

# Introduction to Operating Systems

## Introduction to Operating Systems

Course?level overview – suitable for beginners

---

## 1. What Is an Operating System (OS)?

| Aspect        | Description                                                                                                                                                    |
|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Definition    | System software that manages hardware resources and provides services for application programs.                                                                |
| Primary Goals | <ul style="list-style-type: none"><li>• Abstract hardware details</li><li>• Allocate resources efficiently</li><li>• Ensure security &amp; stability</li></ul> |
| Typical Users | End?users, developers, system administrators, embedded devices.                                                                                                |

---

## 2. Core Functions of an OS

### 1. Process Management

- Creation, scheduling, and termination of processes.
- Context switching & multitasking (pre?emptive vs. cooperative).
- Inter?process communication (IPC): pipes, message queues, shared memory, sockets.

### 2. Memory Management

- Allocation & deallocation of RAM to processes.
- Virtual memory: paging & segmentation.
- Protection (address space isolation) and swapping.

### 3. File System Management

- Organization of data on storage devices.
- Hierarchical directory structure, file metadata, permissions.
- Common file systems: FAT, NTFS, ext4, APFS, XFS.

# Introduction to Operating Systems

## 4. Device (I/O) Management

- Drivers translate generic OS calls to device-specific actions.
- Buffering, caching, and spooling to improve performance.

## 5. Security & Protection

- Authentication, authorization, and access control lists (ACLs).
- Isolation mechanisms (user vs. kernel mode, sandboxing).
- Auditing and logging.

## 6. User Interface

- Command-line interface (CLI) – e.g., Bash, PowerShell.
- Graphical user interface (GUI) – windows, menus, icons.

---

## 3. OS Architecture Models

| Model                    | Key Characteristics                                                                                                                                            |
|--------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Monolithic Kernel</b> | All services run in kernel space (e.g., Linux, early UNIX). Fast but less fault-tolerant.                                                                      |
| <b>Microkernel</b>       | Minimal kernel (scheduler, IPC, basic memory mgmt); other services run in user space (e.g., Minix, QNX). Improves reliability, may incur performance overhead. |
| <b>Hybrid Kernel</b>     | Mix of monolithic and microkernel ideas; core services in kernel, others modular (e.g., Windows NT, modern macOS).                                             |
| <b>Exokernel</b>         | Exposes hardware resources directly to applications; OS provides only protection and allocation policies. Research-oriented.                                   |

---

## 4. Process vs. Thread

| Concept           | Process                                                | Thread                                                          |
|-------------------|--------------------------------------------------------|-----------------------------------------------------------------|
| <b>Definition</b> | Independent execution unit with its own address space. | Lightweight execution unit sharing the process's address space. |
| <b>Overhead</b>   | Higher (creation, context switch).                     | Lower – faster creation and switching.                          |

# Introduction to Operating Systems

| Concept              | Process                              | Thread                                                                   |
|----------------------|--------------------------------------|--------------------------------------------------------------------------|
| <b>Use Cases</b>     | Isolated tasks, security boundaries. | Parallelism within a program (e.g., UI thread + worker threads).         |
| <b>Communication</b> | Requires IPC mechanisms.             | Can share memory directly; synchronization needed (mutexes, semaphores). |

---

## 5. Scheduling Basics

### 1. Scheduling Goals

- Maximize CPU utilization.
- Minimize response time & turnaround time.
- Ensure fairness & meet real-time constraints.

### 2. Types of Schedulers

- **Long-term (admission) scheduler** – decides which jobs enter the ready queue.
- **Short-term (CPU) scheduler** – selects the