🧩 CHAPTER 1: PLATFORM TECHNOLOGIES

🔹 What is Platform Technology

- A **platform** is a group of technologies used as a base for developing applications, processes, or other technologies.
- It provides **hardware**, **software**, and **network** components that support operations and development.
- It conforms to **standards** that allow developers to build applications on it.

🔹 Components of a Platform

- **Hardware** – Physical devices (CPU, memory, I/O devices).
- **Software** – Operating systems, applications, and tools.
- **Networks** – Connectivity that allows systems to communicate.

💻 GENERATION OF COMPUTERS

1. **First Generation (1940s–1950s)** – Vacuum tubes, very large, slow, and costly.
2. **Second Generation (1950s–1960s)** – Transistors replaced vacuum tubes.
3. **Third Generation (1960s–1970s)** – Integrated circuits (ICs) used.
4. **Fourth Generation (1970s–Present)** – Microprocessors and personal computers.
5. **Fifth Generation (Present and Beyond)** – Artificial Intelligence, quantum computing.

🔹 Modern Computer Components

- One or more **processors (CPU)**

- **Main Memory** (RAM)
- **Disk Storage**
- **Printers**
- **Input/Output devices**

⚙️ OPERATING SYSTEM (OS)

🔹 Definition

- A **system software** that manages computer hardware, software resources, and provides services for computer programs.

🔹 Goals of an Operating System

- **Efficient Utilization** – Optimize CPU, memory, and I/O.
- **Multitasking** – Manage multiple processes at once.
- **Resource Sharing** – Allow sharing of hardware/software resources.
- **System Security** – Protect system with authentication & access control.

🔹 Functions of OS

- Provides **User Interface (CLI/GUI)**.
- **Manages hardware resources** through APIs.
- **Launches and manages** application execution.
- **Process Control, Memory Management, File Management, Security, Job Accounting**, and **Error Detection**.

🔹 SYSTEM CALLS

What is a System Call?

- A **mechanism for programs to request services** from the OS (e.g., file access, device control).
- Acts as a bridge between **user programs** and **kernel** functions.

Why System Calls Are Needed

1. File operations (create, delete, read, write).
2. Network communication (send/receive data).
3. Access hardware devices (printers, scanners).
4. Create and manage processes.

🧠 TYPES OF SYSTEM CALLS

1. Process Control

- End/Abort process, load/execute, create/terminate, allocate/free memory.
- Manages process lifecycle.

2. File Management

- Create, delete, open/close, read/write files.
- Handle file access and attributes.

3. Device Management

- Request/release devices, attach/detach, set attributes.
- Manage device I/O and buffers.

4. Information Maintenance

- Get/set system time/date, device attributes.
- Transfers data between OS and user.

5. Communication

- Create/delete communication links, send/receive messages.
- Used in **interprocess communication (IPC)**.

🖥️ TYPES OF OPERATING SYSTEMS

- **Batch OS** – Executes jobs in batches without user interaction.
- **Multiprogramming OS** – Runs multiple programs in memory.
- **Multitasking OS** – Performs many tasks simultaneously.
- **Time-Sharing OS** – CPU time divided among users.
- **Distributed OS** – Manages group of separate computers.
- **Network OS** – Connects and manages networked systems.
- **Real-Time OS** – Provides immediate response to input.

🧩 CHAPTER 2: PROCESSES AND THREADS

🔹 2.1 The Process

- A **process** = a program in execution.
- The **basic unit of work** managed by the OS.
- Has its own **memory, stack, program counter, and registers**.

Process Memory Structure

- **Stack** – Temporary data, parameters, local variables.
- **Heap** – Dynamically allocated memory.
- **Data Section** – Global/static variables.
- **Code Section** – Program instructions.

🔹 2.2 Process Creation and Termination

Process Creation

- Created by:
  - System initialization
  - User request
  - Batch job
  - Another process (using `fork()`)
- **Parent process** creates **child process**.
- **Child** can be duplicate of parent or load a new program.

Process Termination

- Done using `exit()` system call.
- Releases all resources (memory, files, buffers).
- Parent can check child status using `wait()`.

Causes for Termination

- Time slot expired

- Memory overflow
- I/O failure
- Process request
- Invalid instruction

🔹 2.3 Process States

Common Process States

1. **New** – Created but not ready yet.
2. **Ready** – Waiting to be assigned to CPU.
3. **Running** – Currently executing.
4. **Blocked/Waiting** – Waiting for I/O or event.
5. **Terminated/Exit** – Finished or aborted.

State Transitions

- **New** → **Ready** – Loaded into memory.
- **Ready** → **Running** – Scheduler picks the process.
- **Running** → **Blocked** – Waiting for I/O or resource.
- **Running** → **Ready** – Time slice ended.
- **Blocked** → **Ready** – I/O completed.
- **Running** → **Exit** – Process finishes or ends.

🔹 2.4 Threads

Definition

- A **thread** is the smallest unit of CPU execution inside a process.
- Each thread has its own **program counter, stack, and registers**.
- Multiple threads share the same memory and resources.

Advantages

- Faster context switching.
- Provides concurrency.
- Efficient communication between threads.
- Utilizes multiprocessors better.

Types of Threads

- **User-Level Threads** – Managed by user applications.
- **Kernel-Level Threads** – Managed directly by the OS.

🔹 **2.5 CPU Scheduling and Algorithms**

What is CPU Scheduling?

- Decides which process gets to use the CPU next.
- Improves **efficiency**, **speed**, and **fairness**.

Objectives

- Maximize CPU utilization.
- Maximize throughput (completed processes/time).
- Minimize turnaround, waiting, and response time.
- Ensure fairness.

Key Terms

- **Arrival Time** – When process enters ready queue.
- **Burst Time** – CPU time required by process.
- **Completion Time** – When process finishes execution.
- **Turnaround Time** – Completion − Arrival.
- **Waiting Time** – Turnaround − Burst time.

⚙️ Types of Scheduling Algorithms

1. FCFS (First Come, First Serve)

- Processes executed in order of arrival (FIFO).
- Simple but can cause long waiting times.

2. SJF (Shortest Job First)

- Process with shortest burst time executes first.
- Efficient but may cause starvation of long jobs.

3. LJF (Longest Job First)

- Opposite of SJF. Longest process first.
- Non-preemptive.

4. Priority Scheduling

- CPU given to highest-priority process.
- Can be **preemptive or non-preemptive**.

5. Round Robin (RR)

- Each process gets fixed time slice (quantum).
- Suitable for **time-sharing** systems.

6. SRTF (Shortest Remaining Time First)

- Preemptive version of SJF.
- Chooses process with least remaining burst time.

7. LRTF (Longest Remaining Time First)

- Preemptive version of LJF.

8. HRRN (Highest Response Ratio Next)

- Process with highest response ratio executes next.
- Formula: `(Waiting Time + Burst Time) / Burst Time`

9. Multiple Queue Scheduling

- Processes divided into queues:
    - **System**, **Interactive**, **Batch**
- Each queue has its own scheduling method.

10. Multilevel Feedback Queue

- Processes can move between queues.
- More flexible and efficient than fixed multilevel queue.