OS ASSIGNMENT 3
REPORT

Low level design of the programs:

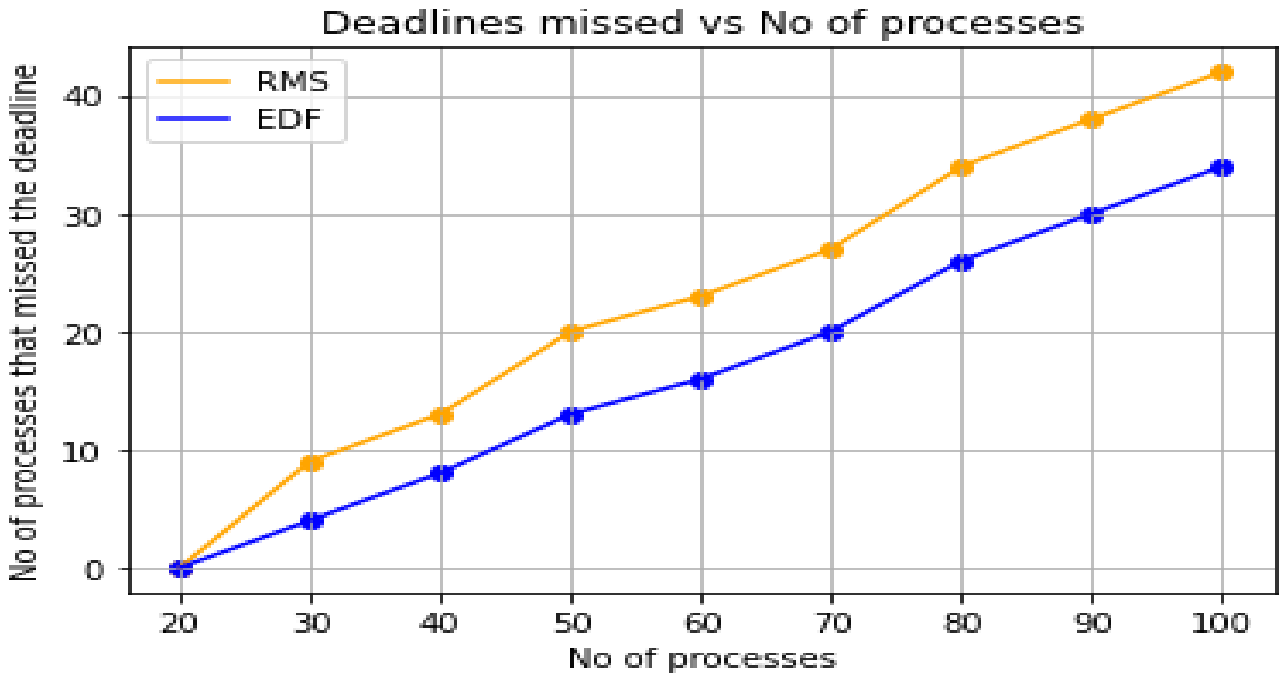
1. Basically the RMS program schedules the processes according to the RMS scheduling algorithm and an EDF program schedules the processes according to the EDF scheduling algorithm.
2. First I created a data type process to represent a process which contains attributes like pid,wait time,cpu burst,period,next deadline,count(to keep track of no of remaining iterations of the process),etc..
3. Then I created three functions: arrange,more_priority and check_complete. The arrange function sorts the process array in ascending order according to the period of the processes in the array,the more_priority function gives the highest priority process in the process array at a given time and check_complete function returns 1 if all iterations of all processes are completed and if not it returns 0 .
4. Now coming to the main function it reads the file input and creates a process array with attributes according to the values of input. Then control goes to the while loop(iterates according to the time) which uses check_complete function to confirm the iterations of processes are not completed.
5. Then it goes to a for loop which checks whether any process other than the running process is missing its deadline,If yes it will be terminated.
6. Then it checks the running process if it can complete its processing before the deadline and if not it will be terminated. Now we have two cases, if it terminates then it calls the more_priority function which returns a process with pid=-1(not valid) if no processes are available else it returns the highest priority process available.
7. If the running process is not terminated then the control goes to an else statement in which it checks whether the running process completed its processing time, If yes it updates the attributes of the process accordingly.
8. Coming out of the if statement it calls the more_priority function to get the highest priority process available at that time. If no processes are available then it prints “cpu is idle” , if pid of the process returned by the more_priority function is equal to pid of the running process then it checks whether this process is new(just started) or old. If it is new it prints “process starts execution” and updates the processing time left else it just updates the processing time left.
9. If the pid of the process returned by the more_priority function is not equal to pid of the running process control goes to an if else statement which checks whether processing time of running process is completed.
10. If it is then it again goes to a if else statement which checks whether the process returned by more_priority function has processing time left equal to cpu_burst,if

yes running process is replaced(not preempted) by new process and prints “process starts execution” else it prints “process resumes execution”

11. If not, the running process is preempted by the process returned by the more_priority function and prints “process x is preempted by process y” and then we have two cases whether the new running process has its processing time left is equal to the cpu_burst. If yes it prints “process starts execution” else it “prints process resume execution”.
12. I have calculated the average wait time of all processes using a for loop in which it increments the wait_time of a process if it is not running and it has a start time greater than time.
13. Coming to the design of the EDF program everything remains the same compared to the RMS program except the more_priority function and arrange function.
14. The arrange function here sorts the process array in the ascending order according to their deadlines and the more_priority function internally calls the arrange function and gives the highest priority process at a given time.

COMPARISON OF RMS AND EDF ALGORITHMS:

Graph 1:



GRAPH 2: