OS Algorithms Project

How to Run

1. Unzip the project folder
2. Pip install requirements.txt file using the following command:

   ```
   pip install -r requirements.txt
   ```

3. When inside the OS_Algos folder, run the following command:

   ```
   python algo.py
   ```

4. The program will ask you if you want to run all the algorithms.
5. The input processes will be stored in the processes_list.csv file.
6. The output will be stored in the output.csv file.
7. If the python file stalls as in it does not show any output, then just cntrl+c and run the file again.

Questions and Answers

This is for the questions asked in the project assignment.

The fully hand analyzed answers to the processes given in the assignment doc are in a examSim.csv file, with all the work shown in examSim.pdf

Q1. The possibility of starvation:

- Round Robin: There is a very small possibility of starvation as the processes are given a time quantum that is extremely small and thus are given a chance to run.
- HRRN: There is a low chance of starvation if a processes since the waiting time in the equation balances out everything pretty well.
- SRT: There is a high chance of starvation if a process has a very long service execution time, it would be pushed back and never executed if there are always processes that have a smaller execution time.
- FCFS: There is a very small chance of starvation as the processes are executed in the order they arrive. And even with them exiting for I/O, after they are done with the IO jobs, they will still be executed.

Q2. The effect of context switching on the throughput:

- Round Robin: If the time quantum was chosen a bit more appropriately the overhead of context switching might have been smaller, but due to constant shuffling of which process is running, context switching overhead is extremely large.
- HRRN: Since Highest Response Ratio has a non-preemtive method of scheduling, it suffers little in terms of context switching overhead. It often does better than its counter parts due to how little shuffling of processes is involved.
- SRT: SRT can suffer from high overhead caused by context switching, as the algorithm requires constant preemption, and sorting to find the process with the shortest execution time remaining. Because of this

every time a new processes is added or if a process comes back from IO the SRT has to shuffle the processes to find the one with lowest remaining service time.
- FCFS: With how FCFS operates, there is little problem with context switching as it does it only when it is changing from one process to the next after completing execution. Even though it might have high wait times, it does not have too much in overhead because of context switching.

Q3. Fairness of each algorithm:

- Round Robin: Round Robin is a very fair algorithm as it gives each process a chance to run. It is also a very efficient algorithm as it does not have to sort the processes to find the next one to run.
- HRRN: One of the most fair algorithms as it does not discriminate between IO bound processes and purely CPU processes. It also does not give too much preference to late coming processes.
- SRT: Is not very fair as long service execution processes are often overlooked for shorter running processes. This can become a problem if shorter processes keep entering the runnnig queue.
- FCFS: FCFS is not a very fair algorithm as it can cause smaller processes to wait for longer processes to finish execution.

Q4. How to improve each algorithm:

- Round Robin: For Round Robin, one could utilize a dynamic time quantum. This would allow the time quantum to be adjusted based on the current state of the processes. Instead of having a 1 second time quantum, you could set the time quantum to the lowest service execution time of the processes in the running queue. This way you could decrease the overhead for context switching and could also decrease the finish times for all the processes.
- SRT: For SRT, you could utilize a method of aging to make sure that no process starves. This could be done by implementing a priority system for the scheduling algroithm. This way you could make sure that the processes that have been waiting for a long time get a chance to run.
- HRRN: For HRRN you could implement a machine learning algorithm to more accurately predict the service execution time of the processes. This way you could make sure that the processes that have a higher response ratio are given a higher priority.
- FCFS: For FCFS you could implement a round robin system where after a certain amount of time, the process is pushed back to the end of the queue. This way you could make sure that if there is a shorter process waiting behind a larger process, it would get a chance to run.