What is scheduling, Types of scheduling, Scheduling parameters

---

Scheduling is a **core function of an operating system (OS)** that ensures smooth execution of processes by efficiently managing system resources like the **CPU**, **memory**, and **I/O devices**. Without scheduling, multiple tasks would compete for resources, leading to delays, system slowdowns, or crashes.

Scheduling follows specific **rules, algorithms, and priority levels** to maximize **performance, fairness, and efficiency**. Let's explore its types, parameters, algorithms, and real-world applications in depth.

1. Types of Scheduling in OS

Scheduling is broadly classified into three categories based on the timeframe in which they operate:

A. Long-Term Scheduling (Job Scheduling)

- Decides which **processes** enter the system for execution.
- Controls the **degree of multitasking** by limiting the number of active processes.
- Helps in **load balancing** to prevent the system from becoming overwhelmed.

Example:

- If a user launches multiple programs at once (e.g., browser, music player, word processor), long-term scheduling **decides which ones should start executing first** while others wait.

B. Medium-Term Scheduling (Swapping)

- Temporarily **removes processes from RAM** to free space and improve system performance.
- Stores inactive processes in **secondary memory (swap space)** until needed again.

- Prevents unnecessary memory congestion by **prioritizing active processes**.

Example:

- If too many apps are running, the OS may **move inactive ones** (e.g., a paused game) to disk storage until the user resumes playing.

C. Short-Term Scheduling (CPU Scheduling)

- Determines **which process** will get the CPU next.
- Executes processes **based on priority, waiting time, or efficiency**.
- Uses different **CPU scheduling algorithms** to optimize task execution.

Example:

- While browsing the internet and listening to music, the OS **switches the CPU's focus between the browser and the music player** so both work smoothly.

D. I/O Scheduling

- Manages **input and output requests** from various processes.
- Prevents multiple programs from **competing for storage access**, reducing delays.
- Uses **disk scheduling algorithms** to improve efficiency (e.g., First-Come-First-Serve, Shortest Seek Time First).

Example:

- When downloading multiple files, **I/O scheduling organizes disk read/write operations efficiently** to speed up the process.

E. Process Scheduling

- Ensures all processes **get fair access** to system resources.
- Balances **background processes** (e.g., antivirus scan) and **user applications** (e.g., typing in a word processor).
- Works alongside **CPU scheduling** for effective multitasking.

Example:

- While watching a video, process scheduling ensures **the video player has enough CPU and memory**, even if other apps (e.g., a file transfer) are running in the background.

F. Thread Scheduling

- Manages execution **order of threads** within a process.
- Helps programs run **multiple tasks simultaneously** using threads.
- Can be **preemptive** (OS decides execution) or **non-preemptive** (threads control execution).

Example:

- In a web browser, multiple tabs run as separate threads. **Thread scheduling ensures each tab gets processing time** without freezing others.

2. Key Scheduling Parameters in OS

Every scheduling technique depends on specific **parameters** that define how processes are executed:

1. **Burst Time:** The time a process needs to complete execution on the CPU.
2. **Arrival Time:** The moment a process enters the scheduling queue.
3. **Priority:** Defines importance—higher-priority processes run before lower-priority ones.
4. **Turnaround Time:** The total time taken from process arrival to completion.
5. **Waiting Time:** The duration a process spends waiting before execution.
6. **Response Time:** The time from a request being made to the first response.
7. **Throughput:** The number of processes completed in a specific timeframe.

Each of these factors is **important for optimizing system performance** and ensuring fair allocation of resources.

3. CPU Scheduling Algorithms

To efficiently schedule CPU tasks, different **algorithms** are used:

A. First-Come-First-Serve (FCFS)

- Processes are executed **in the order they arrive**.
- Simple and fair but **can cause delays** if a long process arrives first.

Example:

- A printer queue processes documents **in the order** they were submitted.

B. Shortest Job Next (SJN)

- The process with the **shortest burst time** is executed first.
- Minimizes waiting time but **is unfair to longer processes**.

Example:

- If one document takes **5 seconds to print** and another **50 seconds**, the **shorter** one prints first.

C. Round Robin (RR)

- Each process is given **a fixed time slice** before switching to the next.
- Ensures **fair multitasking** and prevents one task from dominating.

Example:

- A web browser **switches between multiple active tabs** so all tabs load properly.

D. Priority Scheduling

- Higher-priority processes **run before lower-priority ones**.
- Prevents delays for **critical tasks** but may cause **starvation** for lower-priority ones.

Example:

- An **urgent system update** might pause a background file download to ensure security.

E. Multilevel Queue Scheduling

- Divides processes into **different priority levels** (e.g., system tasks vs. user applications).
- Each level has a **separate scheduling algorithm**.

Example:

- An **operating system prioritizes security updates and kernel tasks** over user apps like a music player.

4. Real-World Applications of OS Scheduling

Scheduling is used in **various fields** to ensure efficiency:

- **Computers & Smartphones:** Manages multitasking between applications.
- **Servers & Cloud Computing:** Balances resource allocation between multiple users.
- **Manufacturing Systems:** Optimizes machinery operations based on scheduling rules.
- **Healthcare & Robotics:** Ensures real-time execution of critical processes.

5. Summary

Scheduling is the backbone of an OS, ensuring **fairness, efficiency, and responsiveness**. By managing how tasks are executed, scheduling prevents system overload and enhances performance. Different scheduling algorithms optimize execution, balancing speed and priority.