Operating Systems–II: CS3523
Spring 2021
Programming Assignment 3:
Implementing Rate-Monotonic Scheduling & Earliest Deadline First Scheduling through Discrete Event Simulation

Submission Date: **13th February 2022, 9:00 pm**

Goal: The goal of this assignment is to implement a program to simulate the Rate-Monotonic & Earliest Deadline First (EDF) scheduling algorithms. Then compare the average waiting time and deadlines missed of both algorithms. Implement both the algorithms in **C++** using discrete event simulation.

Details: You have to simulate the two Real-time scheduling algorithms discussed in Chapter 6 of the book: Rate-Monotonic Scheduling & Earliest Deadline First Scheduling in C++.

Develop a program to simulate both the algorithms. Your program will read the input from the file and generate the correct sequence of events (for each process execution), occurring in the system.

Events to be considered are when a process:

1. starts its execution.
2. finishes its execution.
3. is preempted by another process.
4. resumes its execution.

Your program will schedule and execute each process (and output the events for each process to the output file) **'k' times** based on the corresponding scheduling algorithm. Your program should output each process that is able to meet its deadline. It should also detect if any process is missing any deadlines, output it to the file and terminate. The details are given below.

Input: The input to the program will be a file, named inp-params.txt, consisting of the following parameters: **n**, the number of processes; followed by n lines consisting of $\mathbf{P_i}$ (process id), **t** (processing time), **p** (period) and **k** (the number of times each process repeats).

For simplicity, you can assume that the period(**p)** and the deadline(**d)** for each process is the same as in the examples shown in the book.

Output: For each algorithm, you must generate two files. The first output file is essentially a log of all the events. It should be named as RM-Log.txt & EDF-Log.txt.

A sample log file output is as follows:

Process P1: processing time=20; deadline:50; period:50 joined the system at time 0
Process P2: processing time=35; deadline:50; period:100 joined the system at time 0
Process P1 starts execution at time 0.
Process P1 finishes execution at time 20.
Process P2 starts execution at time 21.
Process P2 is preempted by Process P1 at time 50. Remaining processing time:5
Process P1 starts execution at time 51.
Process P1 finishes execution at time 70.
Process P2 resumes execution at time 71.
Process P1 finishes execution at time 75.
CPU is idle till time 99.
Process P1 starts execution at time 100.
.....
.....
.....

As you can see this output is the same as the example given in the book.

The second output file will consist of statistics about the current execution and it is denoted as: RM-Stats.txt & EDF-Stats.txt. It will consist of (1) number of processes that came into the system; (2) number of processes that successfully completed; (3) number of processes that missed their deadlines; (4) Average waiting time for each process

Your program should output the following files:

1. RMS-Log.txt and EDF-Log.txt consisting of discrete events, of each algorithm, as described above.
2. RM-Stats.txt & EDF-Stats.txt, consisting of the average waiting time and deadlines missed of both the algorithms.

Report: You have to submit a report for this assignment. The report should consist of two things:

The report should first explain the design of your program while explaining any complications that arose in the course of programming RM & EDF algorithms.

The report should contain a comparison of the performance of RM & EDF algorithms. You must run both these algorithms multiple times to compare the performances and display the result in form of a graph. You must have the following graphs:

- Graph1: Deadline missed vs Number of processes: In this graph, Y-axis is the number of processes that miss the deadline, while the x-axis is the number of processes varying from 20 (2 processes each repeating 10 times) to 100 (10 processes each repeating 10 times) in the increments of 10. This graph will have two curves one for each algorithm. Assume that all the remaining

parameters are the same for both the algorithms - **t**, **p** and **k** for each of the process executed.

- Graph2: Average Waiting time vs Number of processes: In this graph, Y-axis is the average waiting time, while the x-axis is the number of processes varying from 20 (2 processes each repeating 10 times) to 100 (10 processes each repeating 10 times) in the increments of 10 like in Graph1. Again, this graph will have two curves one for each algorithm. Assume that all the remaining parameters are the same for both the algorithms - **t**, **p** and **k** for each of the process executed.

A sample input file is provided for you to generate the reports.

Submission Format:
You have to submit the following deliverables for this assignment.

1. The source code: Assgn3-RMS<RollNo>.cpp & Assgn3-EDF<RollNo>.cpp
2. Readme: Assgn3-Readme-<RollNo>.txt. This file will explain how the Tas should execute your program.
3. Report: Assgn3Report-<RollNo>.pdf as explained above.

Name the zipped document as: Assgn3-<RollNo>.zip. Please follow the naming convention strictly. Otherwise, your assignment will not be evaluated. Upload all this in the classroom by the above mentioned date.

Grading Policy:

1. Design as described in the report and analysis of the results: 50%
2. Execution of the programs based on the description in the readme: 40%
3. Code documentation and indentation: 10%

Extra Credit: You will be given an extra credit of 20% if you accurately model and include the time incurred in selecting and switching to the next process. Assume that the context switch time is 10 microseconds. Note that the time taken by processes for execution as shown in the input file is in the order of milliseconds.

You must clearly describe in your report, how you are computing the time taken to select the next process.

As mentioned before, all assignments for this course have the late submission policy of a penalty of 10% each day after the deadline. We will consider a late assignment for a maximum of 6 days. Any submission beyond that will not be considered.

Kindly remember that all submissions are subjected to plagiarism checks.