Embedded OS Implementation, Fall 2023
Project #3 (due Dec. 13, 2023 (Wednesday) 12:00)

[PART I] NPCS Implementation

Objective:

Implement the non-preemptible critical section (NPCS) based on the RM scheduler in uC/OS-II.

Problem Definition:

uC/OS-II uses a variation of the priority inheritance protocol to deal with priority inversions. In this assignment, you are going to implement the NPCS based on the RM scheduler in uC/OS-II.

Consider the two examples and observe how the task suffers the scheduler delay.

Periodic Task Set = { $task_{ID}$ (ID, arrival time, execution time, period, R1 lock, R1 unlock, R2 lock, R2 unlock) }

※ L(R#): Lock resource #, U(R#): Unlock resource #

Example Task Set 1 = { $task_1$ (1, 2, 8, 20, 2, 7, 4, 6),
$task_2$ (2, 0, 11, 40, 5, 8, 1, 9) }

Example Task Set 2 = { $task_1$ (1, 2, 4, 15, 1, 2, 0, 0),
$task_2$ (2, 10, 3, 20, 0, 0, 0, 0),
$task_3$ (3, 0, 7, 22, 0, 0, 1, 6)}

The input file format:

Task ID	Arrival Time	Execution Time	Task Period	R1 Lock Time	R1 Unlock Time	R2 Lock Time	R2 Unlock Time
##	##	##	##	##	##	##	##

Example of the input file:

```
1 2 4 15 1 2 0 0
2 10 3 20 0 0 0 0
3 0 7 22 0 0 1 6
```

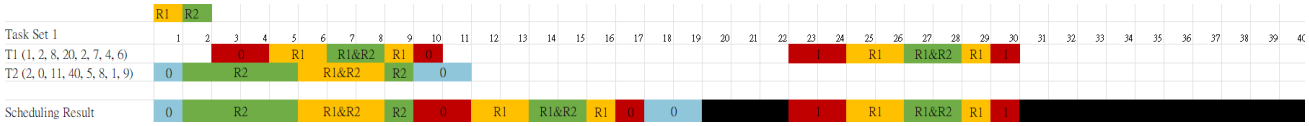
※ Lock time and unlock time are relative to the task start time.

Evaluation:

The output format:

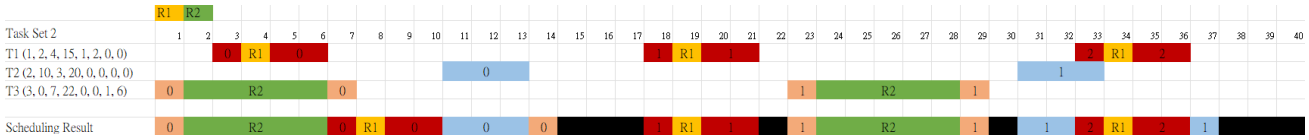
Tick	Event	CurrentTask ID	NextTask ID	Response Time	Blocking Time	Preemption Time	Resource Name
##	Preemption	task(ID)(job number)	task(ID)(job number)				
##	Completion	task(ID)(job number)	task(ID)(job number)	##	##	##	
##	LockResource	task(ID)(job number)					R#
##	UnlockResource	task(ID)(job number)					R#

The output results of Example 1:




```
  1       LockResource      task( 2)( 0)    R2
  5       LockResource      task( 2)( 0)    R1
  8       UnlockResource    task( 2)( 0)    R1
  9       UnlockResource    task( 2)( 0)    R2
  9       Preemption        task( 2)( 0)    task( 1)( 0)
 11       LockResource      task( 1)( 0)    R1
 13       LockResource      task( 1)( 0)    R2
 15       UnlockResource    task( 1)( 0)    R2
 16       UnlockResource    task( 1)( 0)    R1
 17       Completion        task( 1)( 0)    task( 2)( 0)            15              7               0
 19       Completion        task( 2)( 0)    task(63)                19              0               8
 22       Preemption        task(63)        task( 1)( 1)
 24       LockResource      task( 1)( 1)    R1
 26       LockResource      task( 1)( 1)    R2
 28       UnlockResource    task( 1)( 1)    R2
 29       UnlockResource    task( 1)( 1)    R1
 30       Completion        task( 1)( 1)    task(63)                8               0               0
 40       Preemption        task(63)        task( 2)( 1)
 41       LockResource      task( 2)( 1)    R2
 45       LockResource      task( 2)( 1)    R1
 48       UnlockResource    task( 2)( 1)    R1
 49       UnlockResource    task( 2)( 1)    R2
 49       Preemption        task( 2)( 1)    task( 1)( 2)
 51       LockResource      task( 1)( 2)    R1
 53       LockResource      task( 1)( 2)    R2
 55       UnlockResource    task( 1)( 2)    R2
 56       UnlockResource    task( 1)( 2)    R1
 57       Completion        task( 1)( 2)    task( 2)( 1)            15              7               0
 59       Completion        task( 2)( 1)    task(63)                19              0               8
 62       Preemption        task(63)        task( 1)( 3)
 64       LockResource      task( 1)( 3)    R1
 66       LockResource      task( 1)( 3)    R2
 68       UnlockResource    task( 1)( 3)    R2
 69       UnlockResource    task( 1)( 3)    R1
 70       Completion        task( 1)( 3)    task(63)                8               0               0
 80       Preemption        task(63)        task( 2)( 2)
 81       LockResource      task( 2)( 2)    R2
 85       LockResource      task( 2)( 2)    R1
 88       UnlockResource    task( 2)( 2)    R1
 89       UnlockResource    task( 2)( 2)    R2
 89       Preemption        task( 2)( 2)    task( 1)( 4)
 91       LockResource      task( 1)( 4)    R1
 93       LockResource      task( 1)( 4)    R2
 95       UnlockResource    task( 1)( 4)    R2
 96       UnlockResource    task( 1)( 4)    R1
 97       Completion        task( 1)( 4)    task( 2)( 2)            15              7               0
 99       Completion        task( 2)( 2)    task(63)                19              0               8
```

The output results of Example 2:



1	LockResource	task(3)(0)	R2			
6	UnlockResource	task(3)(0)	R2			
6	Preemption	task(3)(0)	task(1)(0)			
7	LockResource	task(1)(0)	R1			
8	UnlockResource	task(1)(0)	R1			
10	Completion	task(1)(0)	task(2)(0)	8	4	0
13	Completion	task(2)(0)	task(3)(0)	3	0	0
14	Completion	task(3)(0)	task(63)	14	0	7
17	Preemption	task(63)	task(1)(1)			
18	LockResource	task(1)(1)	R1			
19	UnlockResource	task(1)(1)	R1			
21	Completion	task(1)(1)	task(63)	4	0	0
22	Preemption	task(63)	task(3)(1)			
23	LockResource	task(3)(1)	R2			
28	UnlockResource	task(3)(1)	R2			
29	Completion	task(3)(1)	task(63)	7	0	0
30	Preemption	task(63)	task(2)(1)			
32	Preemption	task(2)(1)	task(1)(2)			
33	LockResource	task(1)(2)	R1			
34	UnlockResource	task(1)(2)	R1			
36	Completion	task(1)(2)	task(2)(1)	4	0	0
37	Completion	task(2)(1)	task(63)	7	0	4
44	Preemption	task(63)	task(3)(2)			
45	LockResource	task(3)(2)	R2			
50	UnlockResource	task(3)(2)	R2			
50	Preemption	task(3)(2)	task(1)(3)			
51	LockResource	task(1)(3)	R1			
52	UnlockResource	task(1)(3)	R1			
54	Completion	task(1)(3)	task(2)(2)	7	3	0
57	Completion	task(2)(2)	task(3)(2)	7	0	4
58	Completion	task(3)(2)	task(63)	14	0	7
62	Preemption	task(63)	task(1)(4)			
63	LockResource	task(1)(4)	R1			
64	UnlockResource	task(1)(4)	R1			
66	Completion	task(1)(4)	task(3)(3)	4	0	0
67	LockResource	task(3)(3)	R2			
72	UnlockResource	task(3)(3)	R2			
72	Preemption	task(3)(3)	task(2)(3)			
75	Completion	task(2)(3)	task(3)(3)	5	2	0
76	Completion	task(3)(3)	task(63)	10	0	3
77	Preemption	task(63)	task(1)(5)			
78	LockResource	task(1)(5)	R1			
79	UnlockResource	task(1)(5)	R1			
81	Completion	task(1)(5)	task(63)	4	0	0
88	Preemption	task(63)	task(3)(4)			
89	LockResource	task(3)(4)	R2			
94	UnlockResource	task(3)(4)	R2			
94	Preemption	task(3)(4)	task(1)(6)			
95	LockResource	task(1)(6)	R1			
96	UnlockResource	task(1)(6)	R1			
98	Completion	task(1)(6)	task(2)(4)	6	2	0

[PART II] CPP Implementation

Objective:

Implement the ceiling-priority protocol (CPP) based on the RM scheduler in uC/OS-II.

Problem Definition:

uC/OS-II uses a variation of the priority inheritance protocol to deal with priority inversions. In this assignment, you are going to implement the CPP based on the RM scheduler in uC/OS-II.

Consider the two examples and observe how the task suffers the scheduler delay.

Periodic Task Set = { $task_{ID}$ (ID, arrival time, execution time, period, R1 lock, R1 unlock, R2 lock, R2 unlock) }

※ L(R#): Lock resource #, U(R#): Unlock resource #

Example Task Set 1 = { $task_1$ (1, 2, 8, 20, 2, 7, 4, 6),
$task_2$ (2, 0, 11, 40, 5, 8, 1, 9) }

Example Task Set 2 = { $task_1$ (1, 2, 4, 15, 1, 2, 0, 0),
$task_2$ (2, 10, 3, 20, 0, 0, 0, 0),
$task_3$ (3, 0, 7, 22, 0, 0, 1, 6)}

Evaluation:

The output format:

Tick	Event	CurrentTask ID	NextTask ID	Response Time	Blocking Time	Preemption Time	Resource Name	Priority Inheritance
##	Preemption	task(ID)(job number)	task(ID)(job number)					
##	Completion	task(ID)(job number)	task(ID)(job number)	##	##	##		
##	LockResource	task(ID)(job number)					R#	## to ##
##	UnlockResource	task(ID)(job number)					R#	## to ##

Priority & Resource Ceiling assign:

Task Priority = Multiples of 3 (i.e. 3, 6, 9……)

Resource Index : R1 = 1, R2 = 2

Resource ceiling = The highest priority of all tasks that require R - Resource Index

Example :

Task Set 2 = { task1 (1, 2, 4, 15, 1, 2, 0, 0),

task2 (2, 10, 3, 20, 0, 0, 0, 0),

task3 (3, 0, 7, 22, 0, 0, 1, 6)}

task1 priority = 3, task2 priority = 6, task3 priority = 9 (RM)

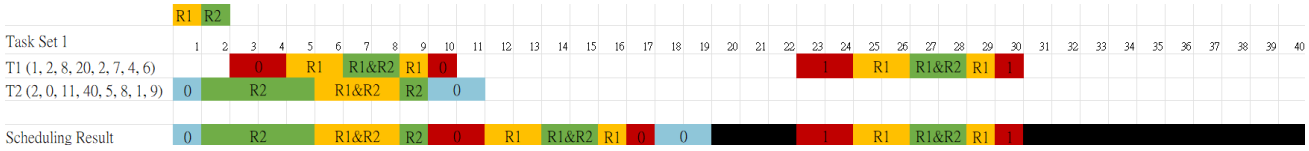
The highest priority of all tasks that require R1 = task1's priority

The highest priority of all tasks that require R2 = task3's priority

R1 ceiling = task1's priority – R1 index = 3 – 1 = 2

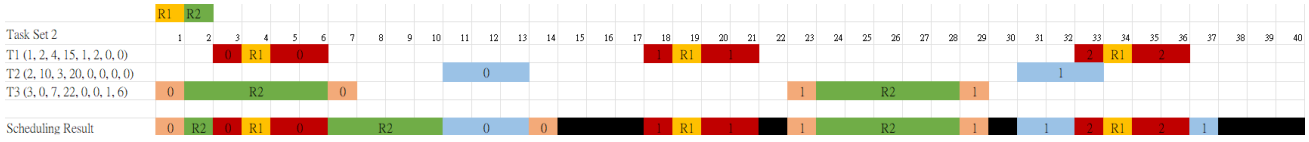
R2 ceiling = task3's priority – R2 index = 9 – 2 = 7

The output results of Example 1:




```
 1      LockResource      task( 2)( 0)    R2                  6 to 1
 5      LockResource      task( 2)( 0)    R1                  1 to 1
 8      UnlockResource    task( 2)( 0)    R1                  1 to 1
 9      UnlockResource    task( 2)( 0)    R2                  1 to 6
 9      Preemption        task( 2)( 0)    task( 1)( 0)
11      LockResource      task( 1)( 0)    R1                  3 to 2
13      LockResource      task( 1)( 0)    R2                  2 to 1
15      UnlockResource    task( 1)( 0)    R2                  1 to 2
16      UnlockResource    task( 1)( 0)    R1                  2 to 3
17      Completion        task( 1)( 0)    task( 2)( 0)        15                7                 0
19      Completion        task( 2)( 0)    task(63)            19                0                 8
22      Preemption        task(63)        task( 1)( 1)
24      LockResource      task( 1)( 1)    R1                  3 to 2
26      LockResource      task( 1)( 1)    R2                  2 to 1
28      UnlockResource    task( 1)( 1)    R2                  1 to 2
29      UnlockResource    task( 1)( 1)    R1                  2 to 3
30      Completion        task( 1)( 1)    task(63)            8                 0                 0
40      Preemption        task(63)        task( 2)( 1)
41      LockResource      task( 2)( 1)    R2                  6 to 1
45      LockResource      task( 2)( 1)    R1                  1 to 1
48      UnlockResource    task( 2)( 1)    R1                  1 to 1
49      UnlockResource    task( 2)( 1)    R2                  1 to 6
49      Preemption        task( 2)( 1)    task( 1)( 2)
51      LockResource      task( 1)( 2)    R1                  3 to 2
53      LockResource      task( 1)( 2)    R2                  2 to 1
55      UnlockResource    task( 1)( 2)    R2                  1 to 2
56      UnlockResource    task( 1)( 2)    R1                  2 to 3
57      Completion        task( 1)( 2)    task( 2)( 1)        15                7                 0
59      Completion        task( 2)( 1)    task(63)            19                0                 8
62      Preemption        task(63)        task( 1)( 3)
64      LockResource      task( 1)( 3)    R1                  3 to 2
66      LockResource      task( 1)( 3)    R2                  2 to 1
68      UnlockResource    task( 1)( 3)    R2                  1 to 2
69      UnlockResource    task( 1)( 3)    R1                  2 to 3
70      Completion        task( 1)( 3)    task(63)            8                 0                 0
80      Preemption        task(63)        task( 2)( 2)
81      LockResource      task( 2)( 2)    R2                  6 to 1
85      LockResource      task( 2)( 2)    R1                  1 to 1
88      UnlockResource    task( 2)( 2)    R1                  1 to 1
89      UnlockResource    task( 2)( 2)    R2                  1 to 6
89      Preemption        task( 2)( 2)    task( 1)( 4)
91      LockResource      task( 1)( 4)    R1                  3 to 2
93      LockResource      task( 1)( 4)    R2                  2 to 1
95      UnlockResource    task( 1)( 4)    R2                  1 to 2
96      UnlockResource    task( 1)( 4)    R1                  2 to 3
97      Completion        task( 1)( 4)    task( 2)( 2)        15                7                 0
99      Completion        task( 2)( 2)    task(63)            19                0                 8
```

The output results of Example 2:



1	LockResource	task(3)(0)	R2	9 to 7		
2	Preemption	task(3)(0)	task(1)(0)			
3	LockResource	task(1)(0)	R1	3 to 2		
4	UnlockResource	task(1)(0)	R1	2 to 3		
6	Completion	task(1)(0)	task(3)(0)	4	0	0
10	UnlockResource	task(3)(0)	R2	7 to 9		
10	Preemption	task(3)(0)	task(2)(0)			
13	Completion	task(2)(0)	task(3)(0)	3	0	0
14	Completion	task(3)(0)	task(63)	14	0	7
17	Preemption	task(63)	task(1)(1)			
18	LockResource	task(1)(1)	R1	3 to 2		
19	UnlockResource	task(1)(1)	R1	2 to 3		
21	Completion	task(1)(1)	task(63)	4	0	0
22	Preemption	task(63)	task(3)(1)			
23	LockResource	task(3)(1)	R2	9 to 7		
28	UnlockResource	task(3)(1)	R2	7 to 9		
29	Completion	task(3)(1)	task(63)	7	0	0
30	Preemption	task(63)	task(2)(1)			
32	Preemption	task(2)(1)	task(1)(2)			
33	LockResource	task(1)(2)	R1	3 to 2		
34	UnlockResource	task(1)(2)	R1	2 to 3		
36	Completion	task(1)(2)	task(2)(1)	4	0	0
37	Completion	task(2)(1)	task(63)	7	0	4
44	Preemption	task(63)	task(3)(2)			
45	LockResource	task(3)(2)	R2	9 to 7		
47	Preemption	task(3)(2)	task(1)(3)			
48	LockResource	task(1)(3)	R1	3 to 2		
49	UnlockResource	task(1)(3)	R1	2 to 3		
51	Completion	task(1)(3)	task(2)(2)	4	0	0
54	Completion	task(2)(2)	task(3)(2)	4	0	1
57	UnlockResource	task(3)(2)	R2	7 to 9		
58	Completion	task(3)(2)	task(63)	14	0	7
62	Preemption	task(63)	task(1)(4)			
63	LockResource	task(1)(4)	R1	3 to 2		
64	UnlockResource	task(1)(4)	R1	2 to 3		
66	Completion	task(1)(4)	task(3)(3)	4	0	0
67	LockResource	task(3)(3)	R2	9 to 7		
70	Preemption	task(3)(3)	task(2)(3)			
73	Completion	task(2)(3)	task(3)(3)	3	0	0
75	UnlockResource	task(3)(3)	R2	7 to 9		
76	Completion	task(3)(3)	task(63)	10	0	3
77	Preemption	task(63)	task(1)(5)			
78	LockResource	task(1)(5)	R1	3 to 2		
79	UnlockResource	task(1)(5)	R1	2 to 3		
81	Completion	task(1)(5)	task(63)	4	0	0
88	Preemption	task(63)	task(3)(4)			
89	LockResource	task(3)(4)	R2	9 to 7		
90	Preemption	task(3)(4)	task(2)(4)			
92	Preemption	task(2)(4)	task(1)(6)			
93	LockResource	task(1)(6)	R1	3 to 2		
94	UnlockResource	task(1)(6)	R1	2 to 3		
96	Completion	task(1)(6)	task(2)(4)	4	0	0
97	Completion	task(2)(4)	task(3)(4)	7	0	4

Credit:

[PART I] NPCS Implementation [40%]

- The correctness of schedule results of examples. Note the testing task set might not be the same as the given example task set. (20%)
- A report that describes your implementation (please attach the screenshot of the code and **MARK** the modified part). (20%)

[PART II] CPP Implementation [45%]

- The correctness of schedule results of examples. Note the testing task set might not be the same as the given example task set. (20%)
- A report that describes your implementation (please attach the screenshot of the code and **MARK** the modified part). (25%)

[PART III] Performance Analysis [15%]

- Compare the scheduling behaviors between NPCS and CPP with PART I and PART II results. (5%)
- Compare the different between blocking time and preemption time. (5%)
- Explain how NPCS and CPP avoid the deadlock problem. (5%)

※ **You must modify the source code.**

※ **Please set the ticks per second in order to run the project quickly. (in os_cfg.h)**

```
#define OS_TICKS_PER_SEC 100u
```

※ **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 parameter, INFO, as 10 to read more task information.**

```
#define INFO 10
```

※ **Please set the system end time as 100 seconds in this project.**

```
#define SYSTEM_END_TIME 100
```

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

※ **We only use two share resources in this project.**

※ **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: Dec. 13, 2023 (Wednesday) 12:00

File name format: RTOS_Myyyddxxx_PA3.zip

RTOS_Myyyddxxx_PA3.zip includes:

- The report (RTOS_Myyyddxxx_PA3.pdf).
- Folder with the executable µC/OS-II project (**RTOS_Myyyddxxx_PA3_NPCS**).
- Folder with the executable µC/OS-II project (**RTOS_Myyyddxxx_PA3_CPP**).

※ Plagiarizing is strictly prohibited.

Hints:

1. In the application region, we define the priorities of tasks and shared resources.

```
#define R1_PRIO 2
#define R2_PRIO 7
#define TASK1_PRIORITY 3
#define TASK2_PRIORITY 6
#define TASK3_PRIORITY 9
```

2. We also declare shared resources, as follows:

```
OS_EVENT* R1;
OS_EVENT* R2;
```

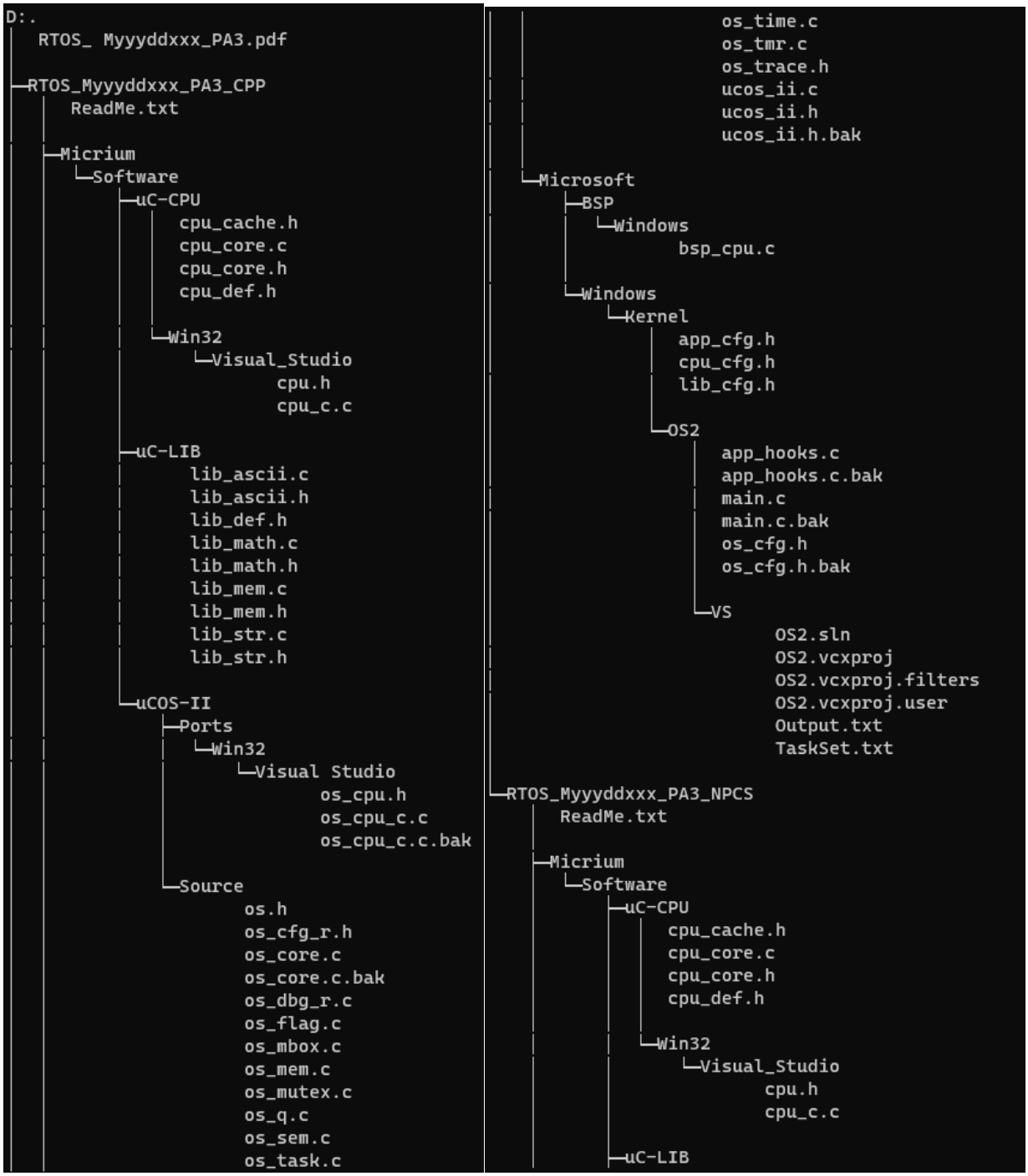
3. In the main function, we not only create tasks but also create shared resources.

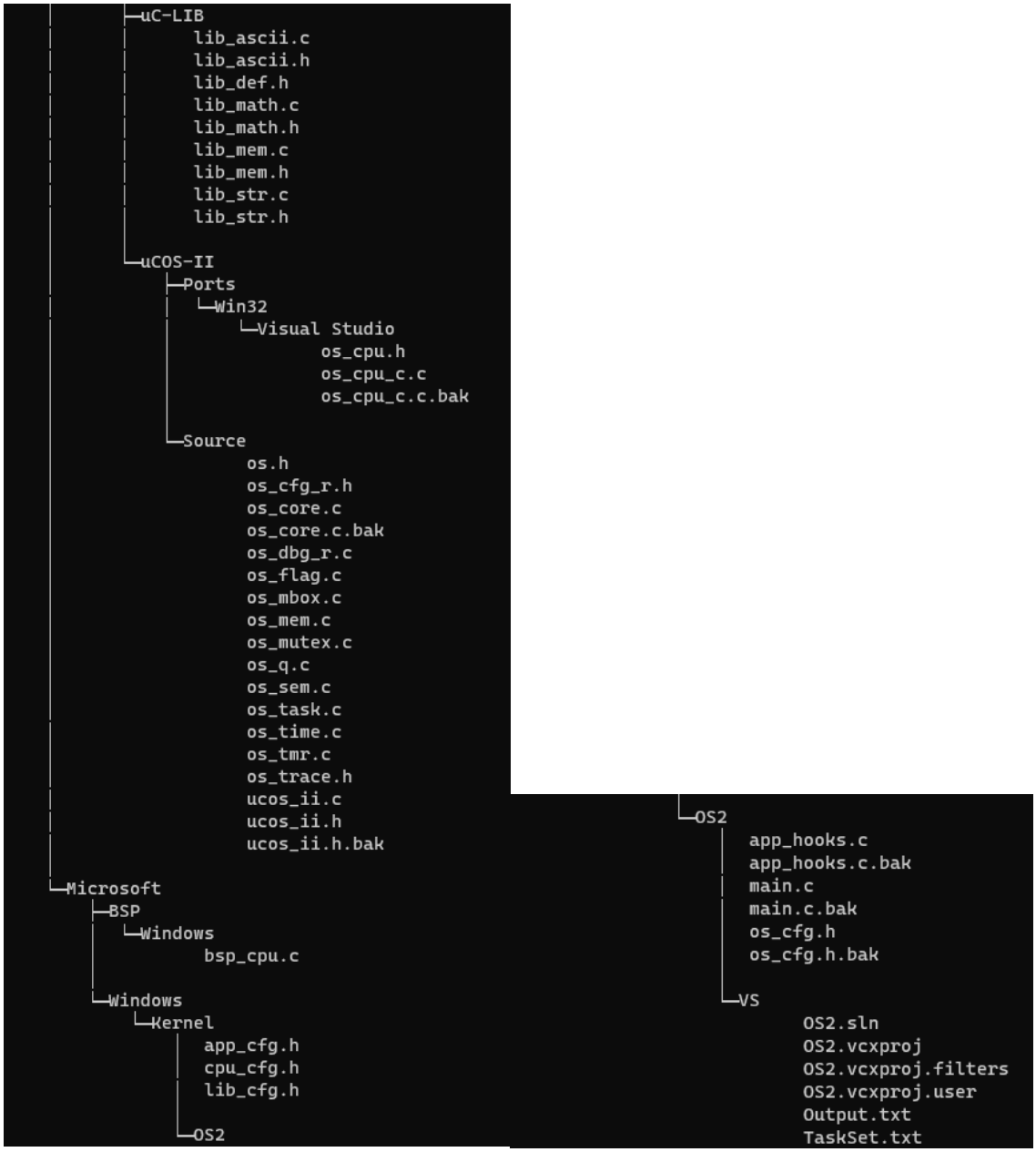
```
INT8U err;
R1 = OSMutexCreate(R1_PRIO, &err);
R2 = OSMutexCreate(R2_PRIO, &err);
```

4. To simulate the duration that a resource is held, we can program a function to implement it:

```
void mywait(int tick)
{
#if OS_CRITICAL_METHOD==3
    OS_CPU_SR cpu_sr = 0;
#endif
    int now, exit;
    OS_ENTER_CRITICAL();
    now = OSTimeGet();
    exit = now + tick;
    OS_EXIT_CRITICAL();
    while (1) {
        if (exit <= OSTimeGet())
            break;
    }
}
```

5. File Tree of PA3:




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
OS2
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