1 Overview

1.1 What Operating Systems Do

$$\left.\begin{array}{r}\text{hardware}\\ \text{software}\\ \text{data}\end{array}\right\}\text{Computer System}\begin{cases}\text{Hardware} & \begin{cases}\text{CPU (Central Processing Unit)}\\ \text{Memory}\\ \text{I/O (Input/Output) Devices}\end{cases}\\ \text{Application Programs} & \text{defines how resources are used}\\ \textbf{Operating System} & \text{controls and coordinates hardware}\end{cases}$$

Components of a Modern Computer System

An **operating system** is a software that

- manages and controls a computer's hardware;
- coordinates and optimizes utilization of hardware;
- provides a basis for application programs.

An operating system is similar to a *government*, who performs no useful function, but provides an environment within which other programs can do useful work.

$$\text{User View}\begin{cases}\textbf{Ease of Use} & \text{One user monopolizes all resources.}\\ \textbf{Maximizing Utilization} & \text{Multiple users utilize resources simutaneously.}\\ \textbf{Compromise} & \text{OS assigns and shares resources to different users.}\\ \textbf{Little or no User View} & \end{cases}$$

$$\text{System View}\begin{cases}\textbf{Resource Allocator} & \text{processes numerous and possibly conflicting requests.}\\ \textbf{Control Program} & \text{manages the execution of user programs}\end{cases}$$

1.2 Computer-System Organization

1.2.1 Computer-System Operation

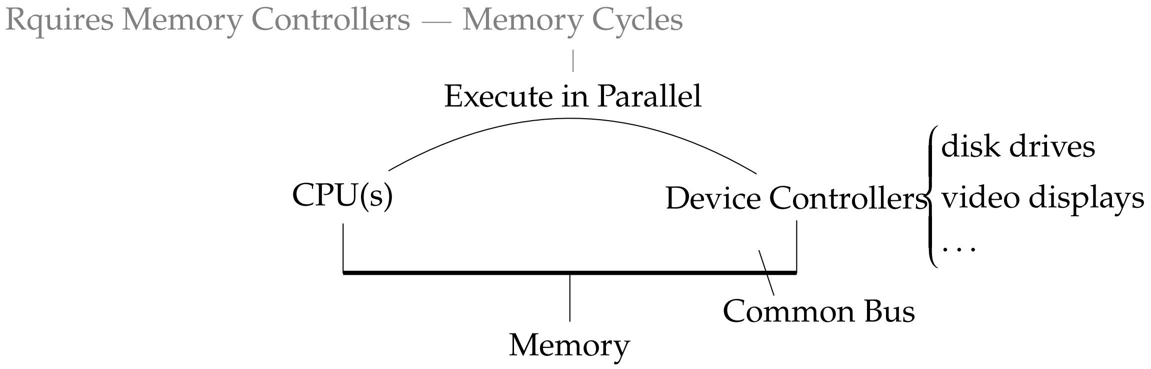


Figure: Components of Modern General-Purpose Computer

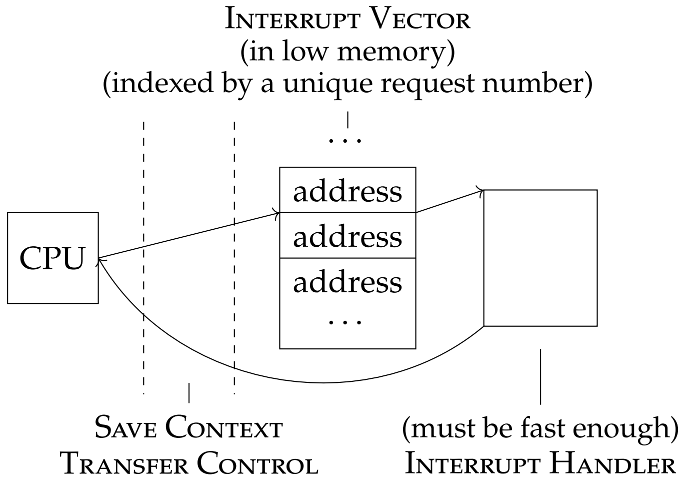
For a computer to start running, it

1. runs **bootstamp program**, which
   - tends to be simple.
   - is stored in **read-only memory** (ROM), or Electrically Erasable Programmable ROM

- initializes all aspects of the OS, from CPU registers to device controllers to memory.
- locates the operating system and loads it to memory (⇐ know how to load and start)

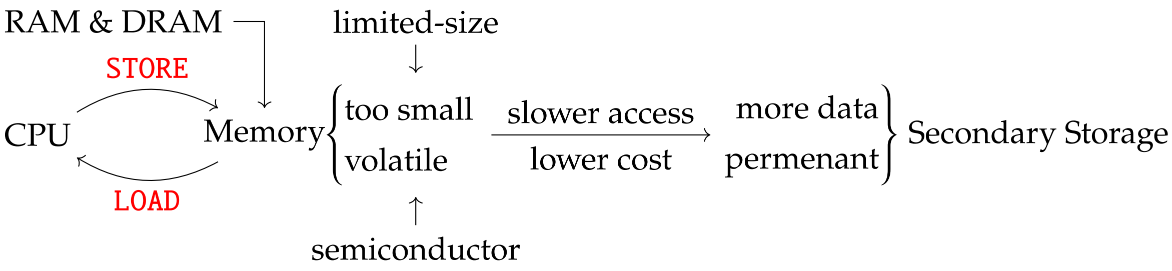
2. loads service programs (**system daemons**: outside kernel, loaded at boot, runs entire time)

The event is signaled by an **interrupt** from either hardware or software.



1.2.2 Storage Structure

All forms of memory provide **an array of bytes**. Each byte has its own address.



Other types of memory:

- Cache: stores data to reduce time cost of further request for that data.
- ROM: cannot be changed ⇒ ONLY static programs (e.g., bootstamp program).
- EEPROM: change is slow ⇒ mostly static programs (e.g., factory-installed programs).

Hierarchy	Magnitude	Volatility	Implementation
Registers	bytes	√	MOSFET
Cache	16KB ~ 50MB	√	MOSFET
Main Memory	8GB ~ 64GB	√	MOSFET
Solid-state Disks	≥ 100 GB	○ / ×	Flash Memory
Magnetic Disks	≥ 500 GB	×	Magnetic Poles
Optical Disks		×	Pits & Lands
Magnetic Tapes	TB	×	Magnetic Memory

Table: Information and Hierarchy of Storage
(higher in hierarchy ⟹ larger capacity, more expensive, and faster)

1.2.3 I/O Structure

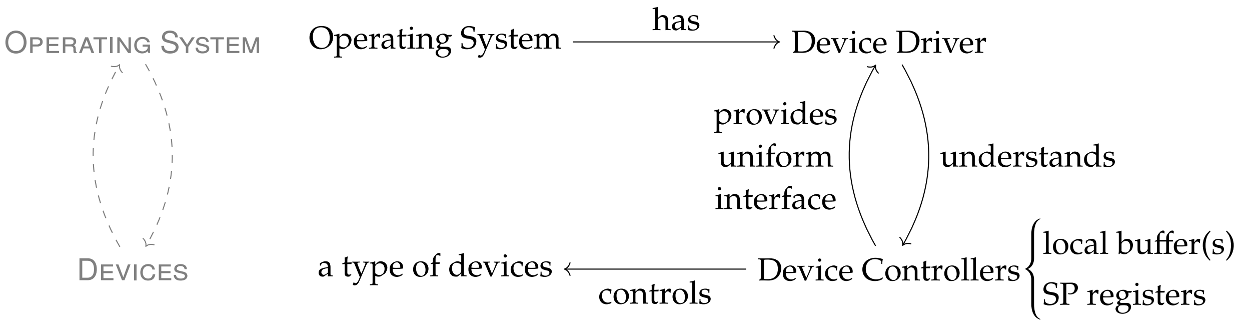


Figure: I/O Structure

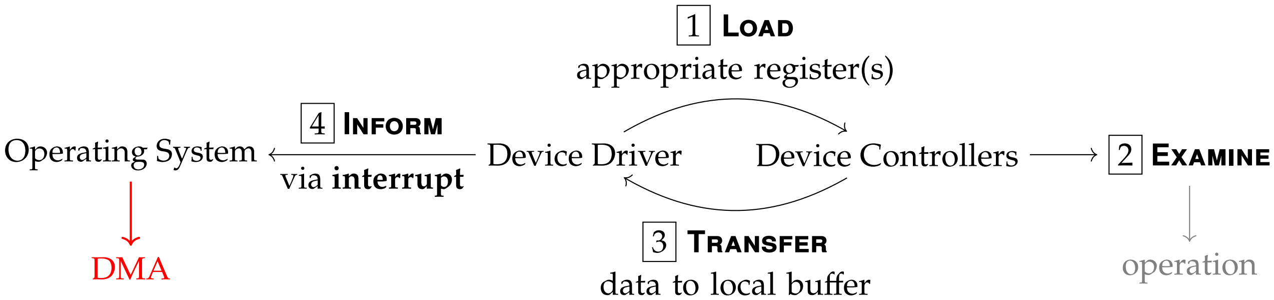


Figure: I/O Operation

This form creates overhead when bulk and/or frequent data movement, like disk and keyboard. By **direct memory access** (DMA), the driver fires only one interrupt and transfers a block of data from its local buffer the main memory, without CPU's intervention.

1.3 Computer-System Architecture

1.3.1 Single-Processor Systems

S.P.S.	1 GP CPU & GP registers	GP instruction set	runs user program
	SP registers (from processors)	limited SP instruction set	is managed by OS only autonomously

1.3.2 Multiprocessor Systems

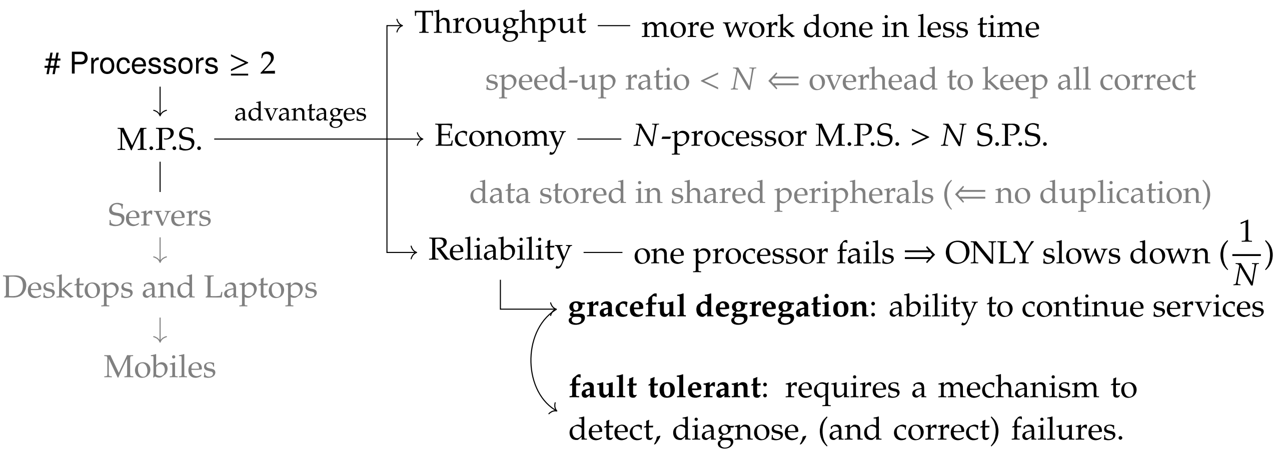

# Processors ≥ 2
M.P.S.
advantages
Servers
Desktops and Laptops
Mobiles
Throughput — more work done in less time
speed-up ratio < N ⇐ overhead to keep all correct
Economy — N-processor M.P.S. > N S.P.S.
data stored in shared peripherals (⇐ no duplication)
Reliability — one processor fails ⇒ ONLY slows down (1/N)
graceful degregation: ability to continue services
fault tolerant: requires a mechanism to detect, diagnose, (and correct) failures.

Figure: Multiprocessor System Concepts

- M.P.S.
  - asymmetric MP
    - boss controls the system
    - worker looks for boss instruction or haspredefined tasks
  - (different implementation in hardware and/or software)
  - symmetric MP
    - peer-to-peer relationships
    - Each CPU owns registers and caches; CPU shares memory.
    - ensures I/O data to reach the appropriate processor.
    - one is overloaded, some others are idle ⇒ dynamically shares

Types of Multiprocessor System

Multicore: includes multiple computing cores (owns registers and local cache) on a single chips; on-chip communication is faster and uses significantly less power than between-chip communication.

1.3.3 Clustered Systems

A **clustred sytem** are composed of two or more individual systems, or nodes, joined together.

- **Clustering**
  - provides
    - high-availability service (one fails ⇒ continues) ⇐ Redundancy
    - high-performance computing environment ⇐ Parallelization
  - type
    - assymmetric
      - hot-standby does nothing, monitors active server
        fails ⇒ becomes another active server
      - other runs applications
    - symmetric: ≥2 machines run and monitor each other.

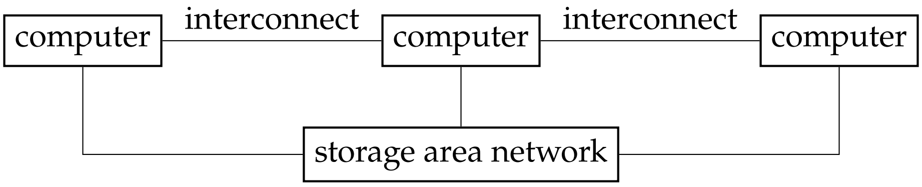


Figure: General structure of a clustered system

Parallelization: divides a program into separate components to run on individual computers in the cluster ⟹ much greater computational power (significantly greater than multiple single-processor systems or even symmetric multiprocessor systems).
Parallel clusters: multiple hosts to access data on shared storage ⇒ access control and locks

1.4 Operating-System Structure

$$\underset{\text{(jobs in disk (job pool))}}{J_D} \supseteq \underset{\text{(jobs in memory)}}{J_M} \ni \underset{\text{(the executing job)}}{j_E}$$

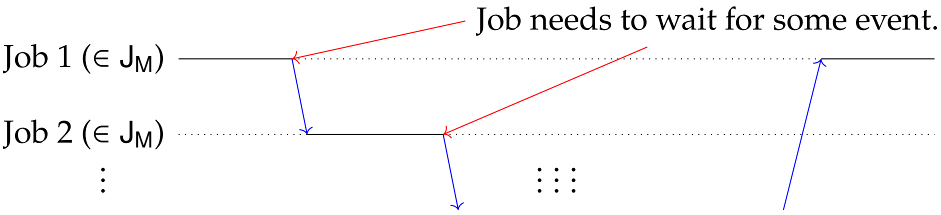


Figure: Multiprogramming Idea Interpretation

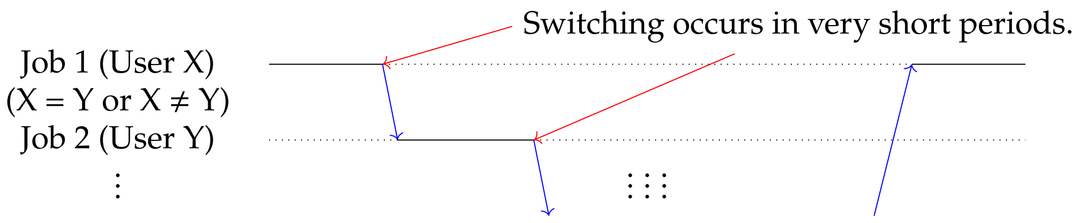


Figure: Multitasking Idea Interpretation
(Multitasking requires an **interactive** system, with short device responsive time.)

Job scheduling: OS needs to choose ready jobs from disk because memory is too small.
CPU scheduling: OS needs to choose a job in memory when multiple jobs are ready to run.
Swapping: processes are swapped in and out of main memory.

1.5 Operating-System Operations

The OS must ensure that incorrect or malicious behaviors in a program cannot cause other programs and the OS to execute incorrectly.

1.5.1 Dual-Mode and Multimode Operation

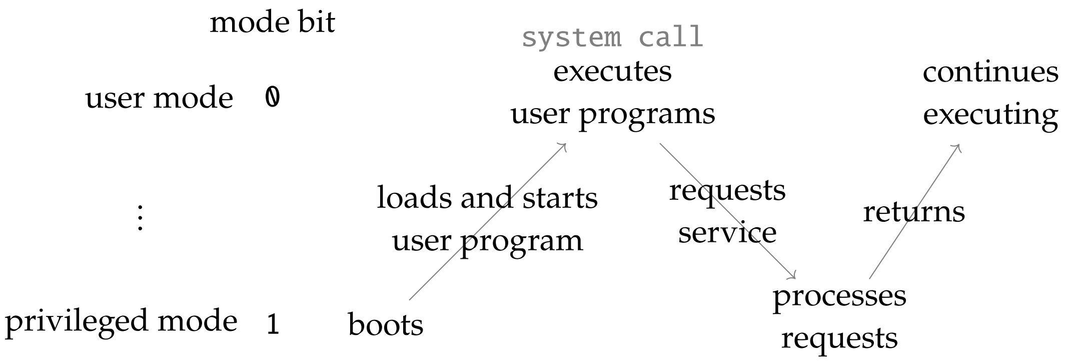


Figure: Dual-Mode Interpretation

Some instructions that might cause harm are called privileged instructions. The hardware only supports privileged instructions and some other operations, like I/O controls, interrupt management, etc., in privileged mode. Some modern architectures support more than two modes.

The lack of hardware-supported dual mode causes serious shortcomings. For example, a user program can overwrite the OS, which might make the system crash or behave oddly.

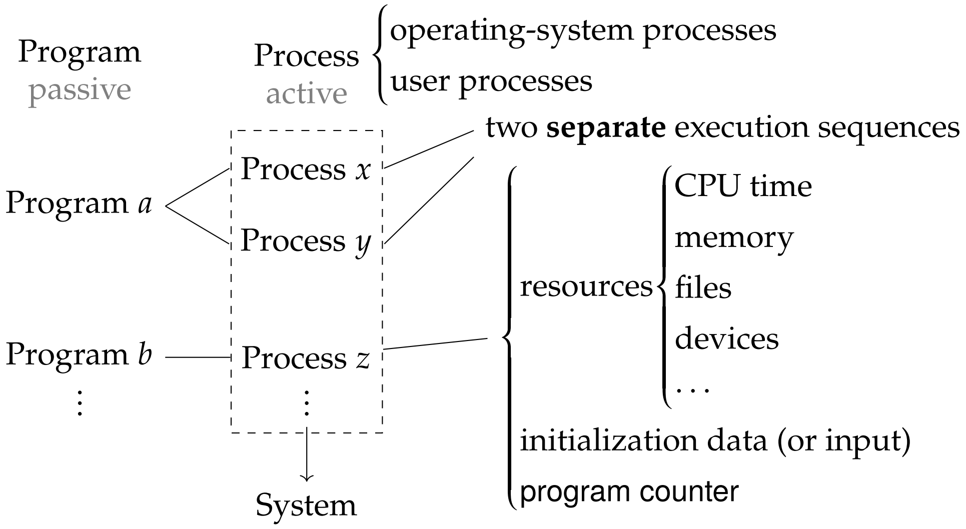
If a privilege violation is occurred, these errors are handled by the OS. The OS must terminate the program abnormally, probably with an appropriate error message.

1.5.2 Timer

Timer is to detect infinite loops, failed returns, user programs running too long, etc. It can interrupt the computer after a specified period. It is usually implemented by a fixed-rate clock and a counter set by the OS. The OS might terminate the program or give more time. Instructions modifying the timer are privileged.

1.6 Process Management

A process is a program **in execution**.



1.7 Memory Management