Assignment 2 OS-2

Vibhanshu Jain
CS19B1027

Rate Monotonic Scheduling

This is a preemptive scheduling algorithm in which a lower priority process is preempted by a higher priority process. The priority of the process is dependent upon the period of the process. The shorter the period the higher the priority. In my approach, I am trying to implement this algorithm using arrays.
I created a class named Process which contains the details of the process added to the system. Now we started the for loop which will run till the time each process ended, which is equal to the max time upto which the system should run.
Now for each incidence of time, we are checking whether there is any process that has a higher priority or effectively a lower period than the currently running process. If we found any such process then we will stop this process and start executing the current process.
In this way, we are implementing the RMS.

Earliest Deadline First

This is also a preemptive process scheduling algorithm in which a lower priority process is preempted by a higher priority process. The priority is dependent on the deadline, the closer the deadline the higher priority.
In my approach, I am trying to implement this using the priority queues. Initially, I am inputting all the processes in a priority queue, which is named as ‘waiting_queue’. Once the process arrival time is equal to the current time, then we will shift the process from waiting_queue to ready_queue. Now, this ready queue is the one that is implemented and used for the scheduling process. Now for each of the processes in the ready queue, I am checking the process with the nearest deadline and starts its processing. In the processing section, we are increasing the current time. Once we found that there is one more process that has a near deadline than the current running process we will replace this process with the new one and update its details. We are also keeping the record of the processes which missed the deadline, in that case, the current time will become equal to its deadline and is still not completed.

Some key points during implementation:

- These methods can be implemented using arrays as well priority queues. (RMS in array and EDF in queues)

- We are using the ‘ifstream’ for input files & ‘ofstream’ for output files. In this case, I defined ins-params.txt for input & EDF-Log.txt and EDF-Stats.txt for throwing output.
- In EDF we are using a priority queue from the STL Library which is very helpful for implementing the algorithm.
- Before starting the RMS algorithm we are checking whether the CPU utilization is greater than n*(2^(1.0/n)-1), where n is the number of processes, and this is the upper bound of the CPU utilization. If the utilization is more than this upper that means these processes cannot be scheduled by RMS and exit the complete process.

Some Results:

- On a general note, EDF misses more deadlines than RMS. But this is more dependent on the details of the process.
- RMF implementation is easier as compared to EDF which makes RMF better for lightweight systems.
- RMF does not provide complete CPU Utilization, (it has an upper bound), whereas in EDF we are able to use the CPU up to 100%, which makes it more efficient for bigger and heavier workloads.
- The RMS has a good number of context switches as compared to EDF. This means if the process numbers and period are more then EDF should work better.