1 Overview

1.1 What Operating Systems Do

- hardware, software, data } Computer System
  - Hardware
    - CPU (Central Processing Unit)
    - Memory
    - I/O (Input/Output) Devices
  - Application Programs — defines how resources are used
  - **Operating System** — controls and coordinates hardware

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.

- User View
  - **Ease of Use** — One user monopolizes all resources.
  - **Maximizing Utilization** — Multiple users utilize resources simutaneously.
  - **Compromise** — OS assigns and shares resources to different users.
  - **Little or no User View**
- System View
  - **Resource Allocator** — processes numerous and possibly conflicting requests.
  - **Control Program** — manages the execution of user programs

1.2 Computer-System Organization

1.2.1 Computer-System Operation

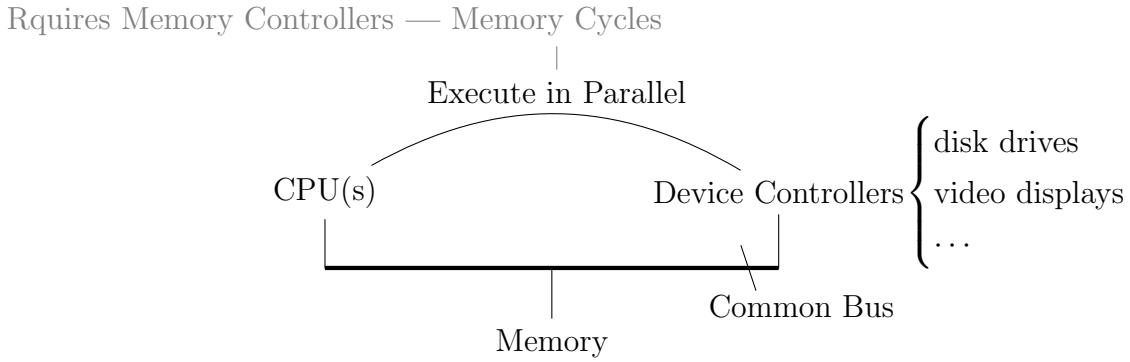


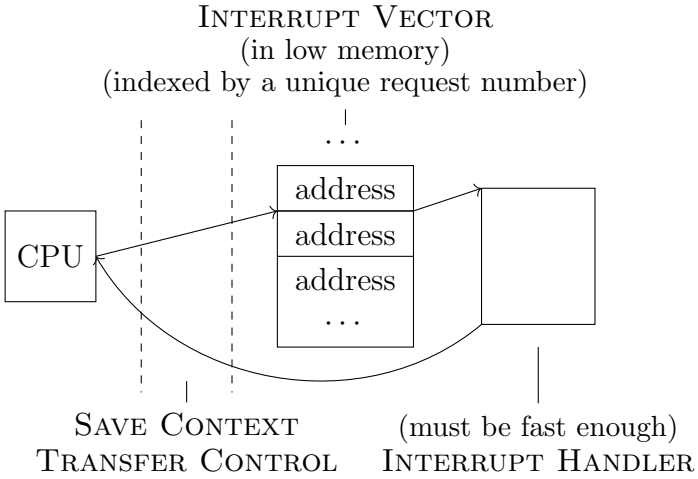
Figure: Components of Modern General-Purpose Computer

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)

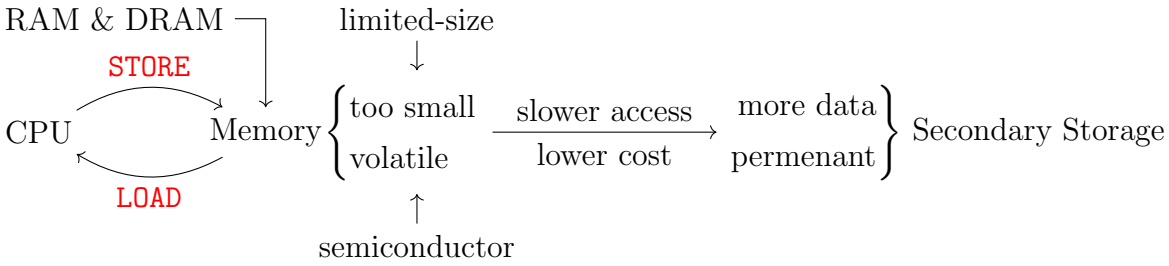
2. loads service programs (**system daemons**: outside kernel, loaded at boot, runs entire time)

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

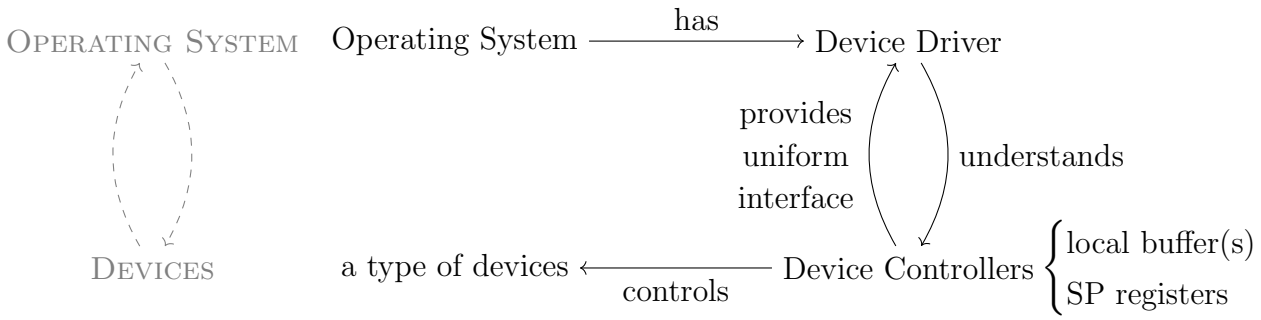


Figure: I/O Structure

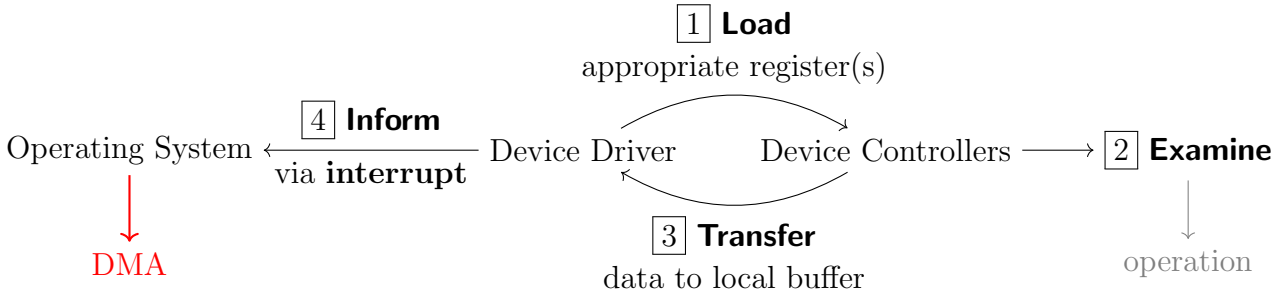


Figure: I/O Operation

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 { is managed by OS / only autonomously
SP registers (from processors) — limited SP instruction set }

1.3.2 Multiprocessor Systems

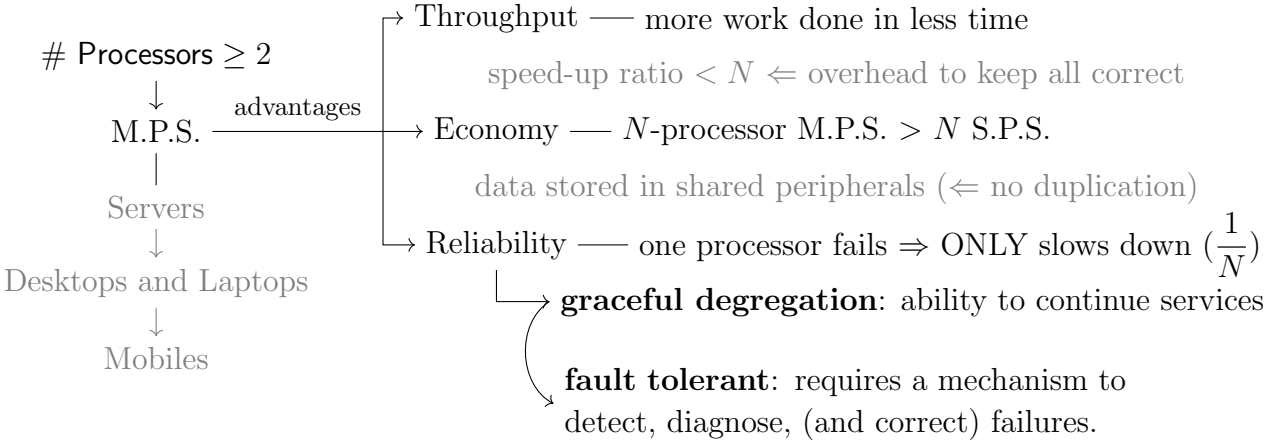


Figure: Multiprocessor System Concepts

- 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.

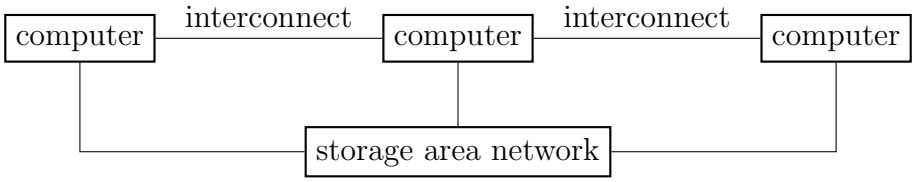


Figure: General structure of a clustered system

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

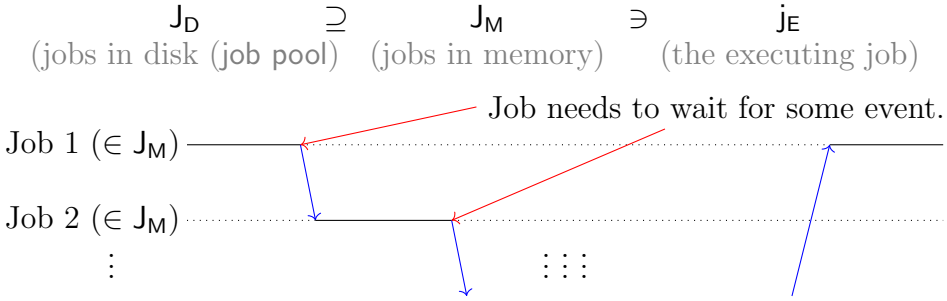


Figure: Multiprogramming Idea Interpretation

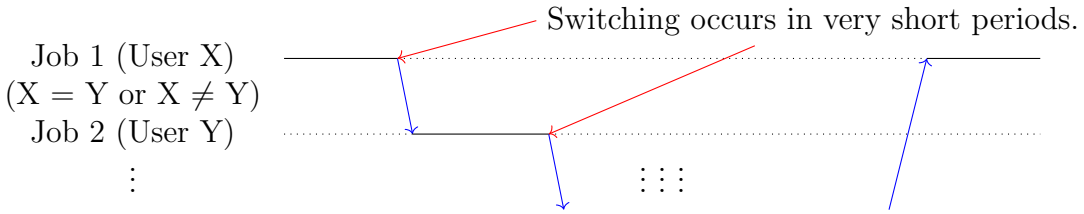


Figure: Multitasking Idea Interpretation
(Multitasking requires an **interactive** system, with short device responsive time.)

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

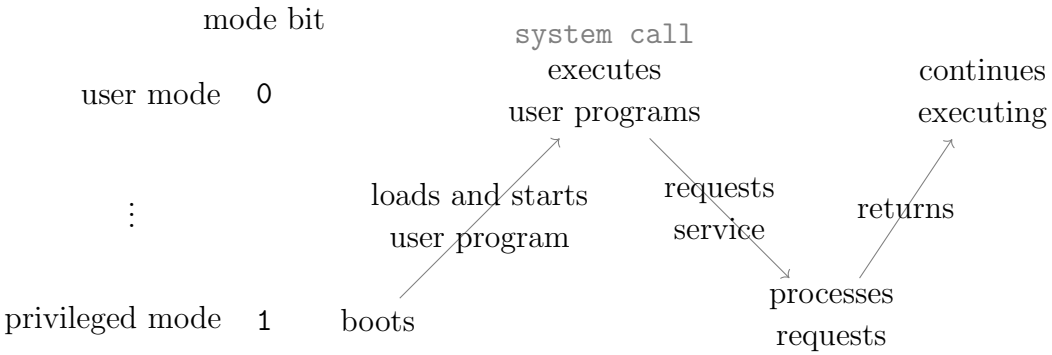


Figure: Dual-Mode Interpretation

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.

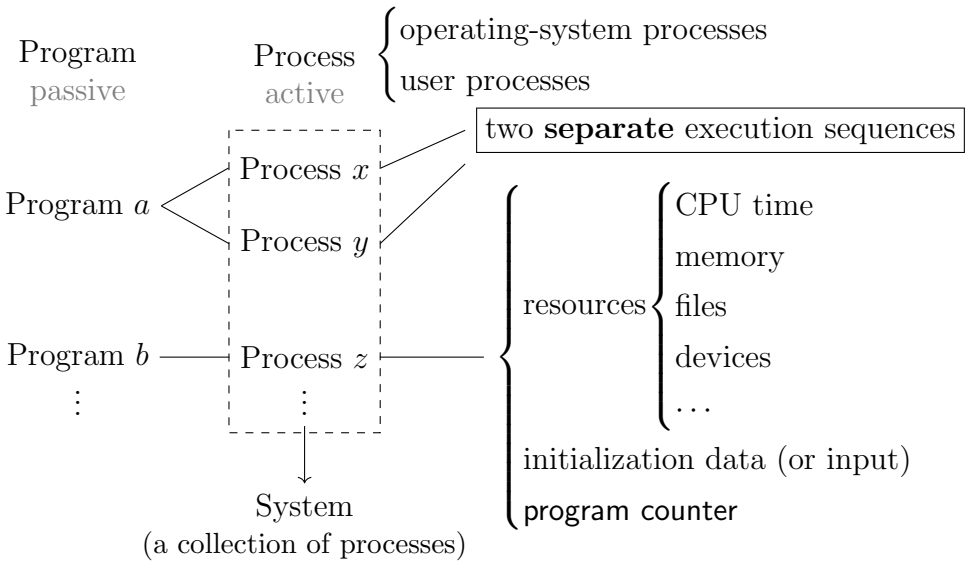
If a privilege violation is occurred, these errors are handled by the OS. The OS must terminate the program abnormally, probably with an appropriate error message.

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

Main memory

- is central to the operation of a modern computer system (only storage available to CPU);
- is a repository of quickly accessible data, shared by CPU I/O devices (different regions);
- contains executable binary fetched by CPU. (terminates $\Rightarrow$ space declared available)

To maintain multiple programs in memory, various schemes are used. The **hardware design** determines the effectiveness of various algorithms.

1.8 Storage Management

1.8.1 File-System Management

A file is the unit of logical storage that the OS abstracts from the physical properties of different storage devices. Each storage device has its own characteristics and physical optimization.

Files can represent programs (executable binary or sources) or data (in various formats).

OS also needs to control whether and how each user may access files.

1.8.2 Mass-Storage Management

Most computer systems use disks to support larger and permanent storage. Most programs use the disk as both the source and destination of their processing. Tertiary storage is slower and lower in cost, with a larger capacity than secondary storage, usually used to backup data, store seldom-use data, etc.

1.8.3 Caching

Caching is to temporarily copy requested information into a faster storage system for faster response to potential further requests. Most caches are implemented purely in hardware. Caches

have limited size, so developers should choose the appropriate cache size and a replacement policy. Main memory can be viewed as a fast cache for secondary storage.

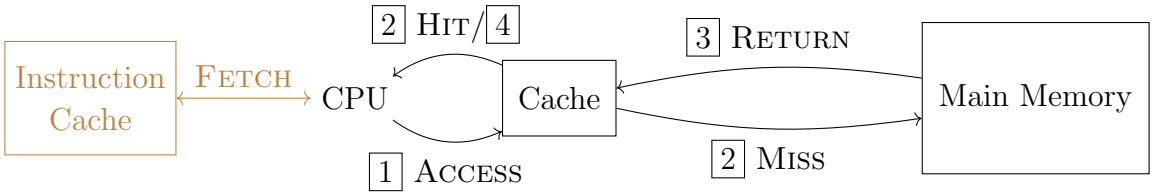


Figure: Caching Principles

1.9 Protection and Security

Protection is any mechanism for controlling the access of processes or users to the resources defined by a computer system. Protection and security require the systems to distinguish among all its users. In some cases, we need to distinguish the set of users, we need to define a group. A user can be in more than one group. Some OSs also support privilege escalation.

1.10 Kernel Data Structures

- `List`
  - singly linked list
  - doubly linked list
  - circularly linked list
- `Stack` (LIFO, `push`, `pop`)
- `Queue` (FIFO, `enqueue`, `dequeue`)
- `Tree`
  - ⇒ `Binary Tree`
  - ⇒ `Binary Search Tree`
  - ⇒ `Balanced Binary Search Tree`
- `Hash Table`
- `Bitmap`

2 Operating-System Structures

2.1 Operating-System Services

Here is a set of functionality that is *helpful to the user*.

- User Interface
  - Command-Line Interface: uses text commands
  - Batch Interface: TODO
  - Graphical User Interface: a window system with a pointing device.
- **Program Execution**: be able to load and execute the program (can end its execution).
- **I/O Operations**: supports programs' demand for I/O (users usually can't directly control).
- **File-system Manipulation**: read, write, create, delete, search, list, permission check.
- Communication: one process needs to exchange information with another process.
- Error Detection & Correction (in hardware like CPU, memory; I/O devices; user programs.

A set of functionality for efficient operation of the system:

- Resource Allocation: CPU cycles, main memory, file storage to different users/jobs.
- Accounting: to keep track of usage of various resources, to help users optimize usage.
- Protection and Security (authentication of users & users' access to system resources)

2.2 User and Operating System Interface

2.2.1 Command Interpreters

Command interpreters, or **shells**, understands and executes the next user-specified command.

- CI contains the code of command ⇒ number of commands determines the size of CI.
- CI tries to identify an executable file to the command ⇒ CI is small and static.

2.2.2 Graphical User Interfaces

The user employs a *mouse-based window-and-menu system*. For mobile systems, users interact by making gestures on the touch screen.

2.2.3 Choice of Interface

Command-Line Interface	Graphical User Interface
more efficient	
makes repetitive tasks easier	
shell scripts	

2.3 System Calls

The demand for system calls in programs is very heavy. Systems usually execute thousands of system calls per second. System-provided application programming interface, or API, encapsulates system calls for programmers. API ensures **portability** (different versions and implementations of the same API) and ease of use (some system calls are difficult to work).

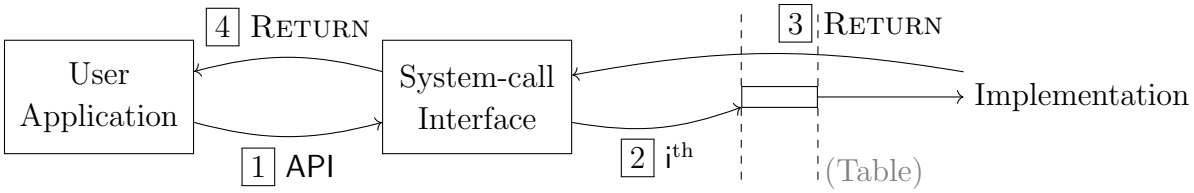


Figure: System-call Interface

Some system calls require extra information. Information can be passed to the parameters (limited) or pushed onto the stack.