Booting: System Startup

Pradipta De
pradipta.de@sunykorea.ac.kr
Booting (Pictorially)
Booting Overview

- BIOS: Basic Input/Output System
  - POST (Power-on Self Test)
  - Loads Bootloader

- Bootloader:
  - Loads the Operating System into memory
  - Implementation: GRUB (Grand Unified BootLoader)

- OS: starts the first user process /sbin/init
  - Checks init tab and triggers different run levels
Power On → BIOS

• When powered on, main memory is empty
• CPU begins execution by accessing a pre-defined address, 0xffff:fff0, which is mapped to the ROM
• BIOS (firmware) is loaded into memory
  – Addressing is real mode
    • Real mode address is (seg*16 + offset)
  – Provides device drivers for every hardware on the computer, like keyboard, video card, system board, memory, and other I/O devices
• BIOS checks the hardware devices – POST step – and then initializes the hardware
• BIOS locates the bootable device and loads the MBR
  – On disk, the sector 0 is called the Master Boot Record (MBR) and contains address of the bootloader for the active partition
  – MBR: Windows MBR, LILO or GRUB on Linux, even a virus 😞
MBR contains
1. Master Boot Code/boot loader code (446 bytes)
2. Drive’s Partition Table (4 16-byte entries)
3. MBR signature (2 bytes)
Bootloader

• BIOS loads the MBR into memory location 0x7c00 and starts executing
  – MBR has information about multiple partitions
  – First sector of a partition is the boot sector

• Bootloader is a two stage process
  – MBR executes the first stage by loading another sector from disk with additional bootstrap code
  – Second stage loads the kernel of the OS
Bootloader → Kernel Initialization

• Bootloader performs
  – Loads OS kernel into memory
  – Initializes RAMdisk
  – Transfer execution to kernel

• Once initialized, the kernel performs
  – Scans hardware configs, initialize device drivers, and puts CPU into protected mode with virtual memory initialized
  – Mount root filesystem
  – Spawn the first user process (pid=1) -- /sbin/init
  – Init process executes the scripts to activate different subsystems
### OS booted (logical view)

<table>
<thead>
<tr>
<th>User Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Standard Libs</strong> (shell, commands, system lib)</td>
</tr>
</tbody>
</table>

**System call interface**

<table>
<thead>
<tr>
<th>Process Management</th>
<th>Memory Management</th>
<th>FileSystems</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Scheduling</td>
<td>- Virtual Memory</td>
<td></td>
</tr>
<tr>
<td>- Synchronization</td>
<td>- Caching</td>
<td></td>
</tr>
</tbody>
</table>

**Kernel interface to hardware (device drivers)**

**Devices** (keyboard, mouse, memory Controllers)
Putting It Together

• How does the bootstrap process works?
  – Understand the steps involved from power on till the OS kernel takes over