|  |  |
| --- | --- |
| Multilevel QUEUE Scheduling | |
|  | |
|  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  | What is the CPU scheduling...?CPU scheduling is a fundamental aspect of computer multitasking operating systems. It's the process by which the operating system decides which tasks (or processes) should be executed by the CPU (Central Processing Unit) and in what order. The primary goal of CPU scheduling is to maximize CPU utilization, throughput, and fairness among processes while minimizing response time, turnaround time, and overhead.In essence, CPU scheduling involves selecting a process from the ready queue (a list of processes waiting to be executed) and allocating the CPU to that process for execution. This selection is typically done using various scheduling algorithms, such as First Come, First Served (FCFS), Shortest Job Next (SJN), Round Robin (RR), Priority Scheduling, etc. Each algorithm has its own characteristics and trade-offs, which affect system performance in terms of efficiency and fairness.Overall, CPU scheduling plays a crucial role in ensuring efficient utilization of the CPU resources and maintaining a responsive system for users and applications. | |  |
|  | What is the Round Robin Robin(RR) algorithms...?What is the Shortest Job First(SJF) algorithms...?What is the First-In-First-Out(FIFO) algorithms...? | |  |
|  | What is the Round Robin Robin(RR) algorithms...? Round Robin (RR) scheduling is one of the most widely used CPU scheduling algorithms in multitasking operating systems. It's designed to provide fair allocation of CPU time among multiple processes while also ensuring responsiveness and preventing starvation.  In Round Robin scheduling, processes are assigned CPU time in a cyclic manner, where each process is given a small unit of time (called a time quantum or time slice) to execute. When a process's time quantum expires, it's preempted and placed back at the end of the ready queue, allowing the next process in the queue to execute. This continues until all processes have had a chance to execute. | Key features of Round Robin scheduling include:  1.Fairness: Since each process gets an equal share of CPU time initially, Round Robin provides fairness among processes. This is particularly important in time-sharing systems where multiple users may be running programs simultaneously.  2.Preemption: Processes are preempted after using their time quantum, ensuring that no single process monopolizes the CPU for an extended period. Preemption allows for better responsiveness and prevents starvation of low-priority processes.  3.Simple Implementation: Round Robin scheduling is relatively simple to implement compared to some other scheduling algorithms like Priority Scheduling or Shortest Job Next (SJN).  However, Round Robin scheduling may suffer from certain drawbacks:  1.High Context Switching Overhead: Preempting processes and switching between them incurs overhead due to saving and restoring process states, which can impact overall system performance.  2.Long Average Response Time: For CPU-bound processes with time slices that are too short, Round Robin scheduling may lead to longer average response times, as processes are constantly being preempted and rescheduled.  Despite these drawbacks, Round Robin scheduling is widely used in operating systems, especially in scenarios where fairness and responsiveness are important considerations. Tweaking the time quantum can help balance between fairness and responsiveness based on system requirements and workload characteristics. |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  | What is the Shortest Job First(SJF) algorithms...? Shortest Job Next (SJN), also known as Shortest Job First (SJF), is a CPU scheduling algorithm that selects the process with the smallest burst time (execution time) to execute next. It belongs to the non-preemptive scheduling category, meaning once a process starts executing, it runs to completion unless it voluntarily relinquishes the CPU or a new process with a shorter burst time arrives.  The key idea behind SJN scheduling is to minimize the average waiting time of processes by prioritizing the shortest jobs first. This strategy aims to reduce the turnaround time of processes and optimize overall system performance.  Here's how Shortest Job Next scheduling works:  Arrival of Processes: As processes arrive in the ready queue, the scheduler selects the one with the smallest burst time. If multiple processes have the same shortest burst time, the tie can be resolved using various methods, such as First Come, First Served (FCFS) for tie-breaking.  Execution: The selected process is then dispatched for execution on the CPU. It continues to execute until it completes its CPU burst.  Completion: Once a process finishes its CPU burst, it's removed from the system, and the scheduler selects the next shortest job from the remaining processes in the ready queue. | SJN scheduling offers several advantages:  1.Optimized Waiting Time: By prioritizing shorter jobs, SJN scheduling tends to minimize the waiting time of processes in the ready queue, leading to efficient resource utilization.  2.Improved Turnaround Time: Since shorter jobs are executed first, processes typically experience shorter turnaround times, which is the total time taken to execute a process from arrival to completion.  However, SJN scheduling also has some limitations:  1.Convoy Effect: Long processes may suffer from starvation if shorter processes frequently arrive, causing them to wait indefinitely in the ready queue.  2.Predicting Burst Time: In practice, accurately predicting the burst time of processes beforehand may be challenging, which can affect the effectiveness of SJN scheduling.  3.Despite these limitations, SJN scheduling is widely used in scenarios where burst times can be estimated accurately or in environments with a predictable workload. Additionally, variations of SJN, such as Shortest Remaining Time First (SRTF), which is preemptive, aim to address some of its limitations. What is the First-In-First-Out(FIFO) algorithms...? First Come, First Served (FCFS) scheduling is one of the simplest CPU scheduling algorithms used in operating systems. In FCFS scheduling, the processes are executed based on their arrival time: the process that arrives first is allocated the CPU first, and so on.  Here's how FCFS scheduling works:  1.Arrival of Processes: As processes enter the system, they are placed in a queue called the ready queue. The processes are ordered in the ready queue based on the order of their arrival, with the process that arrives first at the front of the queue.  2.Execution: When the CPU becomes available, the operating system selects the process at the front of the ready queue (the one that arrived first) and allocates the CPU to it for execution.  3.Completion: The selected process continues to execute until it completes its CPU burst or is interrupted by an event such as I/O request or a higher-priority process becoming available.  FCFS scheduling is easy to implement and understand, but it has some notable characteristics and limitations:  1.Fairness: FCFS scheduling ensures fairness among processes in the sense that each process gets a turn to execute in the order it arrived. However, it may not be the most fair in terms of response time or overall performance.  2.Non-Preemptive: FCFS scheduling is non-preemptive, meaning once a process starts executing, it runs to completion unless it voluntarily relinquishes the CPU or is interrupted by an event.  3.Convoy Effect: Long processes may cause shorter processes to wait for an extended period, leading to inefficient resource utilization. This phenomenon is known as the convoy effect.  4.Average Waiting Time: The average waiting time of processes can be relatively high in FCFS scheduling, especially if long processes arrive early.  Overall, FCFS scheduling is simple to implement and suitable for scenarios where process arrival times are known in advance and fairness among processes is the primary concern. However, it may not be the most efficient scheduling algorithm in terms of minimizing average waiting time or maximizing system throughput. |  |
|  |  | |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  | This shows how to work my code. |  |  |