CS3523: Operating Systems - II

Programming Assignment 3 - Report

Soumi Chakraborty

ES19BTECH11017

1 Overview

This assignment includes the source code for two scheduling algorithms: Rate-Monotonic Scheduling (RMS) and Earliest Deadline First (EDF). This report includes the design and implementation of the two algorithms along with a comparison between the two.

1.1 Rate-Monotonic Scheduling

This algorithm schedules processes in order of their static priorities. The priorities assigned to each process doesn't change during the execution of the program. Each process's priority is inversely proportional to its period. This implies that processes with shorter periods get higher priorities. When a higher priority process needs to execute while a lower priority process is already executing, the lower priority process gets preempted in favour of the one with the higher priority.

1.2 Earliest Deadline First Scheduling

This algorithm schedules processes in order of their dynamic priorities. The priority of a process is decided based on its deadline: the earlier its deadline, the higher the priority. This algorithm doesn't necessarily require the processes to be periodic.

2 Implementaion in C++

Since both the programs follow the same basic structure, their design is being described together.

- **`struct Process`**: stores details about each unique process.
- struct ProcessDetails: stores details about each individual process. Includes details like arrival time, deadline, priority, amount of processing time remaining and etc. This is the struct that is used during the majority of the scheduling process.
- **`bool SortProcess()`**: a function to sort two processes based on their priorities and if they are the same, then based on their arrival times.
- **`struct queueCompare`**: struct which includes a custom sorting function for the priority queue.
- **`processList`**: a sorted vector containing a list of all the individual processes.
- **`readyQueue`**: a priority queue that includes all the processes that are ready to be executed.

- **`preemptedList`**: a regular queue to store all the processes that were preempted during execution. All the processes in this queue are eventually put back into the ready queue when it's their turn to execute again.

2.1 The Scheduler Function

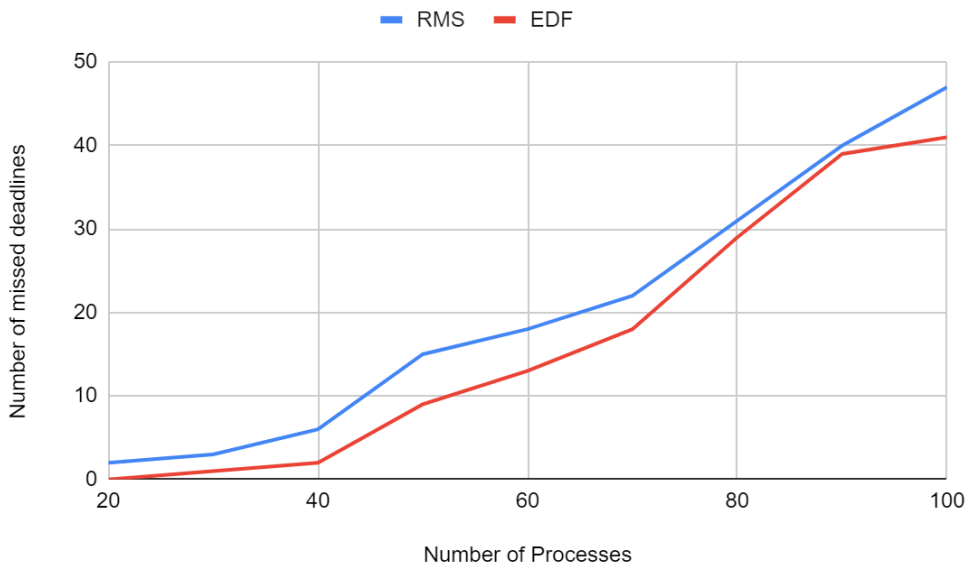
The scheduler functions (RMSscheduler() and EDFscheduler()). The structure of the function has been described below:

1. Set current time as the arrival time of the first process in the sorted process list.
2. Run a loop until either all the processes have been in the ready queue at least once or until the ready queue is completely empty.
3. Add all processes with the same arrival time as the current arrival process to the ready queue.
4. Add all processes from the preempted processes list to the ready queue.
5. Run a loop until the ready queue is empty.
6. Pop an element from the ready queue for processing.
7. Check if there is a possibility of a process being preempted. If the process can finish executing before the arrival time of the next process, then there's no possibility of preemption.
8. If the process can't be preempted, then either finish processing it, or terminate it if it misses its deadline. Update all relevant variables and the log file. Each time a process either finishes or gets terminated, update its waiting time accordingly.
9. If there is a possibility of a process being preempted, check if the priority of the next process is higher. If not, do the same as step 8.
10. If both the above steps evaluated to false, preempt the process by adding it to the preempted list. Update all variables like current time, remaining time, etc. and the log file.
11. If both the ready queue and the preempted list are empty, check if the arrival time of the next process is greater than the current time. If yes, log that the CPU is idle.
12. Repeat from step 3 until all the processes are done executing.

3 Comparisons

The graphs below were generated as directed in the assignment statement. The input file used has also been included in the submission file with the file name inp-params.txt.

3.1 Missed Deadlines



From the graph it is clear that EDF misses fewer deadlines when compared to RMS.

3.2 Average Waiting Time