Assignment 3: Single Processor Scheduling

CIS 4410/5410

Due: 10:15 am, Wednesday, September 28, 2022

This assignment is to implement a real-time scheduler for periodic tasks. There are three parts:

1. implement a schedulability analyzer [35 pts]
2. implement a scheduler [45 pts]
3. discussion questions [20 pts]

For parts 1 and 2, skeleton code has been provided for reading keyboard inputs into variables. Your solution may use these variables however you see fit, but **you may not alter the input prompts**. Additional skeleton code has been provided to help you structure your implementation. You may use, alter, or overwrite this skeleton code as needed, so long as the input prompts remain unchanged and the printed outputs fit the specifications. For part 3, answer the discussion questions in a separate document, and upload the PDF to to Codio (in the code editor, go to File>Upload).

For this assignment, we assume that each task is described as (S, P, E, D), where S is the start time relative to the virtual timer's start time, P is the period, and E is the maximum execution time and D is the relative deadline. We further assume that the deadline (D) of the task is equal to its period (P). Also assume that a job's release occurs at the beginning of a time step (i.e. if the job's release is at time step 4, it can be scheduled for time step 4). Likewise, a job's deadline occurs at the beginning of a time step (i.e., if the job's deadline is at time step 7, it must be scheduled at time step 6 or earlier to be considered on-time). We denote job (i, j) to represent the jth job (instantiation) of the task i.

Part I: Schedulability Analyzer

In this part, you will implement a schedulability analyzer. The schedulability analyzer simulates task executions according to given specifications. Given a set of system specifications and tasks, your analyzer should output a list of time-stamped job executions and information about the simulation.

At each time step, your analyzer should print the following:

```
Time: [CurrentTime]
Scheduler: picking task [TaskId]
Task [TaskID], Job [JobId]: starts executing [ExecTimeSoFar] of
    [maxExecutionTime]
```

For example: when task 2 = (0,10,3,10) is executed for the second block during the 1st period, it prints:

```
Time: 1
Scheduler: picking task 2
Task 2, Job 1: starts executing 2 of 3
```

If a task i job j misses its deadline at a time step, a note should be printed with its identifiers i and j. If not jobs are ready to run at a time step, this should also be printed. An example output is shared in the appendix. At the end of execution, your analyzer should print the total number of deadline misses and the CPU utilization.

Inputs are as follows:

- Priority policy (for your assignment, it is either RM or EDF)
- A set of tasks, $\{(S_i, P_i, E_i, D_i) | i = 1 \dots N\}$
- Start time of virtual timer (e.g., counter)
- Maximum total execution time (if 0 is specified, the max execution time should be the least common multiple of all the periods)
- Termination flag indicating the system's behavior when a job misses its deadline
  - True: if any job of a task misses its deadline, that job is terminated when the deadline is missed, and the next job of the task is started as specified at the next period
  - False: if any job misses its deadline, that job is not terminated and can be scheduled again

Outputs are as follows:

- Execution trace as described above (a sample output is shown at the end of this document)
- Number of deadline misses
- CPU Utilization (% of time 'CPU' is active)

Your schedulability analyzer should describe a sequence task executions but not actually execute any task, meaning your solution should not create or run any threads. Additionally, your solution may use simulated time rather than a clock time. In other words, you may represent time in terms of arbitrary time units, rather than true milliseconds or seconds.

Part II: Scheduler Implementation/Simulator

In this part, you will implement a simulator of a real-time scheduler that can schedule tasks according to their priorities (e.g., using RM for static priority and EDF for dynamic priority). In real-life, at each scheduling time, the scheduler picks the highest priority task that is ready to execute and then lets it execute until one of the following conditions become true: 1) the task finishes its execution and begins idling until the start of its next period, 2) it becomes blocked waiting for a condition (such as resource availability, i/o completion), 3) a new higher priority thread becomes ready to execute.

You are to design and implement a simulator of a priority-based scheduler for tasks, where each task is implemented as a thread. That is, for each task (S, P, E, D), create a thread (see this tutorial for pthread synchronization tips) that executes as specified. Each task will instantiate a job starting at time S which executes every period, P, for the duration of E. How you mimic the execution for the duration of E is up to you (e.g., infinite loop), but you must make sure that at most one task is "actively executing" at any given time (since we assume single processor).

More specifically, you are to implement: 1) a data structure for maintaining the scheduling queue, where each element is a Task Control Block consisting of relevant information such as the task id, thread id, priority, start time, period, max execution time, execution time within the current period, etc., 2) the main scheduling thread that executes every 1/10th of a second to update task priorities and determine which task to execute next, 3) the appropriate number of tasks (i.e., thread), where the active thread executes for 1/10th of a second before returning execution back to the main scheduling thread.

Your output should be similar to part (1), but additionally whenever a thread is executing, it should print a status. Specifically, whenever task i, (thread i) is scheduled to run, the task will print:

```
Execute current task [TaskId]
```

For example: when task 2 = (0,10,3,10) is executed for the second block during the 1st period, it prints:

```
Time: 1
Scheduler: picking task 2
Task 2, Job 1: starts executing 2 of 3
Execute current task 2
```

At the end of execution, your simulator should print the total number of deadline misses and the CPU utilization. An example output is shared in the appendix. The exact inputs and outputs are the same as in part I.

Part III: Discussion Questions

1. Give step-by-step instructions on how to compile and run your timer in the Codio environment. Briefly describe your implementations for parts 1 and 2. For part 2, what mechanisms did you use for thread synchronization?

2. For this question, we consider what happens when a task misses its deadline.

   (a) For task set A, run your schedulability analyzer with both policies. For each run, set the termination flag to true and simulate from time units 0 to 20. Record the number of deadline misses for RM and EDF.

   A:

Task	S	P	E	D
1	0	2	1	2
2	0	6	2	6
3	0	8	3	8

   (b) Now run your schedulability analyzer on task set A with both policies, but this time with the termination flag set to false. How does the number of deadline misses change for RM? What about for EDF? Explain why this happens.

   (c) Considering your results from part (2b), what is one reason why a system designer would choose the RM policy over the EDF policy?

   (d) Now consider the deadline misses for EDF in parts (2a) and (2b). Explain why a system designer may choose to have tasks terminate once they have missed their deadline. On the other hand, why might a system designer choose to have tasks finish their execution even after missing their deadline?

3. For this question, we consider the theoretical guarantees of the RM and EDF scheduling policies.

   (a) For task sets B and C, calcualate the theoretical CPU utilization

   $$U = \sum_{i=1}^{N} \frac{E_i}{P_i}.$$

   B:

Task	S	P	E	D
1	0	4	1	4
2	0	6	2	6
3	0	8	3	8

   C:

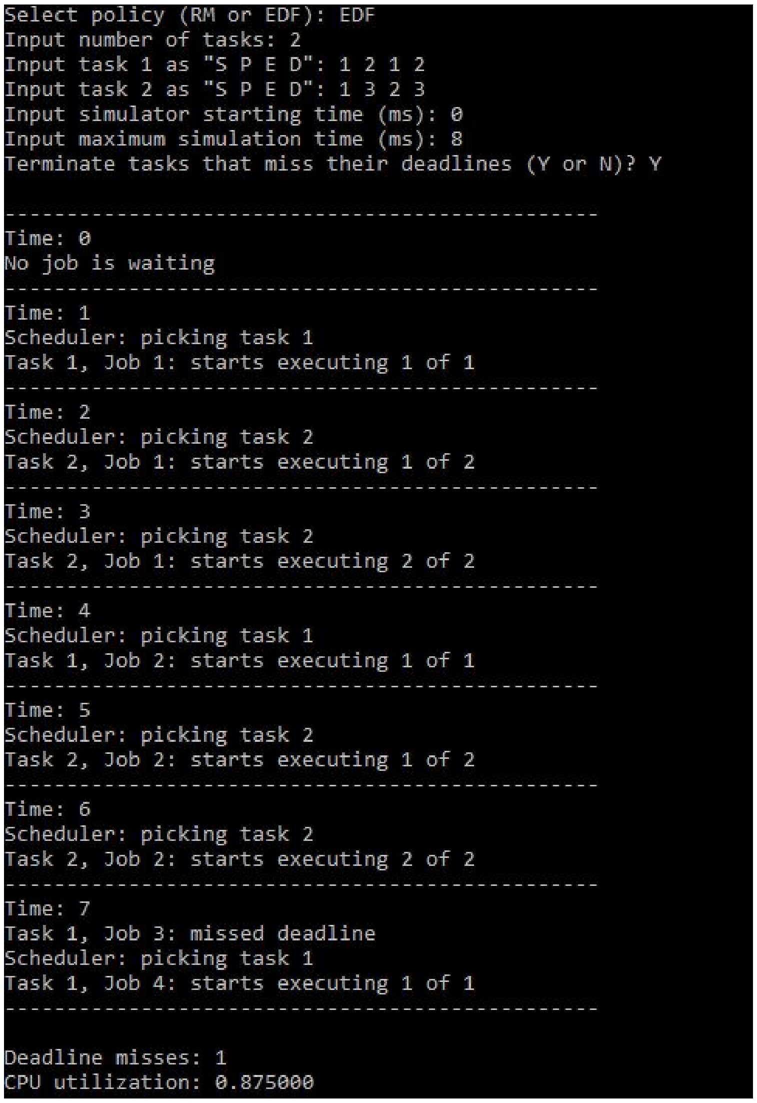
Task	S	P	E	D
1	0	4	1	4
2	0	6	2	6
3	0	8	4	8

   (b) Given the theoretical CPU utilizations you calculated in part (3a), do you expect deadline misses when scheduling task set B with EDF? How about when scheduling task set C with EDF? Explain your reasoning.

   (c) In part (3b), we determined the schedulability of task sets B and C with EDF using only their theoretical CPU utilizations. Can we do the same for scheduling task set B with RM? What about scheduling task set C with RM? If yes, answer whether the task is schedulable. Explain your reasoning.

(d) Run your schedulability analyzer on task sets B and C with both policies (run task set B with RM and EDF, and run task set C with RM and EDF). For each run, set the termination flag to true and simulate from time units 0 to 100. Record the number of deadline misses and actual CPU utilization. Does your analyzer confirm your answers from parts (3b) and (3c)?

0.1 Appendix

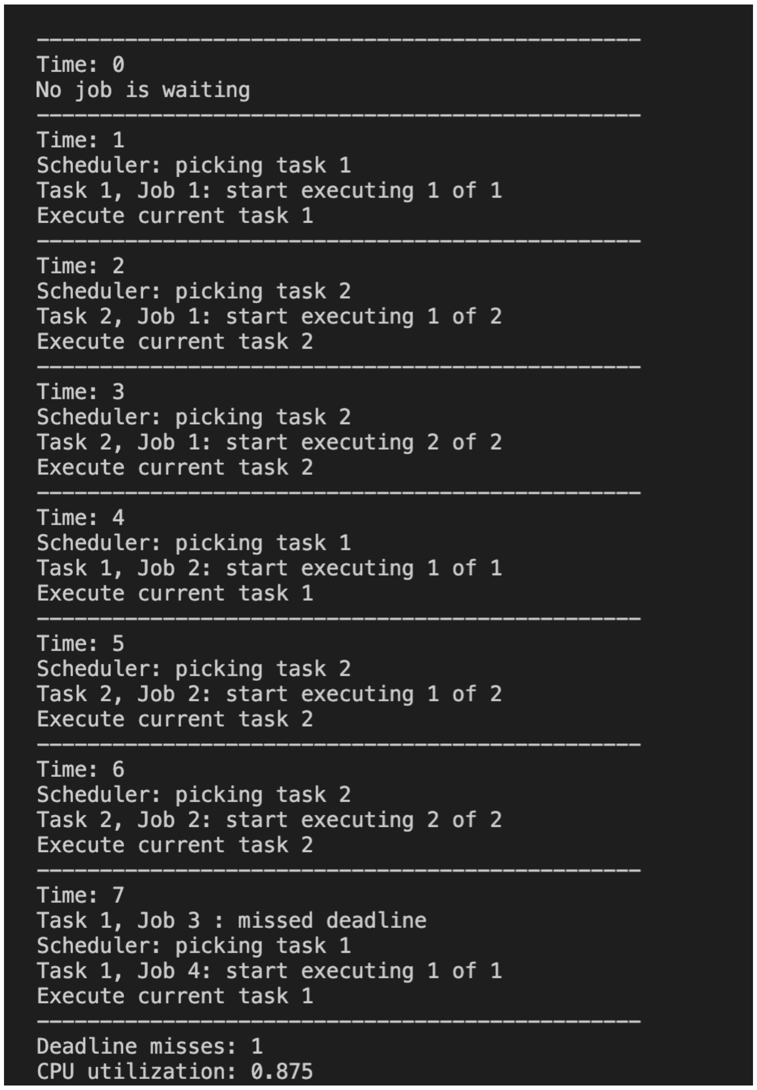


```
Select policy (RM or EDF): EDF
Input number of tasks: 2
Input task 1 as "S P E D": 1 2 1 2
Input task 2 as "S P E D": 1 3 2 3
Input simulator starting time (ms): 0
Input maximum simulation time (ms): 8
Terminate tasks that miss their deadlines (Y or N)? Y

------------------------------------------------
Time: 0
No job is waiting
------------------------------------------------
Time: 1
Scheduler: picking task 1
Task 1, Job 1: starts executing 1 of 1
------------------------------------------------
Time: 2
Scheduler: picking task 2
Task 2, Job 1: starts executing 1 of 2
------------------------------------------------
Time: 3
Scheduler: picking task 2
Task 2, Job 1: starts executing 2 of 2
------------------------------------------------
Time: 4
Scheduler: picking task 1
Task 1, Job 2: starts executing 1 of 1
------------------------------------------------
Time: 5
Scheduler: picking task 2
Task 2, Job 2: starts executing 1 of 2
------------------------------------------------
Time: 6
Scheduler: picking task 2
Task 2, Job 2: starts executing 2 of 2
------------------------------------------------
Time: 7
Task 1, Job 3: missed deadline
Scheduler: picking task 1
Task 1, Job 4: starts executing 1 of 1
------------------------------------------------

Deadline misses: 1
CPU utilization: 0.875000
```

Figure 1: Part 1 Sample Output



```
------------------------------------------------
Time: 0
No job is waiting
------------------------------------------------
Time: 1
Scheduler: picking task 1
Task 1, Job 1: start executing 1 of 1
Execute current task 1
------------------------------------------------
Time: 2
Scheduler: picking task 2
Task 2, Job 1: start executing 1 of 2
Execute current task 2
------------------------------------------------
Time: 3
Scheduler: picking task 2
Task 2, Job 1: start executing 2 of 2
Execute current task 2
------------------------------------------------
Time: 4
Scheduler: picking task 1
Task 1, Job 2: start executing 1 of 1
Execute current task 1
------------------------------------------------
Time: 5
Scheduler: picking task 2
Task 2, Job 2: start executing 1 of 2
Execute current task 2
------------------------------------------------
Time: 6
Scheduler: picking task 2
Task 2, Job 2: start executing 2 of 2
Execute current task 2
------------------------------------------------
Time: 7
Task 1, Job 3 : missed deadline
Scheduler: picking task 1
Task 1, Job 4: start executing 1 of 1
Execute current task 1
------------------------------------------------
Deadline misses: 1
CPU utilization: 0.875
```

Figure 2: Part 2 Sample Output