RTOS PA1

studentID: M11202158

name: 陳昱嘉 Chen, Yu-Chia

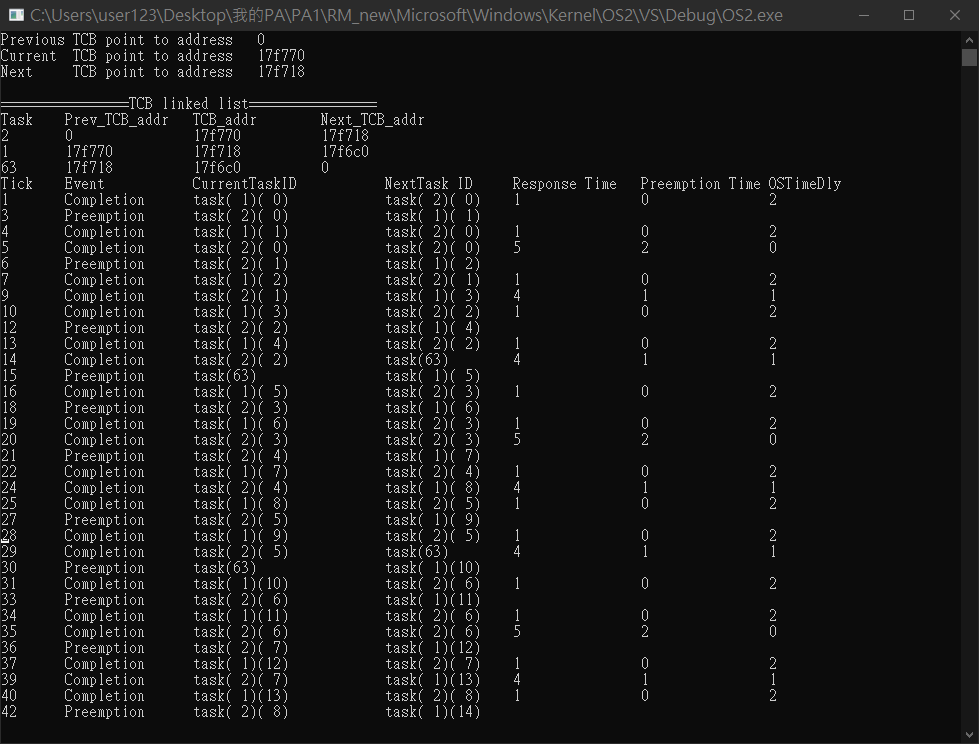
RM Scheduler Implementation

The correctness of schedule results of examples

Taskset:

1 0 1 3

2 0 3 5

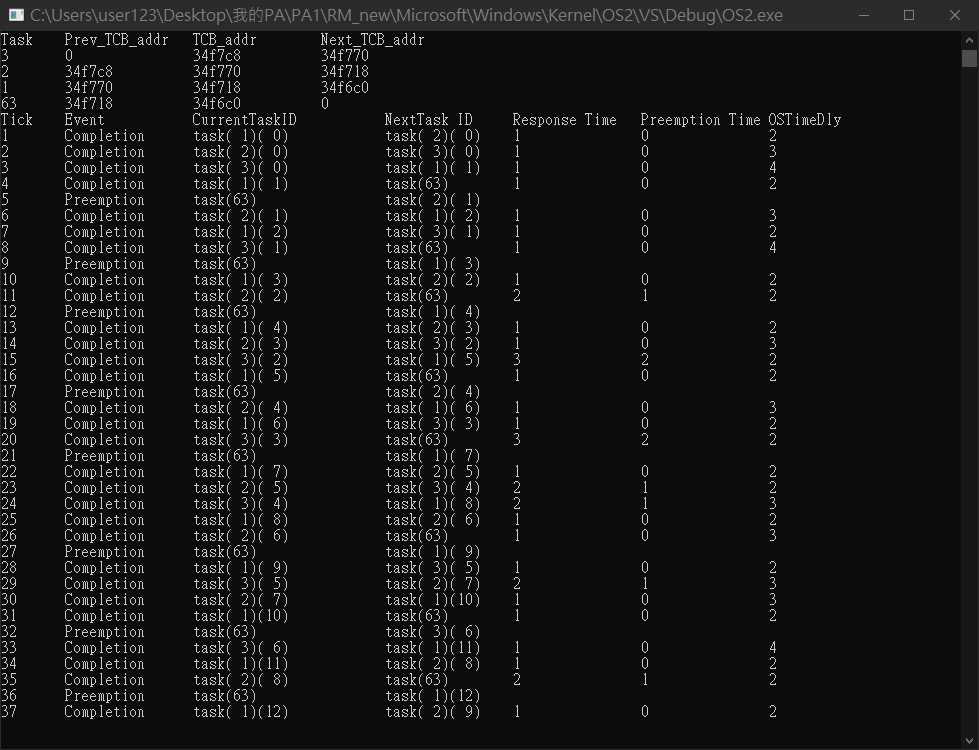


Taskset:

1 0 1 3

2 1 1 4

3 2 1 5

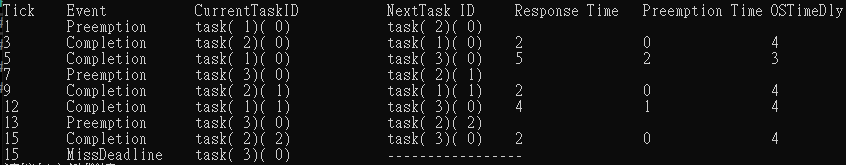


Taskset:

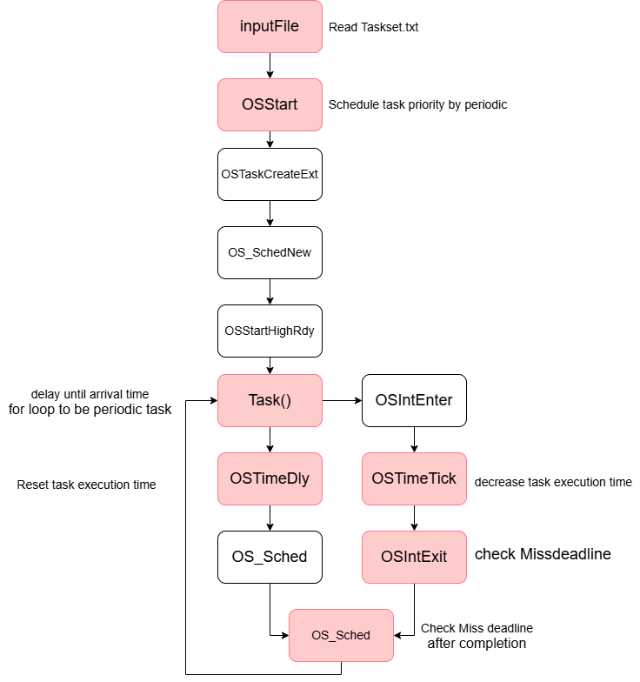
1 0 3 8

2 1 2 6

3 0 4 15



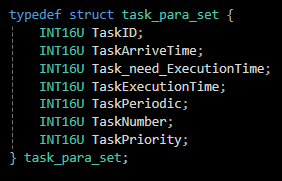
A report that describes your implementation



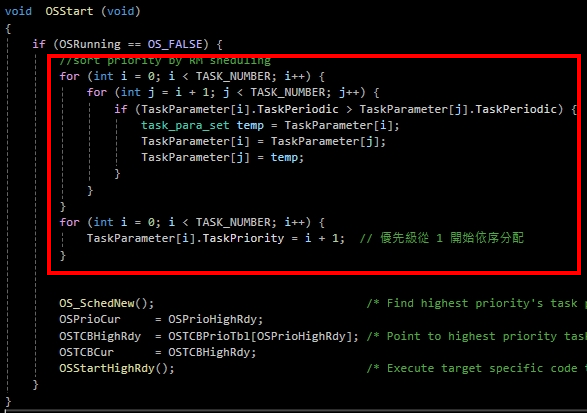
Inputfile



Add the ability to read the Taskset's ID, arrival time, execution time, periodicity, and Task\_need\_ExecutionTime, which represents the number of ticks the task still needs to complete in the current period. A task with a TaskNumber of 0 indicates the number of times the task has been completed.

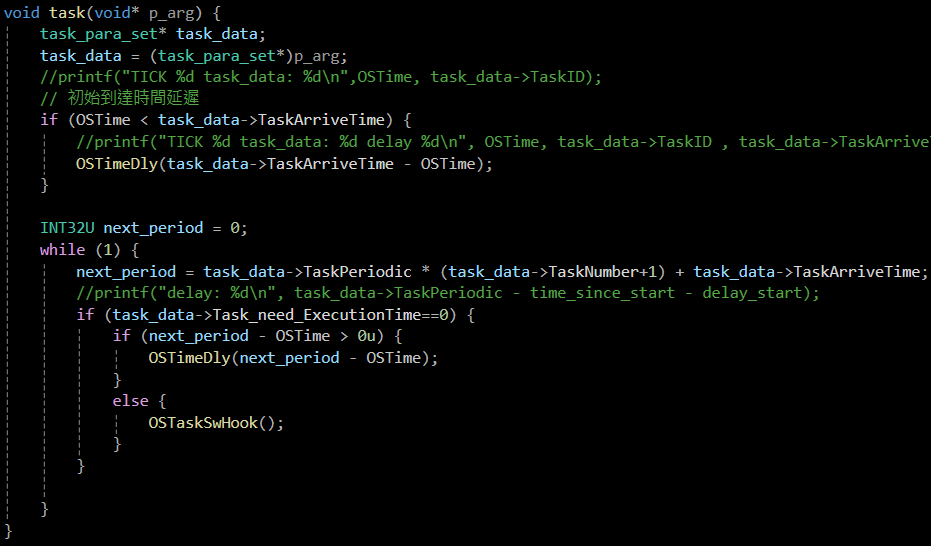


Each TaskParameter stores the Taskset's ID, arrival time, execution time, periodicity, Task\_need\_ExecutionTime (remaining ticks for completion in the current period), and TaskNumber (times the task has been completed).OSSart

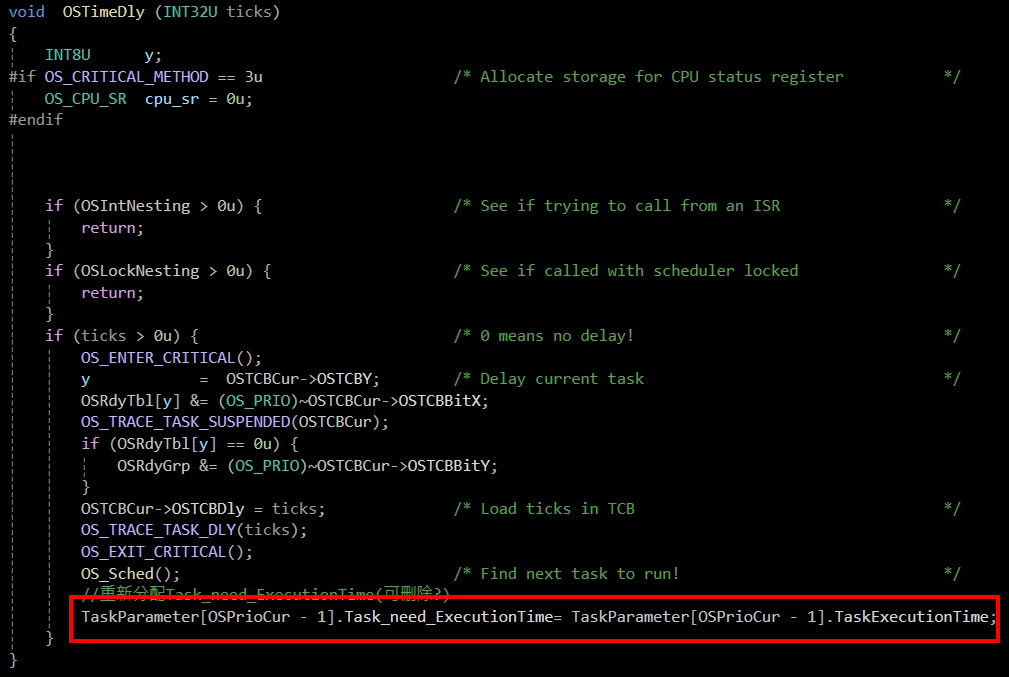


In OSStart, tasks are prioritized based on their periodicity, with shorter periods assigned higher priority.

Task

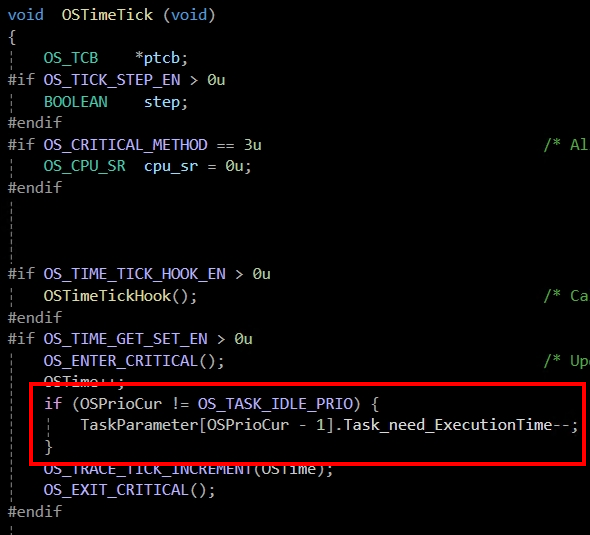


Each task starts after an arrival time delay. In the while loop, next\_period marks the next period's start. After each TimeTick, Task\_need\_ExecutionTime is decremented. If the task completes early, it delays until next\_period - OSTime. If it finishes exactly at the next period, OSTaskSwHOOK increments TaskNumber to record task completion, preventing unnecessary context switches.OSTimeDly



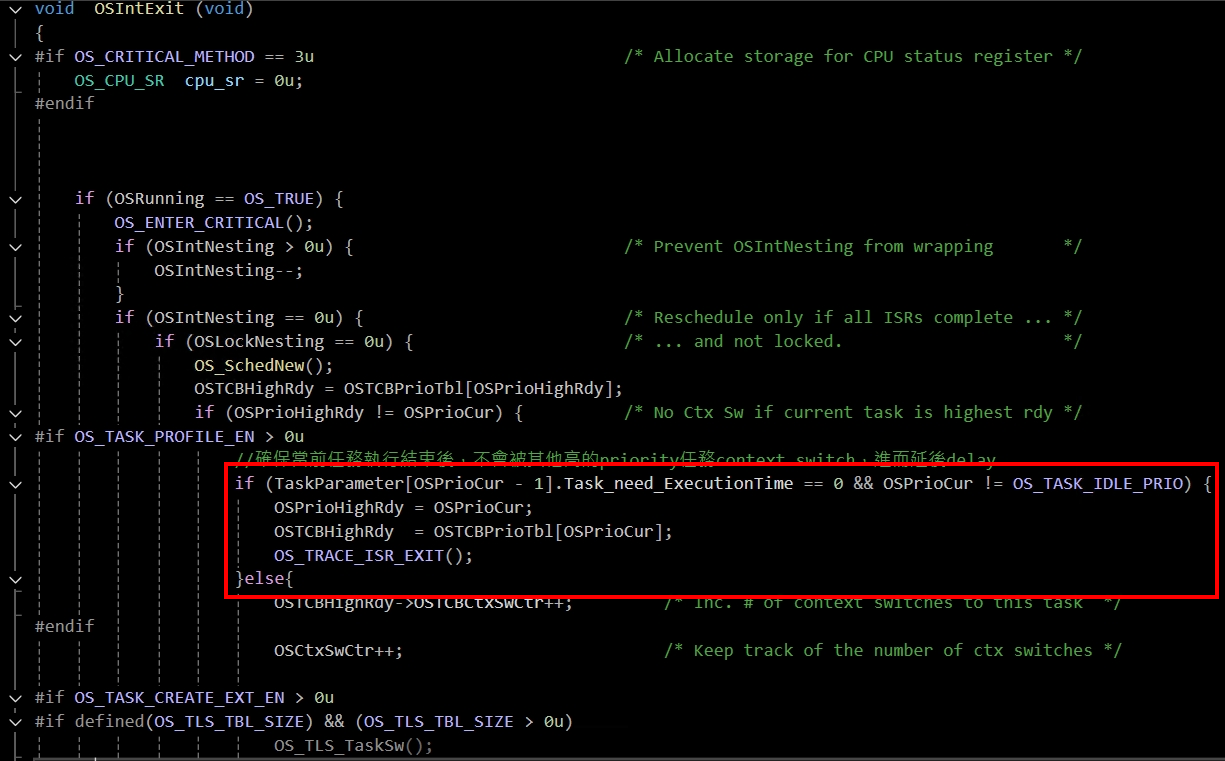
When a task enters a delay, its Task\_need\_ExecutionTime is reset, ensuring it needs to complete the full TaskExecutionTime in the next period.

OSTimeTick



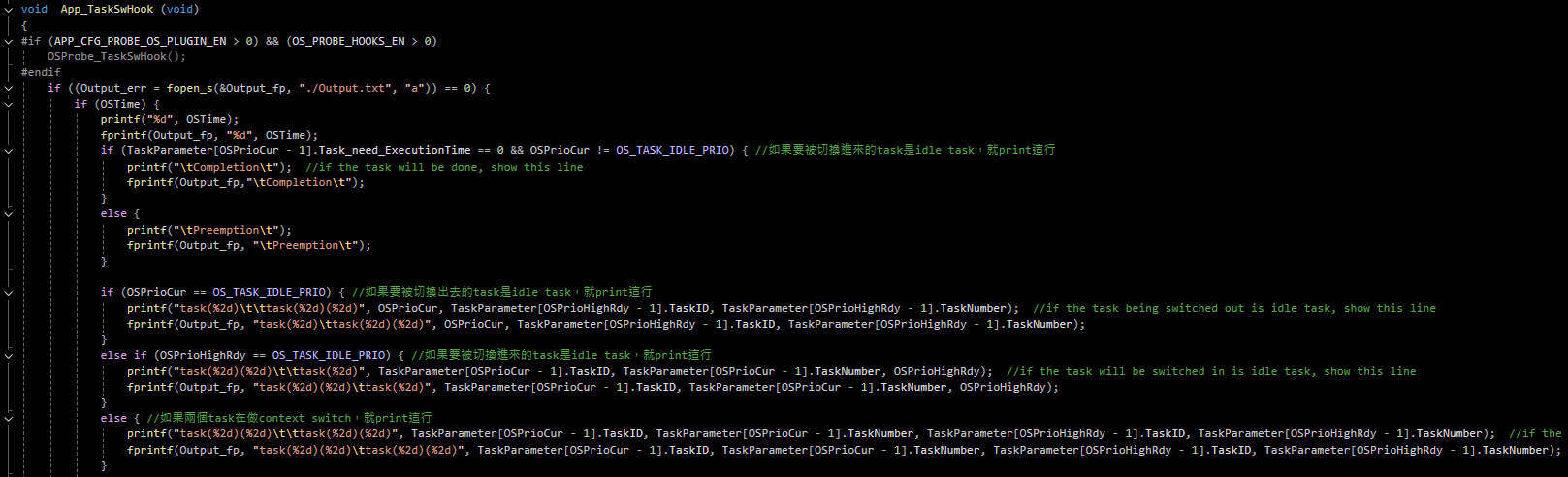
If the current task is not idle, its Task\_need\_ExecutionTime is decremented by 1, and then the task level checks whether the task is completed.

OSIntExit

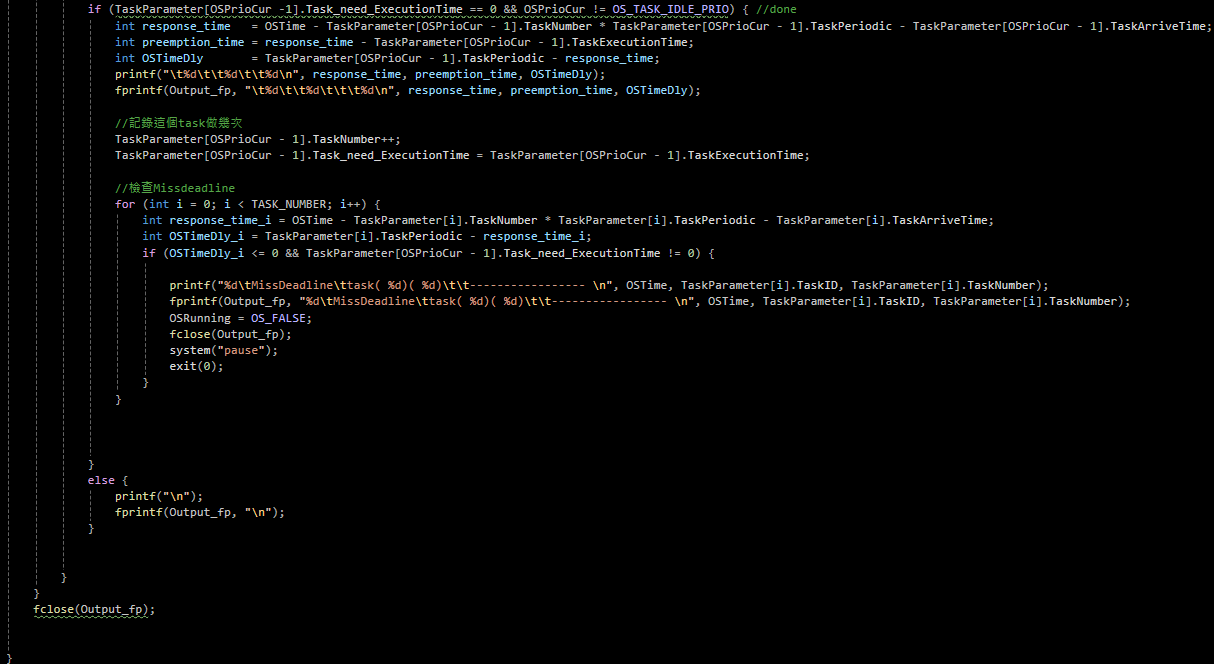


OSIntExit checks if the next task is different from the current task. If so, it performs a context switch. However, if the task just completed, it won't re-enter to finish execution, causing the current task to remain incomplete.

OS\_Sched



OS\_Sched is executed during delays or interrupt context switches. It contains OS\_TASK\_SW, which runs the OSTaskSwHook function. During a context switch, it calls App\_TaskSwHook to check the current task's completion status using TaskParameter[OSPrioCur - 1].Task\_need\_ExecutionTime.



When the condition TaskParameter[OSPrioCur - 1].Task\_need\_ExecutionTime == 0 occurs, it indicates task completion. At this point, the system displays the response time, preemption time, and OSTimeDly.

response\_time = response\_time = OSTime - TaskParameter[OSPrioCur - 1].TaskNumber \* TaskParameter[OSPrioCur - 1].TaskPeriodic - TaskParameter[OSPrioCur - 1].TaskArriveTime;

preemption\_time = response\_time - TaskParameter[OSPrioCur - 1].TaskExecutionTime;

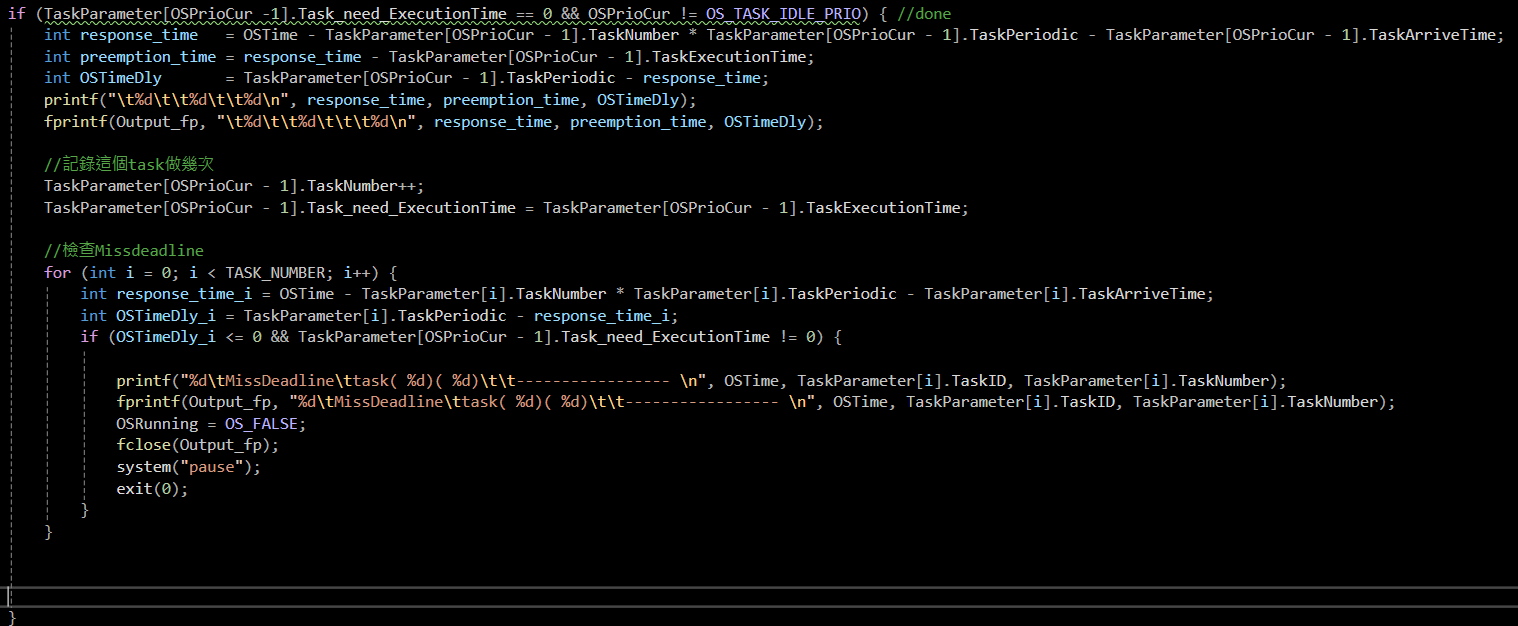
OSTimeDly = TaskParameter[OSPrioCur - 1].TaskPeriodic - response\_time;

After the task completes, if the next period starts immediately, TaskNumber is incremented by 1, and Task\_need\_ExecutionTime is reset.

handle the deadline missing situation under RM

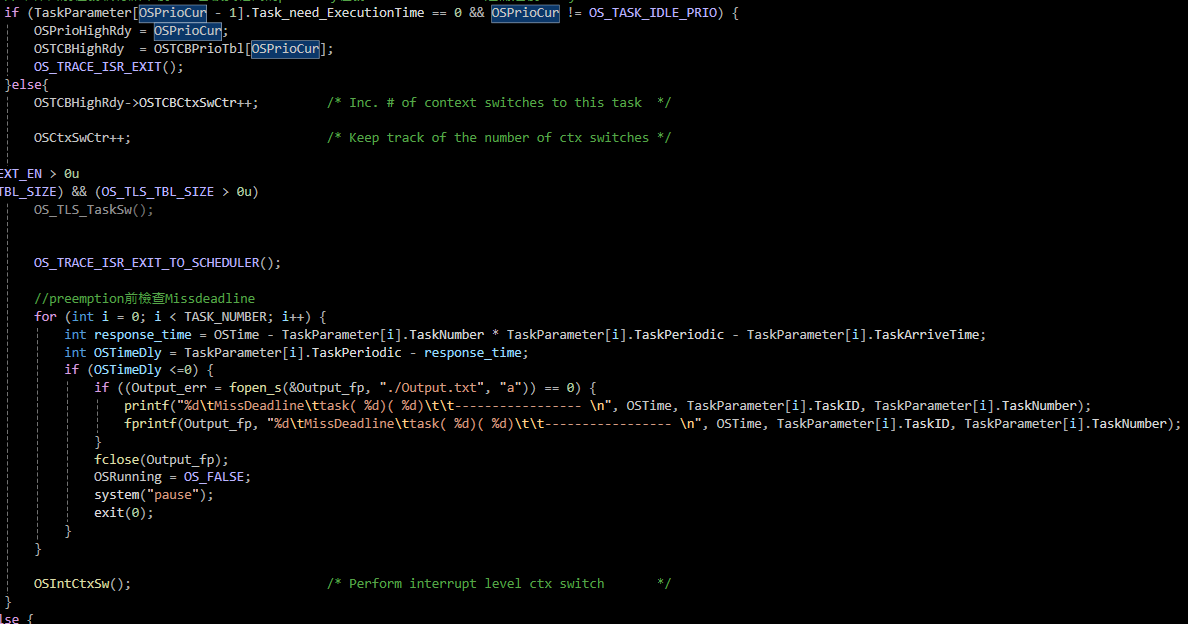
There are three scenarios for checking: after task completion, when a task is preempted, and when a task is currently running.

Completion > Missdeadline > Preemption

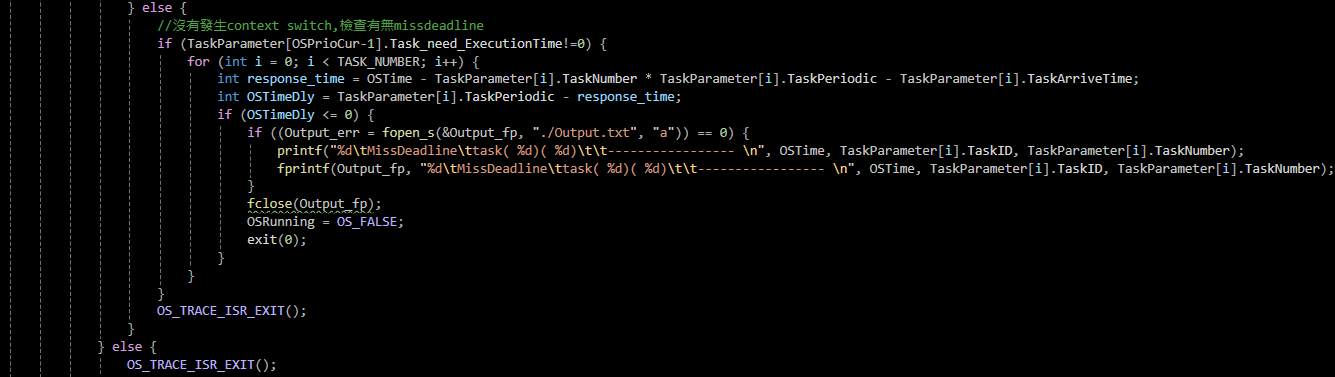


In App\_TaskSwHook, after a task completes, it immediately checks if any task has missed its deadline in the current tick.

The other two scenarios are checked within OSIntExit.



This occurs when a task is preempted, and before the context switch happens, it checks whether any task has missed its deadline.



Finally, in the new tick, if the current task's TaskParameter[OSPrioCur-1].Task\_need\_ExecutionTime != 0, it is first checked. If Task\_need\_ExecutionTime == 0, it means the task has completed, so it proceeds to the task level to handle the completion before checking for missed deadlines.