

Embedded OS Implementation, Fall 2023
Project #1 (due October 25th, 2023(Wednesday) at 12:00)

[PART I] Task Control Block Linked List

Objective:

Following the previous homework (HW1), please add some code to the μ C/OS-II scheduler in the kernel level to observe the operations of the task control block (TCB) and TCB linked list.

※ The TCB address is dynamic.

The output results are shown below:

```
OSTick    created, Thread ID 11760
Task[ 63] created, TCB Address  92e9e0
-----After TCB[63] begin linked-----
Previous TCB point to adress      0
Current  TCB point to adress      92e9e0
Next     TCB point to adress      0

The file 'TaskSet.txt' was opened
Task[  1] created, TCB Address  92ea54
-----After TCB[ 1] begin linked-----
Previous TCB point to adress      0
Current  TCB point to adress      92ea54
Next     TCB point to adress      92e9e0

Task[  2] created, TCB Address  92eac8
-----After TCB[ 2] begin linked-----
Previous TCB point to adress      0
Current  TCB point to adress      92eac8
Next     TCB point to adress      92ea54

=====TCB linked list=====
Task    Prev_TCB_addr    TCB_addr    Next_TCB_addr
  2             0         92eac8         92ea54
  1        92eac8         92ea54         92e9e0
 63        92ea54         92e9e0             0
```

[PART II] RM Scheduler Implementation

Objective:

To implement the Rate Monotonic (RM) scheduler for periodic tasks and observe the scheduling behaviors.

Problem Definition:

Implement the following three task sets of periodic tasks. Add necessary code to the μ C/OS-II scheduler **in the kernel level** to observe how the task suffers from the scheduler. We give the files for the parameter of the task.

Periodic Task Set = $\{\tau_{ID} (ID, \text{arrival time}, \text{execution time}, \text{period})\}$

Example Task Set 1 = $\{\tau_1 (1, 0, 1, 5), \tau_2 (2, 0, 3, 8)\}$

Example Task Set 2 = $\{\tau_1 (1, 2, 2, 6), \tau_2 (2, 0, 5, 15), \tau_3 (3, 1, 2, 10)\}$

Example Task Set 3 = $\{\tau_1 (1, 1, 3, 6), \tau_2 (2, 0, 4, 9), \tau_3 (3, 3, 2, 8)\}$

✂ The priority of the task is set according to the RM scheduling rules.

The input file format:

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

Example of task set file 1:

```
1 0 1 5
2 0 3 8
```

Example output file of task set 1:

1	Completion	task(1)(0)	task(2)(0)	1	0	4
4	Completion	task(2)(0)	task(63)	4	1	4
5	Preemption	task(63)	task(1)(1)			
6	Completion	task(1)(1)	task(63)	1	0	4
8	Preemption	task(63)	task(2)(1)			
10	Preemption	task(2)(1)	task(1)(2)			
11	Completion	task(1)(2)	task(2)(1)	1	0	4
12	Completion	task(2)(1)	task(63)	4	1	4
15	Preemption	task(63)	task(1)(3)			
16	Completion	task(1)(3)	task(2)(2)	1	0	4
19	Completion	task(2)(2)	task(63)	3	0	5
20	Preemption	task(63)	task(1)(4)			
21	Completion	task(1)(4)	task(63)	1	0	4
24	Preemption	task(63)	task(2)(3)			
25	Preemption	task(2)(3)	task(1)(5)			
26	Completion	task(1)(5)	task(2)(3)	1	0	4
28	Completion	task(2)(3)	task(63)	4	1	4
30	Preemption	task(63)	task(1)(6)			

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)*”.

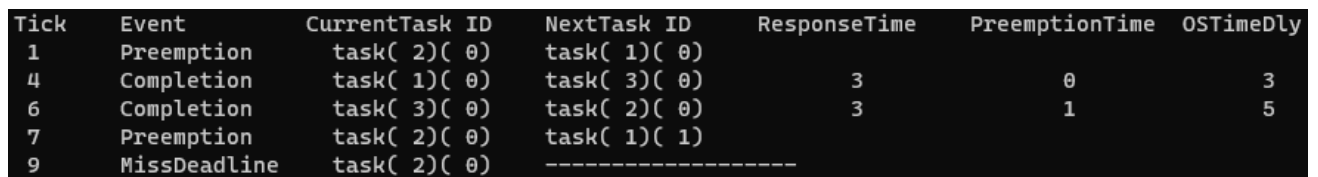
The output results of Task Set 1:

TaskSet 1(RM)	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
$\tau_1(1, 0, 1, 5)$	0					1					2						3				4						5				
$\tau_2(2, 0, 3, 8)$			0							1									2									3			
Result (RM)	0			0		1				1	2	1					3		2		4					3	5		3		

Tick	Event	CurrentTask ID	NextTask ID	ResponseTime	PreemptionTime	OSTimeDly
1	Completion	task(1)(0)	task(2)(0)	1	0	4
4	Completion	task(2)(0)	task(63)	4	1	4
5	Preemption	task(63)	task(1)(1)			
6	Completion	task(1)(1)	task(63)	1	0	4
8	Preemption	task(63)	task(2)(1)			
10	Preemption	task(2)(1)	task(1)(2)			
11	Completion	task(1)(2)	task(2)(1)	1	0	4
12	Completion	task(2)(1)	task(63)	4	1	4
15	Preemption	task(63)	task(1)(3)			
16	Completion	task(1)(3)	task(2)(2)	1	0	4
19	Completion	task(2)(2)	task(63)	3	0	5
20	Preemption	task(63)	task(1)(4)			
21	Completion	task(1)(4)	task(63)	1	0	4
24	Preemption	task(63)	task(2)(3)			
25	Preemption	task(2)(3)	task(1)(5)			
26	Completion	task(1)(5)	task(2)(3)	1	0	4
28	Completion	task(2)(3)	task(63)	4	1	4
30	Preemption	task(63)	task(1)(6)			

TaskSet 2(RM)	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	
$\tau_1(1, 2, 2, 6)$					0						1						2						3							4		
$\tau_2(2, 0, 5, 15)$						0															1											
$\tau_3(3, 1, 2, 10)$				0										1										2								
Result (RM)		0	0		0	0			0		1	0		1			2				1		3		2		1			4		

The output results of Task Set 3:



[Part III] FIFO Scheduler Implementation

Objective:

To implement the non-preemptive First In First Out (FIFO) scheduling for periodic tasks, and handle the miss deadline behaviors.

Problem Definition:

Implement the following task set of periodic tasks. Add necessary code to the μ C/OS-II scheduler **in the kernel level** to observe how the task suffers the schedule delay.

Periodic Task Set = $\{\tau_{ID}(ID, \text{arrival time}, \text{execution time}, \text{period})\}$

Task Set 1 = $\{\tau_1(1, 0, 1, 5), \tau_2(2, 0, 3, 8)\}$

Task Set 2 = $\{\tau_1(1, 2, 2, 6), \tau_2(2, 0, 5, 15), \tau_3(3, 1, 2, 10)\}$

※If tasks arrive simultaneously, the task with the **smaller TaskID** will be executed first.

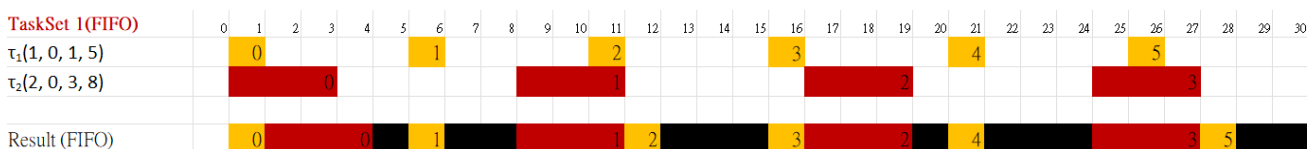
Evaluation:

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)	-----			

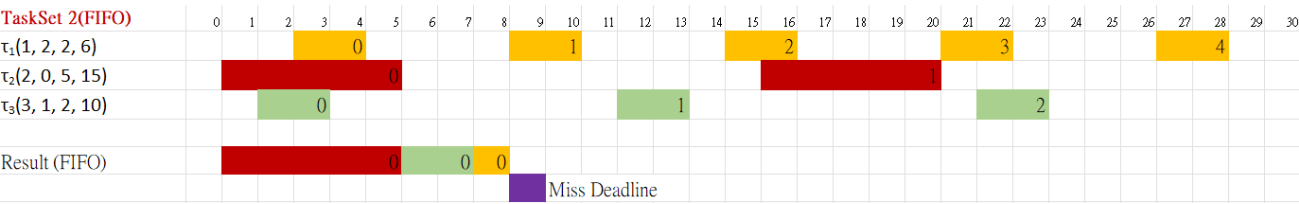
※ If the task is Idle Task, print “*task(priority)*”.

The output results of Task Set 1:



Tick	Event	CurrentTask ID	NextTask ID	ResponseTime	PreemptionTime	OSTimeDly
1	Completion	task(1)(0)	task(2)(0)	1	0	4
4	Completion	task(2)(0)	task(63)	4	1	4
5	Preemption	task(63)	task(1)(1)			
6	Completion	task(1)(1)	task(63)	1	0	4
8	Preemption	task(63)	task(2)(1)			
11	Completion	task(2)(1)	task(1)(2)	3	0	5
12	Completion	task(1)(2)	task(63)	2	1	3
15	Preemption	task(63)	task(1)(3)			
16	Completion	task(1)(3)	task(2)(2)	1	0	4
19	Completion	task(2)(2)	task(63)	3	0	5
20	Preemption	task(63)	task(1)(4)			
21	Completion	task(1)(4)	task(63)	1	0	4
24	Preemption	task(63)	task(2)(3)			
27	Completion	task(2)(3)	task(1)(5)	3	0	5
28	Completion	task(1)(5)	task(63)	3	2	2
30	Preemption	task(63)	task(1)(6)			

The output results of Task Set 2:



Tick	Event	CurrentTask	ID	NextTask	ID	ResponseTime	PreemptionTime	OSTimeDly
5	Completion	task(2)	(0)	task(3)	(0)	5	0	10
7	Completion	task(3)	(0)	task(1)	(0)	6	4	4
8	MissDeadline	task(1)	(0)	-----				

Credit:

[PART I] Task Control Block Linked List [20%]

- The screenshot results. (10%)
- A report that describes your implementation (please attach the screenshot of the code and **MARK** the modified part). (10%)

[PART II] RM 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. (25%)
- A report that describes your implementation (please attach the screenshot of the code and **MARK** the modified part). (40%)
- Implement and describe how to handle the deadline missing situation under RM. (5%)

[PART III] FIFO Scheduler Implementation [10%]

- The correctness of schedule results of examples. Note the testing task set might not be the same as the given example task set. (5%)
- Implement FIFO and compare the schedule results with that of RM (please attach the screenshot of the code and **MARK** the modified part). (5%)

※ 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.

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

※ Please set the system end time as 30 seconds in this project.

```
#define SYSTEM_END_TIME 30
```

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

※ **When the current task is completed, the completion information shall be printed even if there is one task missing its deadline.**

Project submit:

Submit to Moodle2.

Submit deadline: October 25th, 2023(Wednesday) at 12:00

File name format: RTOS_Myyyddxxx_PA1.zip

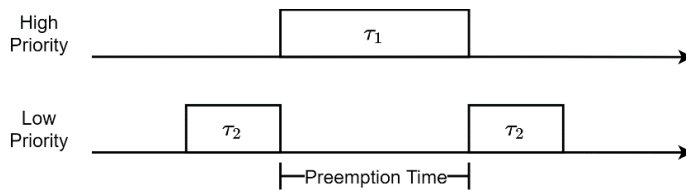
RTOS_Myyyddxxx_PA1.zip includes (The tree structure of files is shown as hints):

- The report (RTOS_Myyyddxxx_PA1.pdf).
- Folder with the executable μ C/OS-II project (RTOS_Myyyddxxx_PA1_RM & RTOS_Myyyddxxx_PA1_FIFO).

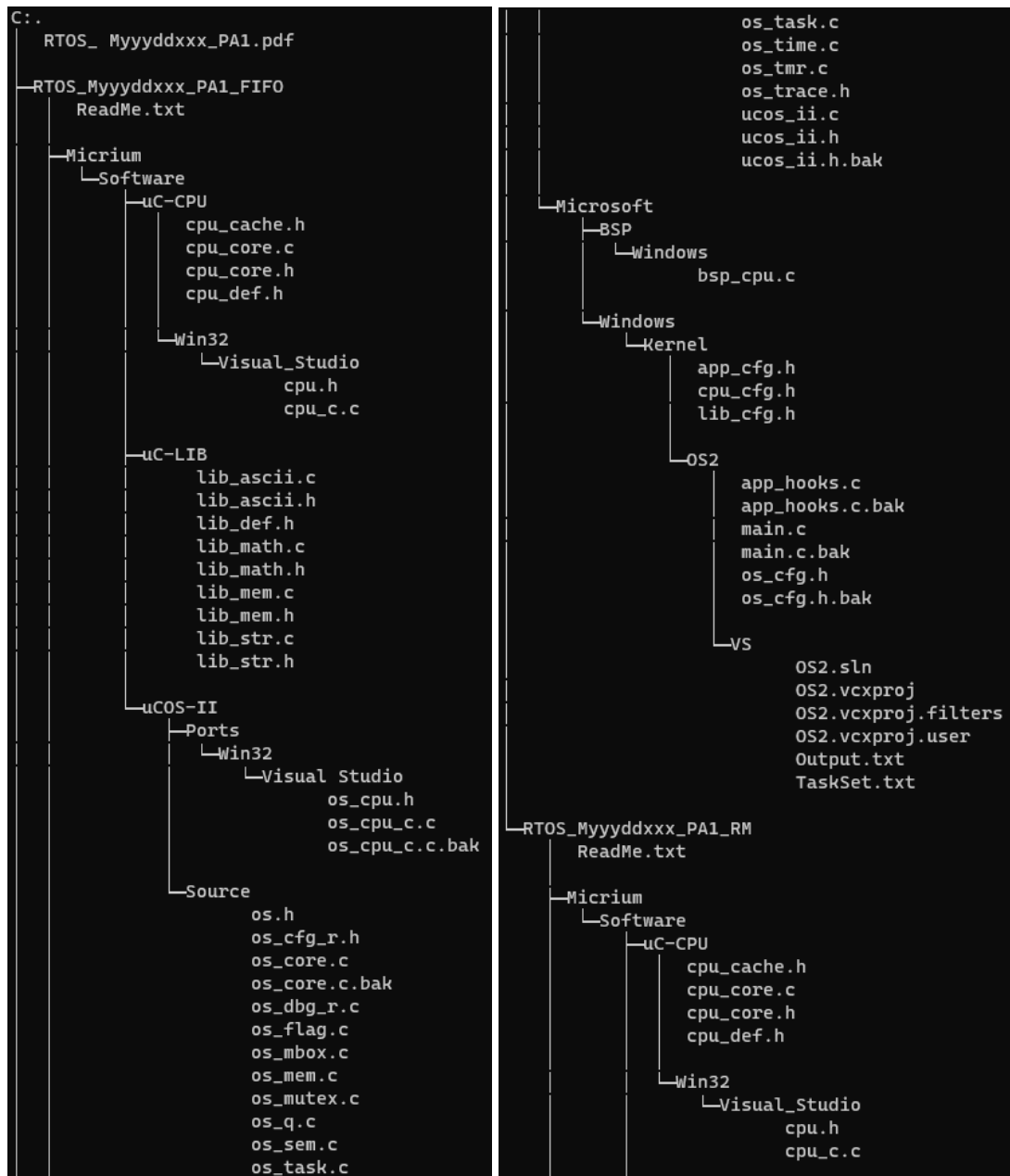
※ Plagiarizing is strictly prohibited.

Hints:

1. Preemption time is introduced in multiple tasking.



2. RTOS_Myyyddxxx_PA1.zip include files as follows:



```

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
                os_cpu.c.c.bak

    └─Source
        os.h
        os_cfg_r.h
        os_core.c
        os_core.c.bak
        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
        ucos_ii.h.bak

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

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

```

```

lib_cfg.h

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

└─vs
    OS2.sln
    OS2.vcxproj
    OS2.vcxproj.filters
    OS2.vcxproj.user
    Output.txt
    TaskSet.txt

```