put processor in REAL Mode]

② Copy program from FIRMWARE storage (EEPROM) to RAM & call into firmware

[Layout, entry point depend on processor]

PROCESSOR MODES (HISTORIC ACCIDENT)

REAL (No paging, 20-bit memory ...) → PROTECTED (PAGING, 32-bit Addresses) → LONG (x86-64/AMD64)

FIRMWARE

Responsible for

- Initializing processor + devices
- Collecting device info [DEVICE TREE, COMING UP] (Device Enumeration)
- Figuring out where to boot from (Disk/USB stick/Network)
- Loading boot loader
- Running bootloader

Many firmware

- BIOS
- UEFI
- uboot
- ...

} Previously used to prevent competitors from booting OS

Featured in a TV show!!

(Halt & Catch Fire)

We will focus on UEFI

- Switches processor from real → long mode
- Identity maps pages.
- Creates initial IDT, GDT, etc.
- Includes drivers for
  - Disk
  - Sound
  - USB
    - ↳ Disk
    - → KBD
    - → Mouse
  - Display (VGA)
- Provides a library of functions to access these devices
- Provides a file system
  - Vfat
- And a loader for PE programs

An OS!

Loads & runs the bootloader from VFAT partition on boot device

↳ Hard disk : Bootloader location on CMOS

→ USB/Removable: /EFI/boot/bootx64.efi

BOOTLOADER

- THE ENTRY POINT TO THE KERNEL

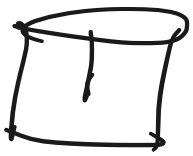
```
void EntryPoint(EFI_HANDLE handle,
                EFI_SYSTEM_TABLE *table)
```

Required for EFI calls

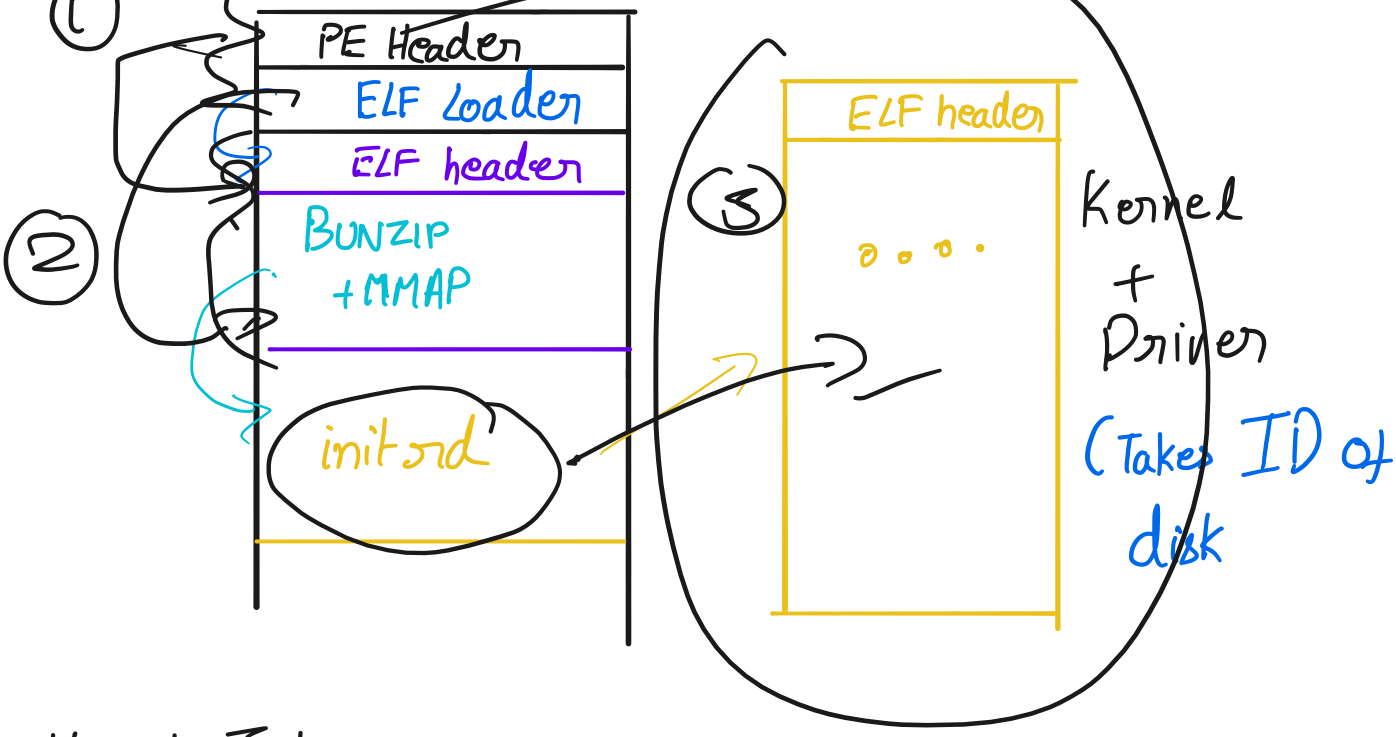
↳ Information about the system
- UEFI services - reboot, halt, print...
- Device tree
- ...

- UEFI requires bootloader is PE binary (REMEMBER USED IN WINDOWS)

- Many bootloaders available
  - EFISTUB
  - GRUB
  - systemd-boot
  - ...



/boot/



Kernel Init

ⓐ Switch away from identity mapped page table

ⓑ Switch IDT (interrupt descriptor table)

ⓒ Load drivers & populate `/dev`

ⓓ Mount `/` [Using disk ID passed as argument]

ⓔ Start initial process

Loading Drivers

- Talked about drivers before

  Software that allows kernel to access hardware

- How to know what drivers to load

for each device?

A: Device tree.

↳ ◦ Address of control registers (BAR)

◦ Type of device

◦ ...

wait(1)

Initial Process

- Many possibilities
  - init.rc
  - systemd
  - init.d
- Not going into details but across all
  - Finish initializing devices
    + NIC: Use DHCP to get address
    + GPU: Set resolution
    + PMU settings
  - Launch daemons

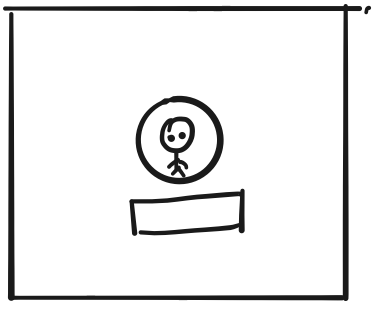
  - sshd
  - ntpd

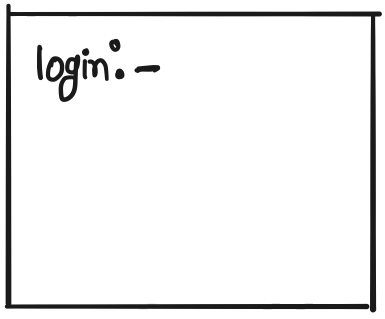
- ..

- Launch session manager

Session Manager

This is the thing you log into. Many choices





login(1): The original session manager

Can only run as root

↳ Uses setuid, setgid

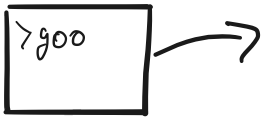
- Can only reduce priv.

① Check password

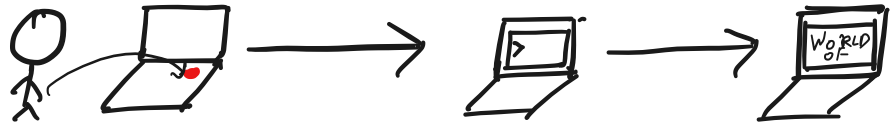
② Fork

↳ setuid, setgid, setsid

→ execve shell




```
...
if (fork() == 0) {
    execve("goo", argv, envp);
}
...
```



Observations

Many OSes loading other OSes

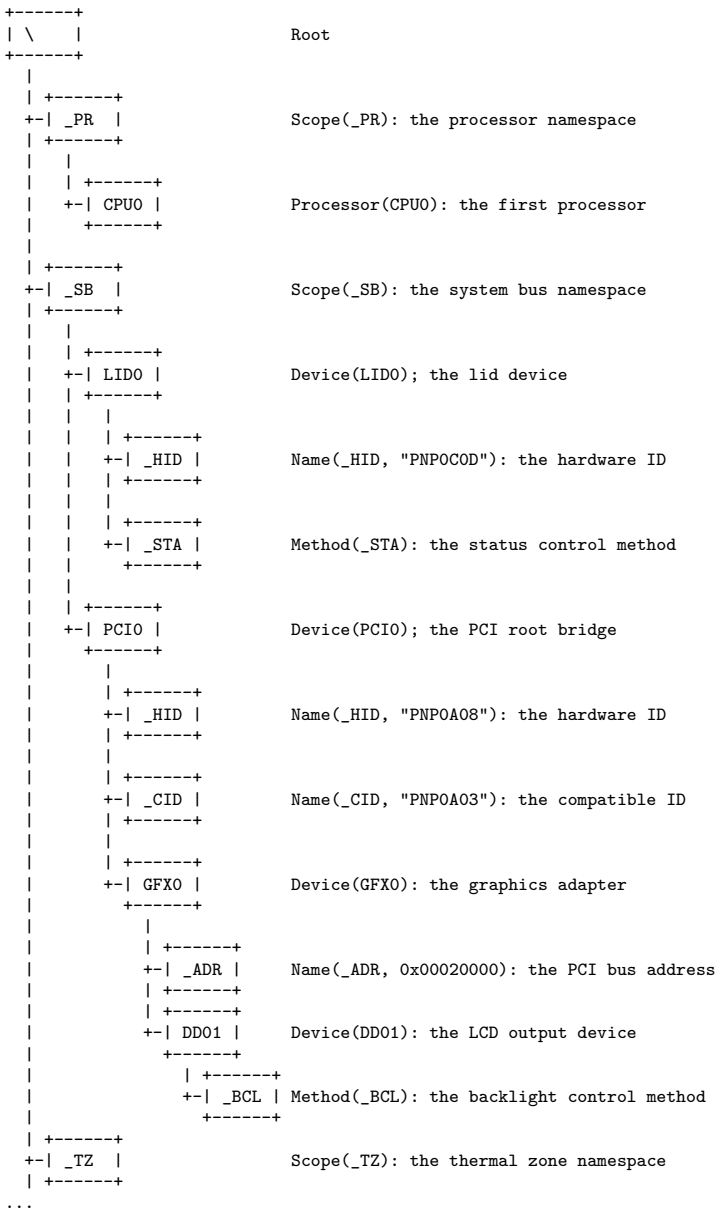
Uefi → bootloader → kernel

There are others in the mix

- System management
- Hypervisor
- ...

CS202

Example Device Tree



Credit: Copied and modified from: https://www.kernel.org/doc/html/latest/firmware-guide/acpi/namespace.html.