COMPUTER ARCHITECTURE AND SOFTWARE EXECUTION PROCESS

OPERATING SYSTEMS - OS

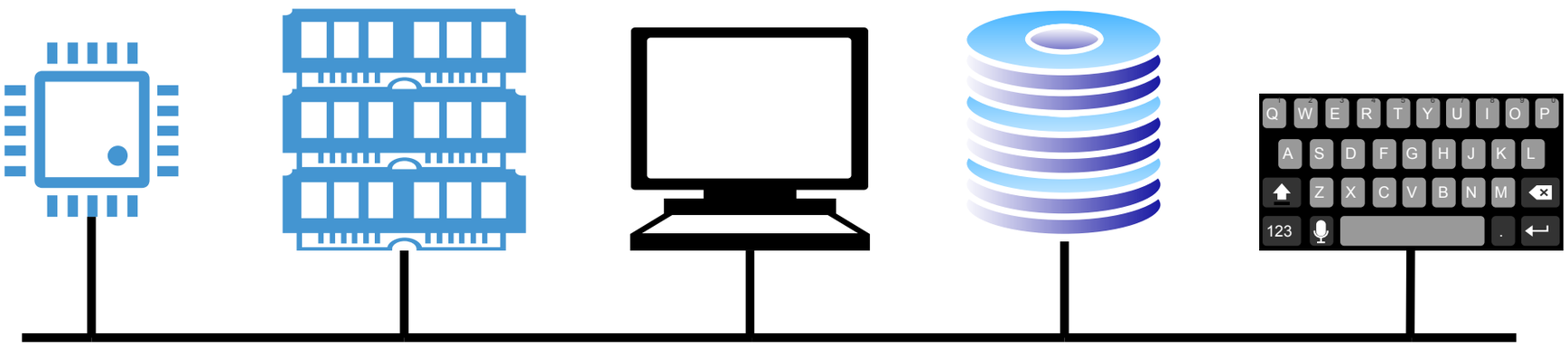
🎓 Bachelor in Artificial Intelligence, Data and Management Sciences
🏛 CentraleSupelec and ESSEC Business School - 2023/2024

Idir AIT SADOUNE
idir.aitsadoune@centralesupelec.fr

OUTLINE

Back to the outline - Back to the begin

HOW DOES THE COMPUTER WORK?



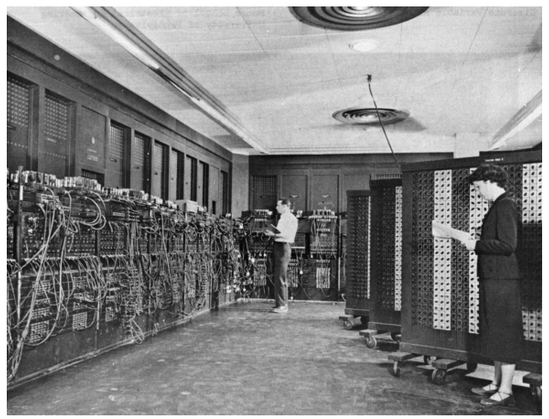
It's all just electrical wires...
... that we turn **ON** and **OFF**.

OUTLINE

Back to the outline - Back to the begin

FORMERLY: ENIAC

- First **fully electronic** computer.
- **Programming** was done directly **in machine language**.
- **Only one program** can run at a time.
- The lack of an operating system forced the programmer to **manually load the program**.

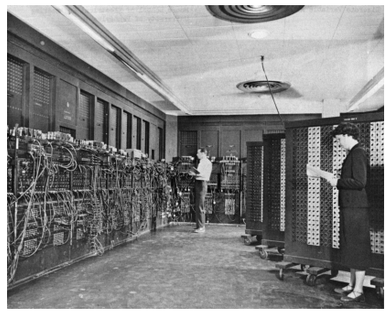


FORMERLY: IBM RAMAC 305

- The **First hard disk computer** (the **IBM 350**) was marketed in September 1956 by **IBM**.
- It consists of the following elements:
  processing unit, printer, console, power supply, hard drive, and 5MB memory.

- **The processing unit** is based on a magnetic drum where the program is stored.
- **An operator** programs using **punched cards** and writes the data onto the drum.

AUTOMATE TASKS

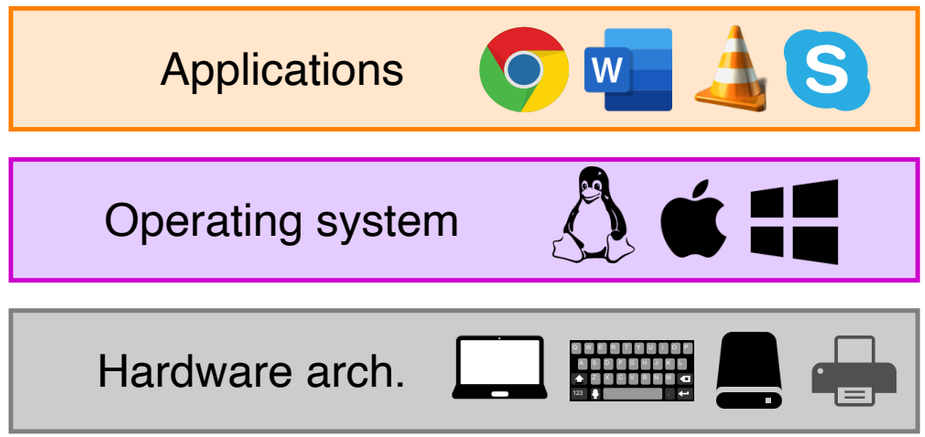


- How to **automate the tasks** of operators and programmers?
- Write a **computer program** that:
  - decides who does what and when.
  - bridges software applications and hardware

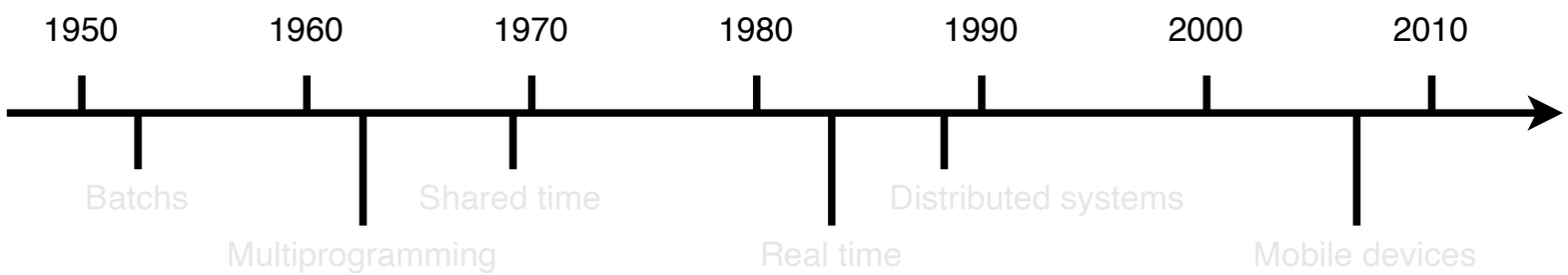
DEFINITION

*"**An operating system** is a set of programs providing the interface between hardware and users."*

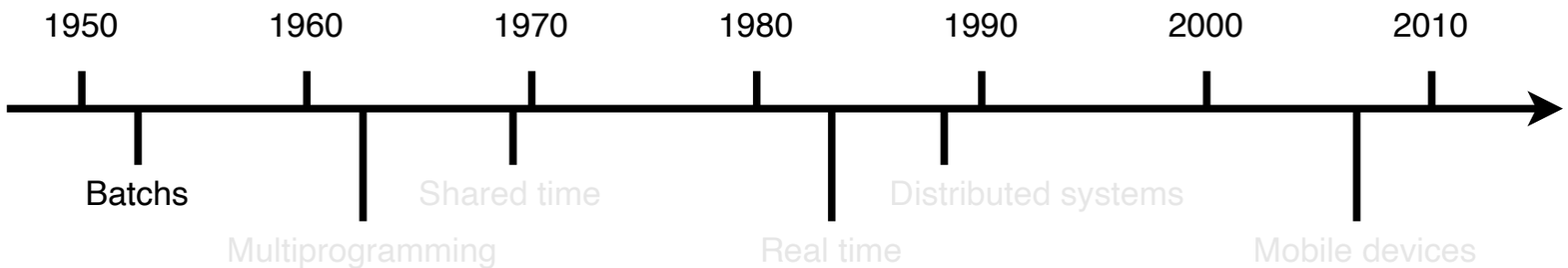
- manages the hardware part.
- serves as a base for software applications



HISTORY/TYPES OF OS

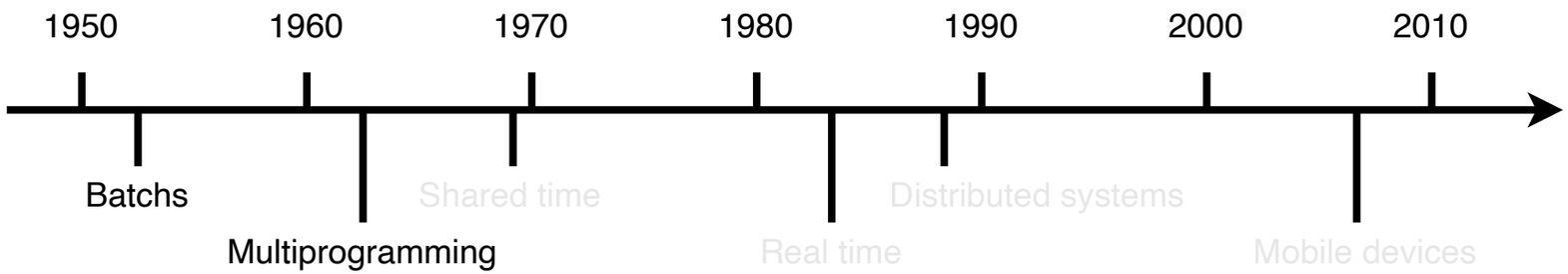


HISTORY/TYPES OF OS



- **Batch systems** are based on two programs:
  1. the loader: loads the programs into the main memory from punch cards.
  2. the processing monitor: executes the sequence of tasks one by one in place of the operator.

HISTORY/TYPES OF OS

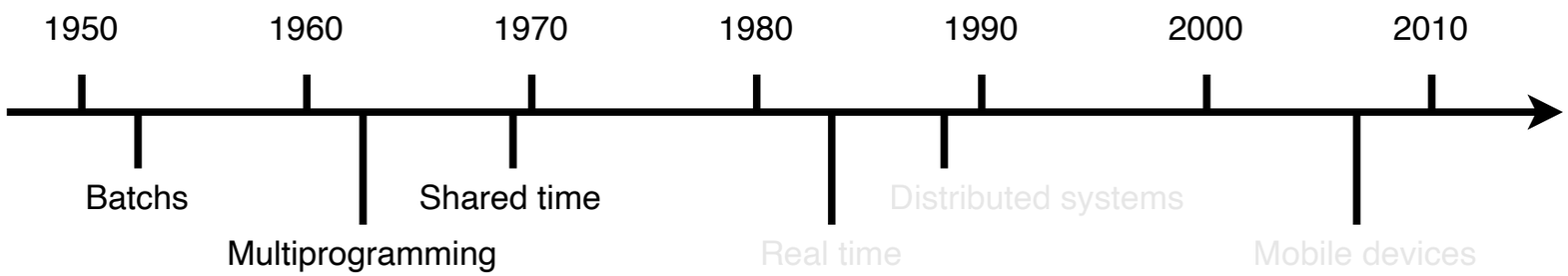


Use multiple components in parallel, which requires:

- **Priority management** (*which process can use the resource*)
  - scheduling
- **Shared memory** (*manage information from multiple processes*)
  - addressing and memory

Example : MULTICS

HISTORY/TYPES OF OS

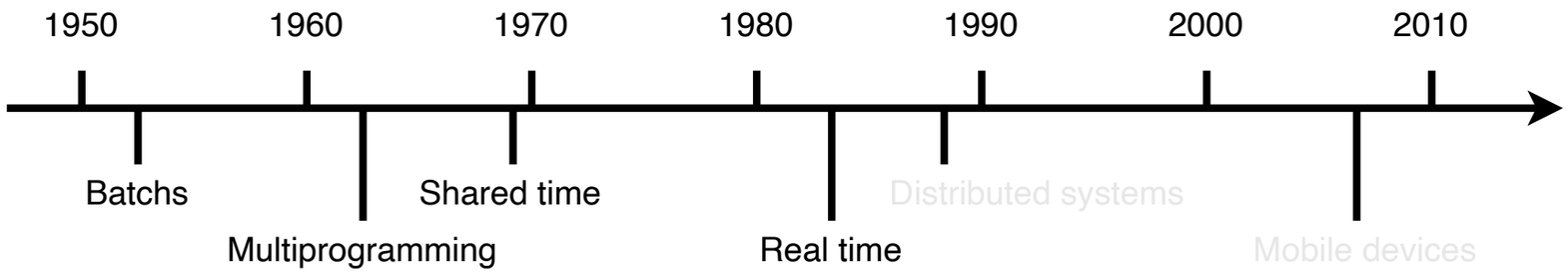


Multiple active processes alternating on the processor

- Interrupt management
- Process synchronization and concurrent programming

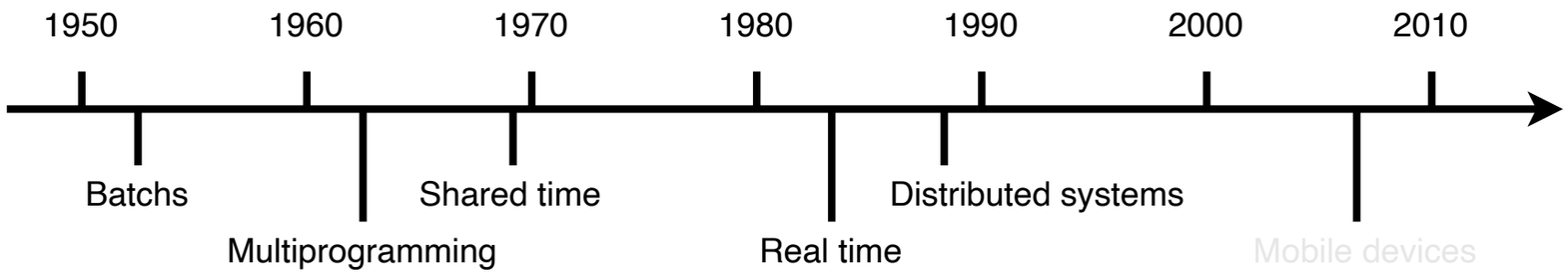
Example : UNICS or UNIX

HISTORY/TYPES OF OS



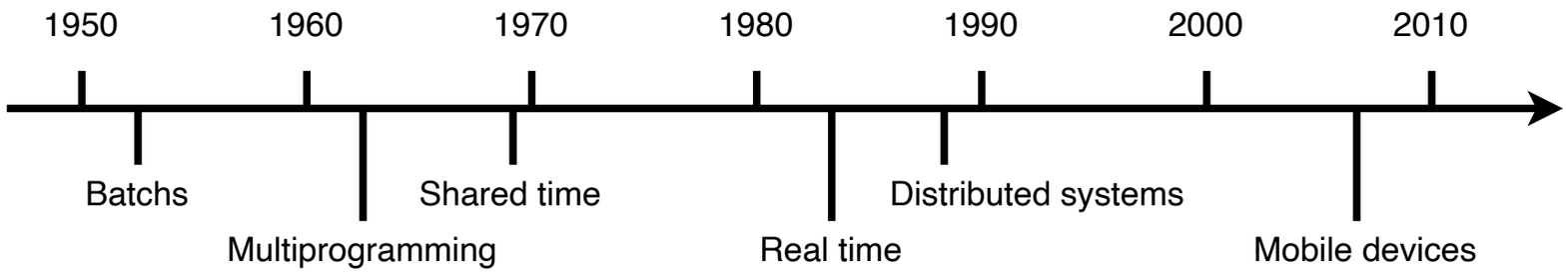
- **Deadline management**: **response time** constraint
  - processes must respond quickly
- Development of **microcomputers**
  - CP/M → IBM PC (MSDOS)
- Appearance of **graphical user interfaces**
  - Xerox → Apple Macintosh 1984, Windows 95, Linux 1991

HISTORY/TYPES OF OS



- **Computers communicate** to exchange data!
  - Arpanet (1967) designed by DARPA
  - E-mail (1972) designed by Ray Tomlinsonn
  - TCP/IP(1972)
  - Client-server $\rightarrow$ NFS - Network File System (Sun, 1984)
  - Arpanet opened late 80s $\rightarrow$ Web early 90s (CERN , Tim Berners-Lee)

HISTORY/TYPES OF OS



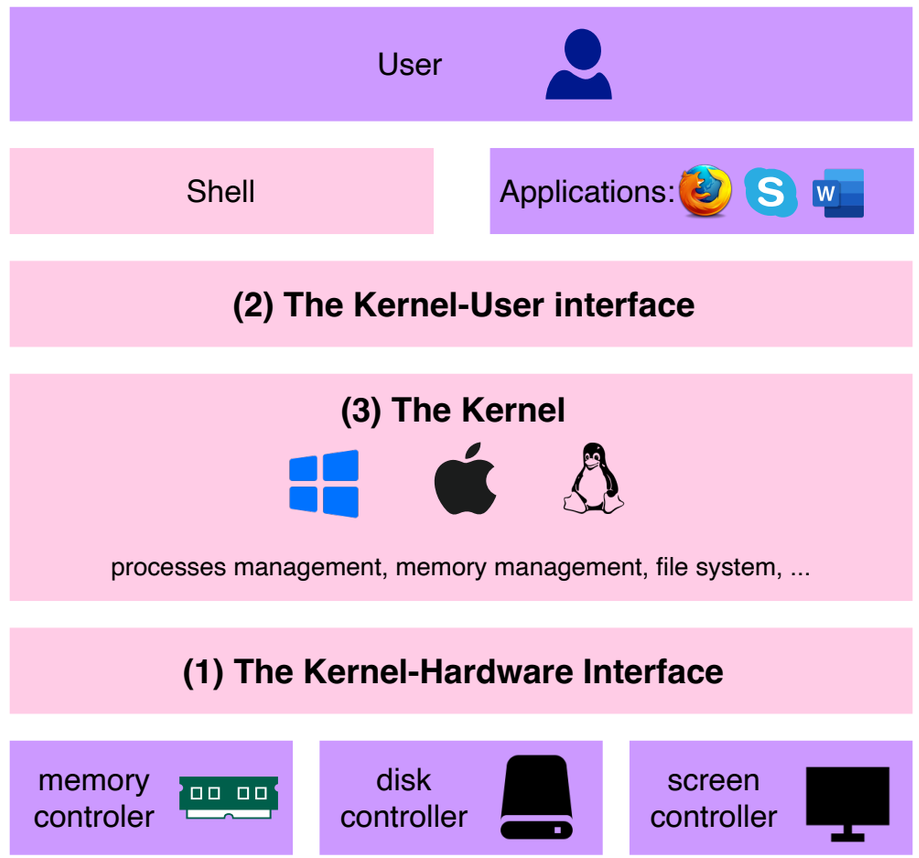
- **Mobile devices** (or handheld computers) have existed since the 80s.
  - 1986 : PDA $\longrightarrow$ PalmOS
  - 2007 : smartphones $\longrightarrow$ android OS
  - 2007 : iPhone $\longrightarrow$ iOS

OUTLINE

Back to the outline - Back to the begin

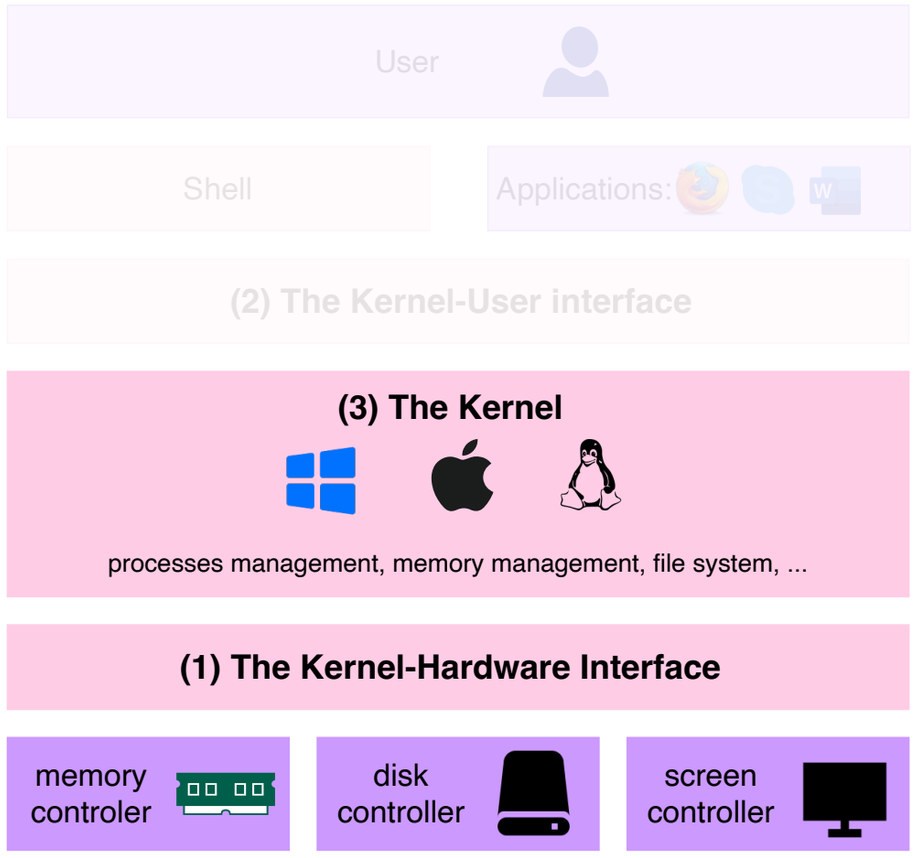
OPERATING SYSTEM ROLES

1. **The kernel-hardware interface** supports the management and sharing of machine resources.
2. **The kernel-user interface** provides an easier-to-use and more user-friendly virtual machine.
3. **The kernel** provides several significant functionalities.



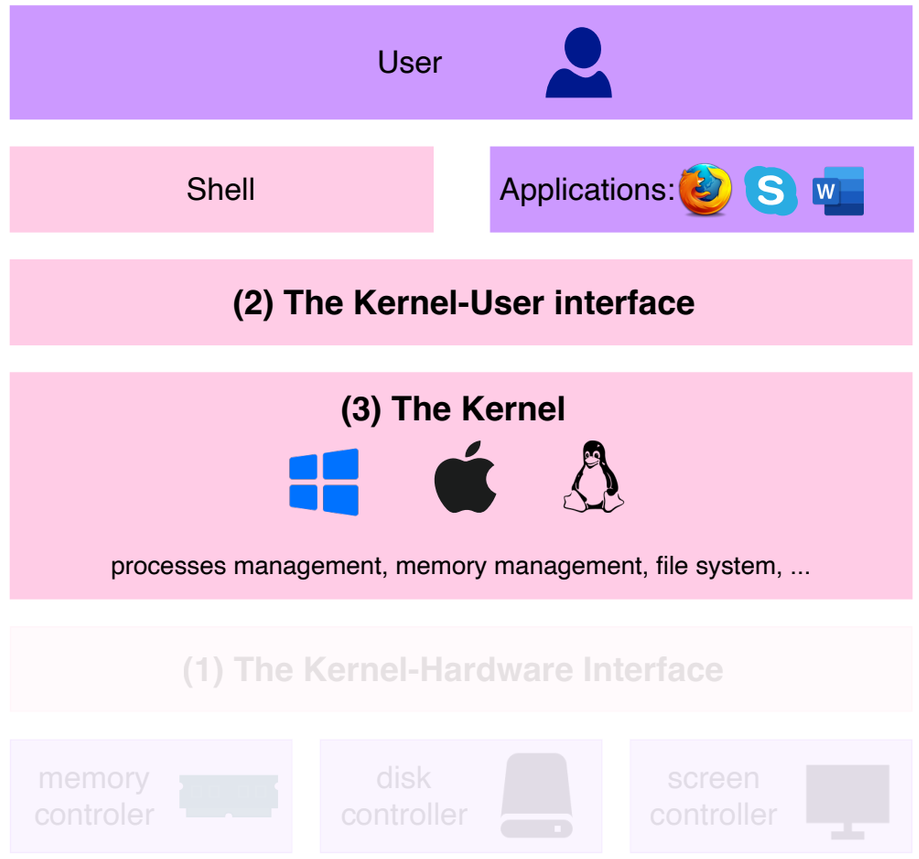
THE KERNEL-HARDWARE INTERFACE

- **Manage access and sharing** of hardware resources.
  - processor
  - memory
  - peripheral devices
  - ...
- This management must ensure:
  - **equity** of access to hardware
  - **protecting** access to hardware
  - **consistency** of hardware states



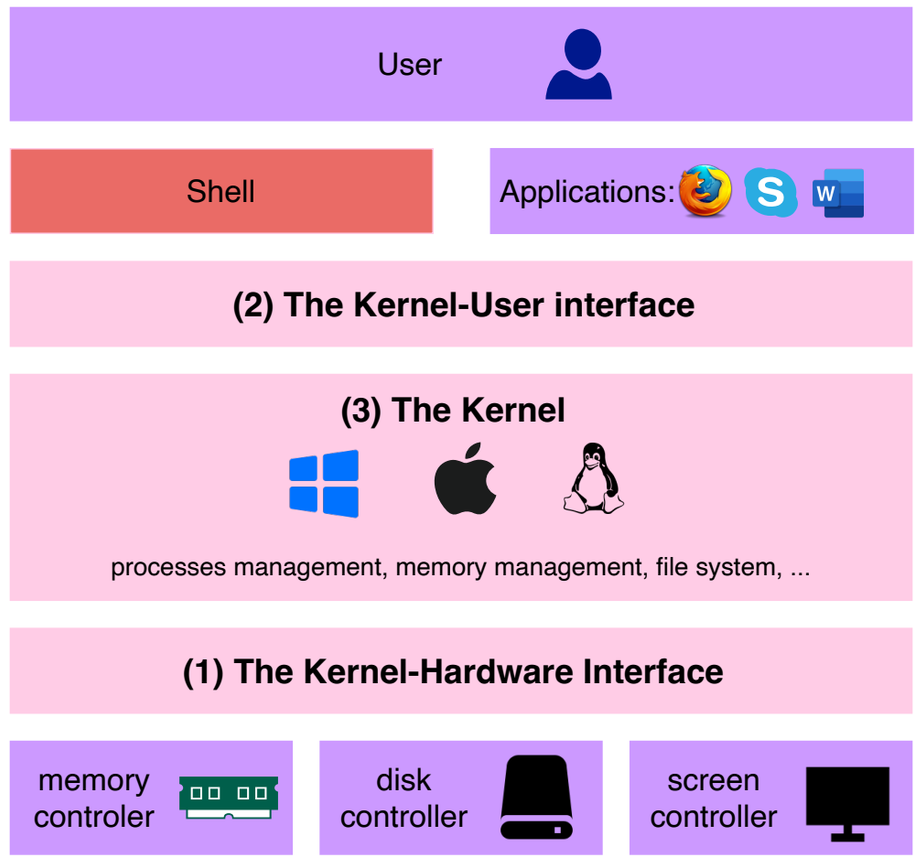
THE KERNEL-USER INTERFACE

- Provide an **interface** between hardware and software applications.
  ➠ a **simplified and unified interface**.
- Present above the physical machine, a more straightforward and more user-friendly **virtual machine**.
- Create **the illusion of real physical resources** (processor, memory, peripheral...).



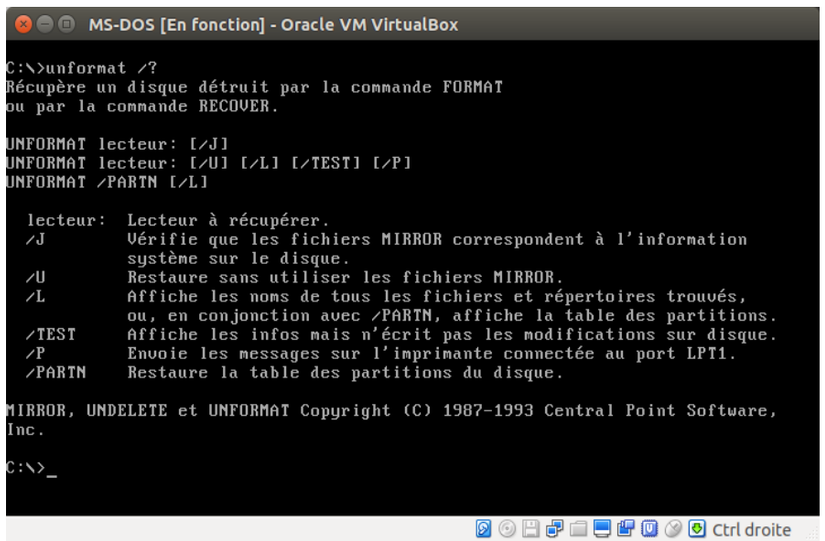
COMMAND INTERPRETER (SHELL)

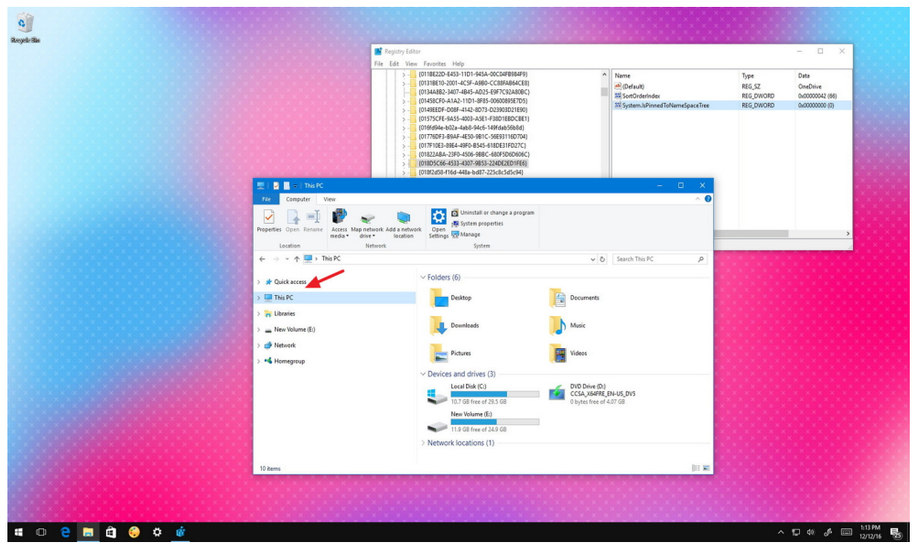
- **Command language**:
  the user interface with the operating system.
- **Command Interpreter**:
  Execute user commands by calling the proper **system routine**.



COMMAND INTERPRETER (SHELL)

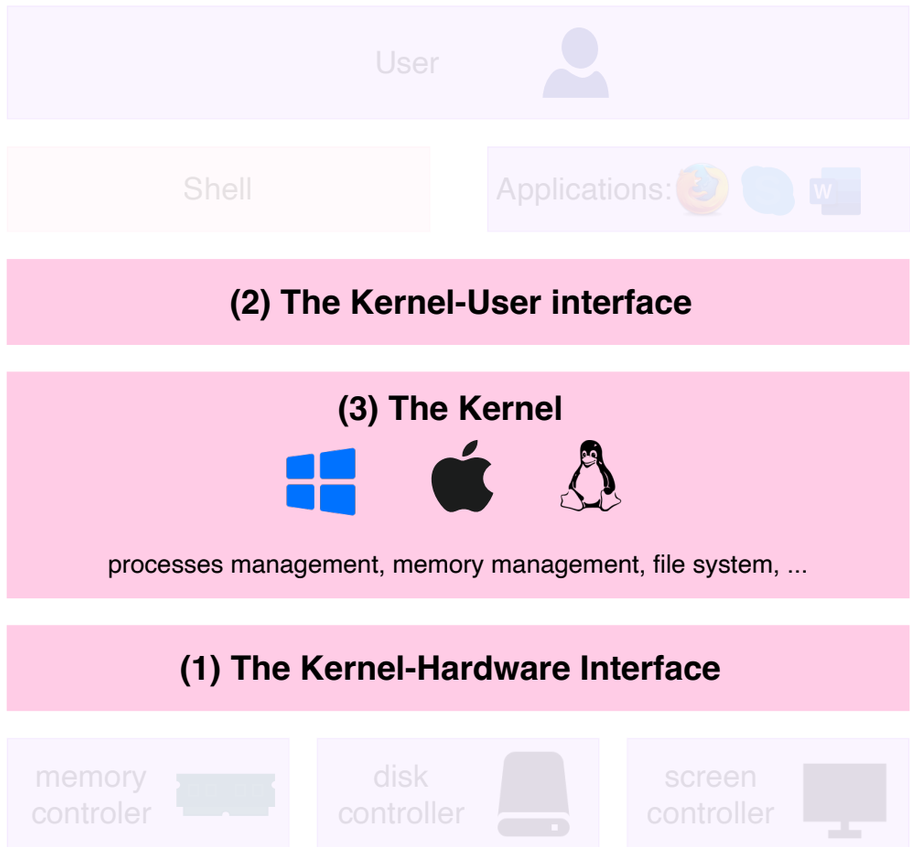
- Each operating system has its **own command language**:
  - **MSDOS/Unix** : console + keyboard
  - **Mac/Windows** : mouse + keyboard
  - **iOS/Android** : buttons + touch screen





THE KERNEL OF AN OPERATING SYSTEM

- **Process management**
  - scheduling, synchronization, ...
- **Memory management**
  - allocation, space management, ...
- **Secondary storage management**
  - file system, ...
- **Input/output management (I/O)**
  - controllers, drivers, ...
- **Security management**

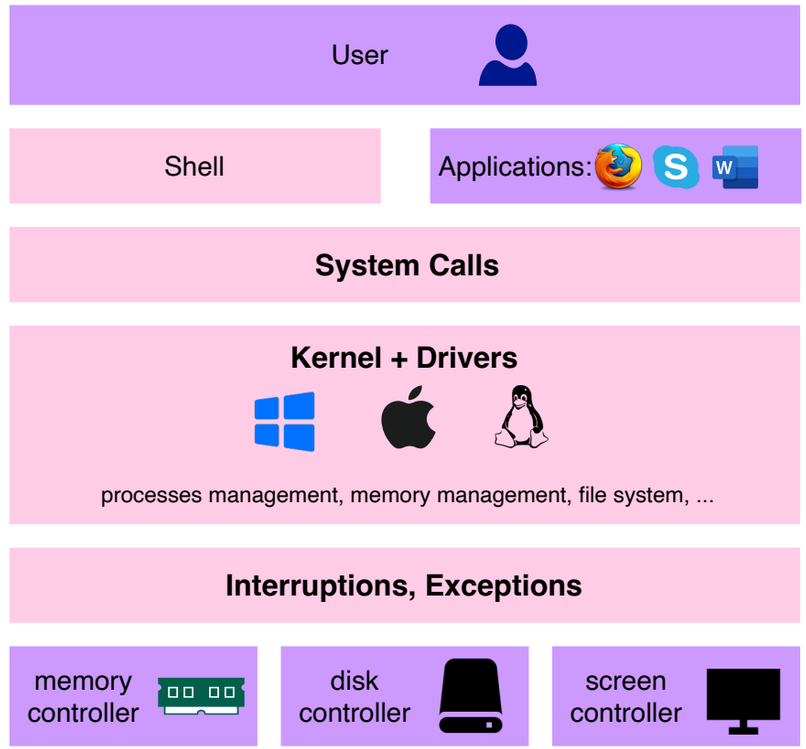


OUTLINE

Back to the outline - Back to the begin

GENERAL STRUCTURE OF AN OS

- **System Calls**: functions allowing user applications to request kernel services.
- **Kernel**: a set of programs providing general services.
- **Drivers**: programs allowing the kernel to interact with external devices.
- **Interruptions** : events produced by the hardware and triggering kernel services.
- **Exceptions** : events produced by the processor and triggering kernel services.



OUTLINE

- What is an Operating System?
- Operating System roles
- Structure of an Operating System
- Loading an Operating System

Back to the outline - Back to the begin

LOADING AN OS

- The **OS** is the first program executed when the computer is turned **ON** after booting (**boot**).
- The **boot** (**bootstrap**) defines the successive stages of startup.

THE BOOT STEPS

1. the **POST** test - **Power On Self Test**
   - after a start or a reset operation, the processor loads the **first instructions** located at the address FFFF0 from the **BIOS ROM** memory.
   - instructions for triggering a **BIOS** program that initializes and tests hardware functions.

2. loading the **MBR** - **Master Boot Record**
   - if the **POST** test succeeds, it will consult RAM CMOS to identify the **system disk** whose first sector is called **MBR**.
   - the **MBR** code tests the partition table to load the partition containing the boot sector with **IPL** - **Initial Program Load**.
     - ➠ the **IPL** loads the **OS** or **bootmanager** into RAM.
     - ➠ the **OS** is launched

THANK YOU

Back to the begin - Back to the outline