B206 Operating Systems

CPU Scheduler

Ahmed Khaled Ebrahim Mohammed

GH1029750

Prof. Mazhar Hameed

GitHub page:

https://github.com/Ahmed-Mohammed-x/CPUScheduler

Guide:

https://youtu.be/tGcD-tqT4fE

Introduction:

This is my CPU Scheduler Simulator that simulates how the CPU operates when it comes to scheduling processes. It is important for the CPU to be able to decide which processes can be processed first as it can heavy impact system performance and affect the user experience.
I implemented 3 CPU scheduling algorithms in this program:

- First Come First Serve (FCFS): a non preemptive algorithm that executes upcoming processes based one who comes first, hence the name, first come first served!
- Shortest Job First (SJF): A non preemptive algorithm as well that selects who has the shortest burst time/ remaining execution time to execute first
- Round Robin (RR): an interesting preemptive algorithm that uses time quantum to order processes. Below is detailed explanation of how it works. Or check the video above!

The goal of this program is to showcase visually how the CPU execute these processes according to the algorithm selected.

System Design:

This scheduler is designed to comply with the functionalities requested in the assessment brief as well as a modular design in mind. It can also help add further features like for example, Right now it only shows as if you have one processor aka one thread to handle. Future implementations will cover multithreading and assigning cores to see how these algorithms work when multiple process come at the same time. 4 main classes are made:

- **Process class:**
    - This class was made to assign attributes as it represents different processes like:
    - **Process ID (Pid):** Unique value to identify each process
    - **Arrival time:** The time in which a new process arrive
    - **Burst time:** The total time the CPU needs to execute a process
    - **Remaining time:** The time left to completely execute a process
    - **Waiting time:** The time where a process waits in ready queue
    - **Turnaround Time:** The total time from the arrival of the process in the CPU to the full completion of the process.
    - **State:** the current state of the process whether it was waiting, readey, running or completed

- **Scheduler class:**
    - This is the brain of the CPU scheduler where:
    - It implements the algorithms selected
    - Maintains the state of the processes
    - Make all necessary calculations and tracks all necessary data

- **CPUSschedulerSimulator class:**
  - This class is the interface that the user interacts with to select the desired algorithm and observe how the CPU handles these processes.
  - It provides a a realistic simulation and display the process as it happens before showing the final output
- **Colours class:** This class just provides colours in the console to help the user digest the process easier as it runs and distinguish between lines

Summary: These classes work together to take the input of the user according to the simulation he wants to run, in which he chooses the desired algorithm, choose the number of processes, fill in the necessary data (pid, arrtime etc) and choose the time quantum (if he chose RR algorithm). Then when users runs the simulation, it will simulate according to their input and show in real time how the cu schedule handles it accordingly.

Implementation:

Different key implementations were done to make the code run as intended and following the rules from the assessment brief as well as professor's feedback input.

- **Lists** were used to store all processes
- **Algorithms** (FCFS, RR, SFJ) were implemented as these are common and show variety (non preemtive, preemtive)
  - They can be found in the Scheduler's class

- **Queues:** Preserves the different states of the processes as time passes and it gets incremented

- **<u>Visual Display</u>** was implemented for the user to observe how each algorithm work when given the same process Input.

Challenges and Solutions:

➢ <u>Round Robin Algorithm:</u>

- There were some problems and learning curves while implementing this algorithm specifically. First was understanding what time quantum in this algorithm mean as well as how to implement it successfully.
- **<u>Time quantum</u>** is the time where a cpu allow a processs to run before it takes it off and bring the next one according to arrival time. If the process still didn't completely execute, it goes to queue again and wait for its turn. The queue system runs based on the arrival time.
- **<u>Main Issue:</u>** processes where going back to queue even when not selecting the Rr algorithm
    - **<u>Solution:</u>** Change the code to make it only do so when selecting the Rr algorithm

➢ <u>Displaying accurate and well aligned results:</u>

- There was a problem with displaying the results and colours the way I want align the box in the end
    - **<u>Solution:</u>** learned how to implement that in Java like the /t/t command for example

- **<u>Miscellaneous:</u>**

    - There were some issues running like producing wrong output, calling wrong variables or functions, wrong colour outputs compared to what I want to view, adding or removing comments, and many small issues like these were debugged and fixed to run the code as intended :)

Conclusion:

This code was such a learning curve as I had no prior experience with java or Intellij but it was interesting as I had to learn a lot of basic as well as some advanced Java implementations. Unfortunately, due to time limitations, only a basic scheduler was made but that only means making it better is a great opportunity for my personal development in the long run. Nevertheless, this program should successfully fulfill all requirements and achieve the following:

- Implementing 3 different CPU scheduling algorithms namely, First come first served, Shortest Job First, and Round Robin algorithms
- A user interface that helps the user configure and view everything seamlessly
- Calculate and show all desired calculations\

Issues that require Immediate fixes

- SJF needs a null check as it may crash with a NullPointerException
- Round robins waiting time needs better accuracy

Future Plans:

- Adding more Algorithms
- Memory management: add memory management to make it seem like a real example of resources being allocated and solved by the cpu
- Graphical interface: for example using the swing library to make it more user-friendly
- Add better graphical visualization of the process like making different diagrams for each processs
- Multi core implementation where more than 1 core can exist so the user can see how the CPU handles processes when it has more than 1 core

Please note that the code for the colour class and implementation were from there resources:

Video: https://youtu.be/MERwLlyIJvs?si=8kEEfp4HlztxR5xv

Source: https://www.w3schools.blog/ansi-colors-java

Thank you!