Computer Scince Brief introduztion

Loriano Storchi

loriano@storchi.org

http::://www.storchi.org/

SOFTWARE

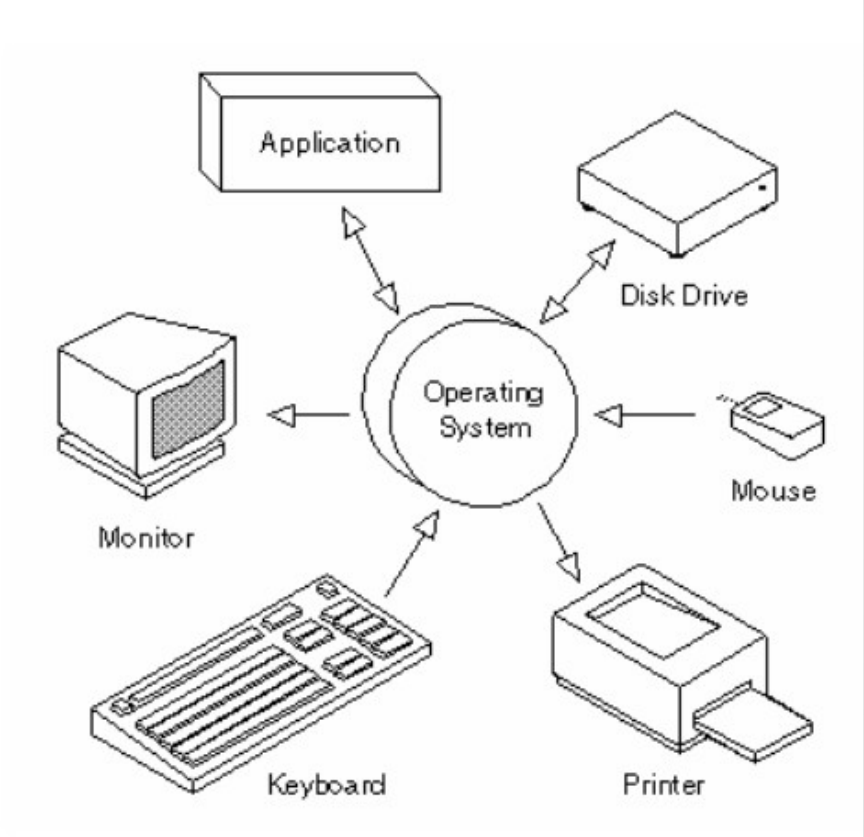
Software

- **Basic software:** dedicated to the management of the computer itself, for example the Operating System
- **Application software:** dedicated to the creation of specific applications, such as internet browsing, word processing or other.

OPERATING SYSTEMS

Operating system

- It is a low-level software that assists the user and high-level applications to interact with the hardware and data that the programs store on the computer
- **An OS performs basic operations, such as recognizing input from the keyboard, sending output to the display, keeping track of directories and files on the disk, and controlling peripherals such as printers**

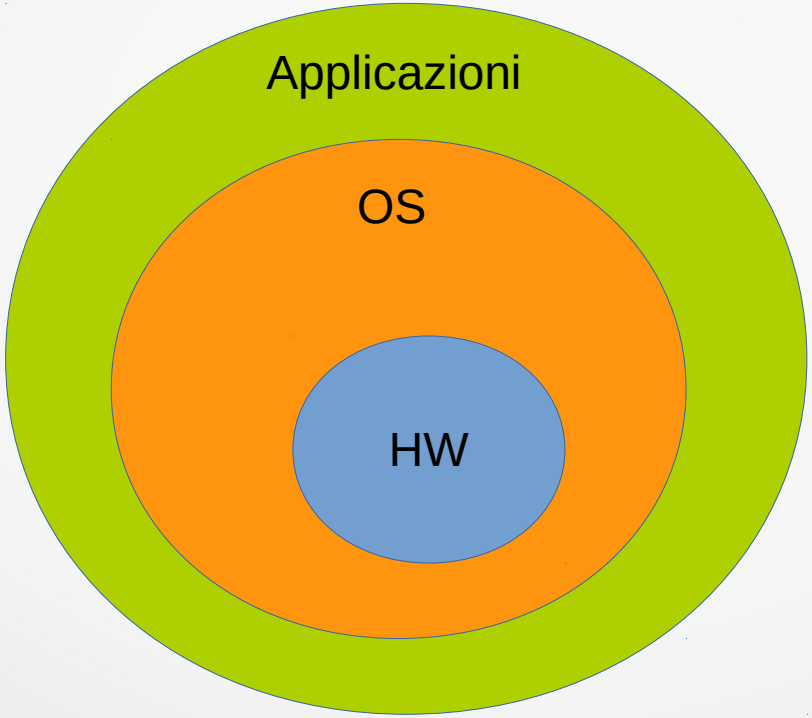


Operating system

- **Program execution:** OSes provide an environment where the user can run application programs without having to worry about **memory allocation or CPU scheduling**
- **I / O Operations**: Every application requires some input and produces some output. **The OS hides the details needed to manage this type of operation with respect to the underlying hardware**
- **Communication**: **There are situations in which two applications must communicate with each other, whether they are on the same computer (different processes), or on different computers.** The OS is responsible for managing this type of **inter-process communication**

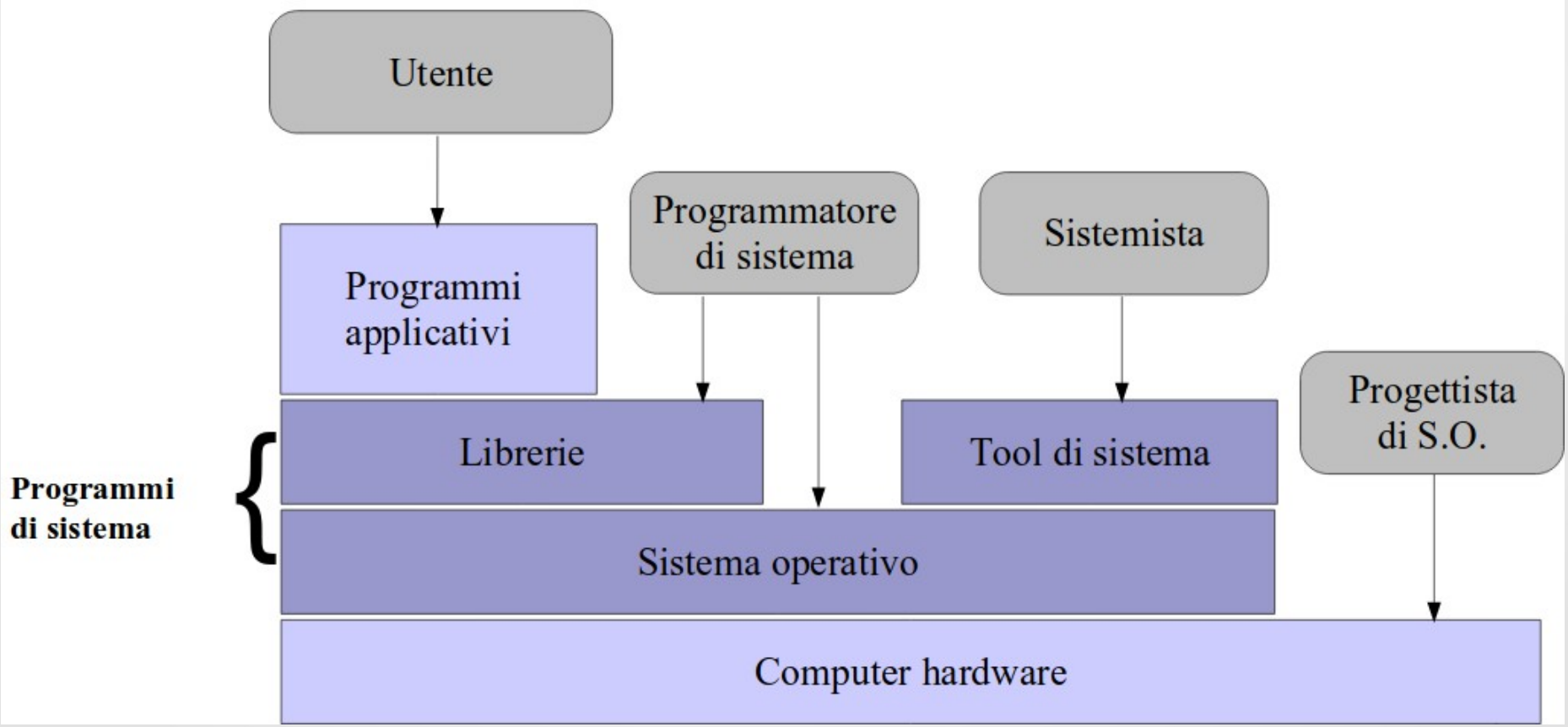
Operating Systems

They make it easier to write application programs that don't have to worry about specific hardware features.



Operating System

Layered view of hardware and software components, for example, provides the programmer with an easy-to-use API, hides hardware details



OPERATING SYSTEMS HISTORY AND MAIN FEATURES

Operating System: history

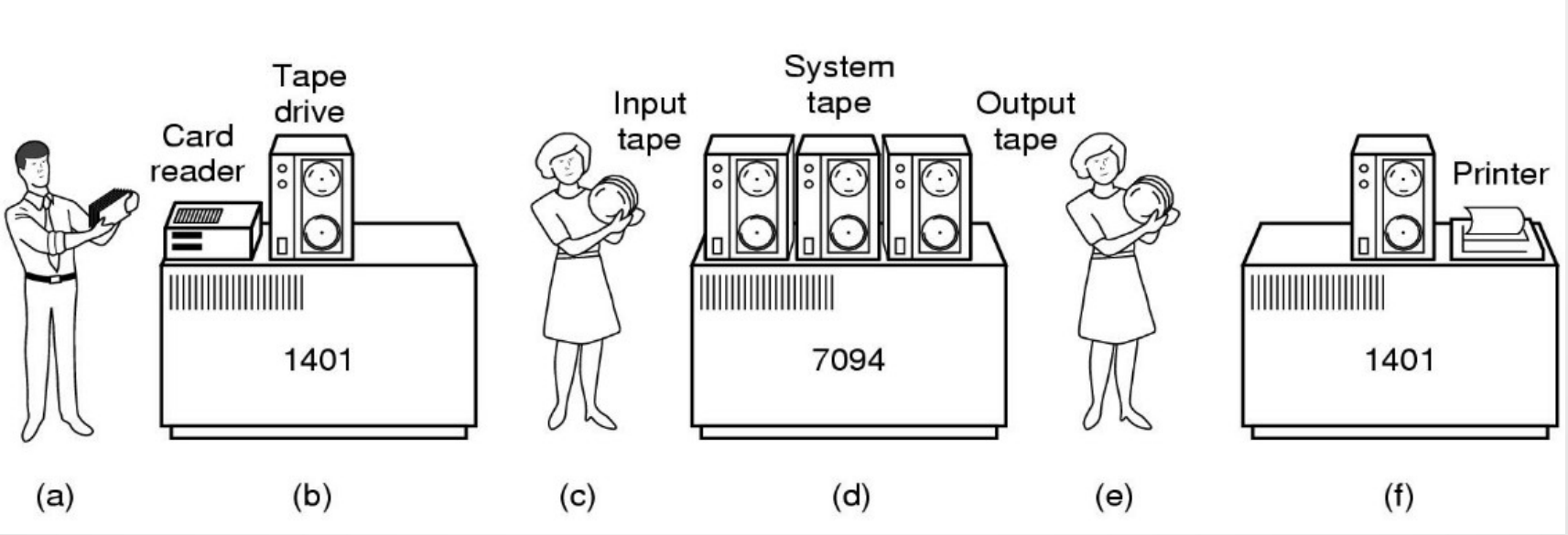
- **Babbage** (1792-1871) tries to build an analytical and **programmable mechanical machine** without an operating system
  - First programmer in history **Lady Ada Lovelace (daughter of Byron)**
- **Valve machines** (1944-1955), are designed, built and programmed by a single group of people
  - Programmed in machine language, used only for numerical calculation
  - There is no OS, there is no distinction between designer, programmer and user
  - The single user writes the program, loads it, loads the data, waits for the output and then moves on to the execution of the next program
  - Nobody imagines that computers will go beyond research labs

Operating System: history

- **Transistors** (1955-1965) can be built more reliable and cheaper machines
  - **They begin to be used for tasks other than basic numerical computation**
  - **Who builds the machine is different from who programs it and therefore uses it (user == programmer)**
  - The first **“high level” languages such as Assembly and FORTRAN** are introduced and punched cards are used
  - **First examples of OS, called Resident Monitors:**
    - Control on the machine is given to the monitor
    - Control is passed to the job being followed
    - Once the job is finished, control returns to the monitor

Operating System: history

- To avoid the downtime between the execution of one job and the other, tapes are introduced for storing jobs



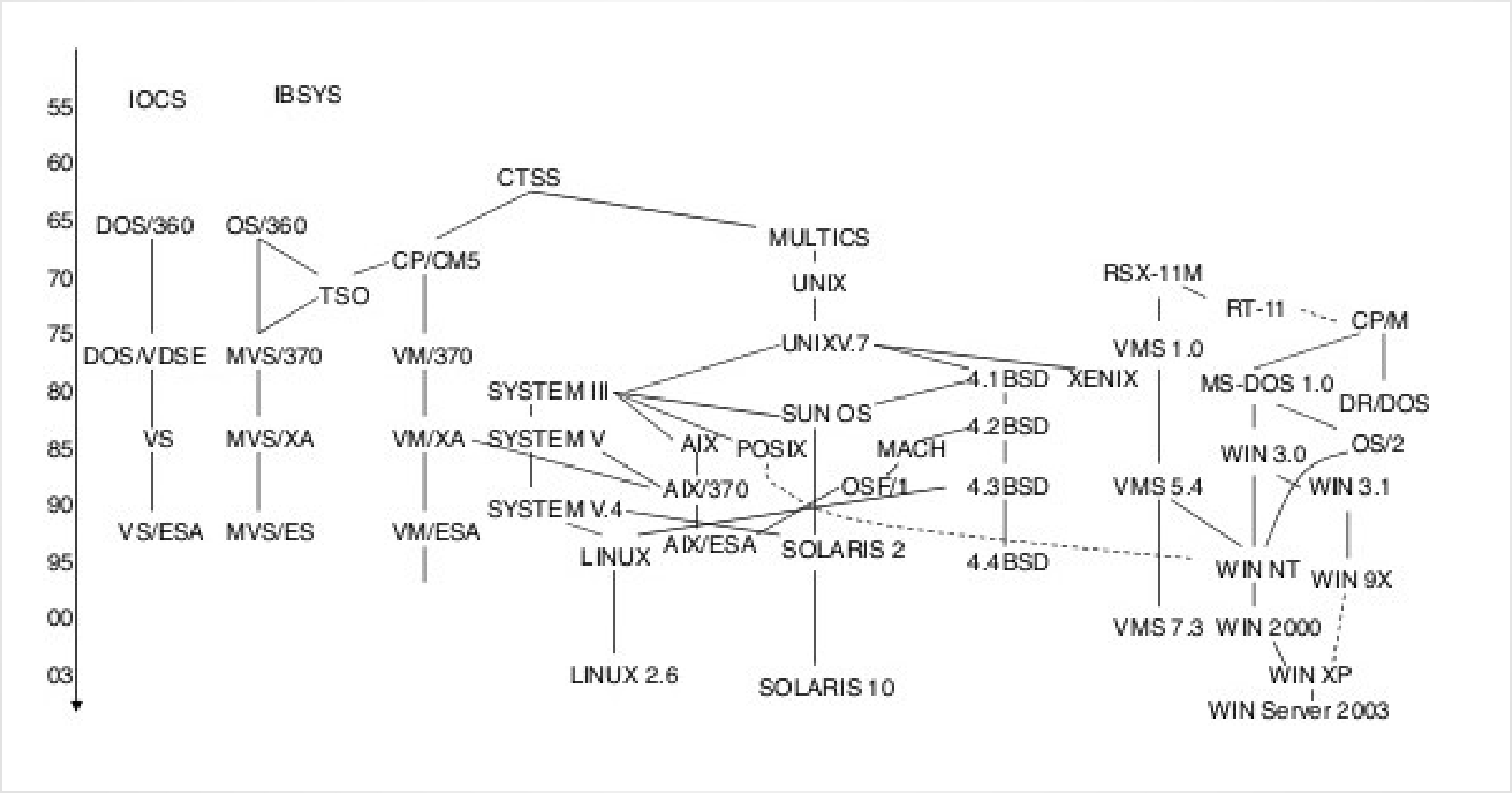
Operating System: history

- **Integrated circuits (1965 - 1980) the figure of the operator disappears as an interface to the computer:**
  - **Programming is mainly done using high-level languages such as C.**
  - Here come the text editors and terminals that allow you to operate
  - **Operating systems** with modern features appear:
    - **Interactive**
    - **Multi-programming**
    - **Time sharing**

Operating System: history

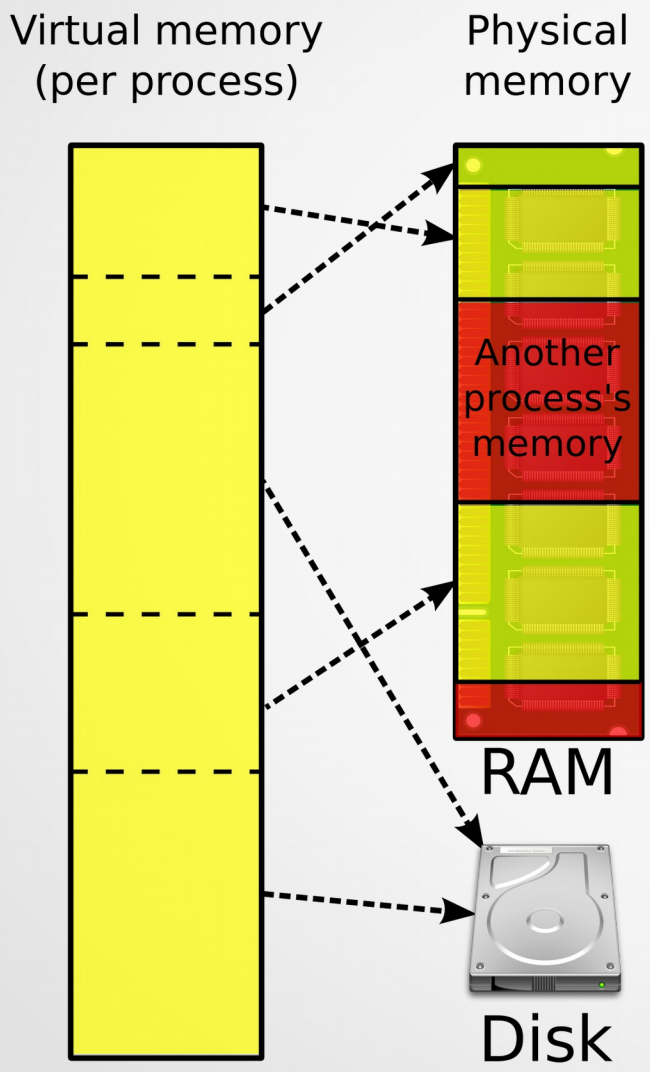
- **CTSS (Compatible Time-Sharing System)** 1965: the concept of multi-programming and the concept of time-sharing are introduced
- **Multics 1965**: introduces the concept of process
- **Unix 1970: derivative of Multics and CTSS initially developed at Bell-labs. Initially developed on two specific architectures, then developed in C (initially all in assembly).**

Operating System: history



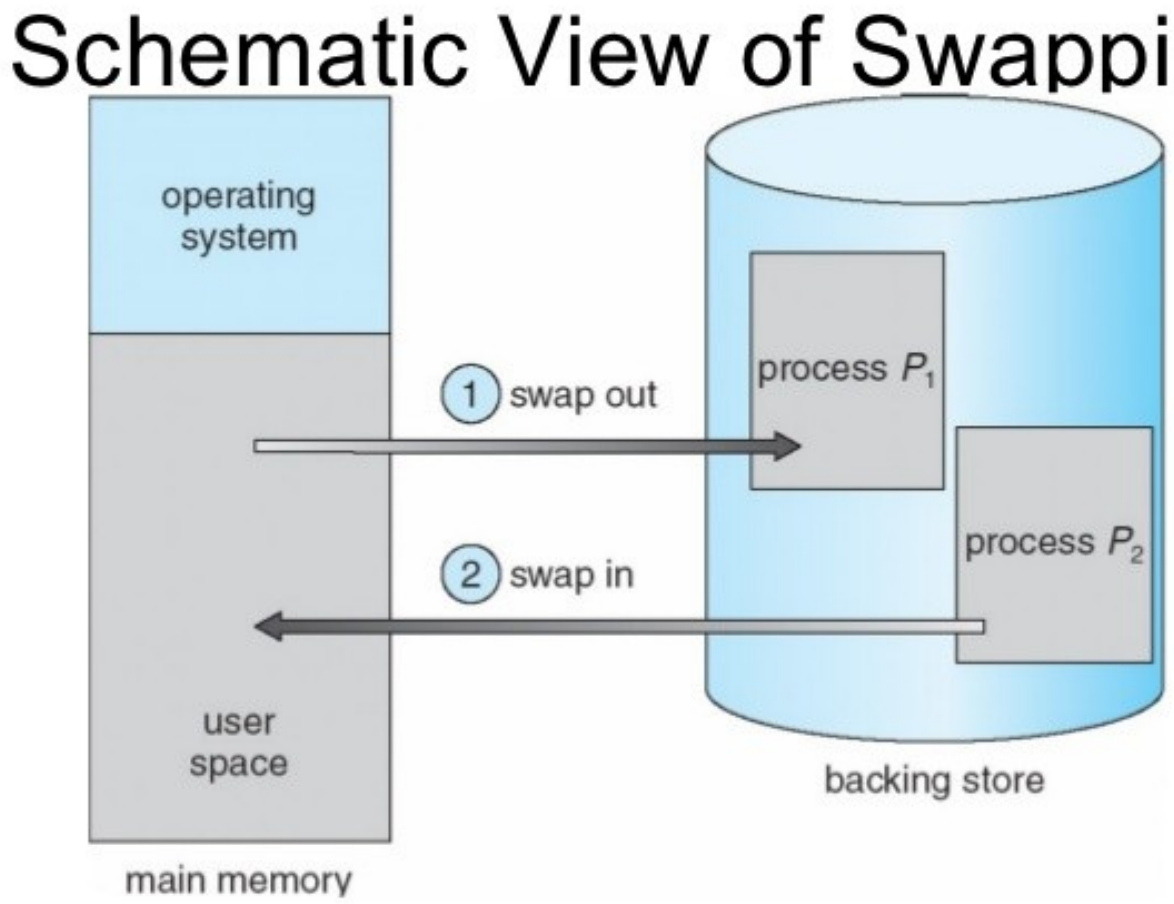
TWO SMALL INSIGHTS

Virtual Memeory



Virtual Memory is a memory management technique that provides an "idealized abstraction of the storage resources actually available on a given machine" which then "creates the illusion for users (processes, programs, applications) of a large memory space. "

Swapping


Schematic View of Swapping
operating system
user space
main memory
1 swap out
2 swap in
process P_1
process P_2
backing store

IN BRIEF FUNDAMENTAL COMPONENTS OF AN OPERATING SYSTEM

Operating System: process management

- **A process is basically a running program,** which therefore uses resources
- The OS must be able to:
  - **Create and terminate the processes themselves**
  - **Manage communication between the processes** themselves
  - **Suspend and reactivate processes**

Operating System: management of the main memory

- **The main memory is basically an array of individually addressable bytes, which can be shared between CPU and I / O devices**
- the OS must be able to:
  - For example, **keep track of which memory areas are used and by whom** (by which process)
  - For example, **decide for which processes to allocate a given memory area** when it is free
  - **Ultimately allocate to free up memory spac**e

Operating System: secondary memory management

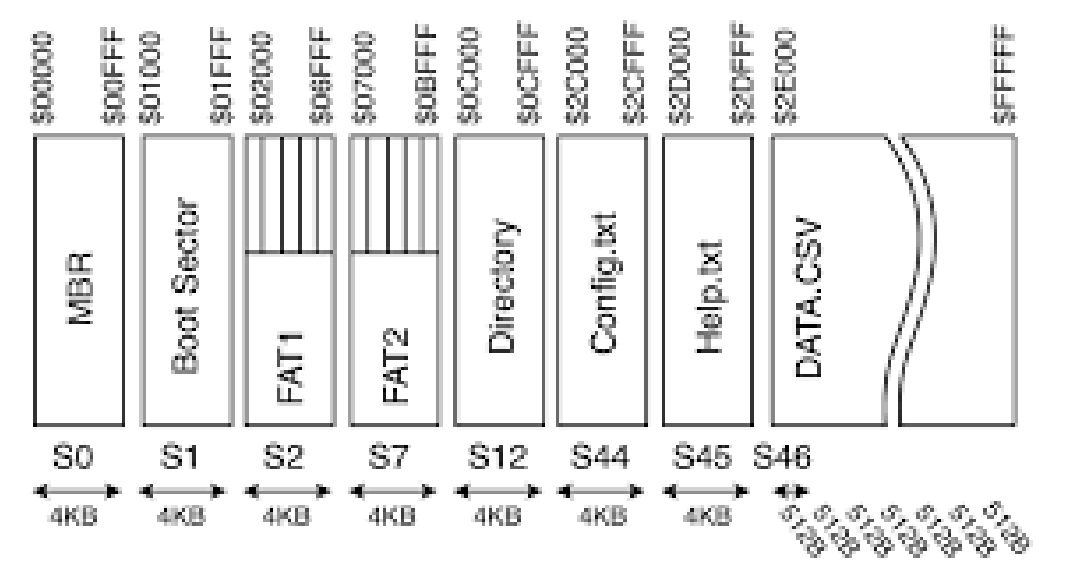
- We know the computers are equipped with a large non-volatile secondary memory (**degrees compared to the main memory which is volatile**)
- The OS must be able to manage the following activities:
  - **Allocate the required space and free it when it is no longer used**
  - **Manage the scheduling of access to devices** (for example hard-disk or CD / DVD or USB sticks)

Operating System: filesystem management

- A f**ile (archive) is the computer abstraction of the concept of information container**, regardless of the device on which it is stored
- **A filesystem is made up of many files (a directory contains references to other files**), the OS must be able to manage the following activities:
  - **Create and delete files and directories**, manipulate them
  - **Manage the filesystem** in the secondary storage
  - Examples, **ext3, ext4, NTFS, FAT32 ...**

Operating System: filesystem management

- **The filesystem: must manage the allocation of disk space,** for example maintaining an index of where the data relating to a given file are stored. Maintain the characteristics of a file such as access data, permissions, names etc.



Operating System: I / O management

- The OS must manage the I / O and therefore **the interaction with the various hardware devices**:
  - **A common interface for managing the various device drivers**
  - **Having different drivers (kernel modules)** for different devices, then **specific components** for the various hardware components of the system
  - A **system of buffering and caching** of information to and from the various devices

Operating System: protection

- **The OS must implement a software protection mechanism. Then manage and control the access of the various programs (processes) to shared system resources (for example memory)**
  - Distinguish between authorized use or not
  - Provide basic mechanisms for implementing protection

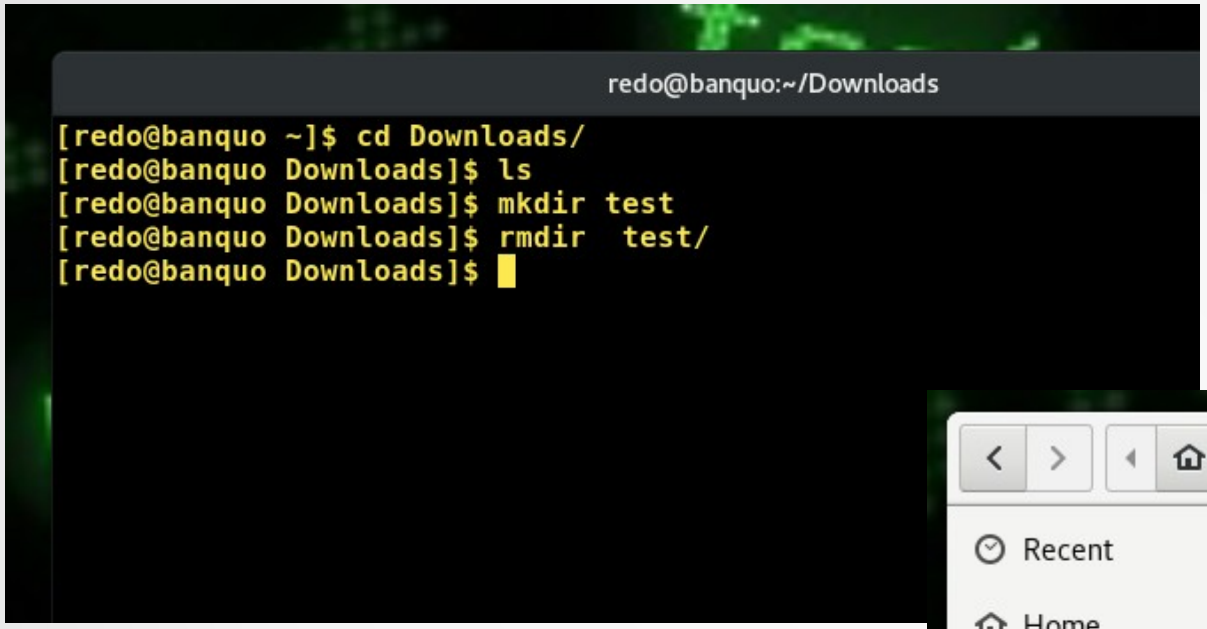
Operating System: networking

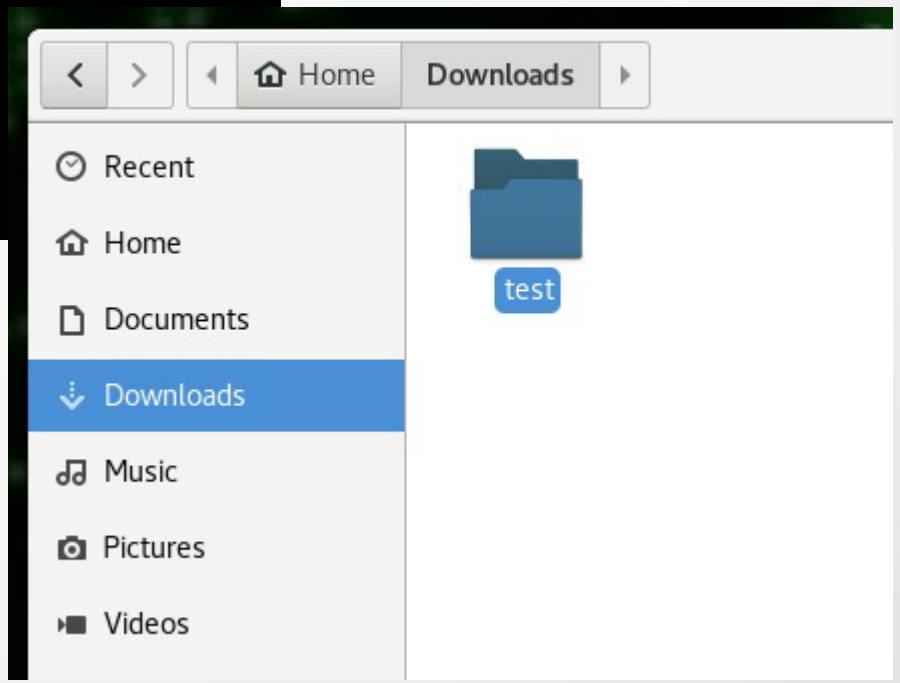
- **By now an essential component of an OS is networking, thus the ability to let two or more computers (processes) communicate and share resources.**
- An OS provides basic communication protocols such as **TCP / IP, UDP**
- In addition to **high-level communication services** such as shared file systems (SMB, NFS) and many others.

Operating System: command interpreter

- Operating systems provide a **user interface:**
  - **Start or end a program**
  - **Interact with the basic components of the operating system, such as the filesystem**
- We can have essentially two types:
  - **Graphics, and therefore icons and windows**
  - **Textual, command line**

Sistema Operativo: interprete dei comandi





CHANGE OF PERSPECTIVE

Operating System: programmer's perspective

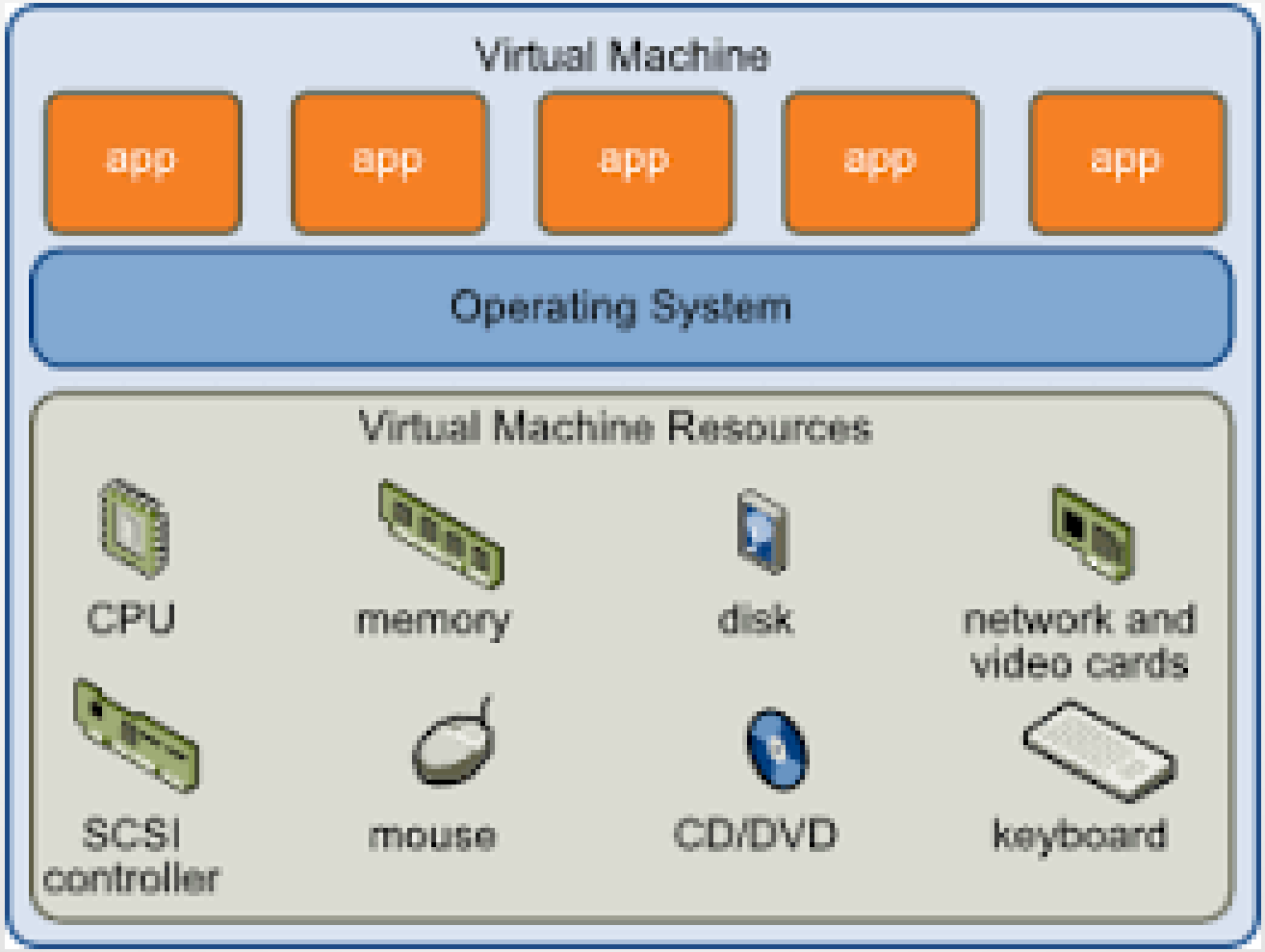
- The System Call for example:

UNIX	WIN32	UNIX	WIN32
fork	CreateProcess	mkdir	CreateDirectory
waitpid	WaitForSingleObject	rmdir	RemoveDirectory
execve	-	link	-
exit	ExitProcess	unlink	DeleteFile
open	CreateFile	mount	-
close	CloseHandle	umount	-
read	ReadFile	chdir	SetCurrentDirectory
write	WriteFile	chmod	-
lseek	SetFilePointer	kill	-
stat	GetFileAttributesEx	time	GetLocalTime

VMs, DOCKER e Cloud Computing

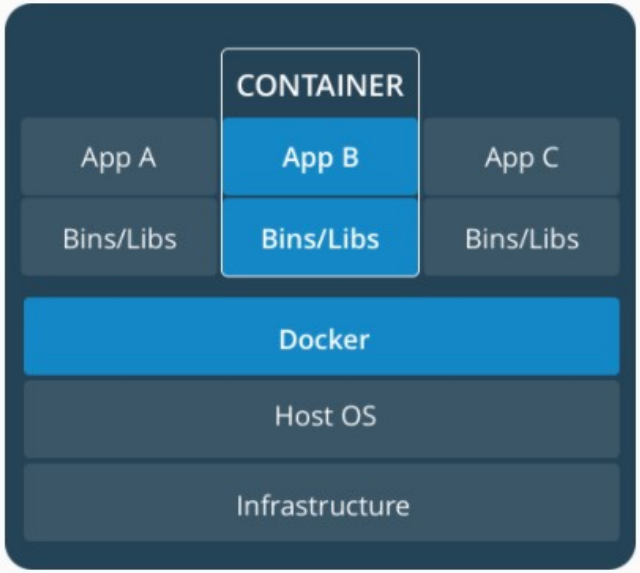
Cloud computing

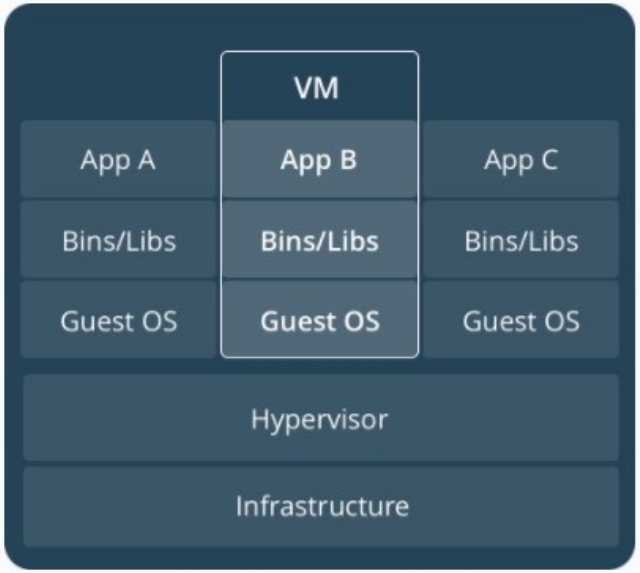
- Virtual Machine



Cloud computing

- Docker vs VM





CONTAINERS

Containers are an abstraction at the app layer that packages code and dependencies together. Multiple containers can run on the same machine and share the OS kernel with other containers, each running as isolated processes in user space. Containers take up less space than VMs (container images are typically tens of MBs in size), and start almost instantly.

VIRTUAL MACHINES

Virtual machines (VMs) are an abstraction of physical hardware turning one server into many servers. The hypervisor allows multiple VMs to run on a single machine. Each VM includes a full copy of an operating system, one or more apps, necessary binaries and libraries - taking up tens of GBs. VMs can also be slow to boot.

Cloud computing

- A virtual infrastructure