

Introduction

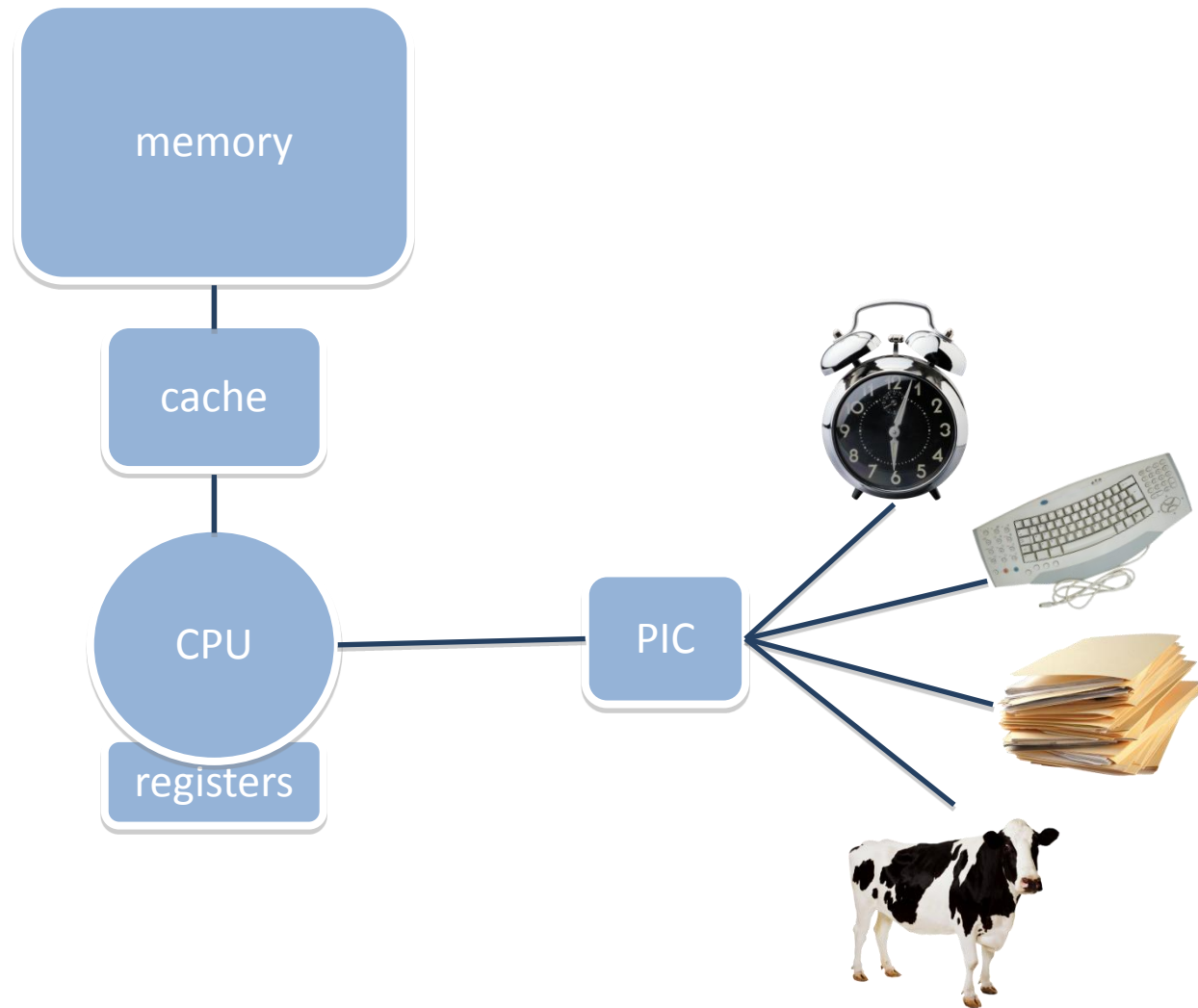
Yolanda Becerra Fontal
Juan José Costa Prats

Facultat d'Informàtica de Barcelona (FIB)
Universitat Politècnica de Catalunya (UPC)
BarcelonaTech
2017-2018 QP

Content

- Previous concepts
 - Architecture
 - Operating Systems
- System Initialization
 - Boot Process
 - Power ON
 - BIOS
 - Bootloader
 - System Initialization
 - Example: Linux & Windows

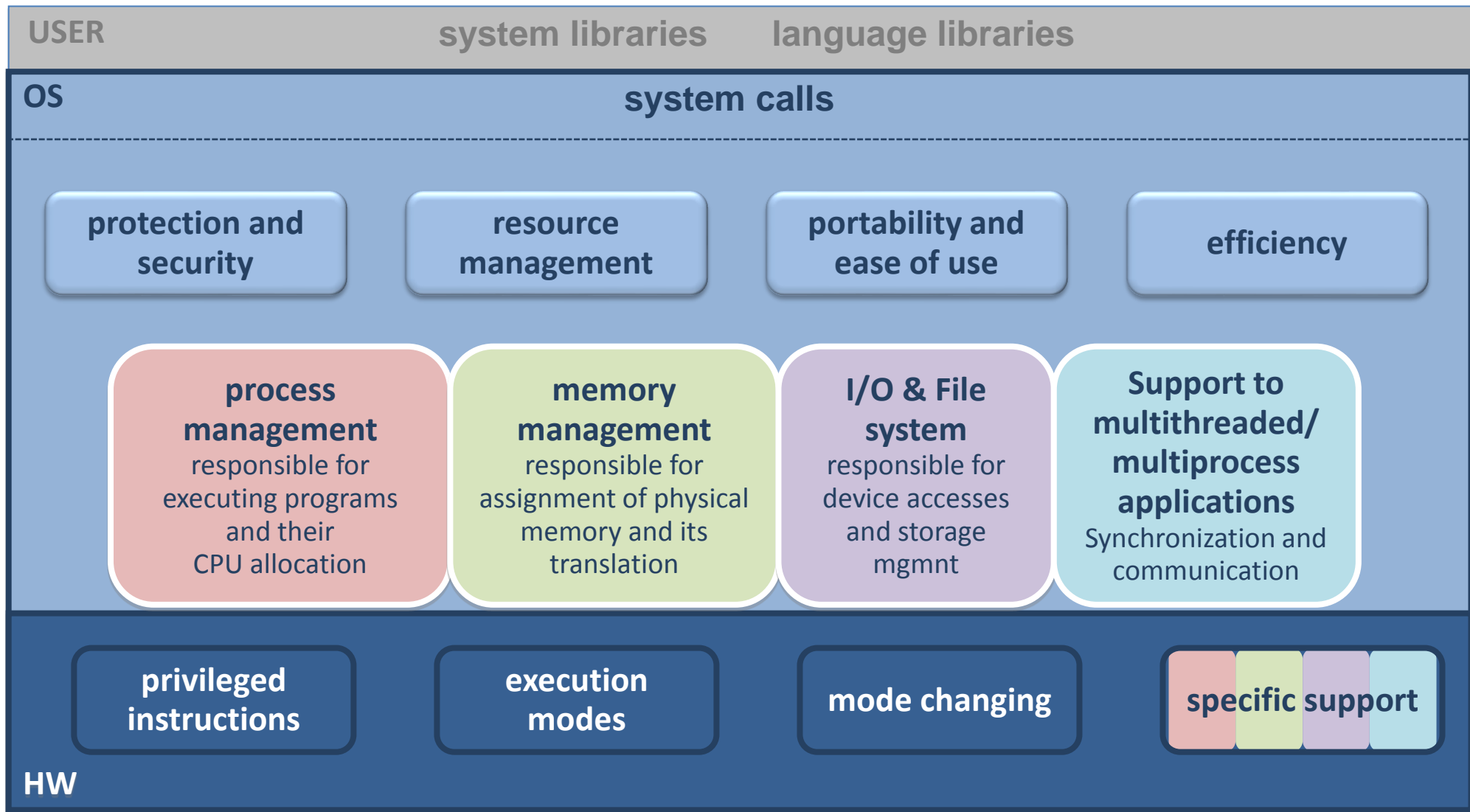
Architecture



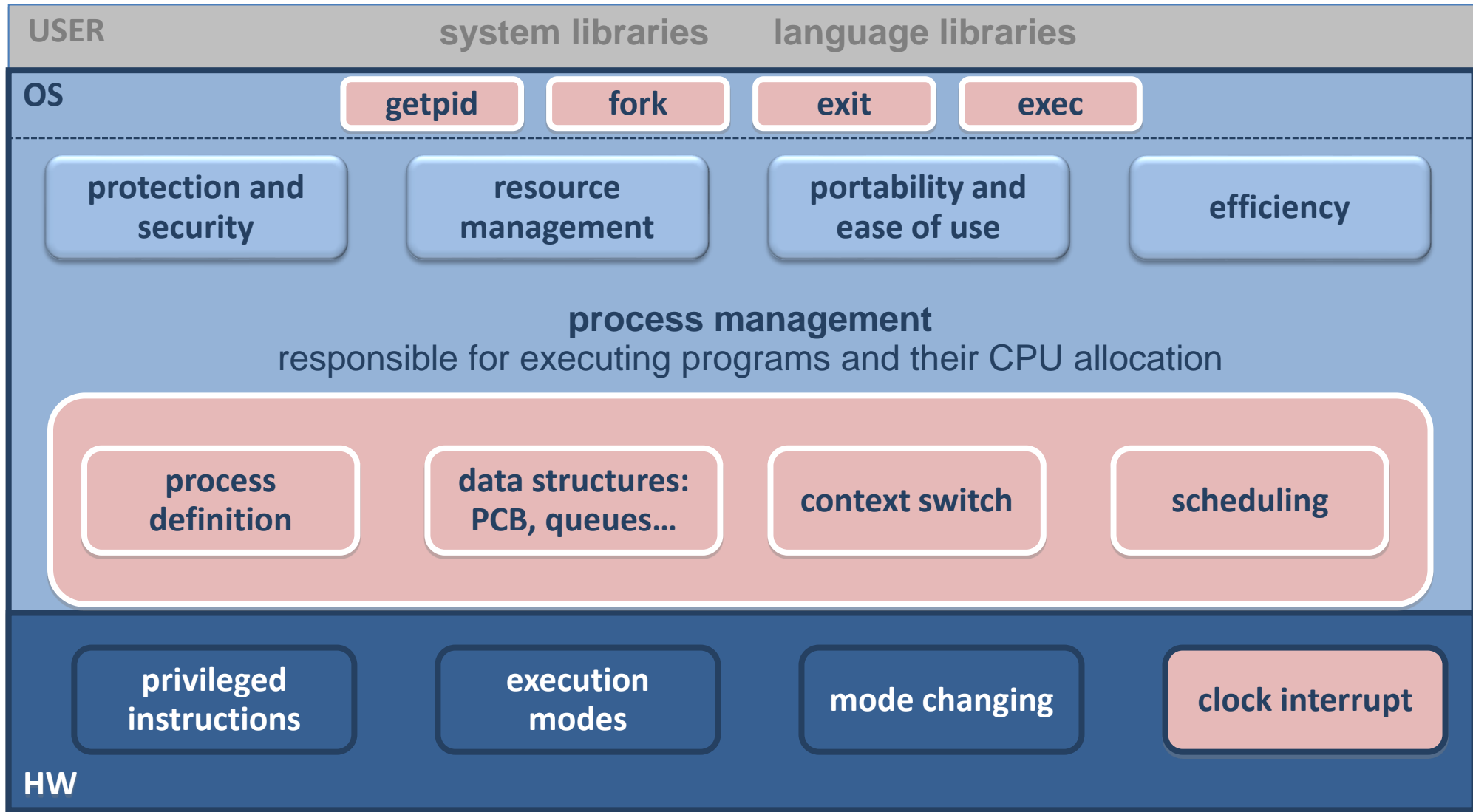
Operating System

- Software between user and HW
- Process management
- I/O & Filesystem
- Support to multithread/multiprocess

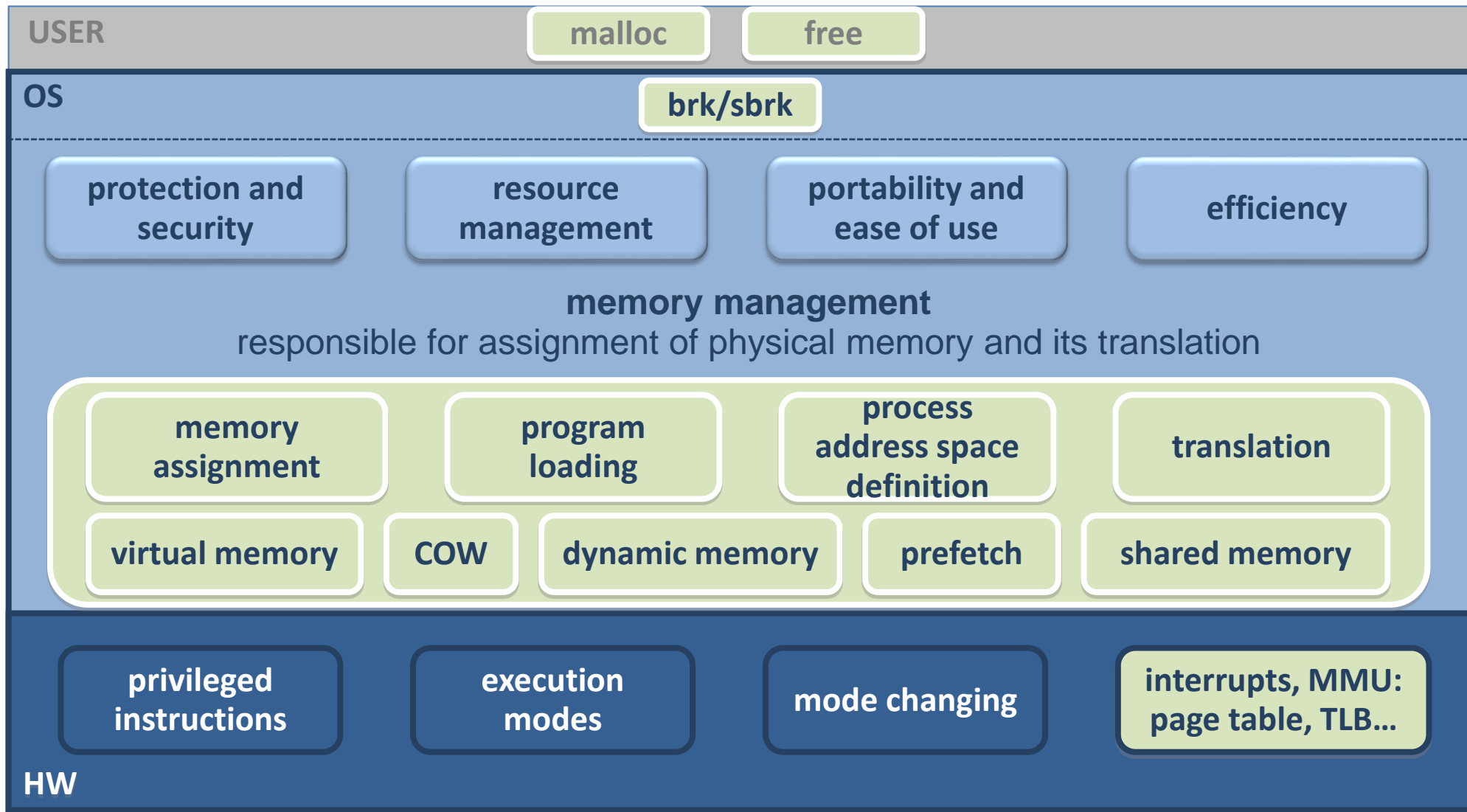
OS: SW between user and HW



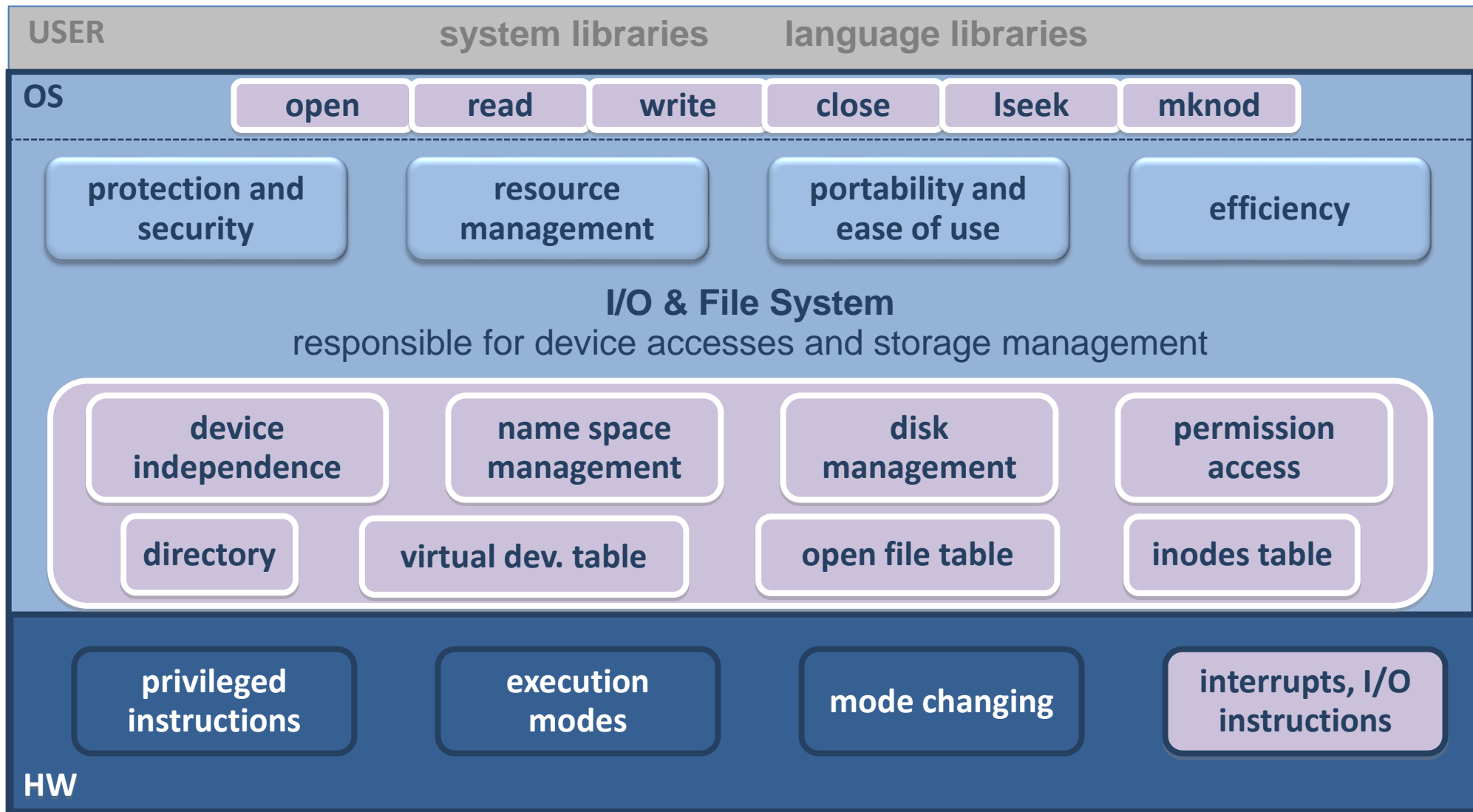
OS: process management



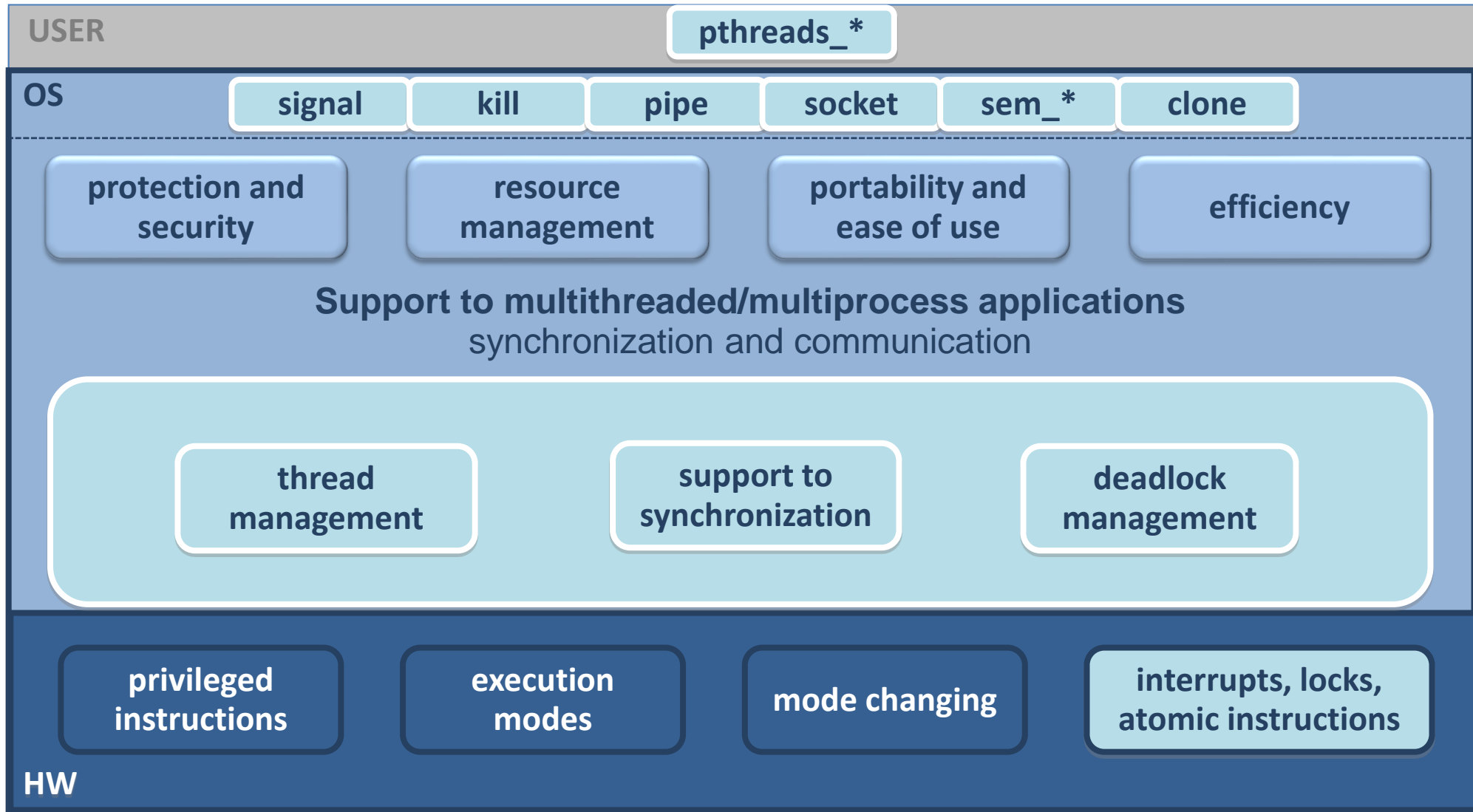
OS: memory management



OS: I/O & File System

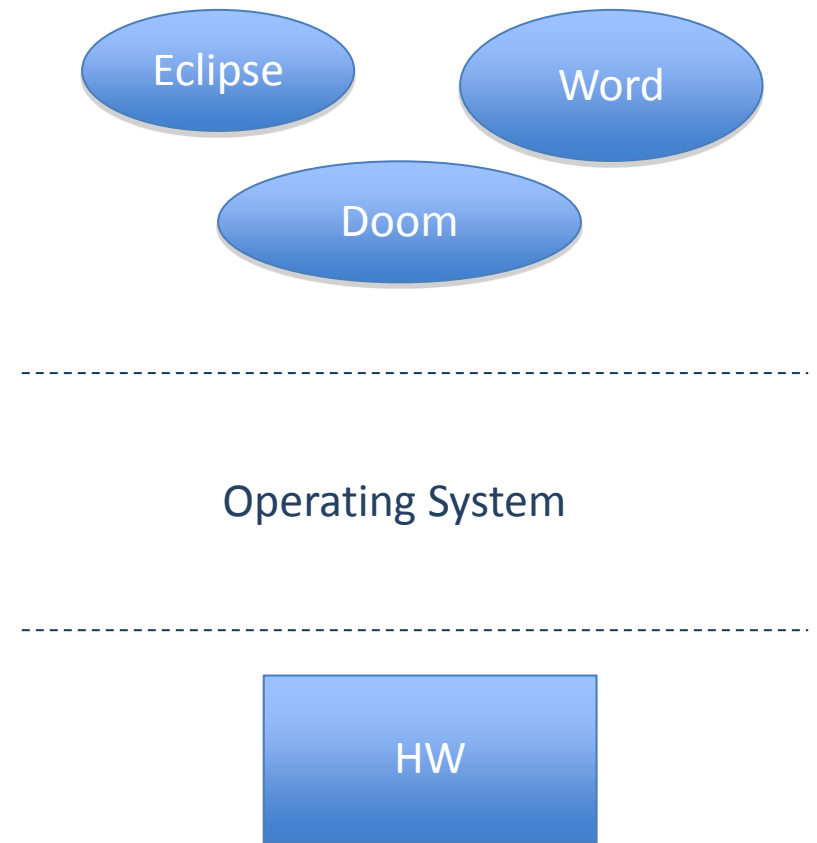


OS: Support to multithreaded/multiprocess



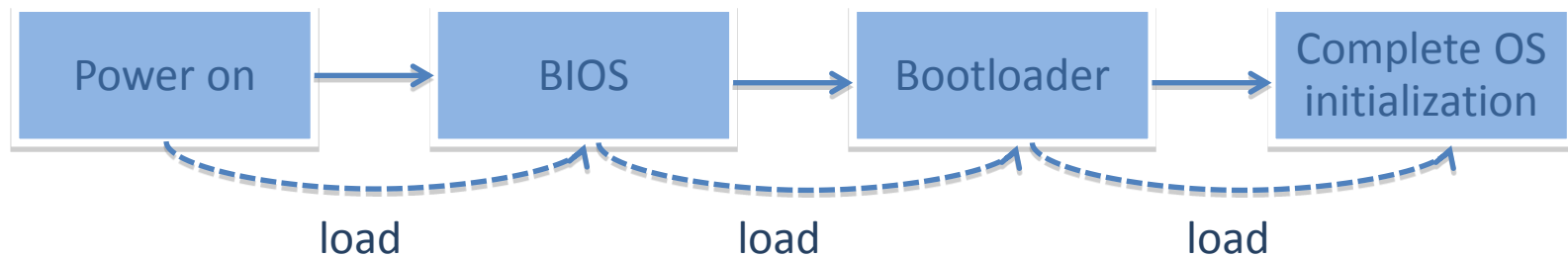
System initialization

- 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



Boot Process

- System Load and initialization
 - Stages



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

BIOS (Basic Input Output System)

- 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)