Embedded OS Implementation, Spring 2025

Project #2 (due Nov 27, 2025 (Thursday) 8:00)

[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 functional data structures for your EDF scheduler.

(2) To cooperate with existing data structures/mechanisms in uC/OS-II.

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

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

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

Example Task Set 2 = {τ_1 (1, 0, 2, 4), τ_2 (2, 0, 4, 7) }

※ The priority of the 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.

※ You have to print "task(ID) is running" message in task() function.

※ Output priority: MissDeadline > Preemption = Completion > task(ID) is running .

The input file format:

Task ID	Arrive Time	Execution Time	Task Periodic
##	##	##	##

Example of file 1:

```
1 0 2 6
2 0 5 9
```

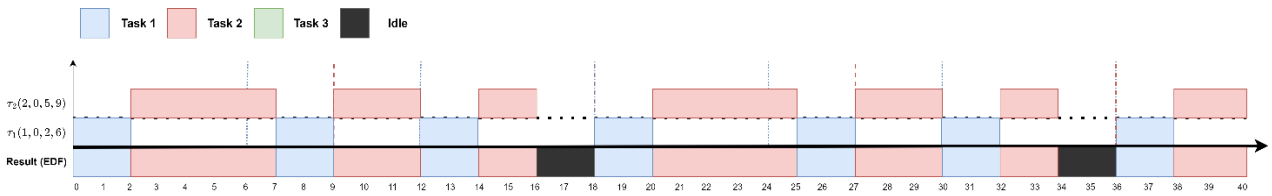
Evaluation:

The output format:

Tick	Event	CurrentTask ID	NextTask ID	Response Time	Preemption Time	OSTimeDly
##	Preemption	task(ID)(job number)	task(ID)(job number)			
##	Completion	task(ID)(job number)	task(ID)(job number)	##	##	##
##	MissDeadline	task(ID)(job number)	----------			
##	task(ID) is running					

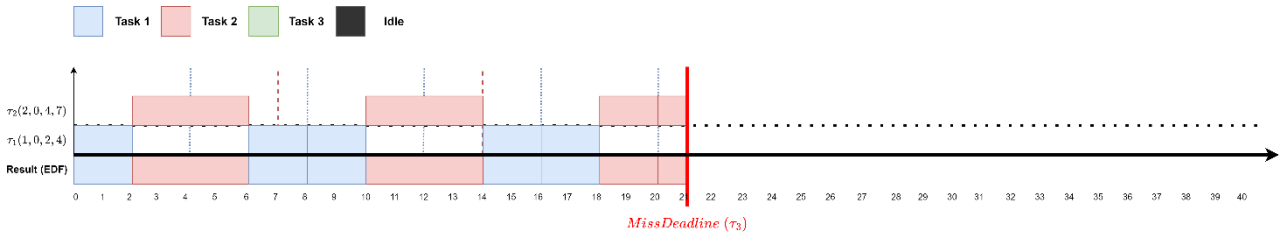
※ If the task is Idle Task, print "*task(priority)*".

The output results of Example 1:




```
 0    task (  1) is running
 1    task (  1) is running
 2    Completion    task(  1)(  0)    task(  2)(  0)       2            0            4
 2    task (  2) is running
 3    task (  2) is running
 4    task (  2) is running
 5    task (  2) is running
 6    task (  2) is running
 7    Completion    task(  2)(  0)    task(  1)(  1)       7            2            2
 7    task (  1) is running
 8    task (  1) is running
 9    Completion    task(  1)(  1)    task(  2)(  1)       3            1            3
 9    task (  2) is running
10    task (  2) is running
11    task (  2) is running
12    Preemption    task(  2)(  1)    task(  1)(  2)
12    task (  1) is running
13    task (  1) is running
14    Completion    task(  1)(  2)    task(  2)(  1)       2            0            4
14    task (  2) is running
15    task (  2) is running
16    Completion    task(  2)(  1)    task( 63)            7            2            2
18    Preemption    task( 63)         task(  1)(  3)
18    task (  1) is running
19    task (  1) is running
20    Completion    task(  1)(  3)    task(  2)(  2)       2            0            4
```

The output results of Example 2:




```
 0    task (  1) is running
 1    task (  1) is running
 2    Completion    task(  1)(  0)    task(  2)(  0)       2            0            2
 2    task (  2) is running
 3    task (  2) is running
 4    task (  2) is running
 5    task (  2) is running
 6    Completion    task(  2)(  0)    task(  1)(  1)       6            2            1
 6    task (  1) is running
 7    task (  1) is running
 8    Completion    task(  1)(  1)    task(  1)(  2)       4            2            0
 8    task (  1) is running
 9    task (  1) is running
10    Completion    task(  1)(  2)    task(  2)(  1)       2            0            2
10    task (  2) is running
11    task (  2) is running
12    task (  2) is running
13    task (  2) is running
14    Completion    task(  2)(  1)    task(  1)(  3)       7            3            0
14    task (  1) is running
15    task (  1) is running
16    Completion    task(  1)(  3)    task(  1)(  4)       4            2            0
16    task (  1) is running
17    task (  1) is running
18    Completion    task(  1)(  4)    task(  2)(  2)       2            0            2
18    task (  2) is running
19    task (  2) is running
20    task (  2) is running
21    MissDeadline  task(  2)(  2)    ------------------
```

[Part II] CUS Scheduler Implementation

Objective:

To implement Constant Utilization Servers (CUS) for serving aperiodic tasks and to observe the scheduling behaviors.

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 Constant Utilization Servers (CUS) to serve 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.

The following two examples include both periodic tasks and aperiodic jobs.

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

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

========== **Example 1**==========

Periodic Task Set = {τ_1 (1, 0, 2, 8), τ_2 (2, 0, 3, 10), τ_3 (3, 0, 4, 15), $\tau_{4\_ServerSize}$ (4, 25%)}

Aperiodic Jobs Set = {j_0 (0, 12, 3, 25), j_1 (1, 14, 2, 33)}

========== **Example 2**==========

Periodic Task Set = {τ_1 (1, 0, 2, 4), τ_2 (2, 0, 3, 9), $\tau_{3\_ServerSize}$ (3, 30%)}

Aperiodic Jobs Set = {j_0 (0, 1, 3, 24), j_1 (1, 11, 2, 39)}

========== **Example 3**==========

Periodic Task Set = {τ_1 (1, 0, 3, 6), τ_2 (2, 0, 4, 10), $\tau_{3\_ServerSize}$ (3, 10%)}

Aperiodic Jobs Set = {j_0 (0, 7, 1, 23), j_1 (1, 10, 2, 25)}

※ 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.

※ If periodic job and aperiodic job have the same deadlines, the **periodic** job will be executed first.

Evaluation:

The additional output format for an aperiodic job:

Aperiodic Jobs Message

Tick	
##	***When aperiodic job arrives:*** ***if arrival time < server deadline:*** **Aperiodic job (job number) arrives. Do nothing.** **else if job *absolute deadline* >= *new server deadline:*** **Aperiodic job (job number) arrives and sets CUS's deadline as ##.** **else:** **Aperiodic job (job number) rejects scheduling.**

##	***When aperiodic job is executing:*** **Aperiodic job (job number) is running.**
##	***When aperiodic job finishes:*** **Aperiodic job (job number) is finished.**
##	***When current tick == server deadline:*** **if aperiodic server is backlogged:** **if job *absolute deadline* >= *new server deadline:*** **Aperiodic job (job number) sets CUS's deadline as ##.** **else:** **Aperiodic job (job number) rejects scheduling.**

※ The time tick of setting the server's deadline is according to the CUS scheduling rules.

Periodic Task Message (include server)

Tick	Event	CurrentTask ID	NextTask ID	Response Time	Preemption Time	OSTimeDly
##	Preemption	task(ID)(job number)	task(ID)(job number)			
##	Completion	task(ID)(job number)	task(ID)(job number)	##	##	## or N/A
##	MissDeadline	task(ID)(job number)	----------			
##	task(ID) is running					

※ Output priority: MissDeadline > Aperiodic job arrive > Aperiodic job (job number) sets … > aperiodic job finished > Preemption = Completion > task(ID) is running = Aperiodic job(job number) is running

※ You should create only one CUS task for all the aperiodic jobs.

The TaskSet.txt format:

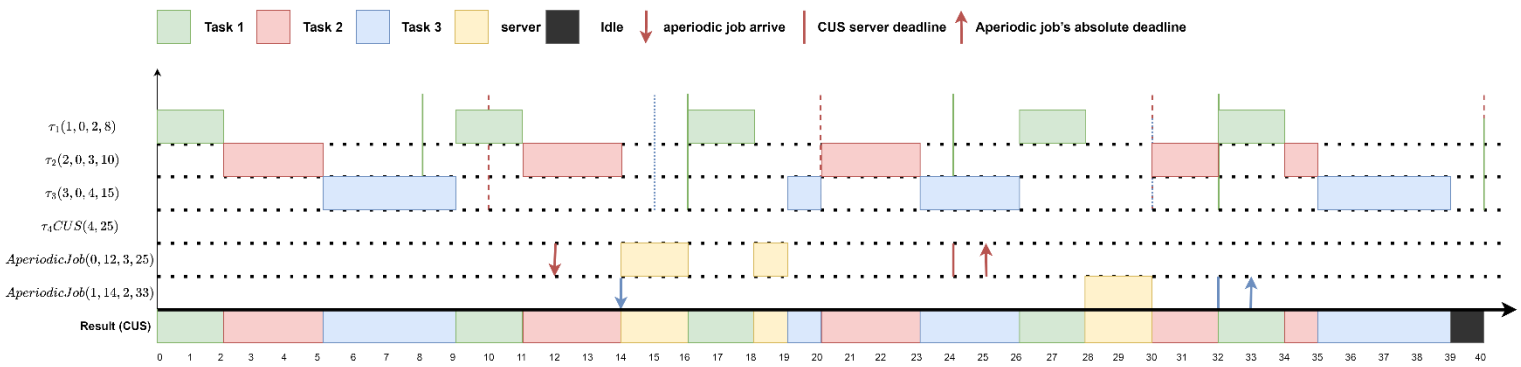
Type	Task ID	Arrive Time	Execution Time	Task Periodic
Periodic Job	##	##	##	##
Type	Server ID	Server Size		
Server	##	##		

```
1 0 2 8
2 0 3 10
3 0 4 15
4 25
```

The AperiodicJobs.txt format:

Jobs No.	Arrive Time	Execution Time	Absolute Deadline
##	##	##	##

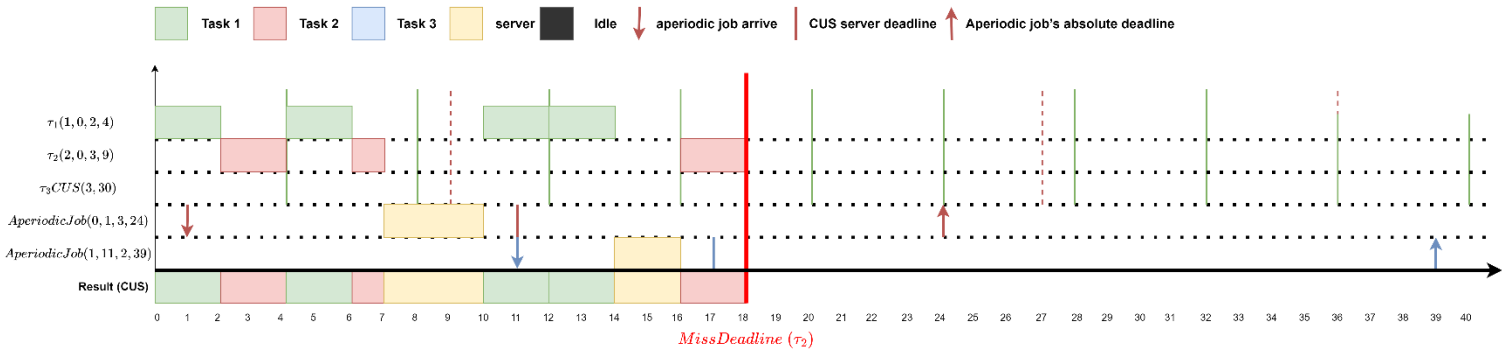
```
0 12 3 25
1 14 2 33
```



The output results of Example 1:

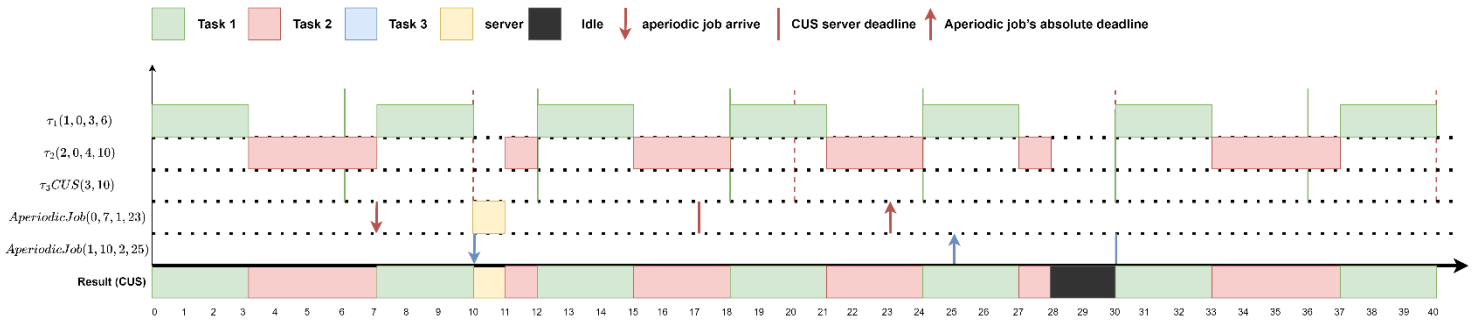
```
0    Task (  1) is running
1    Task (  1) is running
2    Completion    task(  1)(  0)    task(  2)(  0)       2            0            6
2    Task (  2) is running
3    Task (  2) is running
4    Task (  2) is running
5    Completion    task(  2)(  0)    task(  3)(  0)       5            2            5
5    Task (  3) is running
6    Task (  3) is running
7    Task (  3) is running
8    Task (  3) is running
9    Completion    task(  3)(  0)    task(  1)(  1)       9            5            6
9    Task (  1) is running
10   Task (  1) is running
11   Completion    task(  1)(  1)    task(  2)(  1)       3            1            5
11   Task (  2) is running
12   Aperiodic job (  0) arrives and sets CUS's deadline as  24
12   Task (  2) is running
13   Task (  2) is running
14   Aperiodic job (  1) arrives. Do nothing.
14   Completion    task(  2)(  1)    task(  4)(  0)       4            1            6
14   Aperiodic job (  0) is running
15   Aperiodic job (  0) is running
16   Preemption    task(  4)(  0)    task(  1)(  2)
16   Task (  1) is running
17   Task (  1) is running
18   Completion    task(  1)(  2)    task(  4)(  0)       2            0            6
18   Aperiodic job (  0) is running
19   Aperiodic job (  0) is finished.
19   Completion    task(  4)(  0)    task(  3)(  1)       7            4          N/A
19   Task (  3) is running
20   Preemption    task(  3)(  1)    task(  2)(  2)
```

The output results of Example 2:




```
0    Task (  1) is running
1    Aperiodic job (  0) arrives and sets CUS's deadline as  11
1    Task (  1) is running
2    Completion    task(  1)(  0)    task(  2)(  0)       2            0            2
2    Task (  2) is running
3    Task (  2) is running
4    Preemption    task(  2)(  0)    task(  1)(  1)
4    Task (  1) is running
5    Task (  1) is running
6    Completion    task(  1)(  1)    task(  2)(  0)       2            0            2
6    Task (  2) is running
7    Completion    task(  2)(  0)    task(  3)(  0)       7            4            2
7    Aperiodic job (  0) is running
8    Aperiodic job (  0) is running
9    Aperiodic job (  0) is running
10   Aperiodic job (  0) is finished.
10   Completion    task(  3)(  0)    task(  1)(  2)       9            6          N/A
10   Task (  1) is running
11   Aperiodic job (  1) arrives and sets CUS's deadline as  17
11   Task (  1) is running
12   Completion    task(  1)(  2)    task(  1)(  3)       4            2            0
12   Task (  1) is running
13   Task (  1) is running
14   Completion    task(  1)(  3)    task(  3)(  1)       2            0            2
14   Aperiodic job (  1) is running
15   Aperiodic job (  1) is running
16   Aperiodic job (  1) is finished.
16   Completion    task(  3)(  1)    task(  2)(  1)       5            3          N/A
16   Task (  2) is running
17   Task (  2) is running
18   MissDeadline  task(  2)(  1)    ------------------
```

The output results of Example 3:




```
 0    Task (  1) is running
 1    Task (  1) is running
 2    Task (  1) is running
 3    Completion    task(  1)(  0)    task(  2)(  0)       3            0            3
 3    Task (  2) is running
 4    Task (  2) is running
 5    Task (  2) is running
 6    Task (  2) is running
 7    Aperiodic job (  0) arrives and sets CUS's deadline as  17
 7    Completion    task(  2)(  0)    task(  1)(  1)       7            3            3
 7    Task (  1) is running
 8    Task (  1) is running
 9    Task (  1) is running
10    Aperiodic job (  1) arrives. Do nothing.
10    Completion    task(  1)(  1)    task(  3)(  0)       4            1            2
10    Aperiodic job (  0) is running
11    Aperiodic job (  0) is finished.
11    Completion    task(  3)(  0)    task(  2)(  1)       4            3          N/A
11    Task (  2) is running
12    Preemption    task(  2)(  1)    task(  1)(  2)
12    Task (  1) is running
13    Task (  1) is running
14    Task (  1) is running
15    Completion    task(  1)(  2)    task(  2)(  1)       3            0            3
15    Task (  2) is running
16    Task (  2) is running
17    Aperiodic job (  1) rejects scheduling.
17    Task (  2) is running
18    Completion    task(  2)(  1)    task(  1)(  3)       8            4            2
18    Task (  1) is running
19    Task (  1) is running
20    Task (  1) is running
```

Credit:

[PART I] EDF Scheduler Implementation [50%]

- The correctness of schedule results of examples. Note the testing task set might not be the same as the given example task set. (40%)
  - Include and implement missed deadline handling in EDF.
- A report that describes your implementation (please attach the screenshot of the code and **MARK** the modified part). (10%)

[PART II] CUS Scheduler Implementation [50%]

- The correctness of schedule results of examples. Note the testing task set might not be the same as the given example task set. (40%)
  - Include and implement missed deadline handling in CUS.
- A report that describes your implementation (please attach the screenshot of the code and **MARK** the modified part). (10%)

※ You must modify the source code!

※ Standard input and output filenames in the project are necessary for the checker. Please check the file names before submitting. You must print out the result on the Output.txt file.

```
#define INPUT_FILE_NAME "./TaskSet.txt"
#define OUTPUT_FILE_NAME "./Output.txt"
#define APERIODIC_FILE_NAME "./Aperiodicjobs.txt"
```

※ Please set the system end time as 40 seconds in this project.

```
#define SYSTEM_END_TIME 40
```

※ You must check your project can produce the correct output file.

※ We will use different task sets to verify your code.

※ We will check your answer in Output.txt.

※ You must define the algorithm in ucos_ii.h.

```
#define RM 0
#define FIFO 1
#define EDF 2
#define ALGORITHM EDF
```

Project submit:

Submit to Moodle.

Submit deadline: Nov 27, 2025 (Thursday) 8:00

File name format: RTOS_Mxxxxxxxx_PA2.zip

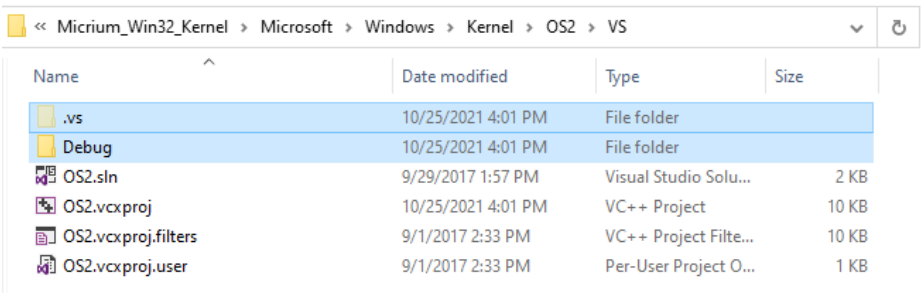
RTOS_Myyyddxxx_PA2.zip includes:

- The report (RTOS_Myyyddxxx_PA2.pdf).
- Folder with the executable project (**RTOS_Myyyddxxx_PA2**).

※ Plagiarizing is strictly prohibited.

Hints:

1. Please delete the ".vs" and "Debug" folders.



2. RTOS_Myyyddxxx_PA2.zip must include files as follow: