Operating Systems – Assignment 1 Report

Group : Mohieddine Farid & Ayman Youss

CPU Scheduling Simulator Web Platform

FCFS

The First-Come, First-Served (FCFS) scheduling algorithm is a simple method used to manage process execution in computing environments. It functions under a straightforward principle: processes are executed in the order they arrive, without preemption. Here’s a step-by-step breakdown of how the FCFS scheduling algorithm works:

1. Sorting by Arrival Time: The algorithm begins by sorting all the processes based on their arrival times.
2. Execution of Processes: Starting with the earliest arrived process, the algorithm checks if the current time (initially set to zero) is less than the arrival time of the process. If the current time is earlier than the process's arrival, the algorithm waits until the process arrives.
3. Process Start and Completion Times:

* Start Time: Each process starts execution at either its arrival time (if the system is idle) or immediately after the previous process has finished, whichever is later.
* Completion Time: This is calculated by adding the burst time (duration the process needs the CPU) to its start time.

1. Calculation of Waiting and Turnaround Times:

* Waiting Time: The duration a process spends waiting in the queue before its execution starts. It is the difference between the start time and the arrival time.
* Turnaround Time: The total time taken from arrival to completion of the process. It is the difference between the completion time and the arrival time.

1. Tracking Time and Performance: After each process is executed, the scheduler updates the current time to the completion time of the recently finished process. It also accumulates the waiting and turnaround times to calculate the average waiting and turnaround times after each process execution.
2. Logging and Historical Data: Throughout its operation, the FCFS algorithm keeps a log of significant events and metrics. This includes the average waiting and turnaround times at the completion of each process, which will be used for plots in the web app.

Priority Scheduling

The processes are initially sorted by their arrival time to manage them in order of their arrivals. The function sets up several variables to track the current time, completed processes, the ready queue, and performance metrics such as waiting and turnaround times.

1. Processing Loop

The outer while loop runs as long as there are unprocessed processes in the main list or processes waiting in the ready queue.

1. Filling the Ready Queue

Processes are moved from the main list to the ready queue as they arrive (i.e., their arrival time is less than or equal to the current time). If the ready queue is empty and there are still processes waiting to arrive, the current time is advanced to the arrival time of the next process.

1. Sorting the Ready Queue

The ready queue is sorted by priority. In this implementation, a lower numerical value represents a higher priority.

1. Executing the Process

From the ready queue, the process with the highest priority (after sorting) is selected and executed:

* Start Time: Set to the current time.
* Completion Time: Calculated by adding the burst time (time needed for execution) to the start time.
* Waiting Time: Time the process has waited in the queue, which is the difference between the start time and the arrival time.
* Turnaround Time: Total time from arrival to completion.

After executing the process, it is moved to the completedProcesses list, and its waiting and turnaround times are added to the total metrics.

1. Logging History

The history log captures the current time, average waiting time, and average turnaround time after each process execution.

SJF

The processes are first sorted by their arrival time. This sorting ensures that they are initially considered in the order they arrive.

1. Processing Loop

The algorithm operates within a loop that continues as long as there are processes that have not been completed or are waiting to be processed.

1. Sorting by Burst Time: The ready queue is sorted based on the burst time, with the shortest burst time first. This is the core of the SJF algorithm, where the shortest job is given preference.
2. Executing the Process:

* Start Time: The process begins execution at the current time.
* Completion Time: This is calculated by adding the burst time to the start time.
* Waiting Time: The interval the process has waited in the queue, calculated from the start time minus the arrival time.
* Turnaround Time: Total time from the process's arrival to its completion.

After processing, the current time is updated to the completion time of the current process.

1. Performance Tracking: Each completed process is added to the completedProcesses list, and its waiting and turnaround times are accumulated to calculate averages.
2. Historical Data Logging: The algorithm logs historical data such as the current time and average waiting and turnaround times after each process is completed.

Round Robin

The processes are sorted by their arrival time to handle them in the order they come.

1. Scheduling Loop

The function operates within a loop that continues as long as there are unprocessed processes either in the main list (processList) or waiting in the queue. If the queue is empty and there are processes waiting to arrive, the current time (t) is set to the arrival time of the next process, and that process is moved to the queue.

1. Processing a Process:

* Start and End Time: A process is picked from the queue, and its execution is simulated from the current time (t) for a duration up to the defined timeQuanta or the remaining burst time of the process, whichever is less.
* First Time Processing: If this is the first time the process is getting CPU time, its start time is set.

1. Handling Process Arrival During Execution: Any new processes arriving during the current process's execution are added to the queue.
2. Updating Process and Time: The remaining burst time of the current process is reduced by the time it was allowed to run. The global time (t) is updated to the end time of the current execution.
3. Queue or Complete: If the process still requires more CPU time (remaining burst time > 0), it is added back to the end of the queue. If it's completed, various metrics like completion time, waiting time, and turnaround time are calculated and recorded. The process is then moved to the completed list.
4. Recording History: After each process completion, the function logs the current time and the average waiting and turnaround times up to that point.

Priority Round Robin

**Class Structure and Initialization**

Constructor: Initializes internal properties for managing processes, such as separate queues for each priority level (queues), a global clock (currentTime), accumulative waiting and turnaround times, and the scheduling quantum (quanta). The constructor also calculates the minimum and maximum priorities based on the process list, which helps in iterating through the priorities during scheduling.

enqueueProcess Method: Adds a process to the appropriate queue based on its priority. If the queue for a given priority doesn't exist, it is created.

**Scheduling Processes**

scheduleProcesses Method: This method is the core of the scheduler. It processes an input list of processes based on their arrival times and priorities.

Sorting: Initially, it sorts the process list by arrival time to handle them in order.

Process Handling: As time progresses, processes are added to their respective priority queues when they arrive. If all queues are empty but there are still processes that haven't arrived, the current time is advanced to the next process's arrival time.

Execution: Processes are executed starting from the highest priority (lowest numerical value) to the lowest. For each process:

It calculates the time slice for execution, which is the lesser of the process's remaining burst time or the quantum.

Updates the process's remaining burst time and the scheduler's current time.

Logs the execution details in a Gantt chart log.

If a process finishes (remaining burst time reaches zero), it records its completion time, calculates its waiting and turnaround times, and updates total waiting and turnaround times. Otherwise, it re-queues the process.

History Tracking: After each process's completion, historical data on average waiting and turnaround times is recorded.

**Utility Methods**

allQueuesEmpty Method: Checks if all priority queues are empty, indicating that there are no more processes to schedule.

Choice of technology for the project :

JavaScript (html css and node.js): Chosen for backend processing and scheduling logic, even though this is a non-traditional choice for CPU scheduling simulation typically seen in lower-level languages, this choice make us benefit from JavaScript's event-driven capabilities and easy-to-implement charts which are especially useful in web applications.

Test Case Loading and Validation: The test cases for different scheduling algorithms are loaded from a JSON file, and results are validated against expected outcomes which ensures algorithm correctness.