Difference between scheduling algorithms and when to use each of them

Scheduling algorithms are used in operating systems to manage the allocation of resources, such as CPU time, among multiple processes or threads. Different scheduling algorithms prioritize processes based on various criteria. The choice of scheduling algorithm depends on factors such as the nature of the workload, system goals, and desired performance characteristics. Here's an overview of some common scheduling algorithms and when to use each of them:

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

- In FCFS, processes are executed in the order they arrive in the ready queue.

- Use when fairness is important, and there is no need for optimizing turnaround or response times.

- Not suitable for time-sensitive or interactive applications, as it may lead to poor response times for short jobs.

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

- SJN schedules the process with the shortest burst time next.

- Use when minimizing average waiting time is important.

- May not be practical in real-time systems or when burst time prediction is difficult.

****3. Priority Scheduling:****

- Processes are assigned priorities, and the highest-priority process is scheduled next.

- Use when some processes are more important than others.

- Can lead to priority inversion and starvation issues if not managed properly.

****4. Round Robin (RR):****

- Each process gets a fixed time slice (quantum) to execute before moving to the next process.

- Use when fairness and responsiveness are important.

- Suitable for time-sharing systems and interactive applications.

****5. Multilevel Queue Scheduling:****

- Processes are divided into multiple queues with different priorities, and each queue has its own scheduling algorithm.

- Use when processes have varying levels of importance or different characteristics.

- Helps in managing processes with different requirements, such as foreground and background tasks.

6. Multilevel Feedback Queue Scheduling:

- Similar to multilevel queue scheduling but allows processes to move between queues based on their behavior.

- Use when processes have varying CPU burst characteristics and priorities.

- Adapts to changes in process behavior over time.

7. Priority Inheritance:

- Used in real-time systems to prevent priority inversion.

- A low-priority process inherits the priority of a high-priority process it depends on.

- Ensures that higher-priority processes are not blocked by lower-priority ones.

8. Lottery Scheduling:

- Processes receive lottery tickets and a random ticket is drawn to select the next process.

- Use when fairness and dynamic allocation of resources are important.

- Suitable for situations where different users or processes have different resource requirements.

9. Guaranteed Scheduling:

- Ensures that each process gets a guaranteed amount of CPU time.

- Use in real-time systems or situations where specific performance guarantees are required.

- Ensures predictable and bounded response times for critical tasks.

The choice of scheduling algorithm depends on factors such as the workload characteristics, desired system goals (e.g., fairness, throughput, response time), and specific requirements of the application. Different scheduling algorithms have trade-offs, and the best choice depends on the context in which the operating system is being used.