CS202 - The Last Lecture

- Announcement

Final Exam: Dec 21 (Thursday)
12:00 - 1:50 PM Here
Cumulative, 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 -o hello hello.c
Rare but stale/ oasis/...

Step 1: Understanding binary loading/ links & loadins
WHAT IS GOO (ON LINUX)? ELF File

execute

1. Check file permissions [Where? for what?]
2. Open file, read header. Check format
3. Use mmap 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]
Q: In what process is this executed?

```c
int main(...) {
    printf("Hello!");
    execute("goodbye");
}
```

Q: In what mode (user/supervisor)?

**Power UP**

1. CPU sets registers to known values
   - User accessible: $0 (eax, ...)
   - Control Registers
     - `cn 1-4, gdtr
     - ldtr, fsr, ...`

   [INTER/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) (No paging, 20-bit memory)

REAL → PROTECTED → LONG
(PAGING, 32-bit Addresses) (x86-64/AMD 64)

FIRMWARE

Responsible for
- Initializing processor + devices
- Collecting device info [DEVICE TREE, Coming Up]
- Figuring out where to boot from (Disk/USB stick/Network)
- Loading boot loader
- Running boot loader

Device Enumeration

Many firmware
- BIOS
- UEFI
- u-boot

Previously used to prevent competitors from booting OS

Featured in a TV show!!
We will focus on UEFI

- Switches processor from real→long mode
- Identifies 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
Bootloader

- The entry point to the kernel

```
void EntryPoint (EFL_HANDLE handle,
               EFL_SYSTEM_TABLE *table)
```

- Required for EFI calls

- Information about the system
  - UEFI services - nboott, halty point...
  - Device tree
  - ...

- UEFI requires bootloader is PE binary
  (remember used in Windows)

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

- USB/Removable: /EFI/boot/bootx64.efi
- Hard drive: Bootloader location on CMOS
Kernel Init

1. Switch away from identity mapped page table
2. Switch IDT (interrupt description table)
3. Load drivers & populate `/dev`
4. Mount `/` [Using disk ID passed as argument]
5. Start initial process

Loading Drivers

- Talked about drivers before

Software that allows kernel to access hardware

- How to know what drivers to load for a device?
for each device:

A: Device tree.

- Address of control registers (BAR)
- Type of device
- ...

wait(1)

Initial Process

- Many possibilities
  - init. nc
  - 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
Session Manager

This is the thing you log into. Many choices

\[
\begin{array}{c}
\text{login}\text{(1)}: \text{The original session manager} \\
\text{Can only run as root} \\
\text{Uses \textit{setuid, setgid}} \\
\text{- Can only reduce priv.}
\end{array}
\]

1. Check password
2. Fork
   \[
   \text{\textit{setuid, setgid, setsid}} \\
   \rightarrow \text{execute shell}
   \]
```c
if (fork() == 0) {
    execlp("goo", argv, envp);
}
```

**Observations**

- Many OSes loading other OSes
  - UEFI → bootloader → kernel
- There are others in the mix
  - System management
  - Hypervisor
  - ...

> goo
Example Device Tree

```
+------+
| \   |
+------+
   Root

  +------+
  | \    |
  +------+
     _PR |
  +------+
  |
  +------+
  +-| CPU0 | Processor(CPU0): the first processor
  +------+
  |
  +------+
  +-| _SB | Scope(_SB): the system bus namespace
  +------+
  |
  +------+
  +-| LIDO | Device(LIDO); the lid device
  +------+
  |
  +------+
  +-| _HID | Name(_HID, "PNP0C0D"): the hardware ID
  +------+
  |
  +------+
  +-| _STA | Method(_STA): the status control method
  +------+
  |
  +------+
  +-| PCI0 | Device(PCI0); the PCI root bridge
  +------+
  |
  +------+
  +-| _HID | Name(_HID, "PNP0A08"): the hardware ID
  +------+
  |
  +------+
  +-| _CID | Name(_CID, "PNP0A03"): the compatible ID
  +------+
  |
  +------+
  +-| GFX0 | Device(GFX0): the graphics adapter
  +------+
  |
  +------+
  +-| _ADR | Name(_ADR, 0x00020000): the PCI bus address
  +------+
  |
  +------+
  +-| DD01 | Device/DD01): the LCD output device
  +------+
  |
  +------+
  +-| _BCL | Method(_BCL): the backlight control method
  +------+
  |
  +------+
  +-| _TZ | Scope(_TZ): the thermal zone namespace
  +------+
```
