Process Scheduling:

Process Scheduling is how the operating system (OS) decides which process gets to use the CPU and other resources at any given time. Think of it as a manager in a busy office assigning tasks to employees so everyone gets their turn and work is done efficiently.

When many programs (processes) are running on a computer, the OS uses **scheduling** to decide:

- Which process runs now?
- Which process waits?
- How much time each process gets?

Why is Process Scheduling Important?

- **Maximizes CPU usage:** Ensures the CPU is always working on something.
- **Fairness:** Each process gets its fair share of resources.
- **Multitasking:** Allows multiple processes to run simultaneously, making the computer feel fast and responsive.

Categories of Process Scheduling

There are three main categories of process scheduling based on when and where the scheduling happens:

---

1. Long-Term Scheduling

- **What it does:** Decides **which processes are allowed to enter the Ready Queue** (queue of processes waiting to use the CPU). **New -> Ready**
- **When it happens:** When a process is created.
- **Purpose:** Controls the degree of **multiprogramming** (how many processes are in memory and ready to execute).
- **Example:** If too many processes are loaded, the system could slow down. Long-term scheduling limits the number of active processes.

---

2. Short-Term Scheduling

- **What it does:** Decides **which process gets the CPU next. Ready -> Running**
- **When it happens:** Frequently, whenever the CPU becomes free (like every few milliseconds).
- **Purpose:** Provides quick decisions to maximize CPU usage.
- **Example:** If two processes (A and B) are in the Ready Queue, the Short-Term Scheduler decides if A or B will run next.

---

3. Medium-Term Scheduling

- **What it does:** Temporarily **removes processes from memory** (called suspending) to free up resources and later reactivates them. **Running -> Suspended**
- **When it happens:** When the system is overloaded, or resources are needed for higher-priority processes.
- **Purpose:** Improves system performance by balancing the load.
- **Example:** If a low-priority process is using too much memory, it might be suspended to make room for other important tasks.

Process States and Scheduling

Scheduling works by moving processes between different states (New, Ready, Running, Blocked, Terminated). For example:

1. A new process enters the **Ready Queue** (handled by Long-Term Scheduler).
2. The Short-Term Scheduler picks a process from the Ready Queue and runs it.
3. If the process needs to wait for I/O (like reading from a file), the process goes to the Blocked state.
4. The Medium-Term Scheduler may suspend some processes to reduce memory usage.

Types of Scheduling Algorithms

Scheduling decisions are made using specific **algorithms**. Common types include:

1. First-Come, First-Served (FCFS):

- Processes are executed in the order they arrive.
- **Example:** Like waiting in line at a grocery store.

2. Shortest Job Next (SJN):

- Processes with the shortest execution time are run first.
- **Example:** If a task takes 5 seconds and another takes 2 seconds, the 2-second task goes first.

3. Round Robin (RR):

- Each process gets a fixed time slice (e.g., 2ms) before moving to the back of the queue.
- **Example:** Like everyone taking turns during a group discussion.

4. Priority Scheduling:

- Processes with higher priority are executed first.
- **Example:** Emergency services are prioritized over regular tasks.

5. Multilevel Queue:

- Processes are divided into queues based on priority or type, and each queue has its own scheduling algorithm.

Real-Life Analogy:

Imagine a theme park with rides:

1. **Long-Term Scheduler:** Decides how many people can enter the park to avoid overcrowding.
2. **Short-Term Scheduler:** Decides who gets to ride the rollercoaster next.
3. **Medium-Term Scheduler:** Asks people to step out of the line temporarily if the ride breaks down or there's a VIP group.

This ensures smooth operation, fairness, and efficiency.

Types of Scheduling Methods:

There are two types of **CPU scheduling methods** that define how the operating system allocates the CPU to processes. Let's break them down in simple terms.

---

1. Preemptive Scheduling

- **What it means:** The operating system can **interrupt** a running process and assign the CPU to another process if needed (based on priority, time limits, etc.).
- **Control:** The operating system is in control, and a process **must give up the CPU** when required.

How it Works:

- If a process with a **higher priority** arrives or if the current process has used up its allowed time (time slice), the CPU switches to another process.
- The interrupted process is paused and put back in the **Ready Queue** to wait for its turn again.

Example:

Imagine you're playing a game (Process A), and suddenly a phone call comes in (Process B). The game is paused, and you take the call. Once the call is finished, you return to the game. This is **preemptive scheduling**: one task interrupts another.

Advantages:

- Ensures **better responsiveness** for high-priority or time-critical tasks.

- More suited for **multitasking** systems.

Disadvantages:

- Frequent context switching (changing between processes) can slow down the system.
- More complex to implement.

Scheduling Algorithms:

- Round Robin (RR)
- Shortest Remaining Time First (SRTF)
- Priority Scheduling (Preemptive)

---

2. Non-Preemptive Scheduling

- **What it means:** Once a process starts running, it **cannot be interrupted** until it finishes or voluntarily gives up the CPU (e.g., waits for input/output).
- **Control:** The process is in control, and the CPU is not taken away forcefully.

How it Works:

- The CPU is allocated to a process, and it keeps running until:
    1. It finishes execution.
    2. It goes into a waiting state (e.g., waiting for I/O).
- Other processes in the **Ready Queue** must wait their turn.

Example:

Imagine you're watching a movie (Process A) on your laptop, and someone sends you an email (Process B). You finish the movie first before checking the email. No interruptions are allowed while you're watching. This is **non-preemptive scheduling**.

Advantages:

- No context switching, so it's **faster and simpler**.
- Works well for processes that finish quickly.

Disadvantages:

- Long processes can cause **starvation** for shorter or high-priority processes.
- Poor responsiveness in real-time systems.

Scheduling Algorithms:

- First-Come, First-Served (FCFS)
- Shortest Job Next (SJN)
- Priority Scheduling (Non-Preemptive)

Context Switching:

Context Switching is the process of **saving the state of a currently running process** and **restoring the state of another process** so the CPU can switch from one task to another. It's how the operating system (OS) enables multitasking by rapidly alternating between processes.

Why is Context Switching Important?

In a multitasking system, multiple processes run simultaneously. Since the CPU can only execute one process at a time, the OS needs a mechanism to:

1. **Pause a process** when its time is up or when a higher-priority process needs the CPU.
2. **Resume a paused process** later, exactly where it left off.

Context switching ensures this smooth transition between processes.

Steps in Context Switching

1. **Save the current state (context) of the running process:**
    - Save information like the **Program Counter (PC)**, **CPU registers**, and other important details in the **Process Control Block (PCB)** of the current process.
2. **Load the state of the next process:**
    - Fetch the saved context (from the PCB) of the next process that is ready to run.
3. **Switch the CPU to the new process:**
    - Start executing the next process from the exact point where it was paused.

Example:
Imagine you're watching a movie on your laptop, and you pause it to check your email. Later, you resume the movie from where you left off.

- **Pausing the movie:** Save the state of the process (e.g., the exact timestamp).
- **Opening the email:** Switch to a different process (email application).
- **Resuming the movie:** Restore the saved state (timestamp) to continue where it left off.

This is how the CPU switches between tasks using context switching.

When Does Context Switching Occur?

1. **Preemptive Multitasking:**
   The OS forces a process to pause when:
    - A higher-priority process arrives.
    - The time slice (time allocated to the process) expires.
2. **Voluntary Context Switching:**
   A process voluntarily pauses itself, such as:
    - Waiting for input/output (I/O).
    - Entering a "sleep" or "waiting" state.
3. **Interrupt Handling:**
   When hardware interrupts (e.g., a keypress or mouse click) occur, the OS switches to the interrupt handler process.

Advantages of Context Switching

- **Multitasking:** Allows multiple processes to run smoothly by sharing the CPU.
- **Fairness:** Ensures no process monopolizes the CPU.
- **Responsiveness:** Improves system responsiveness for time-sensitive tasks.

Disadvantages of Context Switching

1. **Overhead:**
   Context switching takes time to save and load processes. This time is spent on switching instead of actual work, reducing CPU efficiency.
2. **Resource Usage:**
   It requires extra memory to store the state of processes in the PCB.
3. **Frequent Switching:**
   Too many context switches (e.g., in preemptive scheduling) can lead to high overhead, slowing down the system.

Frequently Asked Questions on Process Scheduling – FAQs

What is the job queue?

A job queue is like a waiting line where tasks are kept until the computer is ready to work on them. When you send a task to the computer, it goes into the queue, and the computer does them one by one, usually in the order they were added.

What is CPU scheduling in OS?

In an operating system, CPU scheduling refers to a technique that permits one process to utilize the CPU while keeping the other programs waiting or put on hold.

What is Inter-Process Communication (IPC)?

IPC is an operating system technique that facilitates data sharing, synchronization, and communication between processes.

What is PCB in OS?

The operating system uses a data structure called a Process Control Block (PCB) to store and handle process-related data.