Embedded OS Implementation, Fall 2023
Project #2 (due Nov 22, 2023 (Wednesday) 12:00)

[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 suffers the schedule delay.

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

Example Task Set 1 = {τ_1 (1, 0, 5, 10), τ_2 (2, 0, 2, 5)}

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

Example Task Set 3 = {τ_1 (1, 0, 2, 5), τ_2 (2, 0, 4, 8), τ_3 (3, 1, 2, 6)}

※ 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.

The input file format:

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

Example of file 1:

```
1 0 5 10
2 0 2 5
```

Evaluation:

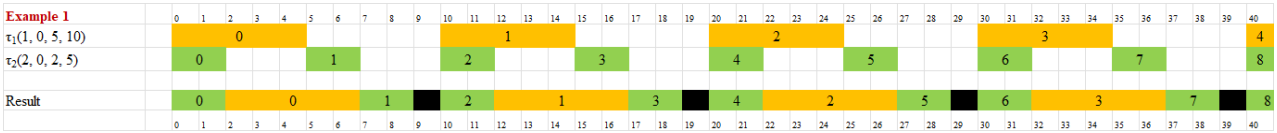
The output format:

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)	----------			

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

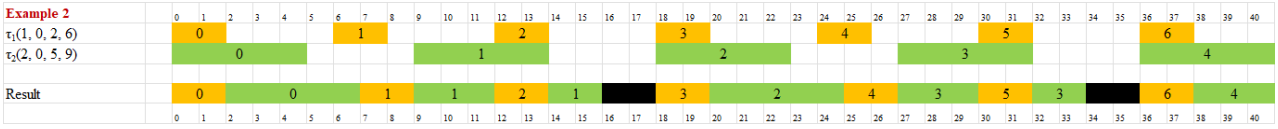
※ When a miss deadline event is triggered, the "Next task ID" format can be modified by yourself.

The output results of Example 1:



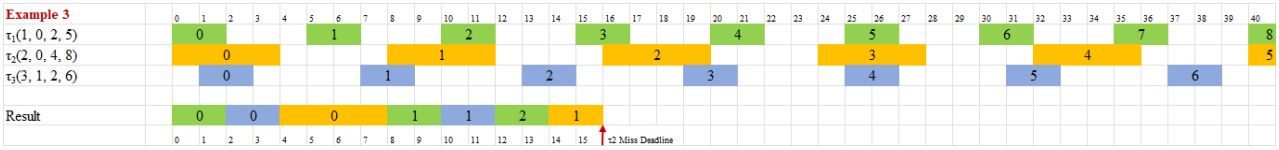
Tick	Event	CurrentTask ID	NextTask ID	ResponseTime	PreemptionTime	OSTimeDly
2	Completion	task(2)(0)	task(1)(0)	2	0	3
7	Completion	task(1)(0)	task(2)(1)	7	2	3
9	Completion	task(2)(1)	task(63)	4	2	1
10	Preemption	task(63)	task(2)(2)			
12	Completion	task(2)(2)	task(1)(1)	2	0	3
17	Completion	task(1)(1)	task(2)(3)	7	2	3
19	Completion	task(2)(3)	task(63)	4	2	1
20	Preemption	task(63)	task(2)(4)			
22	Completion	task(2)(4)	task(1)(2)	2	0	3
27	Completion	task(1)(2)	task(2)(5)	7	2	3
29	Completion	task(2)(5)	task(63)	4	2	1
30	Preemption	task(63)	task(2)(6)			
32	Completion	task(2)(6)	task(1)(3)	2	0	3
37	Completion	task(1)(3)	task(2)(7)	7	2	3
39	Completion	task(2)(7)	task(63)	4	2	1
40	Preemption	task(63)	task(2)(8)			

The output results of Example 2:



Tick	Event	CurrentTask ID	NextTask ID	ResponseTime	PreemptionTime	OSTimeDly
2	Completion	task(1)(0)	task(2)(0)	2	0	4
7	Completion	task(2)(0)	task(1)(1)	7	2	2
9	Completion	task(1)(1)	task(2)(1)	3	1	3
12	Preemption	task(2)(1)	task(1)(2)			
14	Completion	task(1)(2)	task(2)(1)	2	0	4
16	Completion	task(2)(1)	task(63)	7	2	2
18	Preemption	task(63)	task(1)(3)			
20	Completion	task(1)(3)	task(2)(2)	2	0	4
25	Completion	task(2)(2)	task(1)(4)	7	2	2
27	Completion	task(1)(4)	task(2)(3)	3	1	3
30	Preemption	task(2)(3)	task(1)(5)			
32	Completion	task(1)(5)	task(2)(3)	2	0	4
34	Completion	task(2)(3)	task(63)	7	2	2
36	Preemption	task(63)	task(1)(6)			
38	Completion	task(1)(6)	task(2)(4)	2	0	4

The output results of Example 3:



Tick	Event	CurrentTask ID	NextTask ID	ResponseTime	PreemptionTime	OSTimeDly
2	Completion	task(1)(0)	task(3)(0)	2	0	3
4	Completion	task(3)(0)	task(2)(0)	3	1	3
8	Completion	task(2)(0)	task(1)(1)	8	4	0
10	Completion	task(1)(1)	task(3)(1)	5	3	0
12	Completion	task(3)(1)	task(1)(2)	5	3	1
14	Completion	task(1)(2)	task(2)(1)	4	2	1
16	MissDeadline	task(2)(1)	----------			

[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 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 examples.

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

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

========== **Example** ==========

Periodic Task Set = {τ_1 (1, 0, 5, 10), τ_2 (2, 0, 2, 5), $\tau_{3\_ServerSize}$ (3, 10%)}

Aperiodic Jobs Set = {j_0 (0, 3, 1, 18), j_1 (1, 11, 2, 37)}

※ 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.**

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

The TaskSet.txt format:

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

```
1 0 5 10
2 0 2 5
3 10
```

The AperiodicJobs.txt format:

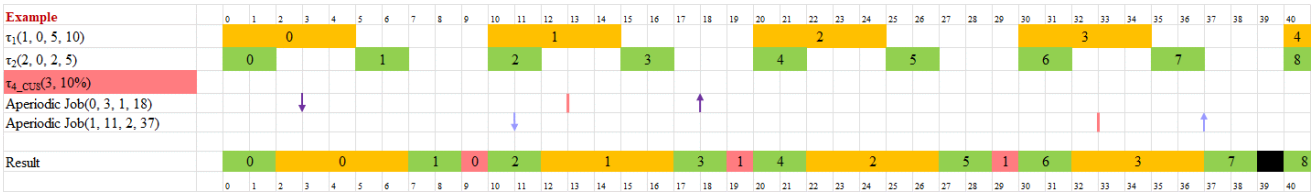
Jobs No.	Arrive Time	Execution Time	Absolute Deadline
##	##	##	##

```
0 3 1 18
1 11 2 37
```

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)	##	##	## or N/A
##	MissDeadline	task(ID)(job number)	----------			

The output results of Example in CUS:



Aperiodic job's arrival time Aperiodic job's absolute deadline CUS server deadline

```
Tick    Event          CurrentTask ID  NextTask ID     ResponseTime    PreemptionTime  OSTimeDly
 2      Completion      task( 2)( 0)    task( 1)( 0)     2               0               3
 3      Aperiodic job(0) arrives and sets CUS server's deadline as 13.
 7      Completion      task( 1)( 0)    task( 2)( 1)     7               2               3
 9      Completion      task( 2)( 1)    task( 3)( 0)     4               2               1
10      Aperiodic job(0) is finished.
10      Completion      task( 3)( 0)    task( 2)( 2)     7               6              N/A
11      Aperiodic job(1) arrives. Do nothing.
12      Completion      task( 2)( 2)    task( 1)( 1)     2               0               3
13      Aperiodic job(1) sets CUS server's deadline as 33.
17      Completion      task( 1)( 1)    task( 2)( 3)     7               2               3
19      Completion      task( 2)( 3)    task( 3)( 1)     4               2               1
20      Preemption      task( 3)( 1)    task( 2)( 4)
22      Completion      task( 2)( 4)    task( 1)( 2)     2               0               3
27      Completion      task( 1)( 2)    task( 2)( 5)     7               2               3
29      Completion      task( 2)( 5)    task( 3)( 1)     4               2               1
30      Aperiodic job(1) is finished.
30      Completion      task( 3)( 1)    task( 2)( 6)    19              17              N/A
32      Completion      task( 2)( 6)    task( 1)( 3)     2               0               3
37      Completion      task( 1)( 3)    task( 2)( 7)     7               2               3
39      Completion      task( 2)( 7)    task(63)         4               2               1
40      Preemption      task(63)        task( 2)( 8)
```

Credit:

[PART I] EDF Scheduler Implementation [70%]

- The correctness of schedule results of examples. Note the testing task set might not be the same as the given example task set. (20%)
- Implement and describe how to handle the missing deadline situation under EDF. (10%)
- A report that describes your implementation (please attach the screenshot of the code and **MARK** the modified part). (40%)

[PART II] CUS Scheduler Implementation [30%]

- The correctness of schedule results of examples. Note the testing task set might not be the same as the given example task set. (15%)
- A report that describes your implementation (please attach the screenshot of the code and **MARK** the modified part). (15%)

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

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

※ You will submit two µC/OS-II projects for PART I and PART II, respectively.

Project submit:

Submit to Moodle.

Submit deadline: Nov 22, 2023 (Wednesday) 12:00

File name format: RTOS_Myyyddxxx_PA2.zip

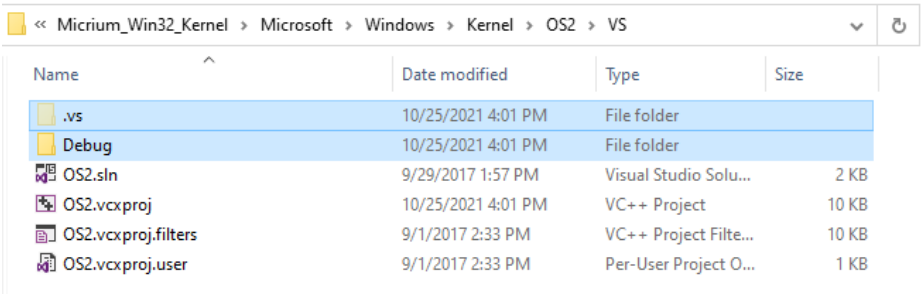
RTOS_Myyyddxxx_PA2.zip includes:

- The report (RTOS_Myyyddxxx_PA2.pdf).
- Folder with the executable µC/OS-II project (**RTOS_Myyyddxxx_PA2_EDF**).
- Folder with the executable µC/OS-II project (**RTOS_Myyyddxxx_PA2_CUS**).

※ Plagiarizing is strictly prohibited.

Hints:

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



2. RTOS_Myyyddxxx_PA2.zip must be including files as follow:

```
C:.
    RTOS_Myyyddxxx_PA2.pdf

├──RTOS_Myyyddxxx_PA2_CUS
    ├──Micrium
        └──Software
            ├──uC-CPU
                    cpu_cache.h
                    cpu_core.c
                    cpu_core.h
                    cpu_def.h

                └──Win32
                    └──Visual_Studio
                            cpu.h
                            cpu_c.c

            ├──uC-LIB
                    lib_ascii.c
                    lib_ascii.h
                    lib_def.h
                    lib_math.c
                    lib_math.h
                    lib_mem.c
                    lib_mem.h
                    lib_str.c
                    lib_str.h

            └──uCOS-II
                ├──Ports
                    └──Win32
                        └──Visual Studio
                                os_cpu.h
                                os_cpu_c.c

                └──Source
                        os.h
                        os_cfg_r.h
                        os_core.c
                        os_dbg_r.c
                        os_flag.c
                        os_mbox.c
                        os_mem.c
                        os_mutex.c
                        os_q.c
                        os_sem.c
                        os_task.c
                        os_time.c
                        os_tmr.c
                        os_trace.h
                        ucos_ii.c
                        ucos_ii.h
```

```
    └──Microsoft
        ├──BSP
            └──Windows
                    bsp_cpu.c

        └──Windows
            └──Kernel
                    app_cfg.h
                    cpu_cfg.h
                    lib_cfg.h

                └──OS2
                        app_hooks.c
                        main.c
                        os_cfg.h

                    └──VS
                            OS2.sln
                            OS2.vcxproj
                            OS2.vcxproj.filters
                            OS2.vcxproj.user
```

```
└──RTOS_Myyyddxxx_PA2_EDF
    ├──Micrium
    │   └──Software
    │       ├──uC-CPU
    │       │   │   cpu_cache.h
    │       │   │   cpu_core.c
    │       │   │   cpu_core.h
    │       │   │   cpu_def.h
    │       │   │
    │       │   └──Win32
    │       │       └──Visual_Studio
    │       │               cpu.h
    │       │               cpu_c.c
    │       │
    │       ├──uC-LIB
    │       │       lib_ascii.c
    │       │       lib_ascii.h
    │       │       lib_def.h
    │       │       lib_math.c
    │       │       lib_math.h
    │       │       lib_mem.c
    │       │       lib_mem.h
    │       │       lib_str.c
    │       │       lib_str.h
    │       │
    │       └──uCOS-II
    │           ├──Ports
    │           │   └──Win32
    │           │       └──Visual Studio
    │           │               os_cpu.h
    │           │               os_cpu_c.c
    │           │
    │           └──Source
    │                   os.h
    │                   os_cfg_r.h
    │                   os_core.c
    │                   os_dbg_r.c
    │                   os_flag.c
    │                   os_mbox.c
    │                   os_mem.c
    │                   os_mutex.c
    │                   os_q.c
    │                   os_sem.c
    │                   os_task.c
    │                   os_time.c
    │                   os_tmr.c
    │                   os_trace.h
    │                   ucos_ii.c
    │                   ucos_ii.h
```

```
└──Microsoft
    ├──BSP
    │   └──Windows
    │           bsp_cpu.c
    │
    └──Windows
        └──Kernel
            │   app_cfg.h
            │   cpu_cfg.h
            │   lib_cfg.h
            │
            └──OS2
                │   app_hooks.c
                │   main.c
                │   os_cfg.h
                │
                └──VS
                        OS2.sln
                        OS2.vcxproj
                        OS2.vcxproj.filters
                        OS2.vcxproj.user
```