Embedded OS Implementation, Fall 2020

Project #2 (due December 16, 2020 (Wednesday) 13:00)

[PART I] EDF Scheduler Implementation

Objective:

To implement the Earliest-Deadline-First (EDF) scheduler for periodic tasks, and to observe the scheduling behaviors.

Problem Definition:

uC/OS-II supports priority-driven scheduling. However, it lacks deadline-driven scheduling. In this assignment, you are going to implement the EDF scheduler in uC/OS-II. To accomplish this assignment, you must know about the scheduler of uC/OS-II. It can be implemented based on the existing data structures of uC/OS-II. The objectives of this assignment are the following:

(1) To add some useful data structures for your EDF scheduler

(2) To co-operate with existing data structures/mechanisms in uC/OS-II

Implement the following two periodic task sets. Add necessary code to the μC/OS-II scheduler **in the kernel level** to observe how the task suffers the schedule delay.

Periodic Task Set = {τ_{ID} (arrival time, execution time, period)}

Task set 1 = {τ_1 (0, 2, 6), τ_2 (0, 5, 9)}

Task set 2 = {τ_1 (0, 1, 4), τ_2 (0, 3, 6), τ_3 (1, 1, 3)}

※ The priority of task is set according to the EDF scheduling rules.

※ If there are tasks with the same deadlines, the task with a lower task ID will be executed first.

Evaluation:

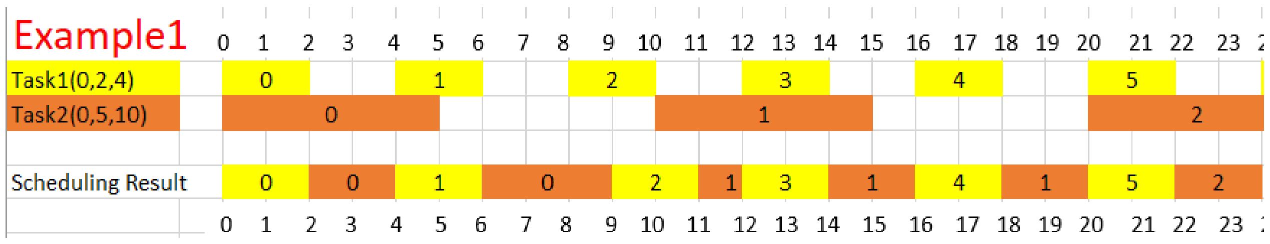
The output format:

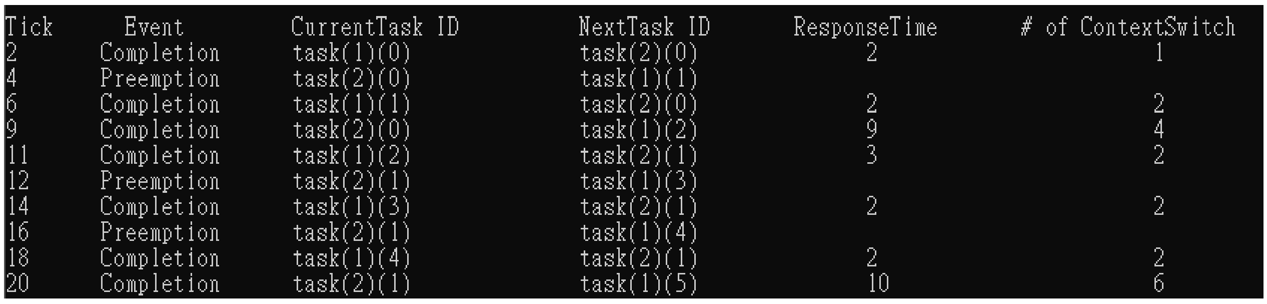
Tick	Event	CurrentTask ID	NextTask ID	ResponseTime	# of ContextSwitch
##	Preemption	task(ID)(job number)	task(ID)(job number)		
##	Completion	task(ID)(job number)	task(ID)(job number)	##	##
##	MissDeadline	task(ID)(job number)	--------------------		

※ If task is Idle Task, print "*task(priority)*".

The Example1 of output results:

Consider two tasks **τ1** (0,2,4) and **τ2** (0,5,10).

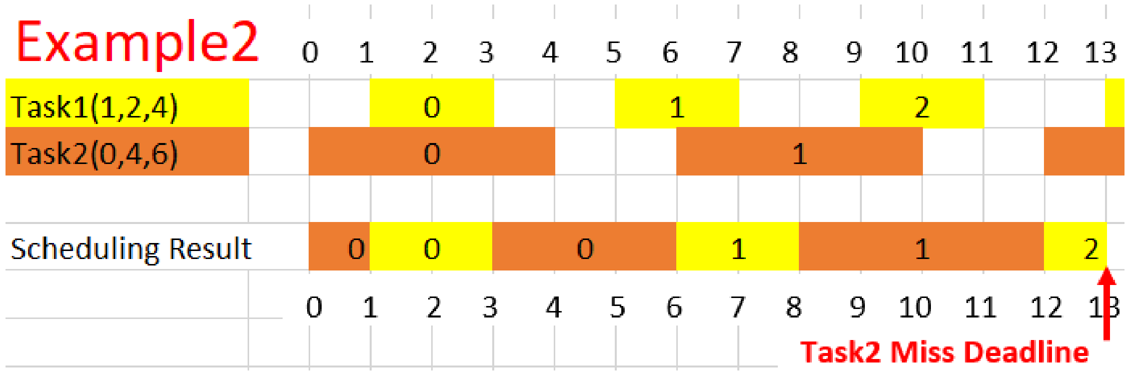


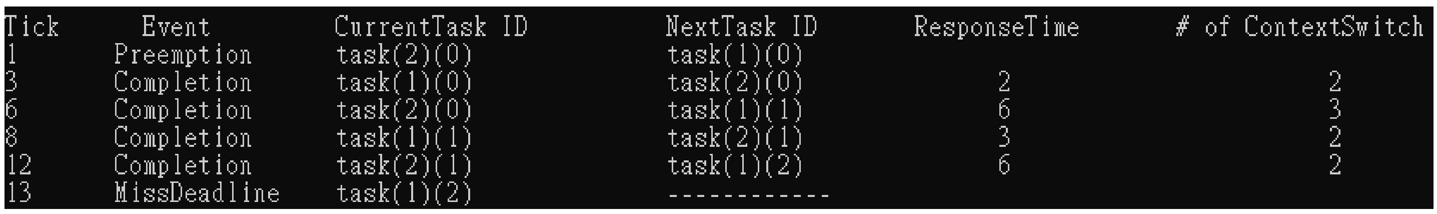


```
Tick       Event          CurrentTask ID         NextTask ID        ResponseTime        # of ContextSwitch
2          Completion     task(1)(0)             task(2)(0)           2                      1
4          Preemption     task(2)(0)             task(1)(1)
6          Completion     task(1)(1)             task(2)(0)           2                      2
9          Completion     task(2)(0)             task(1)(2)           9                      4
11         Completion     task(1)(2)             task(2)(1)           3                      2
12         Preemption     task(2)(1)             task(1)(3)
14         Completion     task(1)(3)             task(2)(1)           2                      2
16         Preemption     task(2)(1)             task(1)(4)
18         Completion     task(1)(4)             task(2)(1)           2                      2
20         Completion     task(2)(1)             task(1)(5)           10                     6
```

The Example2 of output results:

Consider two tasks **τ1** (1,2,4), **τ2** (0,4,6).





```
Tick       Event          CurrentTask ID         NextTask ID         ResponseTime        # of ContextSwitch
1          Preemption     task(2)(0)             task(1)(0)
3          Completion     task(1)(0)             task(2)(0)             2                       2
6          Completion     task(2)(0)             task(1)(1)             6                       3
8          Completion     task(1)(1)             task(2)(1)             3                       2
12         Completion     task(2)(1)             task(1)(2)             6                       2
13         MissDeadline   task(1)(2)             ------------
```

[Part II] CUS Scheduler Implementation

Objective:

To implement Constant Utilization Servers (CUS) for serving aperiodic tasks.

Problem Definition:

As you did in Part I, uC/OS-II supports the EDF scheduling algorithm. Based on your EDF scheduler, you are going to implement the Constant Utilization Servers (CUS) for serving aperiodic tasks.

Implement the following two task sets. Add necessary code to the μC/OS-II scheduler **in the kernel level** to observe how the task suffers the schedule delay.

Some periodic tasks and aperiodic jobs are included in the following two task sets.

Periodic Task Set = {τ_{ID} (arrival time, execution time, period)}

Aperiodic Job Set = {j_{num} (arrival time, execution time, absolute deadline)}

======================== Task Set 1 ========================

Periodic Task Set1 = {τ_1 (0, 1, 4), τ_2 (0, 4, 10), $\tau_{3\_ServerSize}$ (0.3)}

Aperiodic Jobs Set1 = {j_0 (4, 3, 16), j_1 (17, 3, 30)}

======================== Task Set 2 ========================

Periodic Task Set2 = {τ_1 (0, 2, 8), τ_2 (0, 3, 10), τ_3 (0, 5, 20), $\tau_{4\_ServerSize}$ (0.2)}

Aperiodic Jobs Set2 = {j_0 (12, 3, 28), j_1 (14, 2, 39)}

※ The priority of a task is set according to the EDF scheduling rules.

※ If there are tasks with the same deadlines, the task with a lower task ID will be executed first.

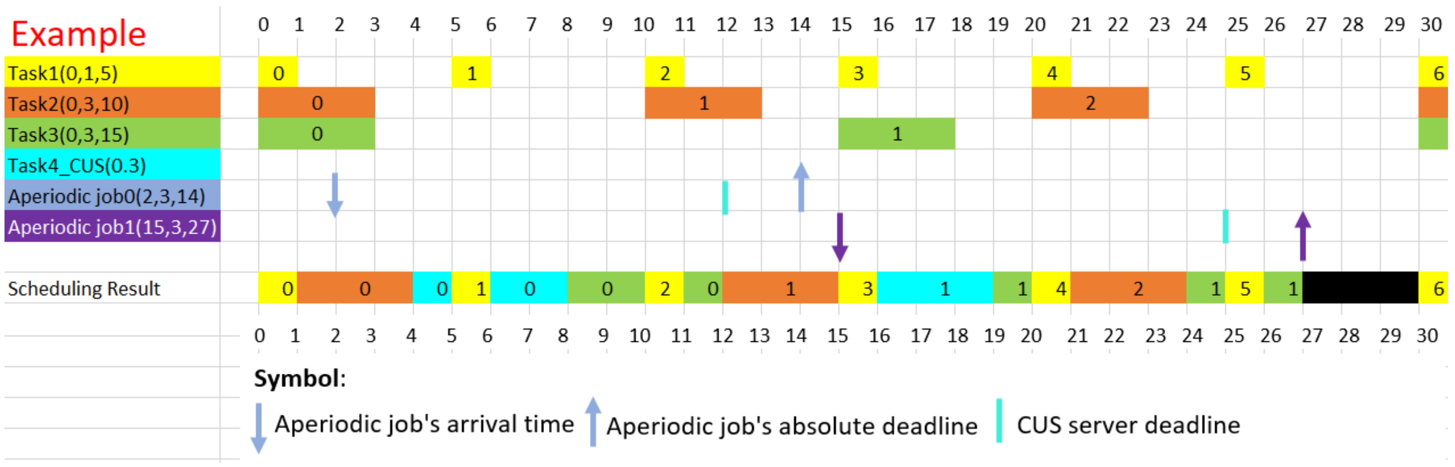
Evaluation:

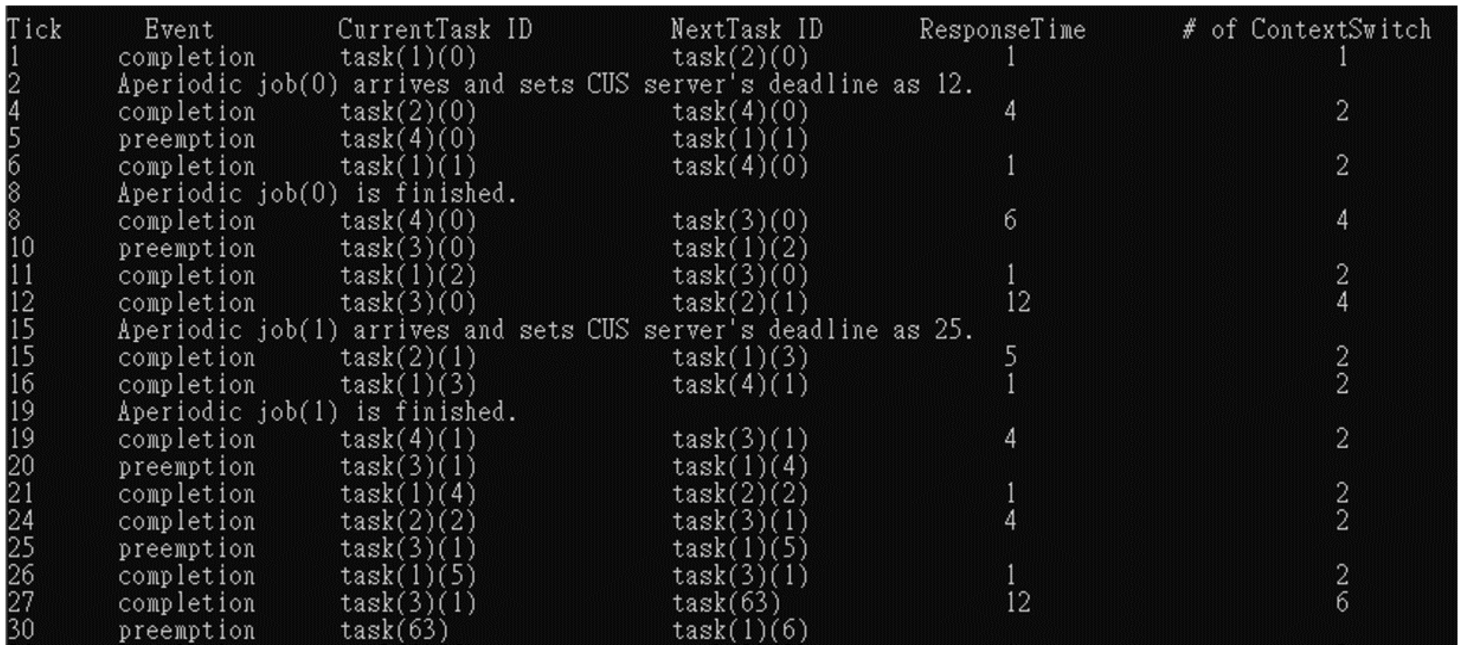
The additional output format for an aperiodic job:

Tick	
##	***if arrive time < sever deadline:*** **Aperiodic job (job number) arrives. Do nothing.** **Aperiodic job (job number) sets CUS server's deadline as ##.**
	else: **Aperiodic job (job number) arrives and sets CUS server's deadline as ##.**
##	**Aperiodic job (job number) is finished.**
	Aperiodic job (job number) misses deadline.

The Example of Output Results in CUS:

Consider a periodic task set = {τ_1 (0, 1, 5), τ_2 (0, 3, 10), τ_3 (0, 3, 15), $\tau_{4\_ServerSize}$ (0.3)} and an aperiodic job set = {j_0 (2, 3, 14), j_1 (15, 3, 27)}





```
Tick      Event          CurrentTask ID          NextTask ID        ResponseTime        # of ContextSwitch
1        completion      task(1)(0)              task(2)(0)          1                       1
2        Aperiodic job(0) arrives and sets CUS server's deadline as 12.
4        completion      task(2)(0)              task(4)(0)          4                       2
5        preemption      task(4)(0)              task(1)(1)
6        completion      task(1)(1)              task(4)(0)          1                       2
8        Aperiodic job(0) is finished.
8        completion      task(4)(0)              task(3)(0)          6                       4
10       preemption      task(3)(0)              task(1)(2)
11       completion      task(1)(2)              task(3)(0)          1                       2
12       completion      task(3)(0)              task(2)(1)          12                      4
15       Aperiodic job(1) arrives and sets CUS server's deadline as 25.
15       completion      task(2)(1)              task(1)(3)          5                       2
16       completion      task(1)(3)              task(4)(1)          1                       2
19       Aperiodic job(1) is finished.
19       completion      task(4)(1)              task(3)(1)          4                       2
20       preemption      task(3)(1)              task(1)(4)
21       completion      task(1)(4)              task(2)(2)          1                       2
24       completion      task(2)(2)              task(3)(1)          4                       2
25       preemption      task(3)(1)              task(1)(5)
26       completion      task(1)(5)              task(3)(1)          1                       2
27       completion      task(3)(1)              task(63)            12                      6
30       preemption      task(63)                task(1)(6)
```

Crediting :

[PART I] EDF Scheduler Implementation [60%]

- The screenshot results (with the given format) of two task sets. (Tick 0 to tick 40 or the tick when a task missing the deadline) (10%)
- Implement and describe how to handle the deadline missing situation under EDF. (10%)
- A report that describes your implementation, including scheduling results of two task sets, modified functions, data structure, etc. (please **ATTACH** the screenshot of the code and **MARK** the modified part) (40%)

[PART II] CUS Scheduler Implementation [40%]

- The screenshot results (with the given format) of two task sets. (Tick 0 to tick 40 or the tick when a task missing the deadline). (10%)
- A report that describes your implementation, including scheduling results of two task sets, modified functions, data structure, etc. (please **ATTACH** the screenshot of the code and **MARK** the modified part). (30%)

[Bonus I] CUS & Button-triggered Aperiodic Job [10%]

- Implement the CUS scheduling and set button-triggered events as aperiodic jobs.

※You must modify source code!

Project submit:

Submit to Moodle.

Submit deadline: December 16, 2020 (Wednesday) 13:00

File name format: RTOS_your student ID_PA2.zip

RTOS_your student ID_PA2.zip includes :

- The report (RTOS_your student ID_PA2.pdf).
- The file you modify(main.c, os_core.c, etc.)