Process Scheduler

How To Run

To run, simply type this commands in Linux:

```
make
./scheduler
```

About The Program

This program simulates a process scheduler. The instructions are read from a file named "instructions.txt". The processes are in folder "processes". The priority and arrival time of processes is read from file "definition.txt". The program will read the instructions, processes and definition files and simulate the process scheduler. The program will output average waiting time and average turnaround time of processes.

Priority Queue and Process Selection:

The scheduler employs a priority queue organized as a heap, utilizing a comparison function that prioritizes platinum processes, followed by higher-priority processes, earlier arrival times, and process names. The priority queue is the key data structure used for selecting the next process to run on the CPU.

Execution and Quantum Time:

Processes, once in the CPU, execute their instructions one by one. The scheduler monitors the quantum burst time of each process during execution. If the quantum burst time exceeds the predefined quantum time for a process, the process is released from the CPU. Quantum burst time, execution time tracked and updated accordingly.

Releasing Processes:

Releasing a process from the CPU involves adding the process back to the priority queue and removing it from the CPU. The quantum count of the released process is increased, and the scheduler checks for potential promotion. Releasing a process from cpu and adding it to priority queue is done by a function. This function is called in 2 cases:

1. After exceeding quantum time
2. The next candidate process in priority queue should preempt the running process

Context Switching:

Context switching occurs when a process is released from the CPU. The released process is added back to the priority queue, and the next candidate process is fetched. If the next candidate process is different from the released process, a context switch occurs, and the new process is added to the CPU. When a platinum process is in the CPU, no context switch occurs.

Process Promotion:

Promotion of processes is checked after quantum count increases. If a process has completed its quantum time or is preempted, it is released, and its quantum count is increased. The scheduler then checks for potential promotion according to quantum count.

Arrival of New Processes:

The scheduler continuously checks for the arrival of new processes. If a new process arrives, it is added to the priority queue. After execution of exit instruction, the process is removed from cpu and not added to priority queue again, leaving cpu empty. CPU gets empty after exit instruction. If the CPU is empty and ready for the next execution, the next candidate process is fetched. If the priority queue is empty and the CPU is idle, the current time jumps to the arrival time of the next process. The program terminates when there are no more processes to execute.

Difficulties

At first, I did not prefer to use a process comparison-based approach to select the next candidate process. I initially attempted to employ process queues with different priorities. However, a challenge arose when multiple processes were added to the queues simultaneously at the same arrival time. This simultaneous addition caused a problem in determining the next process to add to the queue, as the first-added process was set to run first. So, I observe that a comparison is needed. Then, I decided to switch to using a priority queue for processes. It made the process selection easier, since it provides fixed outcome with comparisons independent from different adding permutations of processes with the same arrival time.

Author

- Atakan Yaşar