كلية الهندسة بشبرا
FACULTY OF ENGINEERING AT SHOUBRA

Multiprocessor Process Scheduling Simulation

Project Team:
1-Bassant Ehab Saber
2-Toqa Sameh Salah
3-Rawan Mohamed Fathy
4-Mariam Reda Ebrahim
5-Nermeen Ahmed Fouad

DESIGN CHOICES

Data Structures:

- **Linked List:** A linked list is used to represent the queues for ready processes, running processes, waiting processes, and the overall timeline of executed processes.
- **Structures:** Process and Core structures are used to define the attributes of a process and a core, respectively.

Global Variables:

- **Process Queues:** readyHead, runningHead, waitingHead, and queueHead represent the queues for different states of processes.
- **Time:** Time is a global variable used to track the current time in the simulation.
- **Cores:** An array of Core structures (cores) represents the available cores.
- **Quantum:** The quantum time for the Round Robin scheduling is a predefined constant (quantum).
- **Time Levels (Time_Level):** The time interval after which processes are moved between priority levels.

Constants:

- **Time Quantum:** quantum is set to 2, representing the time quantum used in the scheduling algorithm.
- **Time Level:** Time_Level is set to 5, representing a time level for processing.

Algorithms Implemented

1-FCFS (First-Come, First-Served) scheduling:

The FCFS algorithm is one of the simplest scheduling algorithms, where the tasks are executed in the order of their arrival. In this simulation, processes are scheduled to run on a set of cores based on their arrival times .

Cores are allocated to processes in a First-Come, First-Served manner.
If a core is available, a process is dequeued from the ready queue and assigned to the core.

Running Process Execution:
The running processes decrement their burst time in each time step.
If the burst time becomes 0, the process is marked as completed, and relevant information is updated.

IO Handling and Waiting Processes:
Processes in the waiting queue are checked for completion of IO time.
If IO time is completed, the process is either moved to the running queue (if a core is available) or remains in the waiting queue.

Time Advancement and Printing:
Time is advanced in each iteration
The state of queues and processes is printed for each time step.

2-Shortest Job First (SJF) Scheduling Algorithm:

The SJF scheduling algorithm is a non-preemptive scheduling algorithm that selects the process with the shortest burst time to execute first. The basic idea is to minimize the average waiting time of processes by prioritizing the shortest jobs. The algorithm assumes that the burst time of a process is known in advance.

Algorithms Implemented:

Processes are enqueued to the ready queue based on their arrival time.The simulation loop continues until all queues are empty .
Cores are allocated to processes based on the Shortest Job First algorithm.
The process with the shortest remaining burst time is selected for execution on each core.

Running Process Execution:

The running processes decrement their burst time in each time step.
If the burst time becomes 0, the process is marked as completed, and relevant information is updated.

IO Handling and Waiting Processes:

Processes in the waiting queue are checked for completion of IO time . If IO time is completed, the process is either moved to the running queue (if a core is available) or remains in the ready queue.

Time Advancement and Printing:

Time is advanced in each iteration The state of queues and processes is printed for each time step.

3-Shortest Time to Completion First (STCF) Scheduling Algorithm:

Shortest Time to Completion First (STCF) algorithm, which prioritizes processes with the shortest remaining burst time. This choice aims to minimize the total completion time of all processes

Algorithms Implemented:

New processes are enqueued based on their arrival time. If the arrival time is less than or equal to the current simulation time, the process is moved from the queue to the ready queue.
Processes are assigned to cores based on the STCF algorithm. The core with the shortest remaining burst time is selected for process execution.

Running Process Execution:

Running processes are decremented in burst time. If a process completes its burst time or requires I/O, it is moved to the appropriate queue (waiting or remaining).

IO Handling and Waiting Processes:

Waiting processes are decremented in burst time. If a waiting process completes its burst time or requires I/O, it is moved to the appropriate queue (remaining or ready).

Time Advancement and Printing:

Time is advanced in each iteration
The state of queues and processes is printed for each time step.

4-Round Robin (RR) Scheduler Simulator Documentation:

Algorithms Implemented:

- Processes are scheduled in a circular manner on available cores.
- Each process runs for a time quantum, and if it doesn't complete, it moves to the ready queue.
- Processes in the waiting queue can transition to the running queue based on their IO time and core availability.
- The Round Robin scheduling is implemented using the concept of a time quantum for each process.

Time Advancement and Printing:
Time is advanced in each iteration
The state of queues and processes is printed for each time step.

5-Multi-Level Feedback Queue (MLFQ) Scheduler:

there are three priority levels: High, Mid, and Low. Processes move between these levels based on their completion in the current level and the expiration of the Round Robin quantum.

Algorithms Implemented:

- there are three priority levels: High, Mid, and Low.
- Processes move between these levels based on their completion in the current level and the expiration of the Round Robin quantum.
- Processes are initially placed in the ready queue.
- Cores are allocated to processes based on priority levels (High, Mid, Low) in a Round Robin manner.
- Processes that complete their burst time move to a lower-priority level.
- Processes in the waiting queue may move to the running queue or higher-priority levels based on IO completion and core availability.

6-Priority scheduler:

there are priority for each process and it is determine which one will enter first.

Algorithms Implemented:

- Priority Scheduling: The main algorithm implemented in this code is priority-based scheduling. It assigns priorities to each process and schedules them based on their priorities. Higher priority processes are given preference in execution.
- Create a process queue and initialize it with processes and their attributes.
- Sort the process queue based on priorities (highest to lowest).
- For preemptive scheduling:If a new process with higher priority arrives, preempt the current process.
- For non-preemptive scheduling:Allow the currently running process to complete before switching.
- Select the process at the head of the queue for execution.
- Update process states, waiting times, and turnaround times.
- Repeat steps 3-6 until all processes are completed.

7-Stride Scheduler:

the processes are arranged with respect to there arrival time then passing them to the running queue , the process with the minmim pass will run.

,

Algorithms Implemented:

- **Stride Length Optimization:** This algorithm calculates the optimal stride length

Time Advancement and Printing:

Time is advanced in each iteration

The state of queues and processes is printed for each time step.

Instructions for Using the Simulator

1-Compilation:

Compile the code using a C compiler. For example:
make

2-Execution:

Run the compiled executable, providing an input file as a command-line argument.

ex: ./main file.txt

3-Input File Format:

- The input file should contain process details in the following format:

Number of processes
Number of cores
ArrivalTime
tIOtime
CpuTime
BrustTime
Priority

4-Output:

- The simulator will print the state of queues and processes for each time step and
  the turnround time and response time

5-Analysis:

- Analyze the average response time and average turnaround time printed at the end of the simulation.