Introduction

Yolanda Becerra Fontal
Juan José Costa Prats

Facultat d'Informàtica de Barcelona (FIB)
Universitat Politècnica de Catalunya (UPC)
BarcelonaTech
2019-2020 QT
Content

• Previous concepts
  – Architecture
  – Operating Systems

• System Initialization
  – Boot Process
    • Power ON
    • BIOS
    • Bootloader
    • System Initialization
  – Example: Linux & Windows
Architecture

- Memory
- Cache
- CPU
- Registers
- PIC

SO2/SOA
x86 processor

• Registers
• Operations:
  – Arithmetic
  – Logic
  – Memory
  – Other
Operating System

- Software between user and HW
- Manages:
  - Processes
  - Memory
  - I/O & Filesystem
- Support to multithread/multiprocess
## OS: SW between user and HW

<table>
<thead>
<tr>
<th>USER</th>
<th>System Libraries</th>
<th>Language Libraries</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS</td>
<td>System Calls</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Protection and Security</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Resource Management</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Portability and Ease of Use</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Efficiency</strong></td>
<td></td>
</tr>
<tr>
<td>HW</td>
<td><strong>Process Management</strong></td>
<td>Responsible for executing programs and their CPU allocation</td>
</tr>
<tr>
<td></td>
<td><strong>Memory Management</strong></td>
<td>Responsible for assignment of physical memory and its translation</td>
</tr>
<tr>
<td></td>
<td><strong>I/O &amp; File System</strong></td>
<td>Responsible for device accesses and storage mgmnt</td>
</tr>
<tr>
<td></td>
<td><strong>Support to multithreaded/multiprocess applications</strong></td>
<td>Synchronization and communication</td>
</tr>
<tr>
<td></td>
<td><strong>Privileged Instructions</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Execution Modes</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Mode Changing</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Specific Support</strong></td>
<td></td>
</tr>
</tbody>
</table>
# OS: Process Management

## USER
### OS
- getpid
- fork
- exit
- exec

**Process management**
- Responsible for executing programs and their CPU allocation

## System Libraries
### Data Structures:
- PCB, queues...

## Language Libraries
### Functions:
- getpid
- fork
- exit
- exec

## Protection and Security

## Resource Management

## Portability and Ease of Use

## Efficiency

## Core Components:
### Privileged Instructions

### Execution Modes

### Mode Changing

### Clock Interrupt

## Hardware (HW)
OS: memory management

**USER**
- malloc
- free

**OS**
- protection and security
- resource management
- portability and ease of use
- efficiency

**memory management**
responsible for assignment of physical memory and its translation

- memory assignment
- program loading
- process address space definition
- translation
- virtual memory
- COW
- dynamic memory
- prefetch
- shared memory

**HW**
- privileged instructions
- execution modes
- mode changing
- interrupts, MMU: page table, TLB...
### OS: I/O & File System

<table>
<thead>
<tr>
<th>USER</th>
<th>system libraries</th>
<th>language libraries</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS</td>
<td>open</td>
<td>close</td>
</tr>
<tr>
<td></td>
<td>read</td>
<td>lseek</td>
</tr>
<tr>
<td></td>
<td>write</td>
<td>mknod</td>
</tr>
</tbody>
</table>

**I/O & File System**

Responsible for device accesses and storage management

- **device independence**
- **name space management**
- **disk management**
- **permission access**
- **directory**
- **virtual dev. table**
- **open file table**
- **inodes table**

**HW**

- **privileged instructions**
- **execution modes**
- **mode changing**
- **interrupts, I/O instructions**
OS: Support to multithreaded/multiprocess applications

Support to multithreaded/multiprocess applications
synchronization and communication

- thread management
- support to synchronization
- deadlock management

- privileged instructions
- execution modes
- mode changing
- interrupts, locks, atomic instructions
• Previous concepts
  – Architecture
  – Operating Systems

• System Initialization
  – Boot Process
    • Power ON
    • BIOS
    • Bootloader
    • System Initialization

  – Example: Linux & Windows
System initialization

• Starting from a computer
• A user wants to execute applications on top of it
• The operating system manages the HW resources to execute the apps.
• But, how the operating system is self-loaded and executed?
  – The boot process

Eclipse
Word
Doom

Operating System

HW
Boot Process

- System Load and initialization
  - Stages

Power on $\rightarrow$ BIOS $\rightarrow$ Bootloader $\rightarrow$ Complete OS initialization

load $\rightarrow$ load $\rightarrow$ load
Power ON

- HW reset signal
  - Reset all devices
- Load BIOS code into memory
  - Hardcoded in the motherboard
  - Remember that, in order to execute any code, it must be loaded into memory
- Start BIOS code execution
• **Power-On Self Test (POST)**
  – Detect & Initialize HW devices

• **Load an OS into memory**
  – From a bootable device
  – Disk organization:
    • MBR + Partition table
    • Partitions (with one or more bootable)
  – Copy MBR to a fixed location in memory
    • Single disk sector
  – A single disk sector limits OS features: 512 bytes too small
    • So, OS kernels are stored somewhere else in the disk
    • Bootloader instead

• **Start bootloader code execution**
Bootloader

• Load the image of an OS kernel into memory
  – Search for an OS in disk
  – Load OS sectors from disk to memory
    • Using BIOS
    • Using Real Mode

• Start OS kernel code execution
OS kernel

- Initialize OS itself
  - Internal structures
    - Process management
    - Memory management
    - I/O management
  - HW
    - Keyboard, video adapter card, ...
    - IDT, GDT, ...
- Switch to Protected Mode
- Start Initial Process
Example: Linux & Windows

• Both OS do basically the same things
• The main differences resides in:
  – **Boot loader:**
    • Linux offers the possibility to load different OS from different partitions
    • Windows just uses the first bootable partition found
      – By default, allows to choose other windows versions installed in the machine
      – Workaround is possible to load different os’s: create a file with the boot sector of the partition
• Understanding the linux kernel: Appendix A (Boot process)
• Windows internals: Chapter 13 (Boot process)