RV Educational Institutions ®
RV College of Engineering ®

Autonomous Institution
Affiliated to Visvesvaraya Technological University, Belagavi

Approved by AICTE, New Delhi

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

OPERATING SYSTEMS - CS235AI

REPORT

TITLE : CPU SCHEDULING ALGORITHMS.

Submitted by

SHIVAKUMAR.	**1RV22CS184.**
SUDHANSHU SUMAN.	**1RV22CS206.**
SHREERAM SHIVABASU BADAGANDI.	**1RV22CS188.**

Computer Science and Engineering
2023-2024

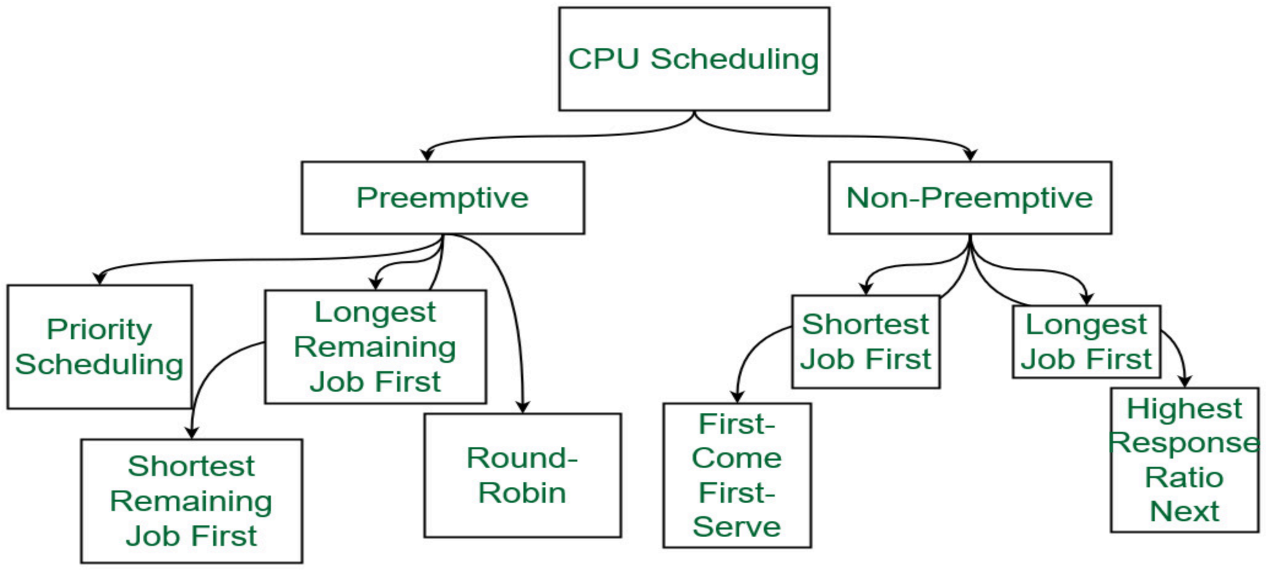
Contents.

Introduction :

CPU scheduling is a fundamental aspect of operating system design, aiming to optimize the utilization of the central processing unit (CPU) by efficiently managing the execution of multiple processes. In a multitasking environment where several processes compete for CPU time, scheduling algorithms play a crucial role in determining the order and duration of execution for each process. The primary goal of CPU scheduling is to achieve high system throughput, minimize response time, and ensure fairness among competing processes.

At the heart of CPU scheduling lies the scheduler, a component within the operating system responsible for selecting which process to execute next from the pool of ready processes. Various scheduling algorithms exist, each with its own set of characteristics and trade-offs. These algorithms can be broadly categorized into preemptive and non-preemptive scheduling. Preemptive scheduling allows a running process to be interrupted and placed back into the ready queue, whereas non-preemptive scheduling allows a process to run until it voluntarily relinquishes the CPU.

The choice of scheduling algorithm depends on the specific requirements of the system and the desired performance metrics. Common scheduling algorithms include First Come, First Served (FCFS), Shortest Job Next (SJN), Round Robin (RR), Priority Scheduling, and Multilevel Queue Scheduling. Each algorithm exhibits unique behavior in terms of CPU utilization, throughput, waiting time, and fairness. Understanding these algorithms and their implications is crucial for system designers and administrators to effectively manage system resources and optimize overall system performance.



First Come First Serve CPU Scheduling :

The First Come, First Serve (FCFS) scheduling algorithm is one of the simplest CPU scheduling algorithms, operating on the principle of processing tasks based on their arrival times. In FCFS, the process that arrives first is allocated the CPU first, hence the name. This algorithm employs a FIFO (First-In-First-Out) queue to manage the order of processes awaiting execution. When a process enters the ready queue, its Process Control Block (PCB) is appended to the tail of the queue, indicating its position in the line of processes. When the CPU becomes available, it is assigned to the process at the head of the queue, which is the one that arrived earliest among the ready processes.

Upon allocation of the CPU to a process, it begins its execution, and the process is subsequently removed from the queue. The FCFS algorithm operates in a non-preemptive manner, meaning that once a process starts executing, it continues until it completes its execution or voluntarily yields the CPU. This lack of preemption implies that once a process gains control of the CPU, it maintains control until it either completes its execution or enters a waiting state, such as I/O operations. Consequently, FCFS scheduling is straightforward to implement and understand, making it suitable for environments where simplicity is prioritized over optimization of system performance metrics.

Despite its simplicity, FCFS scheduling may lead to potential inefficiencies, especially in scenarios where short processes are queued behind long-running processes, causing longer waiting times for shorter jobs. This phenomenon, known as the convoy effect, can result in poor overall system performance, particularly in environments with a mix of short and long processes. While FCFS is easy to implement and intuitive, its drawbacks have led to the development and adoption of more sophisticated scheduling algorithms that aim to address the shortcomings of FCFS and optimize various performance metrics such as response time, throughput, and fairness.

Shortest Job First CPU Scheduling :

The Shortest Remaining Job First (SRJF), also known as Shortest Remaining Time First (SRTF), is a preemptive CPU scheduling algorithm that selects the process with the smallest amount of time remaining until completion for execution. In SRJF, the scheduler constantly compares the remaining burst time of all ready processes and selects the one with the shortest remaining time to execute next. This approach ensures that processes with shorter execution times are prioritized, leading to minimized average waiting times and improved system responsiveness.

Since the currently executing process in SRJF is by definition the one with the shortest remaining time, it will continue execution until it either completes or a new process arrives with a smaller burst time. As execution progresses, the remaining time of the currently running process decreases, ensuring that it maintains its status as the shortest remaining job. However, if a new process enters the ready queue with a smaller remaining time, the CPU is preempted, and the newly arrived process with the shortest remaining time is scheduled for execution. This preemptive nature allows for efficient utilization of CPU resources and ensures that shorter jobs are given priority.

While SRJF scheduling is effective in minimizing average waiting times and improving system responsiveness, it can potentially lead to a situation known as starvation. Starvation occurs when longer processes are continuously preempted by shorter processes, causing them to wait indefinitely for CPU time. To mitigate starvation, priority-based mechanisms or aging techniques can be implemented, where the priority of processes gradually increases over time, ensuring that all processes eventually get a chance to execute. Despite the challenges associated with starvation, SRJF remains a popular scheduling algorithm due to its ability to optimize system performance by prioritizing shorter jobs.

Round Robin CPU Scheduling :

Round Robin (RR) is a popular CPU scheduling algorithm that provides a fair and balanced approach to executing processes. In RR, each process is cyclically assigned a fixed time slot known as the time quantum or time slice. This time quantum dictates the maximum duration a process can execute before being preempted and moved to the back of the ready queue to await its next turn. RR is considered the preemptive version of the First Come First Serve (FCFS) CPU scheduling algorithm, as it ensures fairness among processes by allowing each process to run for a specified time quantum before moving on to the next.

Round Robin CPU scheduling primarily focuses on time-sharing techniques, making it suitable for multitasking environments where multiple processes compete for CPU time. By allocating fixed time slices to each process in a round-robin fashion, RR ensures that all processes get a fair share of the CPU's resources, regardless of their arrival time or execution characteristics. This approach helps prevent any single process from monopolizing the CPU for an extended period, thus improving system responsiveness and fairness.

In RR, if a process completes its execution within the allocated time quantum, it is removed from the ready queue. However, if the process's execution exceeds the time quantum, it is preempted and placed back into the ready queue to await its next turn. This preemptive nature of RR ensures that no process hogs the CPU indefinitely, enabling better resource utilization and response time. Overall, Round Robin CPU scheduling strikes a balance between fairness and efficiency, making it a widely used algorithm in modern operating systems for managing CPU resources in multitasking environments.

Priority Scheduling :

Priority scheduling is a prevalent CPU scheduling algorithm utilized in batch systems, where each process is assigned a priority value. The priority value determines the order in which processes are executed, with higher priority processes given precedence over lower priority ones. This prioritization ensures that critical or time-sensitive tasks are handled promptly. In priority scheduling, the process with the highest priority is selected for execution first, followed by processes with lower priorities in descending order. However, if multiple processes share the same priority, they are typically executed based on a first-come, first-served basis, adhering to the principle of fairness among processes of equal importance.

The assignment of priorities in priority scheduling can be based on various factors, such as memory requirements, time constraints, or resource needs of the processes. For instance, processes requiring extensive memory resources or having strict time deadlines may be assigned higher priorities to ensure their timely execution. Additionally, priorities can be determined by the ratio of average I/O to average CPU burst time, where processes with a higher ratio may be given higher priorities to optimize overall system performance and resource utilization. This flexible approach allows system administrators to tailor priority assignments based on the specific requirements and characteristics of the workload.

While priority scheduling offers advantages in terms of responsiveness and meeting critical task deadlines, it also poses challenges, particularly in scenarios where lower priority processes may suffer from starvation. Starvation occurs when high-priority processes continuously preempt lower-priority ones, leading to prolonged waiting times or even indefinite delays for low-priority tasks. To mitigate starvation, priority aging techniques may be employed, where the priority of waiting processes gradually increases over time, ensuring that all processes eventually receive CPU time. Despite its complexities and potential drawbacks, priority scheduling remains a valuable tool for optimizing system performance and meeting diverse workload requirements in batch processing environments.

Importance of Project :

- **Resource Utilization:** Effective CPU scheduling ensures that the CPU is utilized efficiently. By keeping the CPU busy with processes, the system can maximize throughput and minimize idle time.
- **Fairness:** CPU scheduling ensures fairness among processes. It allocates CPU time fairly among competing processes, preventing any single process from monopolizing the CPU and starving others.
- **Response Time:** Good scheduling algorithms aim to minimize the response time for interactive processes. This improves the overall user experience by ensuring quick responses to user inputs.
- **Turnaround Time:** CPU scheduling affects the turnaround time, which is the total time taken to execute a process from submission to completion. Minimizing turnaround time is important for enhancing system performance and user satisfaction.
- **Throughput:** Throughput refers to the number of processes completed per unit of time. Efficient CPU scheduling can increase throughput by optimizing the execution of processes, leading to better system performance.
- **Priority Management:** Some processes may have higher priority than others, based on factors such as system requirements or user preferences. CPU scheduling allows for the management of process priorities, ensuring that critical tasks are executed in a timely manner.
- **Resource Allocation:** CPU scheduling is essential for allocating system resources effectively. It determines which processes get access to the CPU and for how long, as well as managing other resources such as memory and I/O devices.
- **Adaptability:** Different scheduling algorithms are suited to different types of systems and workloads. CPU scheduling allows for the selection and implementation of appropriate algorithms based on the specific requirements and characteristics of the system. Overhead
- **Reduction:** Efficient CPU scheduling can help reduce overhead associated with context switching and process management. By minimizing unnecessary overhead, system performance can be improved.

- **Predictability:** Predictable behavior is crucial for real-time systems and applications. CPU scheduling plays a role in ensuring predictable performance by maintaining consistent response times and meeting deadlines for time-sensitive tasks.

Conclusion :

In conclusion, CPU scheduling is a vital aspect of operating system design, playing a crucial role in optimizing the utilization of CPU resources and ensuring efficient execution of processes. Throughout the evolution of computing systems, various scheduling algorithms have been developed, each with its own set of characteristics, advantages, and drawbacks. From the simple First Come First Serve (FCFS) algorithm to more sophisticated approaches like Shortest Remaining Job First (SRJF), Round Robin, and Priority scheduling, each algorithm addresses specific system requirements and performance metrics.

The choice of CPU scheduling algorithm depends on factors such as system workload, response time requirements, fairness considerations, and resource constraints. While some algorithms prioritize fairness and responsiveness, others focus on maximizing throughput or minimizing average waiting times. Additionally, preemptive scheduling techniques allow for better resource utilization by allowing processes to be interrupted and rescheduled based on dynamic priorities or time slices.

In modern computing environments, where multitasking and real-time processing are commonplace, the importance of efficient CPU scheduling cannot be overstated. As technology continues to evolve, the development of new scheduling algorithms and optimization techniques remains an active area of research and innovation. Ultimately, effective CPU scheduling contributes to overall system performance, user satisfaction, and the seamless operation of diverse computing applications.

References :

- Simulation of Priority Round-Robin Scheduling Algorithm
  Tri Dharma Putra, Rakhmat Purnomo Universitas Bhayangkara Jakarta Raya rakhmat.purnomo@dsn.ubharajaya.ac.id
  Submitted : Aug 9, 2022 | Accepted : Aug 18, 2022 | Published : Oct 3, 2022
  link : https://jurnal.polgan.ac.id/index.php/sinkron/article/download/11665/1135.

- Performance analysis of FCFS and improved FCFS scheduling algorithms for dynamic real-time computer systems (IEEE).
- An Improved SJF Scheduling Algorithm in Cloud Computing Environment By Mokhtar A. Alworafi,DoS in CS,Mysore University, India