Operating Systems Notes

May 31, 2025

Contents

Chapter 1 – Foundations of Operating Systems

1 Purpose and Core Responsibilities

- **Intermediary:** OS sits between user & applications and hardware, hiding details while exposing services.
- **Goals:**
  - Run programs conveniently (consistent UI).
  - Run programs efficiently (performance, safe sharing).
  - Manage hardware resources (CPU, memory, I/O, storage).

2 Abstraction Layers (high → low)

1. **User** – GUI, CLI, touch, voice.
2. **Application programs** – browsers, compilers, games.
3. **Operating system** – kernel, system programs, middleware.
4. **Computer hardware** – CPU, memory, devices.

3 User View vs. System View

- **User view:** Focus on ease of use; resource details hidden.
- **System view:**
  - **Resource allocator** – arbitrates CPU, memory, I/O, etc.
  - **Control program** – governs execution, manages devices.

4 What Forms an Operating System?

Always	Usually	Optional / Varies
Kernel – always runs after boot.	**Device drivers**, loadable modules. **System programs** (shells, daemons).	**Middleware** (graphics, DB), extra utilities.

5 Why Study Operating Systems?

- All code runs on an OS; knowing its policies & APIs yields safer, faster, portable software.
- OS concepts (processes, memory, I/O, security) recur in servers, clouds, IoT.

6 Glossary Highlights (1.1)

- **Operating system, kernel, system program, middleware, resource allocator, control program, ease of use, resource utilization**

1.2 Computer-System Organization

1 System-Level Hardware Layout

- Shared **system bus** connects CPU cores, main memory, device controllers.
- **Device controller:** logic per device class (disk, GPU, USB) with registers & buffer.
- **Device driver:** kernel API shielding hardware details.
- **Parallelism:** CPU & controllers run concurrently; **memory controller** arbitrates.

2 Interrupts – Hardware-Driven Events

2.1 Lifecycle

1. Controller raises signal on **interrupt-request line**.
2. CPU catches signal, saves state, jumps to handler via **interrupt vector**.
3. Handler services device, restores state, executes `return_from_interrupt`.

2.2 Efficiency Features

- **O(1) dispatch** via vector table.
- **Nonmaskable** vs. **maskable** lines (maskable can be disabled).
- **Priority & interrupt chaining:** high-priority pre-empts low; vector entry may head list of handlers.

2.3 End-to-End I/O Timeline

`User code` → `I/O request` → `controller/DMA busy` → `finish` → `interrupt` → `handler` → `resume user code`

3 Storage Structure

3.1 Memory Hierarchy

Layer	Volatile?	Size	Access	Notes
Registers	Yes	bytes	sub-ns	in CPU
Cache	Yes	KB–MB	ns	SRAM
Main memory	Yes	GBs	∼10 ns	DRAM
NVM/SSD	No	10 GB–TB	μs	electrical
HDD	No	100 GB–TB	ms	mechanical
Optical/tape	No	TB–PB	s–min	archival

3.2 Key Units & Terms

- **Bit → byte (8 bits) → word** (CPU width).
- **KiB / MiB / GiB / TiB / PiB** = powers of 1024.
- **Volatile** memory loses data on power-off; **non-volatile storage (NVS)** persists.
- **Firmware / EEPROM** stores bootstrap program.

3.3 Why Secondary Storage?

- Main memory is finite and volatile; programs & data live on slower persistent media until loaded.

4 I/O Structure

4.1 Direct Memory Access (DMA)

- Driver configures DMA engine; controller moves block without CPU, raises one interrupt on completion.

4.2 Bus vs. Switched Fabrics

- **Shared bus:** one transfer at a time; common on PCs.
- **Switch fabric:** concurrent links; used in servers & SoCs.

4.3 Full I/O Cycle

1. Driver starts I/O.
2. Controller activates device/DMA.
3. CPU handles other tasks, checks interrupts.
4. Device finishes, raises interrupt.
5. Handler validates data, wakes waiting process.

5 Glossary Highlights (1.2)

Term	Definition
Bus	Shared path linking CPU, memory, controllers.
Device driver	Kernel interface to a controller.
Interrupt / vector / request line	Hardware signal and its dispatch mechanism.
Maskable / nonmaskable interrupt	Can or cannot be disabled.
DMA	Controller-driven block transfer to/from memory.
Volatile / non-volatile memory	Data lost or retained on power loss.