SNS COLLEGE OF TECHNOLOGY

An Autonomous Institution

Coimbatore-35

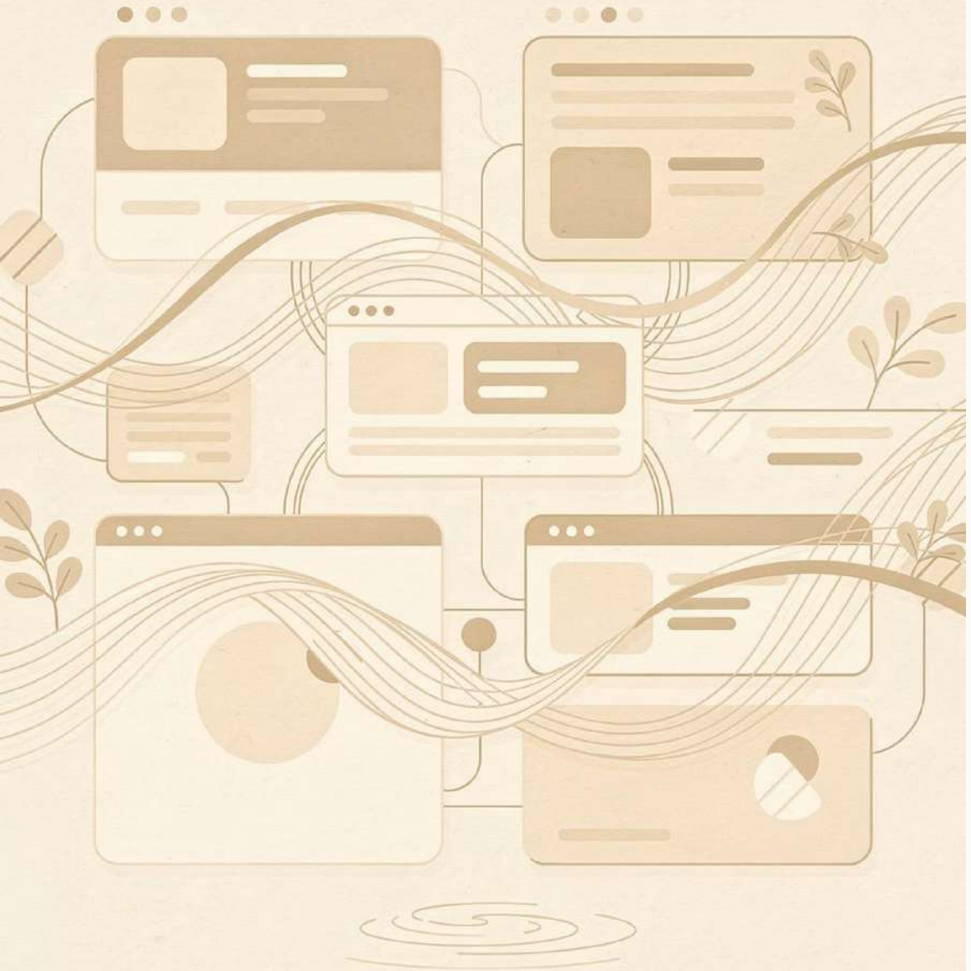
Department of Computer Science and Engineering

23CST206–OPERATING SYSTEMS AND VIRTUALIZATION

B.E- CSE /IV SEMESTER

UNIT – II PROCESS MANAGEMENT

Topic 1:Processes - Process Concept



Processes: Understanding the Process Concept

Explore the fundamental building blocks of modern operating systems and how programs transform into active, executing processes.

What is a Process?

A process is a program in execution. When we write computer programs in a text file and execute them, they become processes that perform all the tasks mentioned in the program.

A process is an active entity,

A program is considered a passive entity.

A single program can create many processes when run multiple times.

Real-Time Example: The Exam Analogy

Question Paper = Program

The same instructions provided to all students, stored and ready to be used.

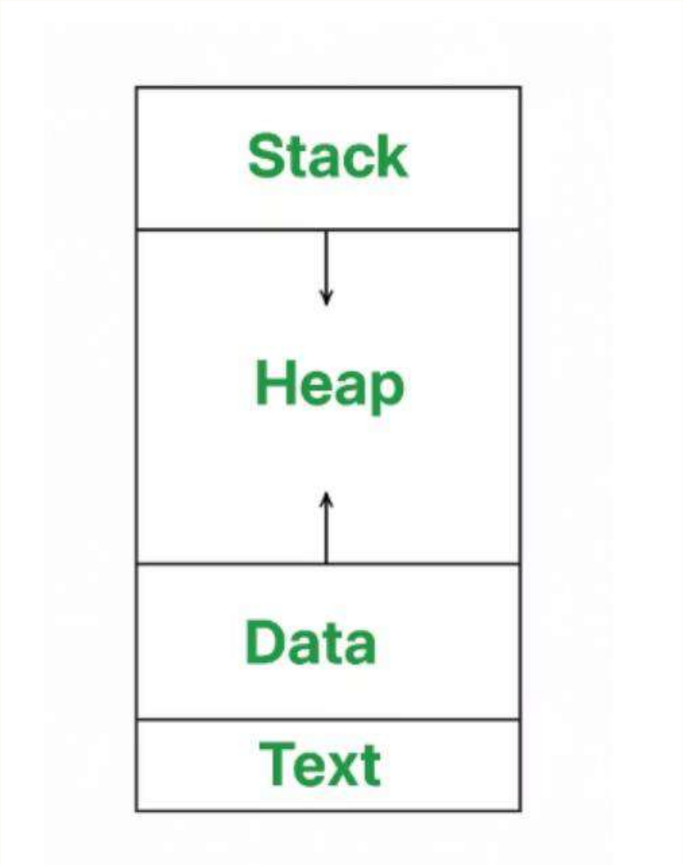
Each Student = Process

Individual execution with unique progress, state, and resources.

Answer Sheets = Memory

Each student has separate workspace, different timing, and independent results.

Process Memory Layout



Text Section

Contains executable instructions. Typically a **read-only** section to prevent accidental modification.

Stack

Stores temporary data including function parameters, return addresses, and local variables.

Data Section

Contains global and static variables used throughout program execution.

Heap Section

Memory dynamically allocated to the process during runtime for flexible storage needs.

Components of a Process

A process is not just program code it consists of several critical components that the operating system must manage carefully to ensure correct execution.

Program Code

The executable instructions of the program stored in the text section.

Program Counter

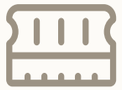
Indicates the address of the next instruction to be executed by the CPU.

CPU Registers

Hold intermediate values and execution context during processing.

Stack

Stores function calls, local variables, and return addresses.



Heap

Used for dynamic memory allocation during program execution.

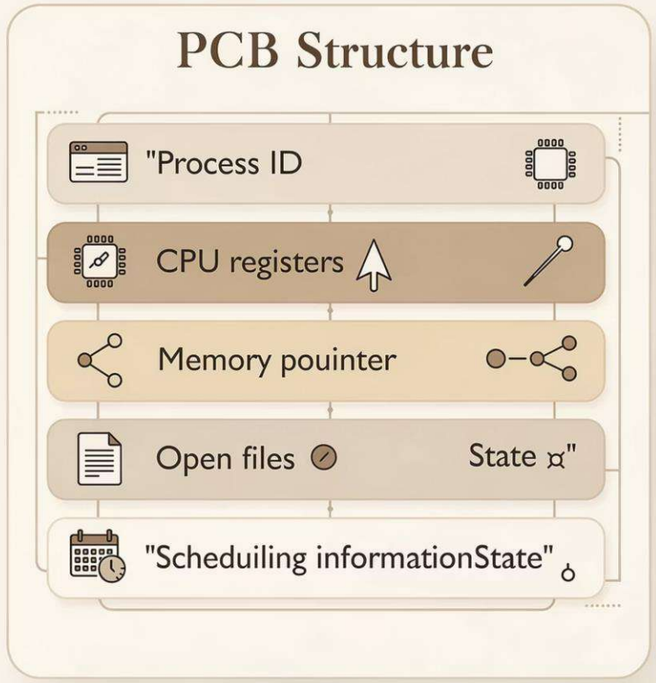
Data Section

Contains global and static variables accessible throughout execution.

CHAPTER 4

Process States: The Process Life Cycle

Real-world example: When you open a music player, it enters the ready state, then running state when CPU is allocated. If it waits for file input, it goes into the waiting state, and after completion, it terminates.



Process Control Block (PCB)

A Process Control Block (PCB) is a critical data structure used by the operating system to track process information and manage execution efficiently.

Unique Identity

Each process is given a unique Process ID (PID) for identification and tracking.

State Management

PCB stores process state, program counter, stack pointer, and open files.

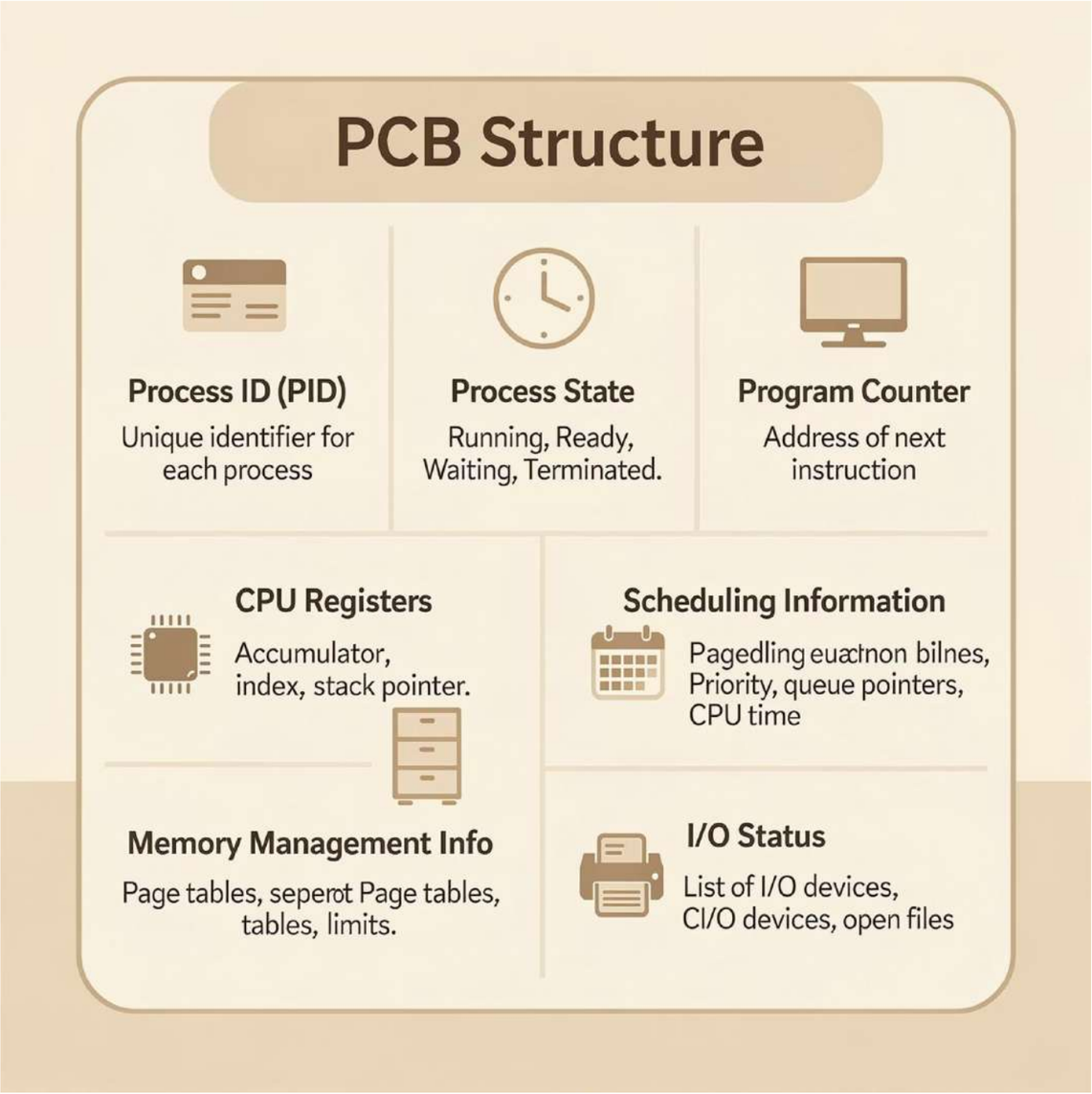
Dynamic Updates

During state transitions, the OS updates the PCB with the latest execution data.

Scheduling Info

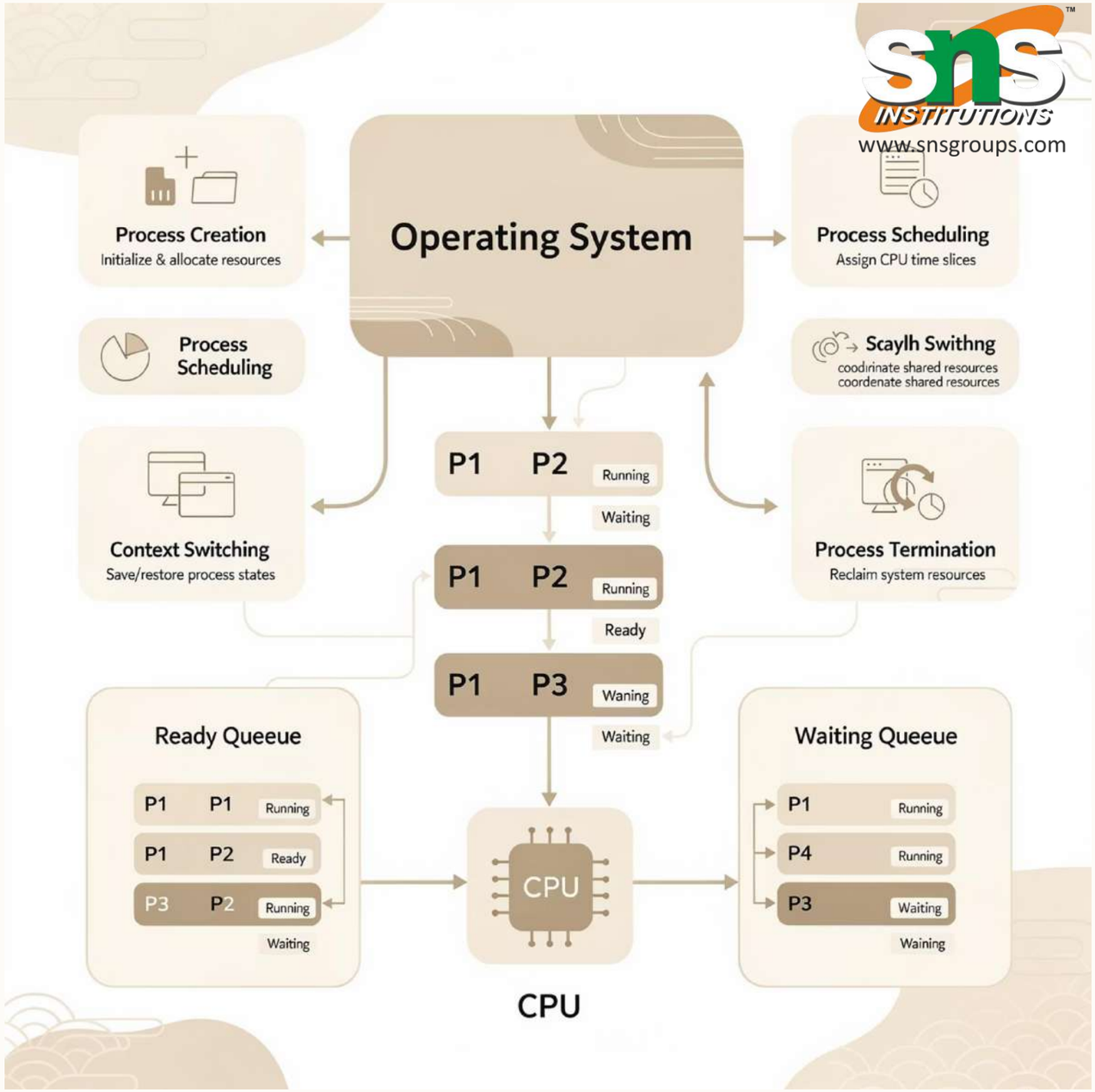
Includes register values, CPU quantum, and process priority for scheduling.

PCB Structure: Inside the Control Block



Operating System's Role in Process Management

The operating system serves as the conductor of the system, orchestrating multiple processes to create the illusion of simultaneous execution through multiprogramming.



Design Thinking Approach: Processes in OS

Problem Statement

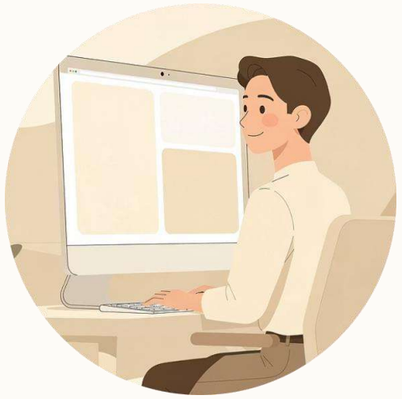
Modern computer systems run multiple applications simultaneously web browser, music player, and code compiler. Each requires CPU, memory, and I/O devices at the same time.

> **The Challenge:** How can the operating system manage multiple programs efficiently, ensuring fairness, correctness, and smooth execution while preventing conflicts and resource starvation?

STEP 1

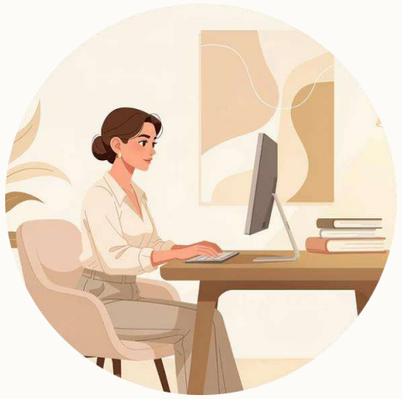
Empathize: Understanding Stakeholders

To address the problem effectively, we must first empathize with the system stakeholders and understand their unique needs and pain points.



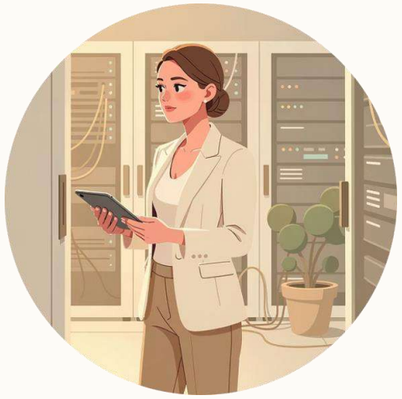
End Users

Want fast, smooth, and responsive applications without crashes or delays. They don't care about internal OS mechanics.



Programmers

Want each program to run predictably and reliably, without interference from other processes.



System Administrators

Want security, controlled access, and resource allocation to prevent monopolization.

OS Developers

Want to design systems that efficiently schedule, track, and manage multiple processes while maintaining stability.

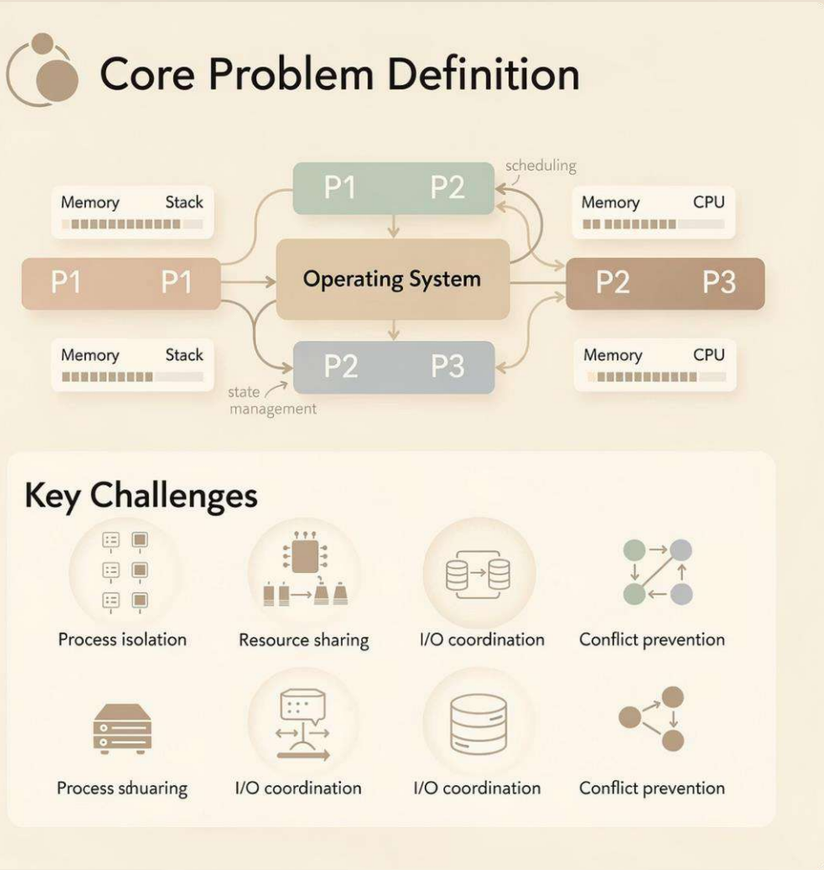
STEP 2

Define: Framing the Core Problem

After empathizing with stakeholders, we can clearly define the fundamental problem that process management must solve.

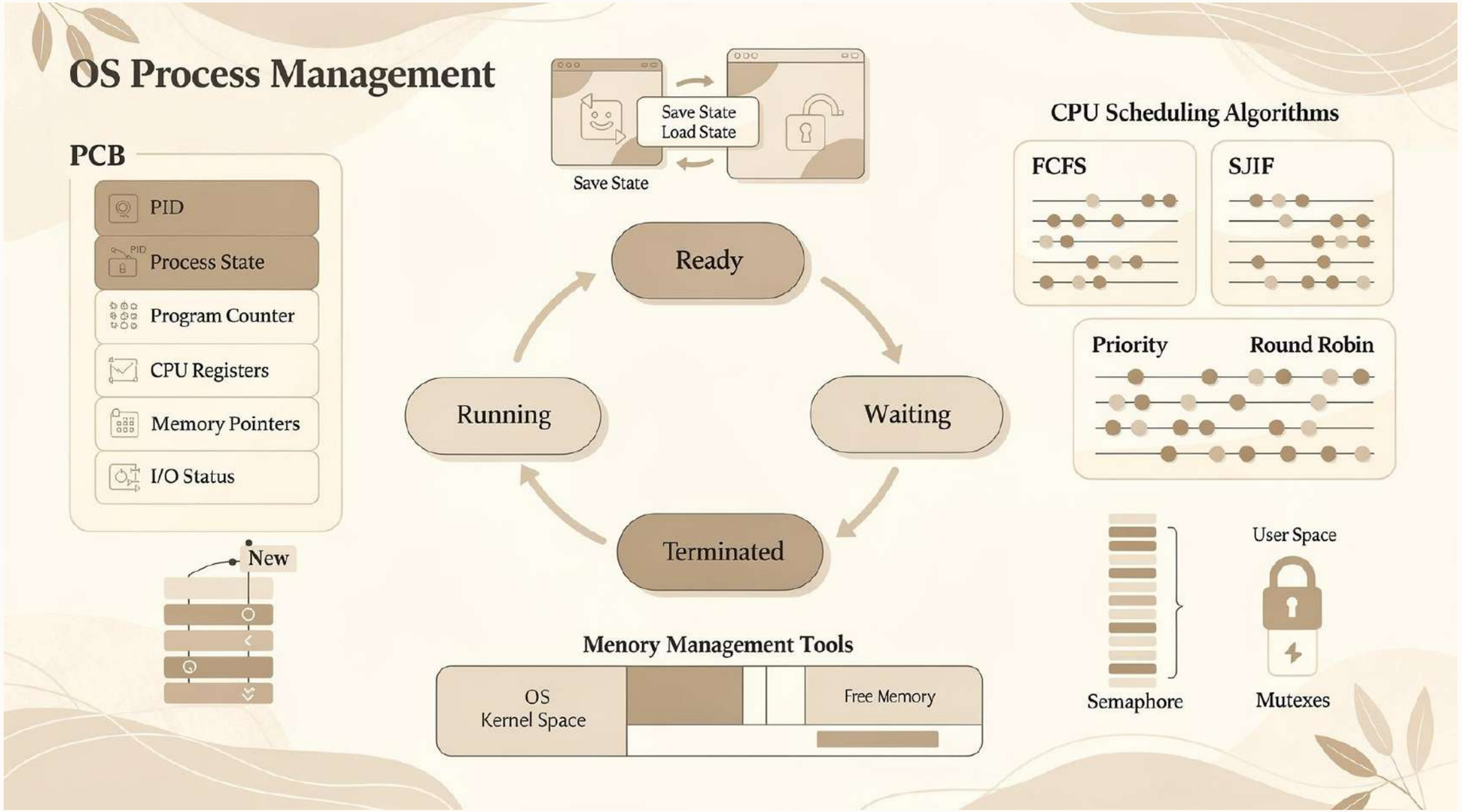
Core Definition

A process is an active entity representing a running program. Modern systems execute multiple processes concurrently, but each has its own memory, stack, CPU usage, and execution state.

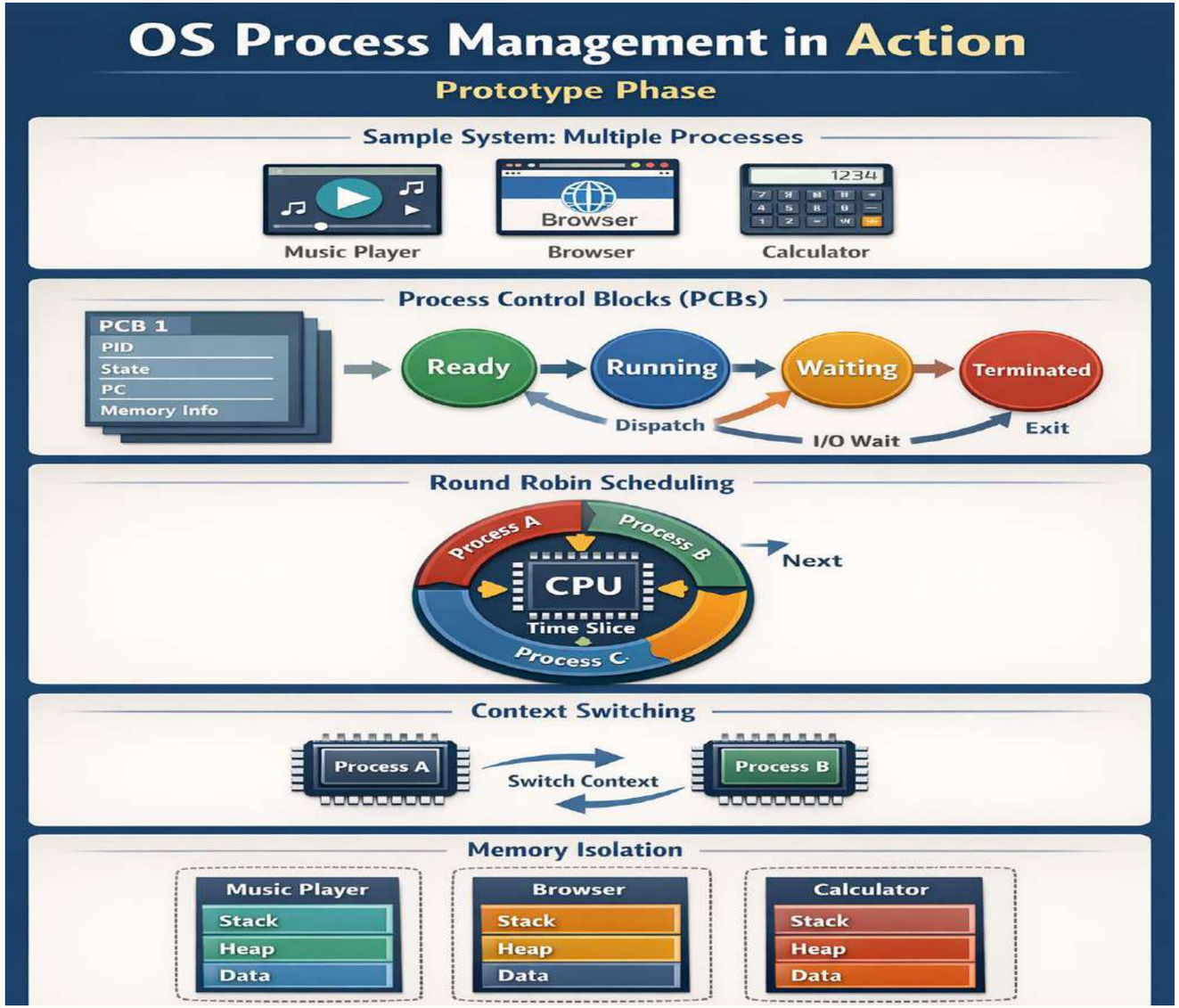


STEP 3

Ideate: Brainstorming Solutions



Prototype: OS Process Management in Action



Test: Validating the Solution

1

Monitor Resource Usage

Open multiple applications and check CPU usage patterns to ensure fair allocation.

2

Verify Fair Scheduling

Confirm that no process monopolizes the CPU and all processes receive appropriate time.

3

Observe State Transitions

Track process states during execution and context switches for correctness.

4

Test Memory Isolation

Validate that one process cannot access another process's memory space.

5

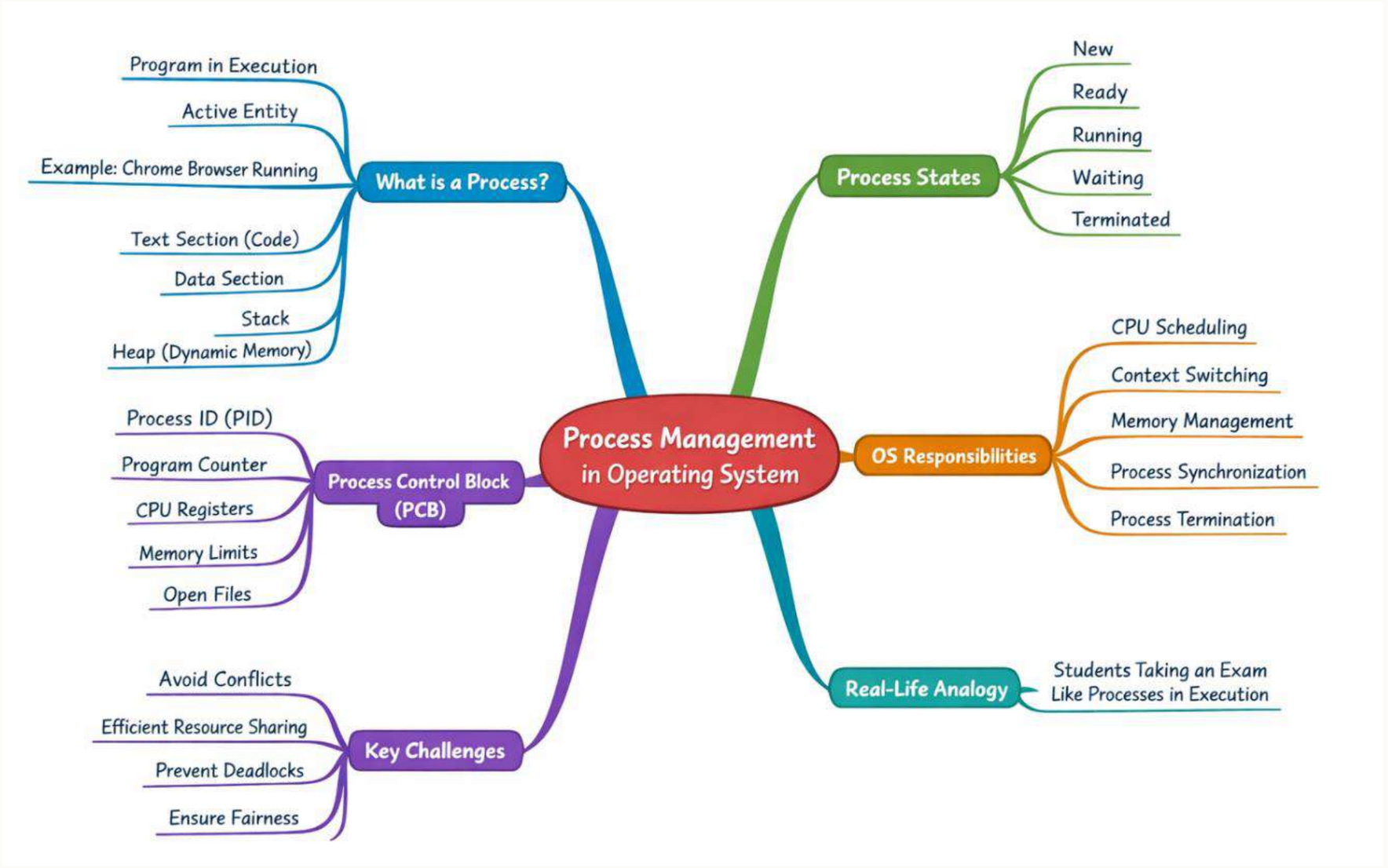
Validate Output

Check output correctness and responsiveness for all running processes.

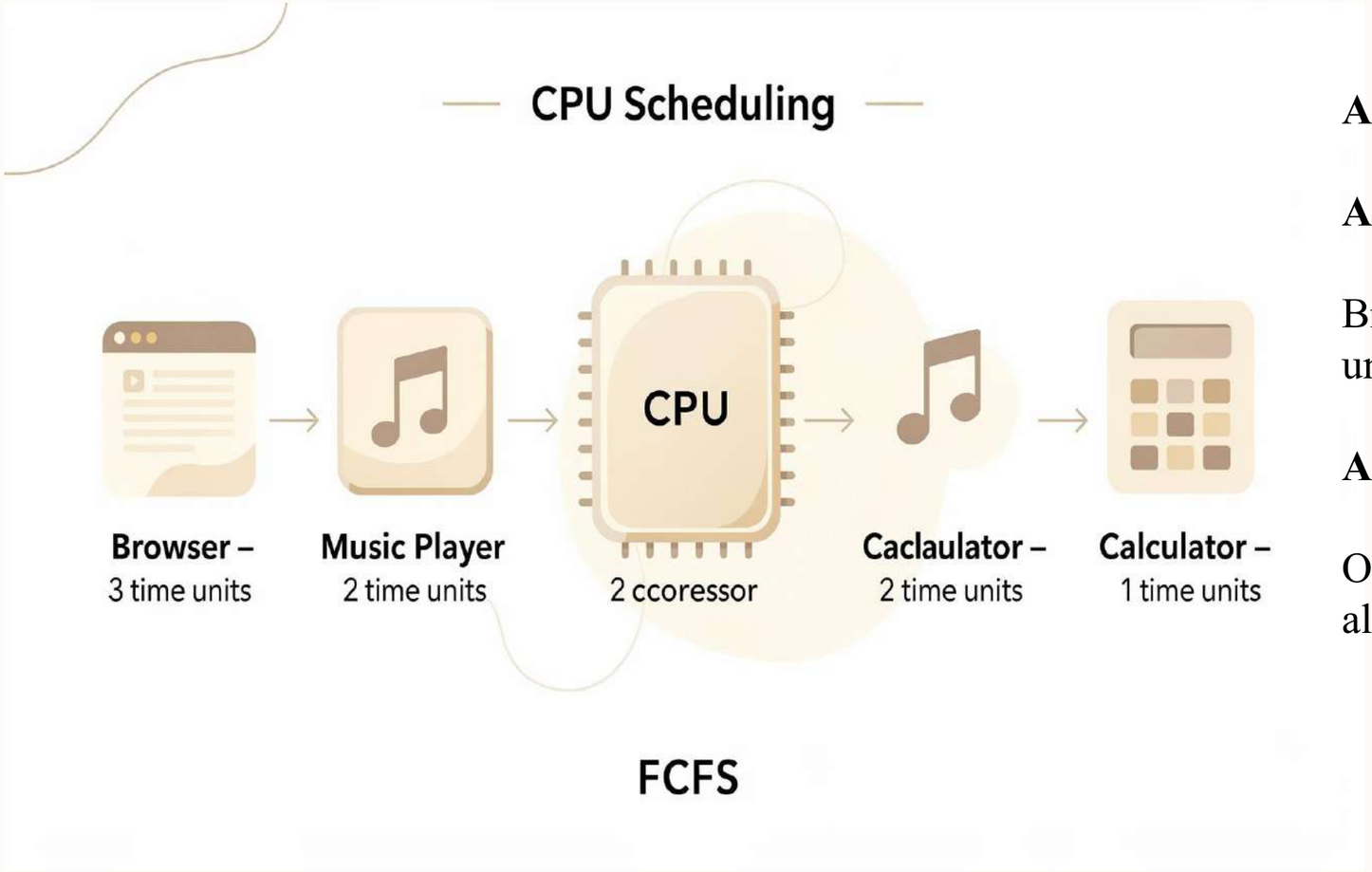
6

Check for Errors

Look for deadlocks, crashes, starvation, or race conditions.


Process Management in Operating System
What is a Process?
Program in Execution
Active Entity
Example: Chrome Browser Running
Text Section (Code)
Data Section
Stack
Heap (Dynamic Memory)
Process States
New
Ready
Running
Waiting
Terminated
Process Control Block (PCB)
Process ID (PID)
Program Counter
CPU Registers
Memory Limits
Open Files
OS Responsibllities
CPU Scheduling
Context Switching
Memory Management
Process Synchronization
Process Termination
Key Challenges
Avoid Conflicts
Efficient Resource Sharing
Prevent Deadlocks
Ensure Fairness
Real-Life Analogy
Students Taking an Exam
Like Processes in Execution

Simple Puzzle: "Who Runs First?"



Answer / Solution

Answer 1

Browser finishes last (needs 3 time units, runs first under FCFS)

Answer 2

OS switches between programs to ensure fair CPU time allocation and enable multitasking.

Thank You