# Embedded OS Implementation, Fall 2020

## Project #3 (due January 6, 2021 (Wednesday) 13:00)

## [ PART I ] NPCS Implementation

***Objective***:

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

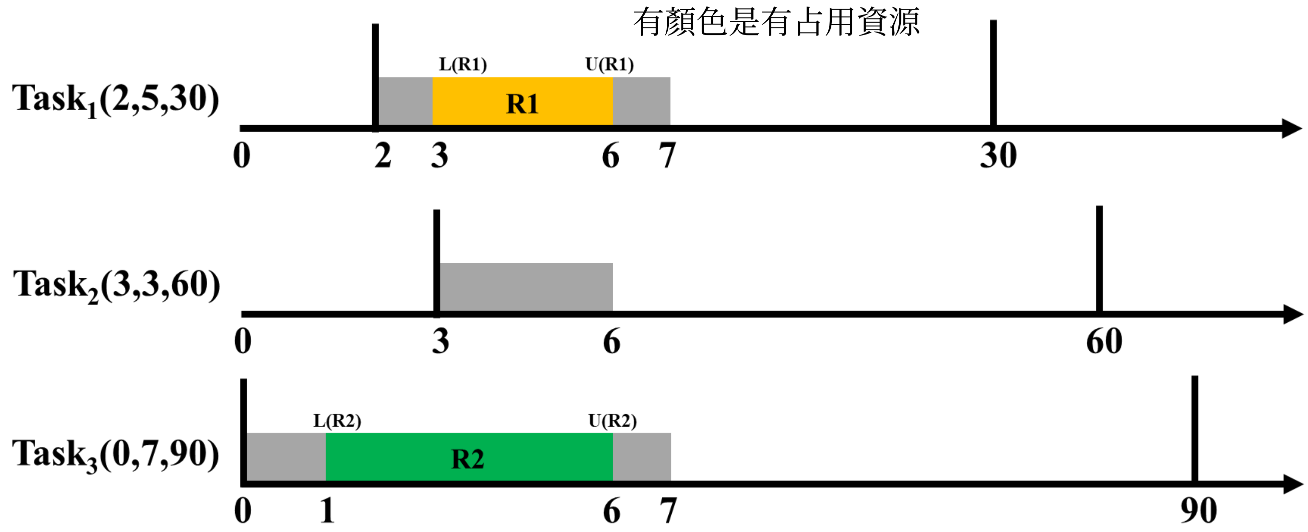
***Problem Definition***:

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

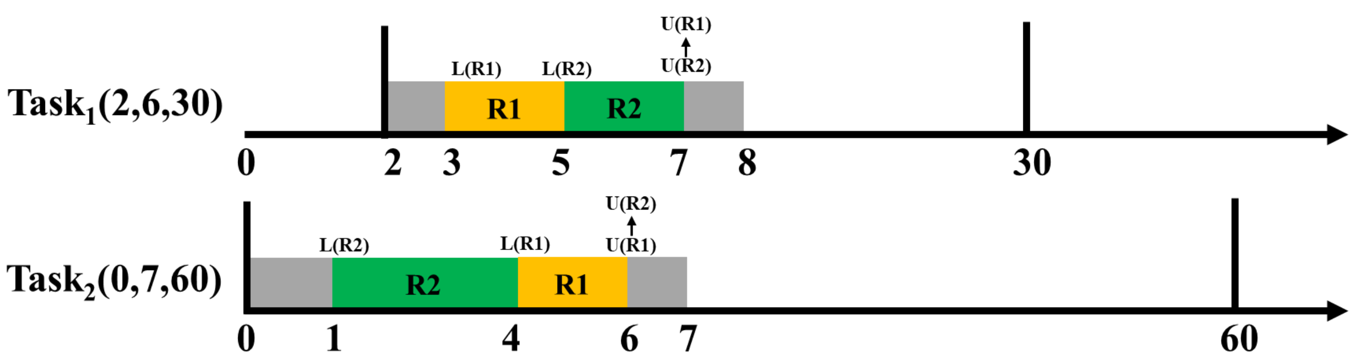
Consider the two periodic task sets and observe how the task suffers the schedule delay.

**Periodic Task Set = {task$_{ID}$ (arrival time, execution time, period)}**

**Task Set 1 = {task$_1$ (2,5,30), task$_2$ (3,3,60), task$_2$ (0,7,90)}**




**Task Set 2 = {task$_1$ (2,6,30), task$_2$ (0,7,60)}**



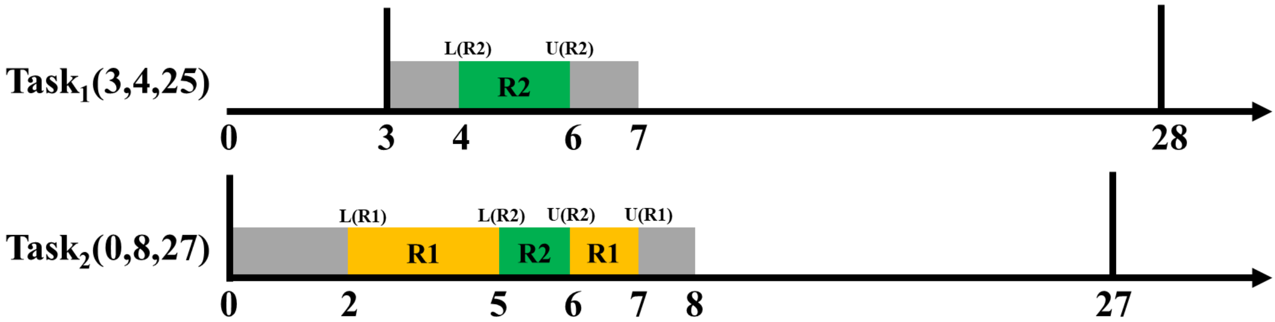
***Evaluation:***

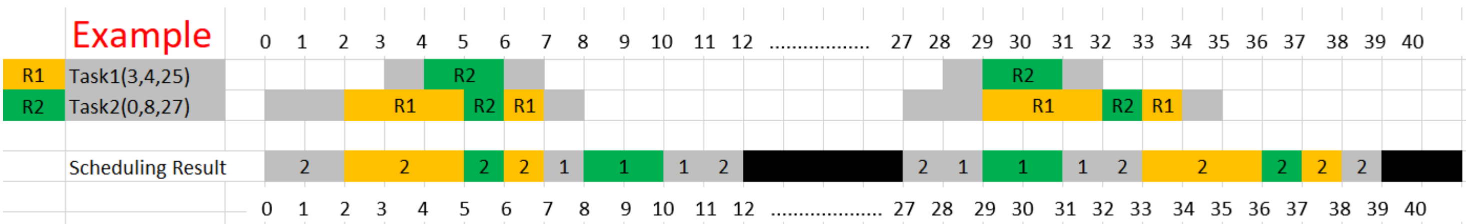
The output format:

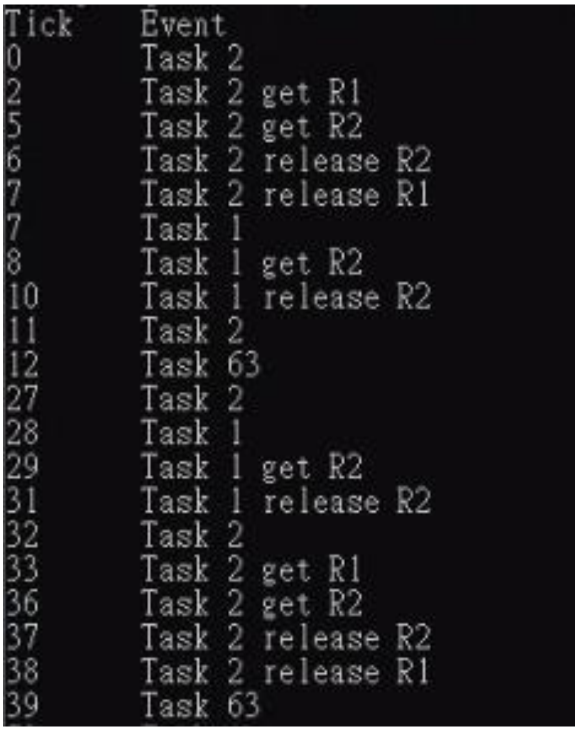
| Tick | Event |
|---|---|
| # | Task ID |
| # | Task ID get R |
| # | Task ID release R |

## The NPCS Example of Output Result:

Consider two tasks, $task_1(3,4,25)$, $task_2(0,8,27)$, and two resources R1, R2.



| Tick | Event |
|---|---|
| 0 | Task 2 |
| 2 | Task 2 get R1 |
| 5 | Task 2 get R2 |
| 6 | Task 2 release R2 |
| 7 | Task 2 release R1 |
| 7 | Task 1 |
| 8 | Task 1 get R2 |
| 10 | Task 1 release R2 |
| 11 | Task 2 |
| 12 | Task 63 |
| 27 | Task 2 |
| 28 | Task 1 |
| 29 | Task 1 get R2 |
| 31 | Task 1 release R2 |
| 32 | Task 2 |
| 33 | Task 2 get R1 |
| 36 | Task 2 get R2 |
| 37 | Task 2 release R2 |
| 38 | Task 2 release R1 |
| 39 | Task 63 |

# [ PART II ] CPP Implementation

***Objective***:

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

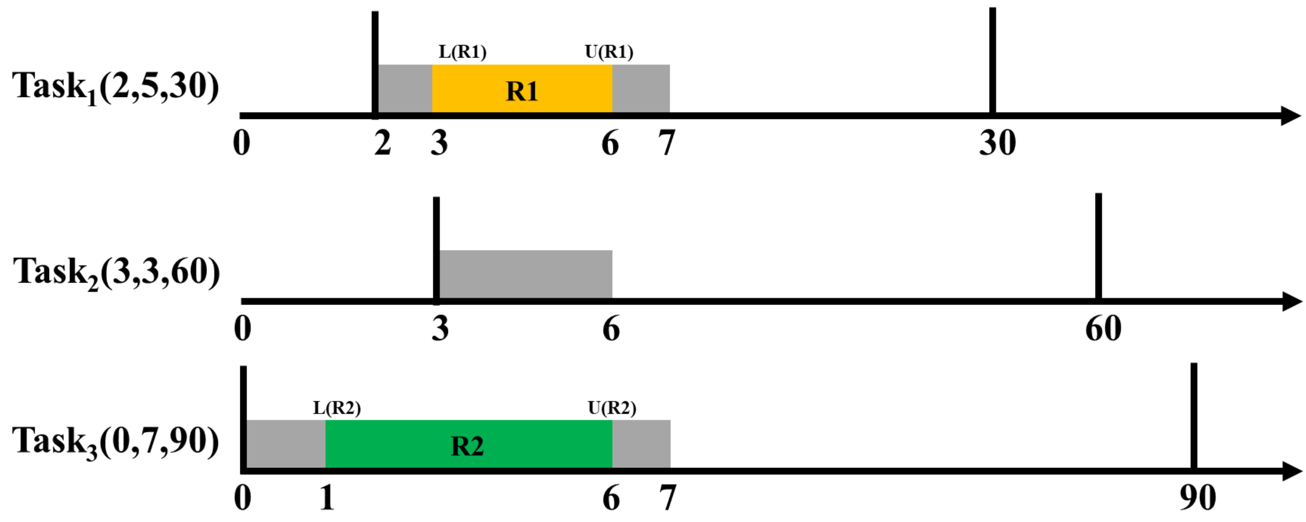
***Problem Definition***:

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

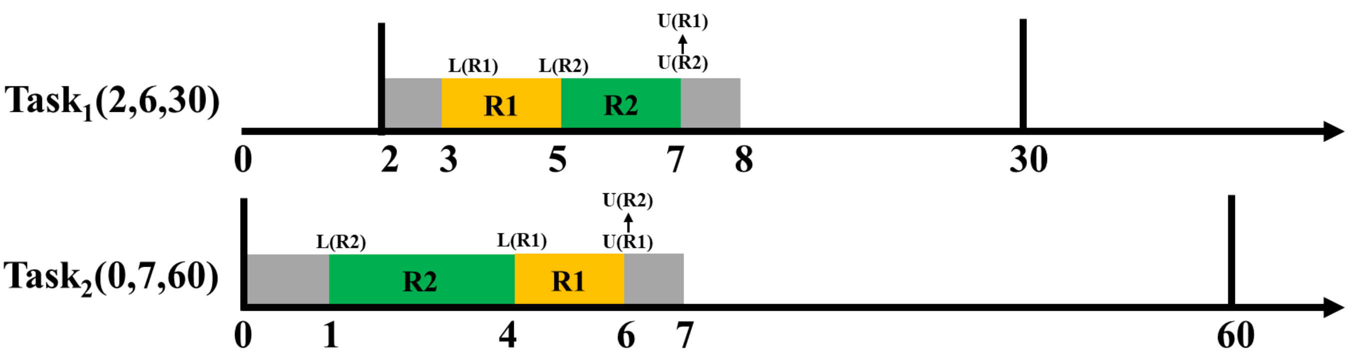
Consider the two periodic task sets and observe how the task suffers the schedule delay.

**Periodic Task Set = {task$_{ID}$ (arrival time, execution time, period)}**

**Task Set 1 = {task$_1$ (2,5,30), task$_2$ (3,3,60), task$_2$ (0,7,90)}**




**Task Set 2 = {task$_1$ (2,6,30), task$_2$ (0,7,60)}**



## ***Evaluation:***

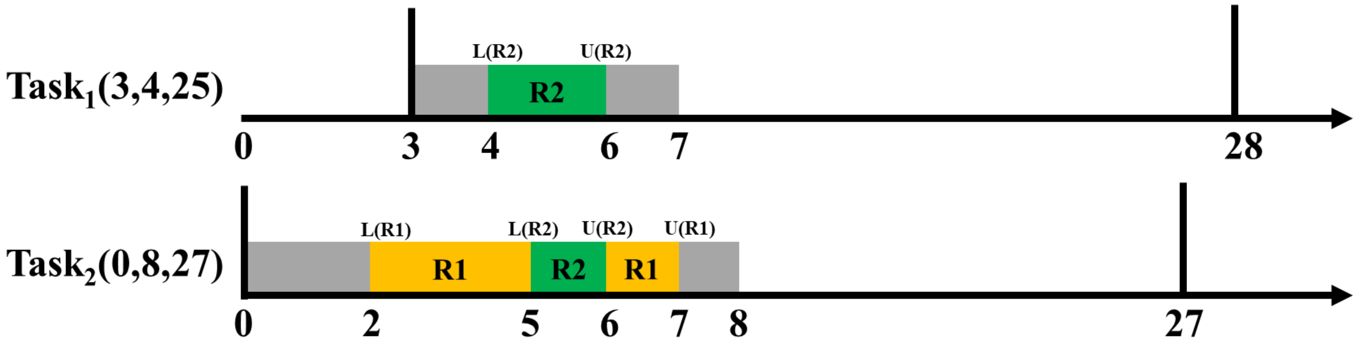
The output format:

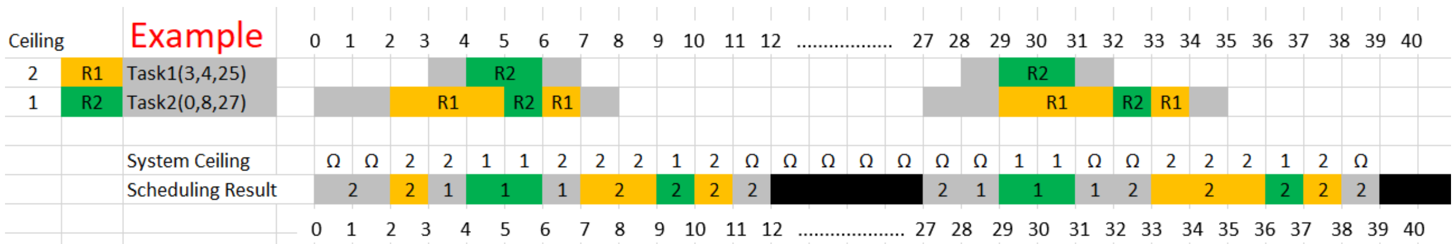
| Tick | Event | SystemCeiling |
|---|---|---|
| # | Task ID | |
| # | Task ID get R | Ceiling_prio→Ceiling_prio |
| # | Task ID release R | Ceiling_prio→Ceiling_prio |

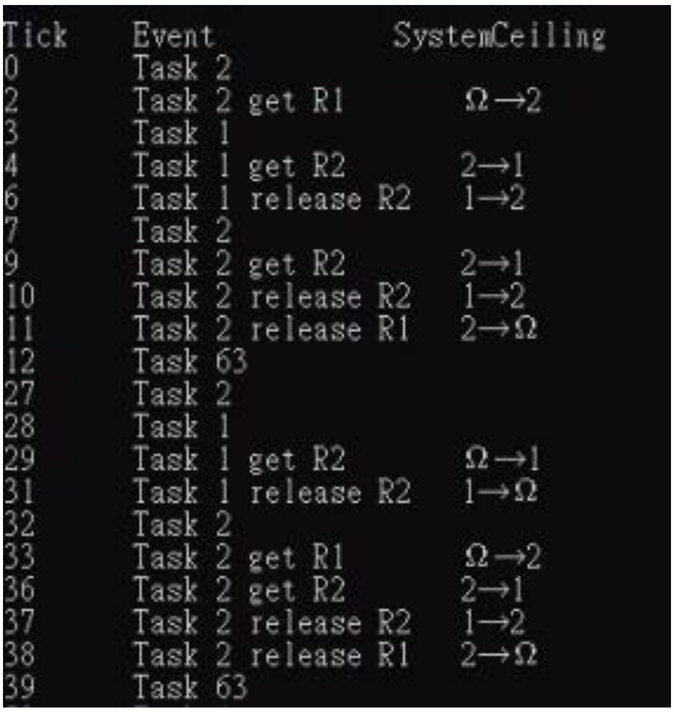
※ If the system ceiling is the lowest, use the symbol "Ω".

## The CPP Example of Output Result:

Consider two tasks, $task_1(3,4,25)$, $task_2(0,8,27)$, and two resources R1, R2.



```
Tick    Event                SystemCeiling
0       Task 2
2       Task 2 get R1          Ω→2
3       Task 1
4       Task 1 get R2          2→1
6       Task 1 release R2      1→2
7       Task 2
9       Task 2 get R2          2→1
10      Task 2 release R2      1→2
11      Task 2 release R1      2→Ω
12      Task 63
27      Task 2
28      Task 1
29      Task 1 get R2          Ω→1
31      Task 1 release R2      1→Ω
32      Task 2
33      Task 2 get R1          Ω→2
36      Task 2 get R2          2→1
37      Task 2 release R2      1→2
38      Task 2 release R1      2→Ω
39      Task 63
```

## ***Crediting :***

### [ PART I ] NPCS Implementation [50%]

- The screenshot result (with the given format) of the two task sets. (Time tick 0-100) (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 ] CPP Implementation [40%]

- The screenshot result (with the given format) of the two task sets. (Time tick 0-100) (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%)

### [ PART III ] Performance Analysis [10%]

比較分析 優缺點 怎麼避免deadlock

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

※ You must modify the source code.

## ***Project submit:***

Submit to Moodle

Submit deadline: January 6, 2021 (Wednesday) 13:00

File name format: RTOS_your student ID_PA3.zip

RTOS_your student ID_PA3.zip includes :

- The report (RTOS_your student ID_PA3.pdf).
- The files you modify( main.c, os_core.c , etc. )

## ***Hints:***

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

```
#define R1_PRIO 1
#define R2_PRIO 2
#define TASK1_PRIORITY      11
#define TASK2_PRIORITY      12
#define TASK3_PRIORITY      13
#define TASK4_PRIORITY      14
```

2. We also declare shared resource, as follows:

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

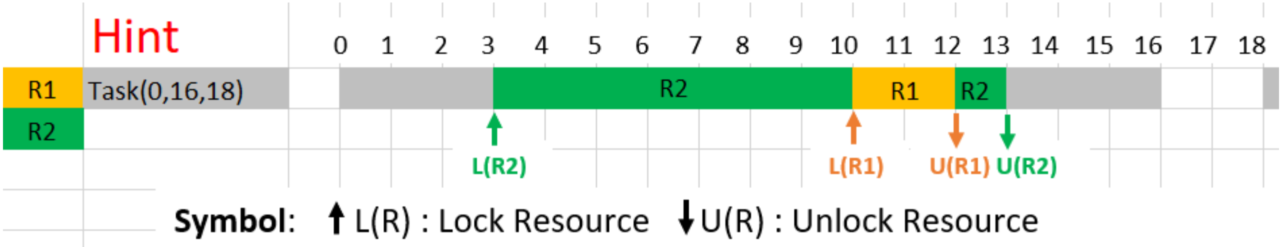
3. In the main function, we not only create tasks but also create shared resources.

```
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. To modeling a task's behavior, we can program the task function as following:




```c
void task1(void* pdata)
{
    INT8U err;
    while (1)
    {
        printf("%d\tTask 1\n", OSTimeGet());
        mywait(3);
        printf("%d\tTask 1 get R2\n", OSTimeGet());
        OSMutexPend(R2, 0, &err);
        mywait(7);

        printf("%d\tTask 1 get R1\n", OSTimeGet());
        OSMutexPend(R1, 0, &err);
        mywait(2);

        printf("%d\tTask 1 release R1\n", OSTimeGet());
        OSMutexPost(R1);
        mywait(1);

        printf("%d\tTask 1 release R2\n", OSTimeGet());
        OSMutexPost(R2);
        mywait(3);
        OSTimeDly( T1_Deadline - OSTimeGet());
    }
}
```