

# Operating Systems

## What is an Operating System

**Giorgio Grisetti**

`grisetti@diag.uniroma1.it`

Department of Computer Control and Management Engineering  
Sapienza University of Rome

# Operating System

Program that

- Controls the execution of other programs
- Provides to the running programs an abstraction of the underlying hardware

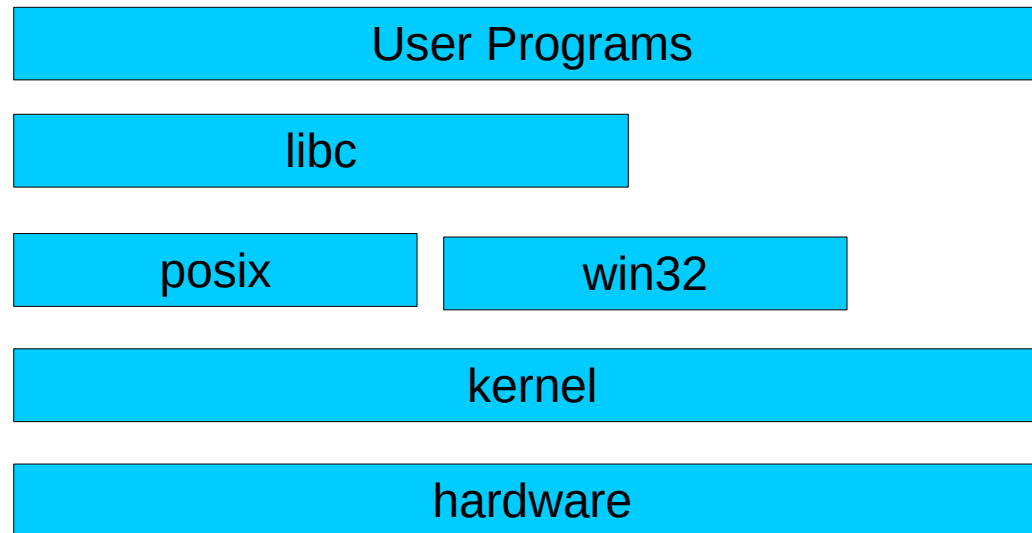
To offer these functionalities an OS

- Manages resources by preventing conflicts
- Deals with error conditions raised by programs or by hardware

From a user perspective an OS should

- Manage resources in an efficient manner
- Make it easy to access these resources
- Run the programs correctly

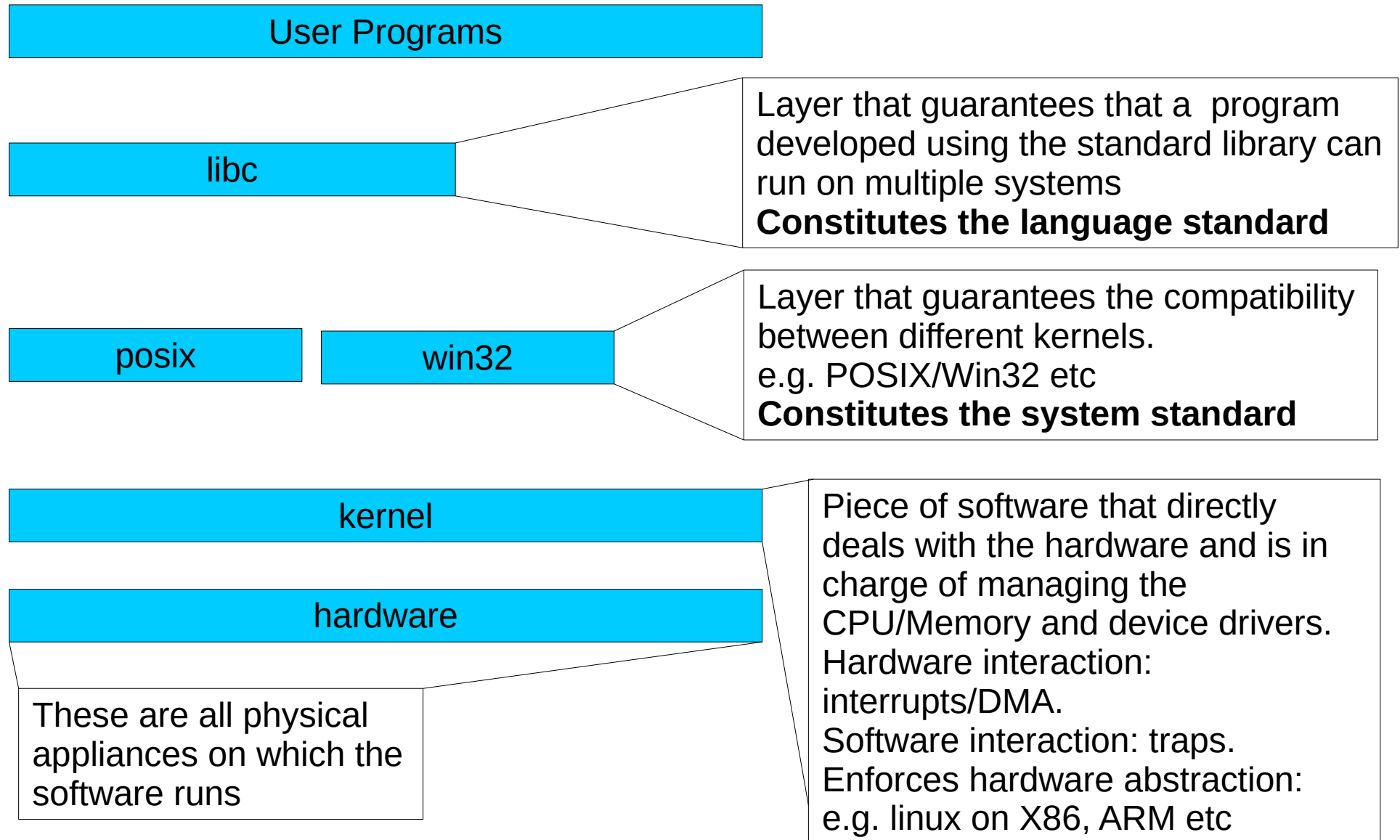
# Computing System



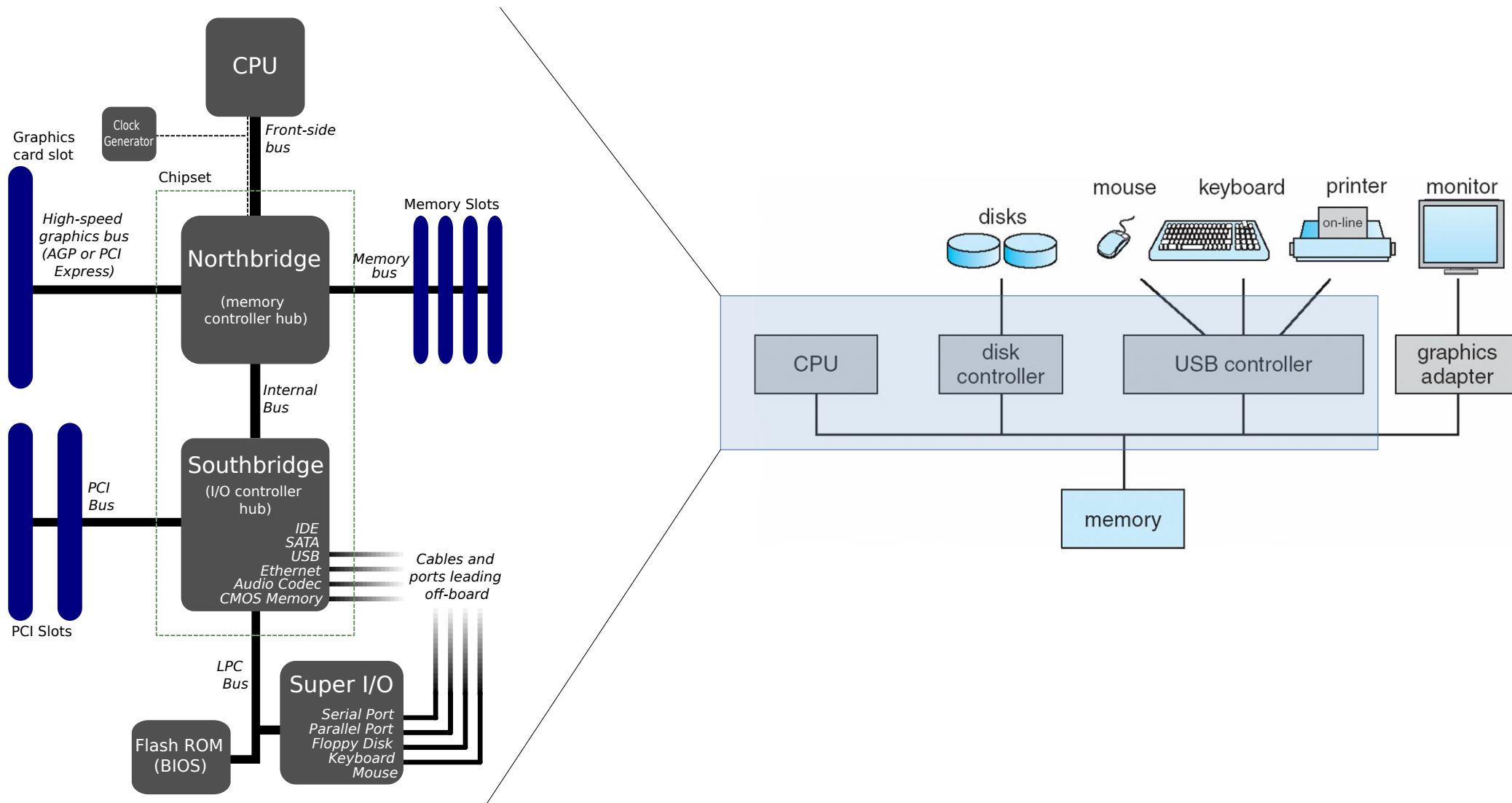
Computing systems are organized in layers

- Typically a higher layer can access to the functionalities of the lower layers only through the API of the layer immediayely below
- This ensures portability

# Standards



# Organization of a Computer



By Original: Gribeco at French Wikipedia  
Derivative work: Moxfyre at English Wikipedia - This file was derived from: Diagramme carte mère.png, CC BY-SA 3.0, <https://commons.wikimedia.org/w/index.php?curid=3789066>

# OS: Services

- Run programs
- Protected access to shared resources (I/O and memory)
- Error Handling
- Collects Statistics (e.g. CPU/Memory/Disk usage and so on)
- Typically an OS comes “packaged” with a bunch of tools: editors, compilers, debuggers, command interpreters, etc..
- These are not part of the kernel, but constitute the OS ecosystem.

# OS: Resources

CPU: Maximize usage

- Multiprogramming

Memory: Implement protection, Reduce access time, Virtualization

- Paging/Segmentation
- Virtual Memory

I/O: Simplify usage Minimize access time, Maximize throughput, Implement Permissions

- Buffering
- I/O Scheduling

# OS: Evolution (overview)

## **Sequential Computing (1945-1955)**

- A program at a time, no OS, machine language

## **Batch Computing(1955-1965)**

- A program after the other, monitor, JCL (Job Control Language), memory protection

## **Multiprogrammed Batch Computing (1965-1980)**

- Better use of the CPU through concurrent program execution
- Fancier Hardware (interrupt, DMA, MMU)
- Process Management

## ▪ **Time-sharing**

- Interactive jobs, CPU virtualization



# OS Evolution (1945-55)

## ▪ **Serial Computation**

- A program at a time
- The user went on the machine to mechanically load the software
- Data entry done through switches, punched card readers, punched tapes
- No Operating System (bare metal)
- Programs written in binary
- Errors reported through light bulbs
- When all goes well → The printer outputs the result
- Unhandy, but it is where we come from..

# OS Evolution (1955-65)

## Batch Computing

- A program after the other.
- The **batch monitor** (a memory resident program) loads and executes the programs one at a time, together with their input data, from an input device (punched card or tapes), printing the results.
- The user does not see the machine but rather handles a bunch of punched cards to an employee that loads them on an input device.
- To manage the execution of programs, between jobs the operator puts instructions for the monitor. These cards contain instructions for the monitor (**JCL - Job Control Language**).

# Evolution: Hardware

## Facts:

- CPU were getting faster, much faster than I/O
- RAM was getting cheaper (thus computers could have larger memories)
- I/O devices were getting smarter
  - Can do the task without CPU supervision through DMA
  - When an event that requires the CPU intervention occurs, they raise an interrupt.

## Issues:

- How to use the CPU while waiting for I/O?

# OS Evolution (1965...)

## Multiprogramming

- A pool of programs is loaded on the machine, sequentially
- When a program requests an I/O operation to the **OS**, the OS assigns the CPU to another process that is ready for execution
- Upon termination of the I/O, a peripheral notifies the **OS** that the operation finished. The OS in turn wakes up the process that requested the operation and puts it in an execution queue.
- Multiprogramming improves
  - CPU usage (cpu time/total time)
  - throughput (processes completed/time).

# Evolution: Hardware

## Facts:

- First interactive I/O systems allow the users to enter directly program/data in the system
- Users want to sit in front of a terminal and shorten the latency in the edit/compile/run loop
- Computers still expensive

# OS Evolution (1965...)

## Time Sharing

- Let multiple users run concurrently their programs on a computer in an interactive fashion
- Single CPU is shared between multiple users/processes with a round robin mechanism on top of the I/O eviction typical for batch systems
- System not controlled by JCL, but by a Terminal/Shell program

## Different performance metrics:

- Response Time: time spent when the process is in the ready state and gets the CPU for the first time

# Evolution: Hardware

## Facts:

- First home computers become available. Multi-user capabilities not initially required
- Rather limited hardware and little RAM (~1KB)

# OS Evolution (1975...)

## OS for Personal computers

- **Single user/single task, ROM resident**

- Apple II OS, Commodore Kernal

They provide a BASIC interpreter that acts as a SHELL

- **OS loaded from disk** (supports hardware from different vendors)

- CP/M: supports multiple computers
  - MS-DOS built on CP/M concepts

- **GUI**

- Apple Lisa
  - Mac-OS
  - AmigaOS (adds preemptive multitasking)
  - Atari TOS
  - Windows 3x
  - RISC-OS (acorn arm)
  - ....



# Evolution: Hardware

## Facts:

- CPUs get more and more powerful. They offer hardware support for memory protection and dual mode (Privileged/User-mode).
- Memory is ridiculously cheap (compared to the ancient times)
- The market is converging to standard ABI for personal computers
  - X86
  - ARM
  - PowerPC (later become a niche market)
- Discrete Video Cards allow for powerful graphics
- Standard buses, support multiple CPUs

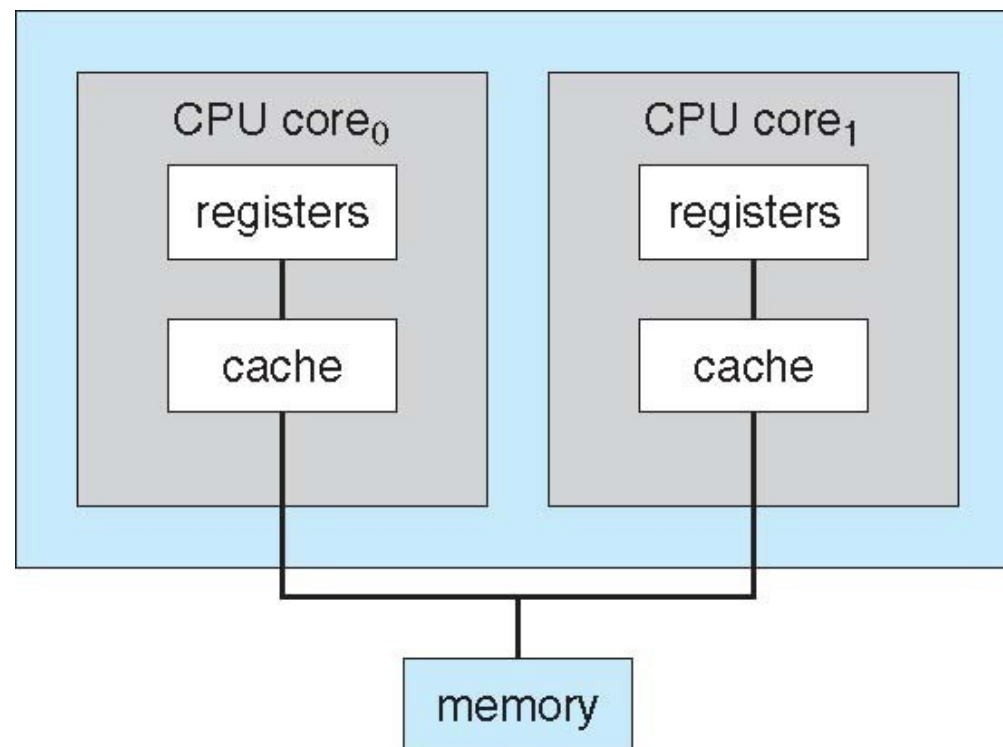
# OS Evolution

- Multi-tasking
  - Windows 95
- Multi-user
  - Windows NT (XP,7,8,9,10...)
  - Unix (Darwin)
  - Linux (Ubuntu/Android)

# Evolution: Hardware

## Facts:

- CPU clock cannot be pimped any longer
- Increase performances by packaging multiple CPUs in a single chip.
- LAN are common



# OS Evolution

- Symmetric Multi Processing
  - Spreads the load over multiple CPUs
  - Preemptive/Non Preemptive Kernel
- Clusters
  - Beowulf
  - Grids

# Modern Operating Systems

## Components

- Process Manager
- Memory Manager
- I/O Managers
  - Off-Core Memory Manager (Disk)
  - Network Card Manager
  - Video Card Manager
  - ...
- File Manager
- Network Manager
- Login Manager
- Command Interpreter

# Modern Operating Systems

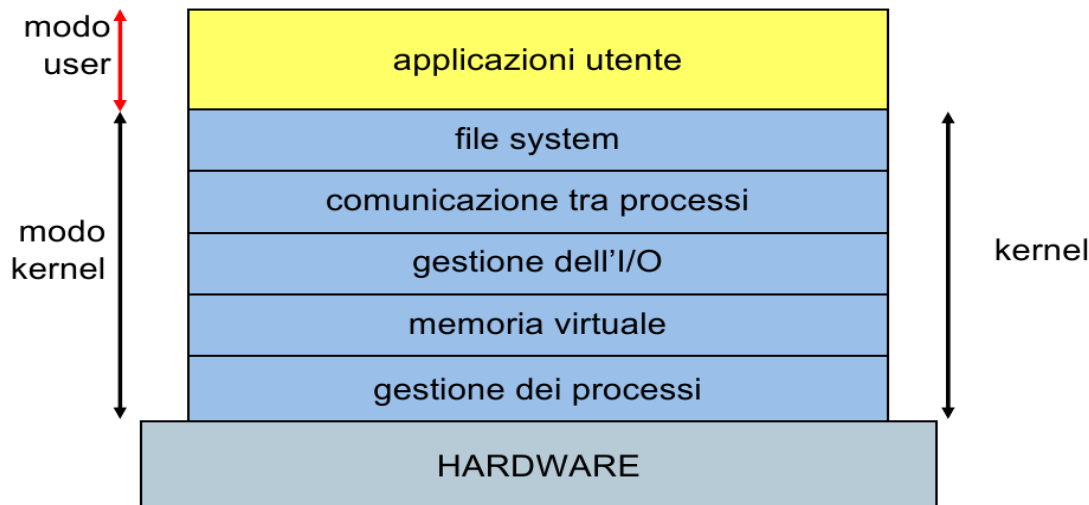
## Services

- Executes Programs
- Provides I/O abstractions
- Provides access to a file system
- Allows for inter process communication/synchronization
- Manages resources
  - RAM
  - CPU
  - Disk
  - Devices
- Offers primitives to enforce protection
- Manages exceptions
- Collects statistics about program/resources usage

# Modern Kernel Structures

## Monolithic

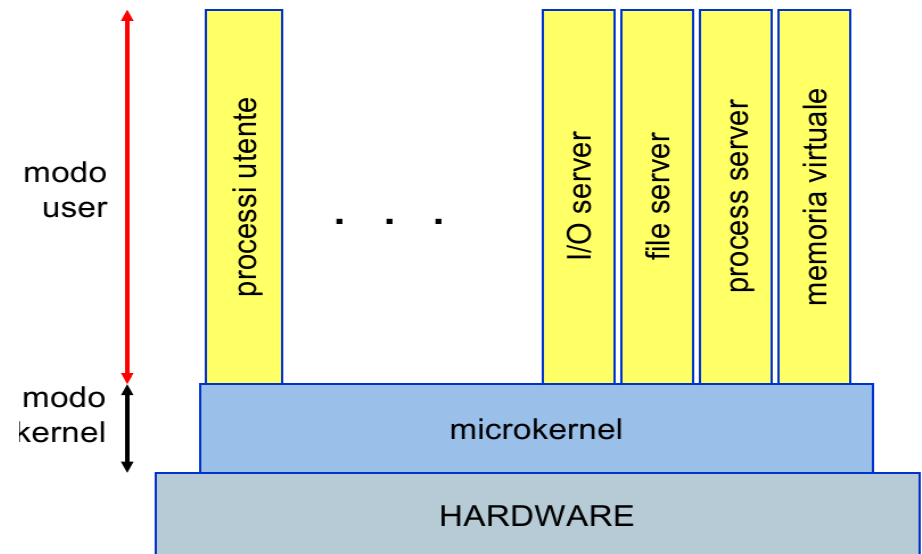
Kernel is one big program that manages all interaction between hardware and programs



## Microkernel

Kernel is composed by

- a light core managing memory and processes and
- a bunch of modules running in less privileged mode that handle all the rest.



# Linux Architecture

