Scheduling algorithms, Evaluation of Scheduling Algorithms

Scheduling algorithms are **sets of rules** used by an OS to manage the execution of multiple processes efficiently. They determine which process gets CPU time, ensuring optimal system performance, fairness, and responsiveness. These algorithms play a crucial role in multitasking systems, preventing delays and resource conflicts.

1. Importance of Scheduling Algorithms

Scheduling algorithms help the OS:

- **Maximize CPU Utilization:** Ensuring the processor is always busy.
- **Minimize Waiting Time:** Reducing time spent in queues.
- **Ensure Fairness:** Preventing process starvation.
- **Optimize Throughput:** Increasing the number of completed processes in a given time.
- **Provide Quick Response Time:** Ensuring interactive programs get immediate attention.

Each algorithm is chosen based on system requirements, balancing between speed, efficiency, and fairness.

2. Types of Scheduling Algorithms

Scheduling algorithms can be broadly classified into **Non-Preemptive** and **Preemptive**.

A. Non-Preemptive Scheduling (Once started, the process cannot be interrupted)

- A process executes until **completion** before moving to another.
- Suitable for batch processing and tasks that must be completed in full.

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

- Processes execute in **arrival order** (first process in, first process out).
- Simple and easy to implement.

- **Disadvantage:** Longer processes can block shorter ones, leading to delays.
- **Example:** A printer processes jobs in the order they were submitted.

2. Shortest Job Next (SJN) / Shortest Job First (SJF)

- The process **with the shortest execution time** runs first.
- Improves **waiting time efficiency** but requires knowing execution time in advance.
- **Disadvantage:** Long processes may face indefinite delay (starvation).
- **Example:** Sorting small file transfers before larger ones.

3. Priority Scheduling (Non-Preemptive)

- **Higher priority processes run first** (based on importance).
- **Disadvantage:** Lower-priority tasks may never execute (starvation).
- **Example:** A system prioritizing emergency software updates before regular tasks.

B. Preemptive Scheduling (Processes can be interrupted)

- The OS can **switch processes based on priority or execution time**.
- Ideal for real-time and interactive systems.

4. Round Robin (RR)

- Each process receives a **fixed time slice (quantum)**.
- Prevents long processes from **dominating CPU time**.
- **Disadvantage:** Shorter quantum values lead to excessive switching (context switching overhead).
- **Example:** A web browser running multiple tabs, ensuring smooth functionality.

5. Shortest Remaining Time First (SRTF)

- The process **with the least remaining execution time** runs next.
- **Disadvantage:** Requires constant monitoring of execution time.
- **Example:** A system prioritizes **quick-loading applications** over slower ones.

6. Priority Scheduling (Preemptive)

- Higher-priority processes **interrupt lower-priority ones**.
- **Disadvantage:** Requires careful priority assignment to avoid starvation.
- **Example:** Emergency security updates pausing routine background tasks.

7. Multilevel Queue Scheduling

- Processes are divided into **priority-based queues**.
- Each queue follows **its own scheduling algorithm**.
- **Example:** System tasks like kernel operations run separately from user applications.

3. Evaluation of Scheduling Algorithms

Each scheduling algorithm is evaluated based on specific **performance criteria**:

Key Performance Metrics for Evaluation:

When analyzing different scheduling algorithms, we consider several important factors:

1. **CPU Utilization**: Measures how effectively the CPU is being used. A well-optimized scheduler ensures that the CPU stays as busy as possible.
2. **Throughput**: Refers to the number of processes completed per unit of time. Higher throughput indicates better efficiency.
3. **Turnaround Time**: The total time taken for a process from submission to completion, including waiting time, execution, and possible delays.
4. **Waiting Time**: The time a process spends in the ready queue before getting CPU execution. Lower waiting times reduce delays.
5. **Response Time**: The time from when a process is submitted until it starts execution. Critical for interactive systems.

The choice of scheduling algorithm depends on the **nature of tasks**—interactive systems prioritize responsiveness, while batch systems favor throughput.

4. Real-World Applications of Scheduling Algorithms

Scheduling plays a critical role in different fields:

- **Operating Systems:** Managing process execution efficiently.
- **Servers & Cloud Computing:** Distributing workloads across multiple users.
- **Manufacturing Systems:** Optimizing machine operations in factories.
- **Healthcare & Robotics:** Managing real-time execution of medical equipment.

Final Thoughts

Scheduling algorithms determine the performance and efficiency of an OS by balancing **speed, fairness, and responsiveness**. Choosing the right algorithm depends on system requirements and workload characteristics.