REPORT-CS20BTECH11035

CLASS FOR A PROCESS:

My code contains a class Process (in public mode) which has

1. Process id Pi.
2. Execution time of process t.
3. Period p.
4. Number of times a process needs to be executed k.
5. Number of times process has executed num_times_executed (initialized to 0 in function priority).
6. Waiting time of process waiting_time (initialized to 0 in function priority).

FUNCTIONS COMMON IN RM AND EDF:

TOTAL_PROCESS:

Function total_process returns total number of processes according to the given input. Function arguments are:

1. An array (pointer) of class process which contains details of all processes.
2. Integer n which is number of distinct process (first line from input file).

PRIORITY:

Function priority returns an array of class process in non-decreasing order of periods (or deadlines). In RMS, priority function assigns priority to processes. The lower the index, the higher the priority. Function arguments are:

1. An array (pointer) of class process which contains details of all processes.
2. Integer i which is the index of a process before it is sorted.
3. Integer n which is number of distinct process (first line from input file).

CHECKCOMPLETE

Function checkcomplete returns 0 if all processes have completed execution, else returns -1. If for all process num_times_executed=k, it returns 0. Function arguments are:

1. An array (pointer) of class process which contains details of all processes.
2. Integer n which is number of distinct process (first line from input file).

NEXTPROCESS

Function next process returns index of process that is to be executed next (not pre-emption in middle of process.). Function arguments are:

1. An array (pointer) of class process which contains details of all processes.
2. Integer t which is time.
3. Integer n which is number of distinct process (first line from input file).

RATEMONOTONICSCHEDULING AND EDFSECHEDULING

Function edfscheduling returns time at which given process completes execution. Function arguments are the same as ratemonotonicscheduling.

Function ratemonotonicscheduling returns time at which given process completes execution. Function arguments are:

1. An array (pointer) of class process which contains details of all processes.
2. Integer priority is the index of process which is to be executed.
3. Integer n which is number of distinct process (first line from input file).
4. Integer time_from indicates the time at which process with given priority starts.
5. Integer time_to indicates the time at which process with given priority ends (but the process may not end at that time due to preemptions in middle).
6. If Integer status is 0 it indicates that there is no process decided to execute after this process (we will decide which process will execute after this process completion using function next process). Else if status is 1, it must be a process which pre-empted another process. So, it is having a process waiting (process which got pre-empted) ready to execute after this.
7. If integer waiting_time_accept is 1, it gives permission to change waiting time otherwise ,it indicates that waiting time already changed.

RMS:

In ratemonotonicscheduling, we update waiting time of the process being executed.

We check if there is any process with priority less than given priority which preempts this process from time_from to time_to. If no process pre-empts, the process completes execution and next process starts. If any process pre-empts, pre-empted process is executed by calling ratemontonicscheduling function and after it's execution the process which got pre-empted completes execution (calls ratemonotonicscheduling function again from time it gets resumed).

EDF:

In edfscheduling, we update waiting time of the process being executed.

We check if there is any process with deadline less than given priority which preempts this process from time_from to time_to. If no process pre-empts, the process completes execution and next process starts. If any process pre-empts, pre-empted process is executed by calling edfscheduling function and after it's execution the process which got pre-empted completes execution (calls edfscheduling function again from time it gets resumed).

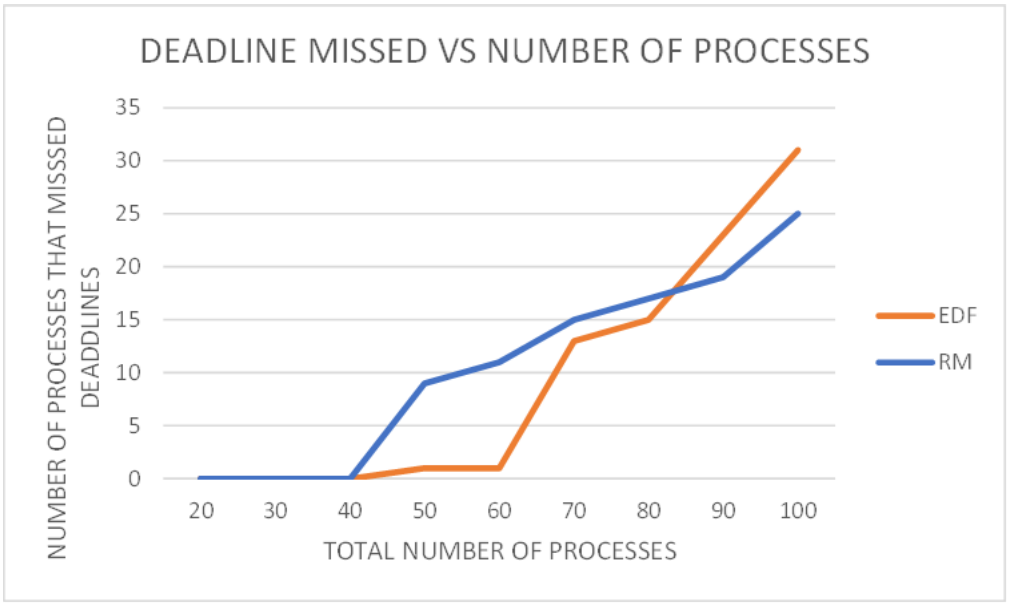
In both, rms and df deadline_miss is a static variable which counts the number of processes that missed their deadline. At the end of these functions we update RM-Stats.txt/ EDF-Stats.txt.

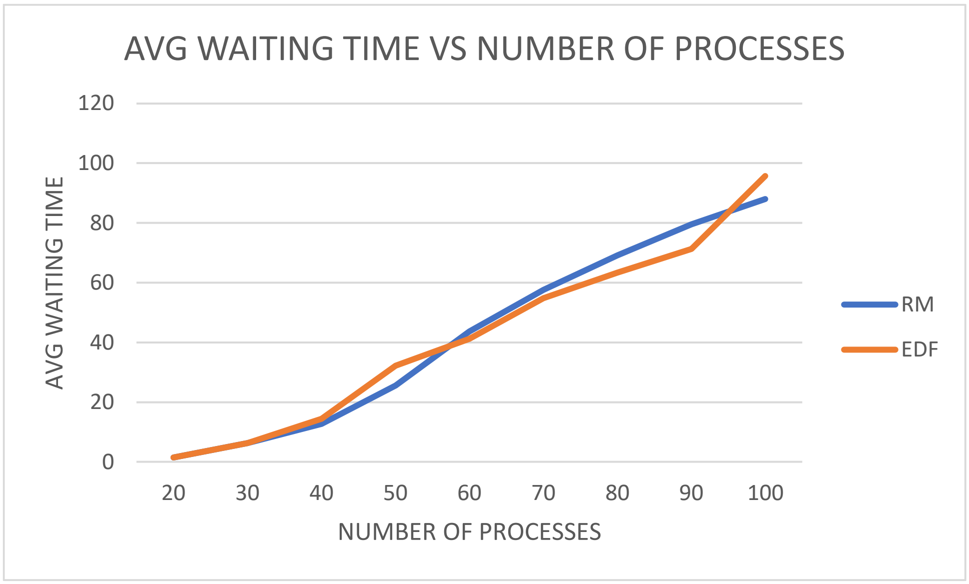
COMPLICATIONS:

1. Complications arose when waiting time was being added twice in some cases(waiting time gets updated but the process does not start as higher priority process needs to execute.). To solve this, I added an integer which checks waiting time is updated only once.
2. num_times_executed used to get incremented every time it resumes after getting pre-empted. So, it used to get incremented multiple times but process gets executed only once. So, to solve this, num_times_executed is decremented whenever it is incremented multiple times.

OBSERVATION:

EDF successfully executes all processes when CPU utilization is less than 100%. But RMS may not be successful every time even if CPU utilization is less than 100%.


DEADLINE MISSED VS NUMBER OF PROCESSES
NUMBER OF PROCESSES THAT MISSSED DEADDLINES
TOTAL NUMBER OF PROCESSES
EDF
RM


AVG WAITING TIME VS NUMBER OF PROCESSES
AVG WAITING TIME
NUMBER OF PROCESSES
RM
EDF