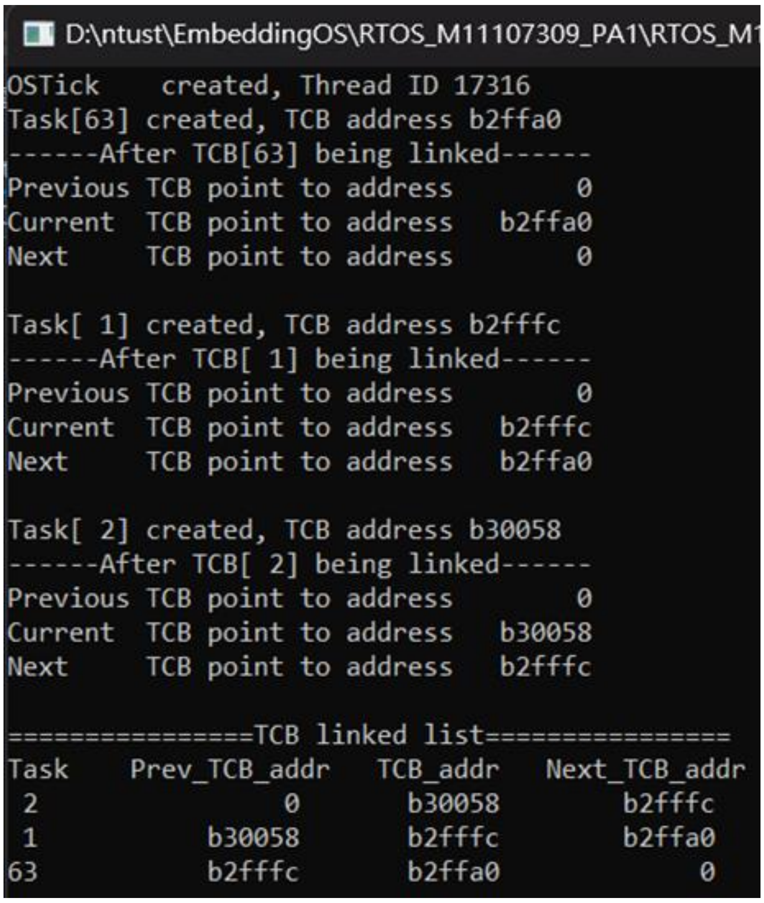
Embedded OS PA1 M11107309 何柏昇

PART I Task Control Block Linked List

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

The screenshot results. (10%)



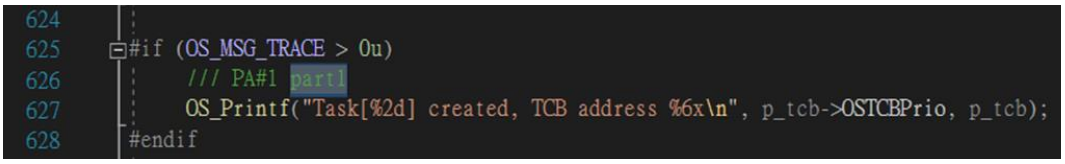
D:\ntust\EmbeddingOS\RTOS_M11107309_PA1\RTOS_M1

```
OSTick    created, Thread ID 17316
Task[63] created, TCB address b2ffa0
------After TCB[63] being linked------
Previous TCB point to address       0
Current  TCB point to address  b2ffa0
Next     TCB point to address       0

Task[ 1] created, TCB address b2fffc
------After TCB[ 1] being linked------
Previous TCB point to address       0
Current  TCB point to address  b2fffc
Next     TCB point to address  b2ffa0

Task[ 2] created, TCB address b30058
------After TCB[ 2] being linked------
Previous TCB point to address       0
Current  TCB point to address  b30058
Next     TCB point to address  b2fffc

===============TCB linked list===============
Task    Prev_TCB_addr    TCB_addr   Next_TCB_addr
 2                 0      b30058          b2fffc
 1            b30058      b2fffc          b2ffa0
63            b2fffc      b2ffa0               0
```

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

(如未標示，整段即為全新 code)

os_cpu_c.c

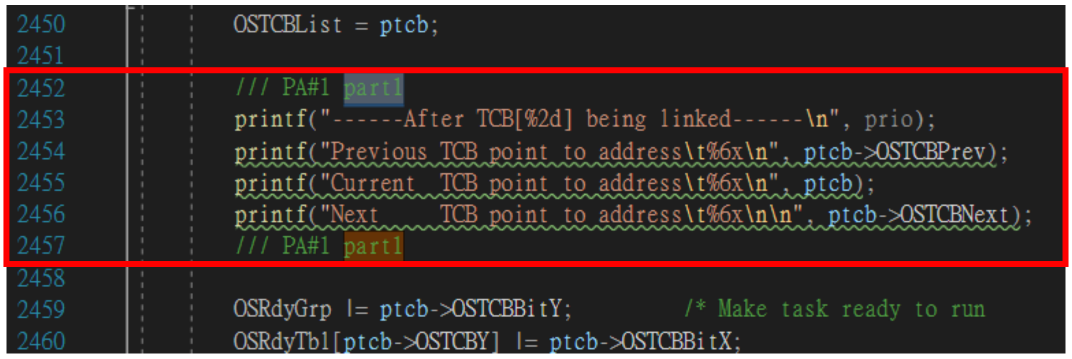


```
#if (OS_MSG_TRACE > 0u)
    /// PA#1 part1
    OS_Printf("Task[%2d] created, TCB address %6x\n", p_tcb->OSTCBPrio, p_tcb);
#endif
```

建立 task 時會進行 trace 並輸出訊息，將其更改為題目要求的 TCB address。

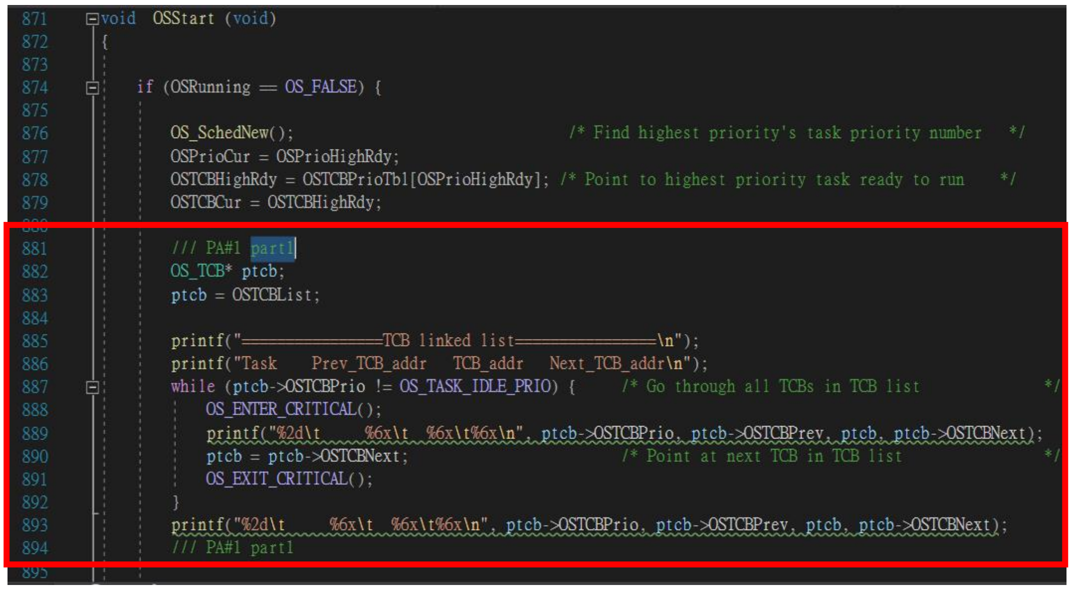
os_core.c

OSTCBInit()



```
        OSTCBList = ptcb;

        /// PA#1 part1
        printf("------After TCB[%2d] being linked------\n", prio);
        printf("Previous TCB point to address\t%6x\n", ptcb->OSTCBPrev);
        printf("Current  TCB point to address\t%6x\n", ptcb);
        printf("Next     TCB point to address\t%6x\n\n", ptcb->OSTCBNext);
        /// PA#1 part1

        OSRdyGrp |= ptcb->OSTCBBitY;         /* Make task ready to run
        OSRdyTbl[ptcb->OSTCBY] |= ptcb->OSTCBBitX;
```

OSStart()



```
void  OSStart (void)
{

    if (OSRunning == OS_FALSE) {

        OS_SchedNew();                               /* Find highest priority's task priority number   */
        OSPrioCur = OSPrioHighRdy;
        OSTCBHighRdy = OSTCBPrioTbl[OSPrioHighRdy]; /* Point to highest priority task ready to run    */
        OSTCBCur = OSTCBHighRdy;

        /// PA#1 part1
        OS_TCB* ptcb;
        ptcb = OSTCBList;

        printf("===============TCB linked list===============\n");
        printf("Task    Prev_TCB_addr   TCB_addr   Next_TCB_addr\n");
        while (ptcb->OSTCBPrio != OS_TASK_IDLE_PRIO) {     /* Go through all TCBs in TCB list            */
            OS_ENTER_CRITICAL();
            printf("%2d\t      %6x\t  %6x\t%6x\n", ptcb->OSTCBPrio, ptcb->OSTCBPrev, ptcb, ptcb->OSTCBNext);
            ptcb = ptcb->OSTCBNext;                        /* Point at next TCB in TCB list              */
            OS_EXIT_CRITICAL();
        }
        printf("%2d\t      %6x\t  %6x\t%6x\n", ptcb->OSTCBPrio, ptcb->OSTCBPrev, ptcb, ptcb->OSTCBNext);
        /// PA#1 part1
```

任務建立好後在 OSStart，呼叫 ptcb，輸出 TCB list，利用 ptcb->OSTCBNext 與 while，歷遍 TCB list。

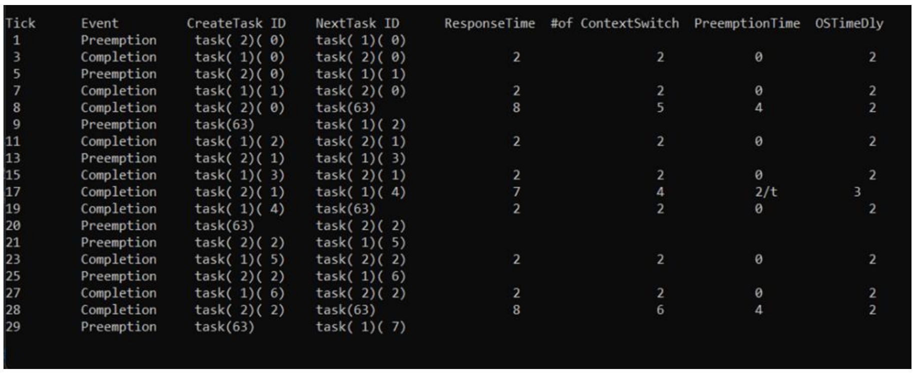
PART II RM Scheduler Implementation

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

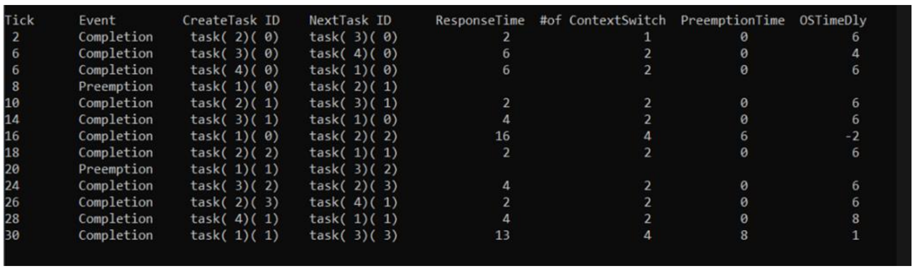
Periodic Task Set = {τ_{ID} (ID, arrival time, execution time, period)}
Example Task Set 1 = {τ1 (1, 1, 2, 4), τ2 (2, 0, 4, 10)}
Example Task Set 2 = {τ1 (1, 3, 4, 14), τ2 (2, 0, 2, 8), τ3 (3, 0, 4, 10), τ4(4, 24, 2, 12)}
Example Task Set 3 = {τ1 (1, 2, 2, 10), τ2 (2, 1, 1, 5), τ3 (3, 0, 8, 15)}

The correctness of schedule results of examples. Note the testing task set might not be the same as the given example task set. (40%)

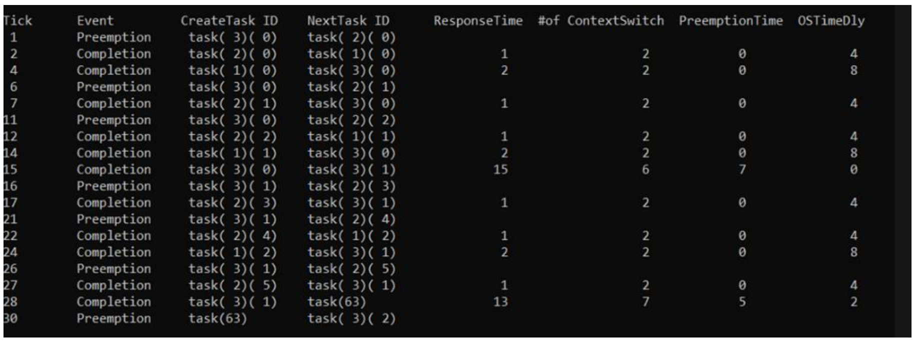
Task Set 1


```
Tick     Event        CreateTask ID    NextTask ID      ResponseTime  #of ContextSwitch  PreemptionTime  OSTimeDly
 1       Preemption    task( 2)( 0)    task( 1)( 0)
 3       Completion    task( 1)( 0)    task( 2)( 0)                2                  2               0          2
 5       Preemption    task( 2)( 0)    task( 1)( 1)
 7       Completion    task( 1)( 1)    task( 2)( 0)                2                  2               0          2
 8       Completion    task( 2)( 0)    task(63)                    8                  5               4          2
 9       Preemption    task(63)        task( 1)( 2)
11       Completion    task( 1)( 2)    task( 2)( 1)                2                  2               0          2
13       Preemption    task( 2)( 1)    task( 1)( 3)
15       Completion    task( 1)( 3)    task( 2)( 1)                2                  2               0          2
17       Completion    task( 2)( 1)    task( 1)( 4)                7                  4               2/t       3
19       Completion    task( 1)( 4)    task(63)                    2                  2               0          2
20       Preemption    task(63)        task( 2)( 2)
21       Preemption    task( 2)( 2)    task( 1)( 5)
23       Completion    task( 1)( 5)    task( 2)( 2)                2                  2               0          2
25       Preemption    task( 2)( 2)    task( 1)( 6)
27       Completion    task( 1)( 6)    task( 2)( 2)                2                  2               0          2
28       Completion    task( 2)( 2)    task(63)                    8                  6               4          2
29       Preemption    task(63)        task( 1)( 7)
```

Task Set 2


```
Tick     Event        CreateTask ID    NextTask ID      ResponseTime  #of ContextSwitch  PreemptionTime  OSTimeDly
 2       Completion    task( 2)( 0)    task( 3)( 0)                2                  1               0          6
 6       Completion    task( 3)( 0)    task( 4)( 0)                6                  2               0          4
 6       Completion    task( 4)( 0)    task( 1)( 0)                6                  2               0          6
 8       Preemption    task( 1)( 0)    task( 2)( 1)
10       Completion    task( 2)( 1)    task( 3)( 1)                2                  2               0          6
14       Completion    task( 3)( 1)    task( 1)( 0)                4                  2               0          6
16       Completion    task( 1)( 0)    task( 2)( 2)               16                  4               6         -2
18       Completion    task( 2)( 2)    task( 1)( 1)                2                  2               0          6
20       Preemption    task( 1)( 1)    task( 3)( 2)
24       Completion    task( 3)( 2)    task( 2)( 3)                4                  2               0          6
26       Completion    task( 2)( 3)    task( 4)( 1)                2                  2               0          6
28       Completion    task( 4)( 1)    task( 1)( 1)                4                  2               0          8
30       Completion    task( 1)( 1)    task( 3)( 3)               13                  4               8          1
```

Task Set 3

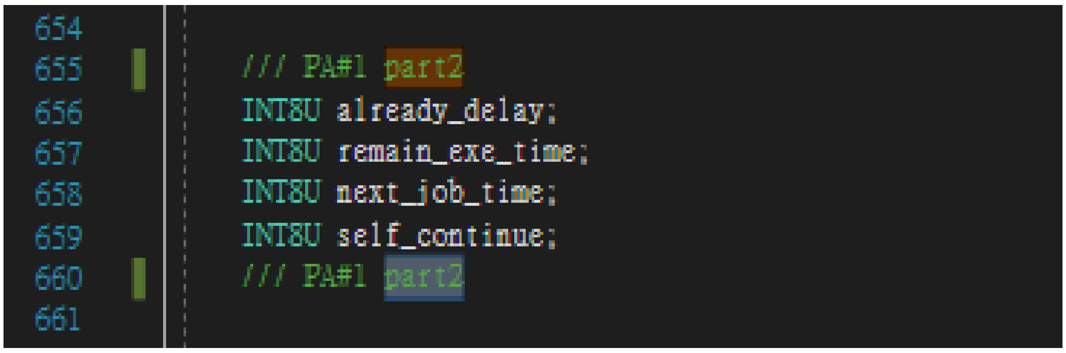

```
Tick     Event        CreateTask ID    NextTask ID      ResponseTime  #of ContextSwitch  PreemptionTime  OSTimeDly
 1       Preemption    task( 3)( 0)    task( 2)( 0)
 2       Completion    task( 2)( 0)    task( 1)( 0)                1                  2               0          4
 4       Completion    task( 1)( 0)    task( 3)( 0)                2                  2               0          8
 6       Preemption    task( 3)( 0)    task( 2)( 1)
 7       Completion    task( 2)( 1)    task( 3)( 0)                1                  2               0          4
11       Preemption    task( 3)( 0)    task( 2)( 2)
12       Completion    task( 2)( 2)    task( 1)( 1)                1                  2               0          4
14       Completion    task( 1)( 1)    task( 3)( 0)                2                  2               0          8
15       Completion    task( 3)( 0)    task( 3)( 1)               15                  6               7          0
16       Preemption    task( 3)( 1)    task( 2)( 3)
17       Completion    task( 2)( 3)    task( 3)( 1)                1                  2               0          4
21       Preemption    task( 3)( 1)    task( 2)( 4)
22       Completion    task( 2)( 4)    task( 1)( 2)                1                  2               0          4
24       Completion    task( 1)( 2)    task( 3)( 1)                2                  2               0          8
26       Preemption    task( 3)( 1)    task( 2)( 5)
27       Completion    task( 2)( 5)    task( 3)( 1)                1                  2               0          4
28       Completion    task( 3)( 1)    task(63)                   13                  7               5          2
30       Preemption    task(63)        task( 3)( 2)
```

A report that describes your implementation (please attach the screenshot of the code and MARK the modified part). (40%)

(如未標示，整段即為全新 code)

ucos_ii.h

global variable

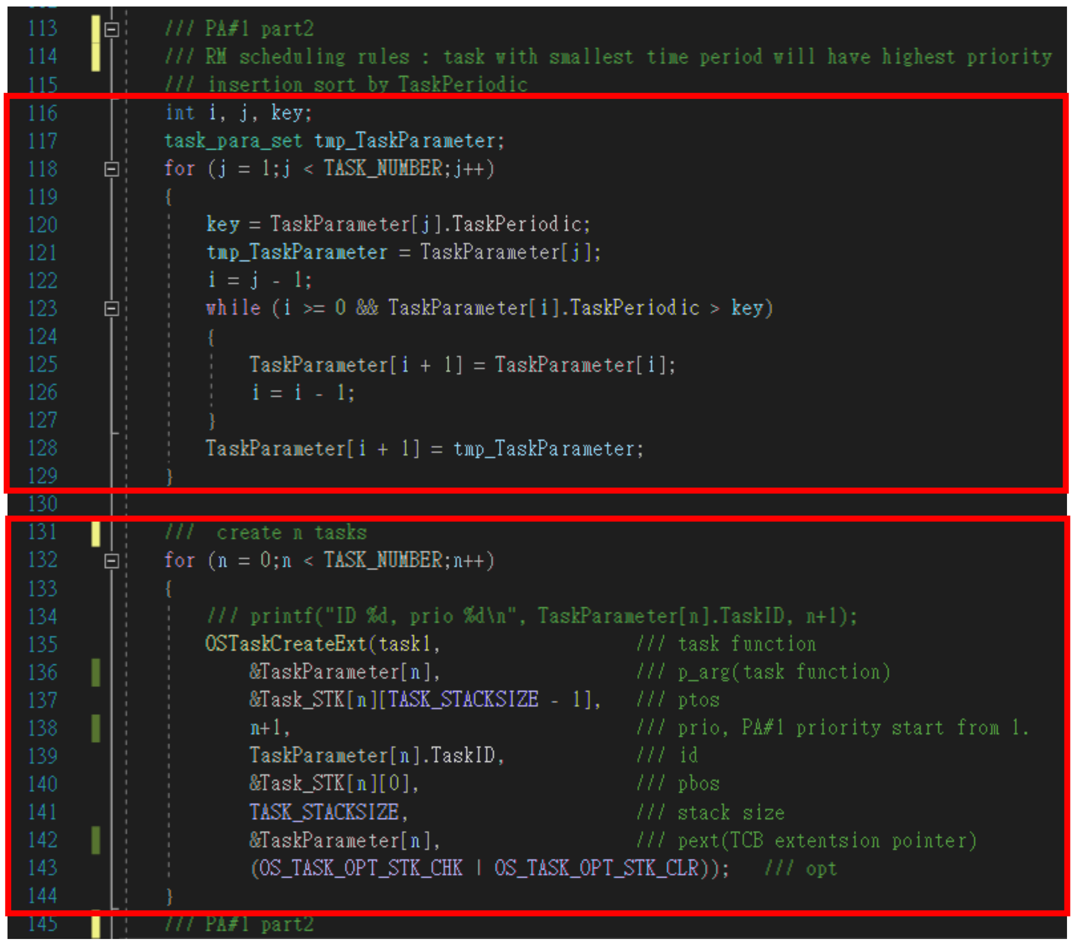


```

/// PA#1 part2
INT8U already_delay;
INT8U remain_exe_time;
INT8U next_job_time;
INT8U self_continue;
/// PA#1 part2

```

建立全域變數，用於後面計算 response time。

main.c

Main function

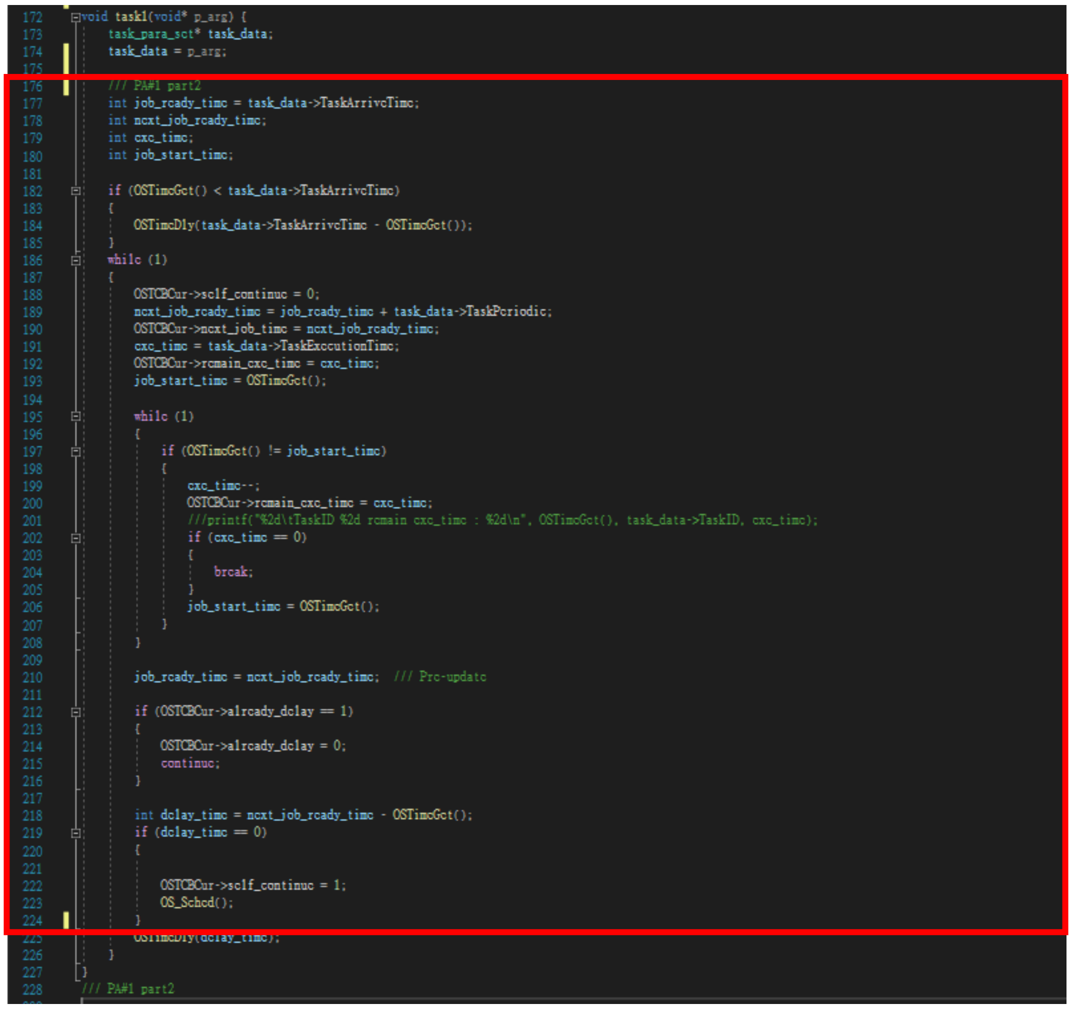


```c
/// PA#1 part2
/// RM scheduling rules : task with smallest time period will have highest priority
/// insertion sort by TaskPeriodic
int i, j, key;
task_para_set tmp_TaskParameter;
for (j = 1;j < TASK_NUMBER;j++)
{
    key = TaskParameter[j].TaskPeriodic;
    tmp_TaskParameter = TaskParameter[j];
    i = j - 1;
    while (i >= 0 && TaskParameter[i].TaskPeriodic > key)
    {
        TaskParameter[i + 1] = TaskParameter[i];
        i = i - 1;
    }
    TaskParameter[i + 1] = tmp_TaskParameter;
}

///  create n tasks
for (n = 0;n < TASK_NUMBER;n++)
{
    /// printf("ID %d, prio %d\n", TaskParameter[n].TaskID, n+1);
    OSTaskCreateExt(task1,                         /// task function
        &TaskParameter[n],                         /// p_arg(task function)
        &Task_STK[n][TASK_STACKSIZE - 1],          /// ptos
        n+1,                                       /// prio, PA#1 priority start from 1.
        TaskParameter[n].TaskID,                   /// id
        &Task_STK[n][0],                           /// pbos
        TASK_STACKSIZE,                            /// stack size
        &TaskParameter[n],                         /// pext(TCB extentsion pointer)
        (OS_TASK_OPT_STK_CHK | OS_TASK_OPT_STK_CLR));    /// opt
}
/// PA#1 part2
```

依照題目要求，需要能夠根據輸入的任務數量，建立任務，所以利用 for loop 進行 n 個 task 的建立。

因為要進行 RM scheduling，所以必須將 task 的 priority 根據 period 進行排列(成反比)，因此針對輸入的 period 將任務重新排列。

Task function

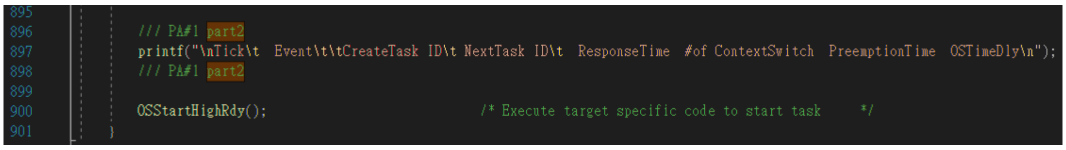


```c
void task1(void* p_arg) {
    task_para_set* task_data;
    task_data = p_arg;

    /// PA#1 part2
    int job_ready_time = task_data->TaskArriveTime;
    int next_job_ready_time;
    int exe_time;
    int job_start_time;

    if (OSTimeGet() < task_data->TaskArriveTime)
    {
        OSTimeDly(task_data->TaskArriveTime - OSTimeGet());
    }
    while (1)
    {
        OSTCBCur->self_continue = 0;
        next_job_ready_time = job_ready_time + task_data->TaskPeriodic;
        OSTCBCur->next_job_time = next_job_ready_time;
        exe_time = task_data->TaskExecutionTime;
        OSTCBCur->remain_exe_time = exe_time;
        job_start_time = OSTimeGet();

        while (1)
        {
            if (OSTimeGet() != job_start_time)
            {
                exe_time--;
                OSTCBCur->remain_exe_time = exe_time;
                ///printf("%2d\tTaskID %2d remain exe_time : %2d\n", OSTimeGet(), task_data->TaskID, exe_time);
                if (exe_time == 0)
                {
                    break;
                }
                job_start_time = OSTimeGet();
            }
        }

        job_ready_time = next_job_ready_time;  /// Pre-update

        if (OSTCBCur->already_delay == 1)
        {
            OSTCBCur->already_delay = 0;
            continue;
        }

        int delay_time = next_job_ready_time - OSTimeGet();
        if (delay_time == 0)
        {

            OSTCBCur->self_continue = 1;
            OS_Sched();
        }
        OSTimeDly(delay_time);
    }
}
/// PA#1 part2
```

根據給定的 arrive time, execution time, period 進行運作的 task funtion。execution 期間卡在 while 迴圈中不斷檢查執行時間是否結束,結束後則使用 OSTimeDly()進入 waiting 狀態,直到下個周期開始。

os_core.c

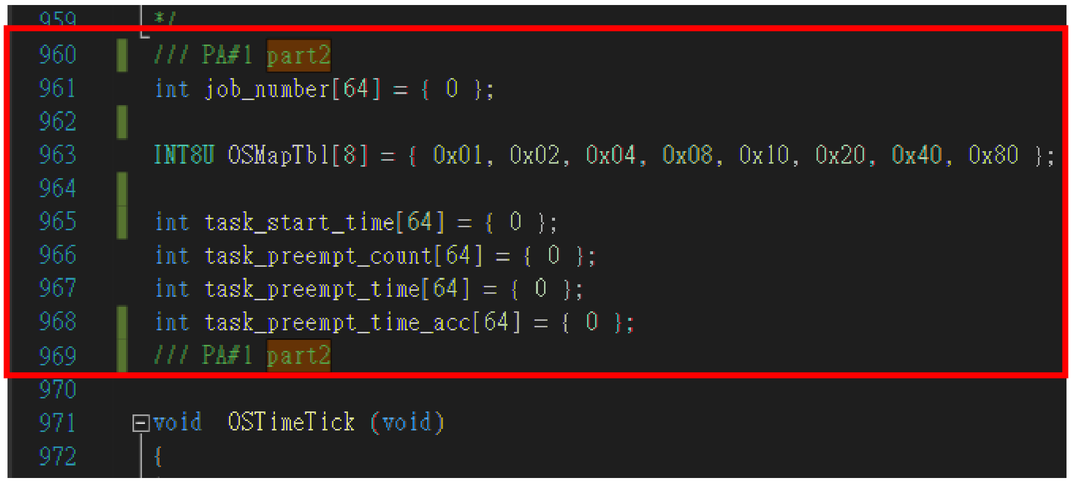
OSStart()



```c
        /// PA#1 part2
        printf("\nTick\t  Event\t\tCreateTask ID\t NextTask ID\t  ResponseTime  #of ContextSwitch  PreemptionTime  OSTimeDly\n");
        /// PA#1 part2

        OSStartHighRdy();                            /* Execute target specific code to start task     */
    }
```

根據題目要求，輸出標題。

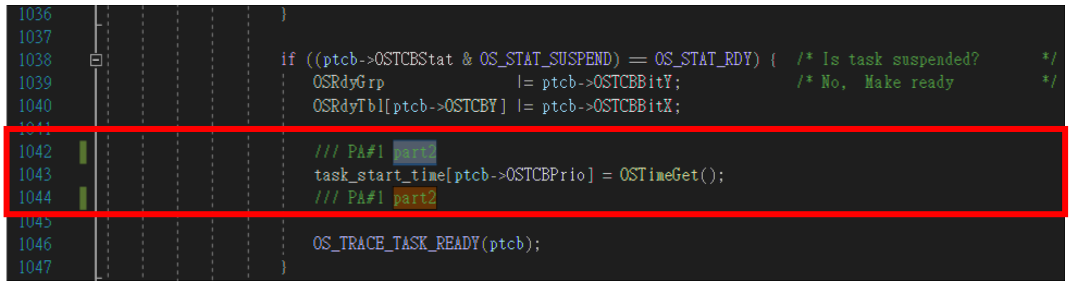
variable



```c
*/
/// PA#1 part2
int job_number[64] = { 0 };

INT8U OSMapTbl[8] = { 0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80 };

int task_start_time[64] = { 0 };
int task_preempt_count[64] = { 0 };
int task_preempt_time[64] = { 0 };
int task_preempt_time_acc[64] = { 0 };
/// PA#1 part2

void  OSTimeTick (void)
{
```

建立變數，等會用於計算 response time、preemtion time、job number 與 CtxSw number。

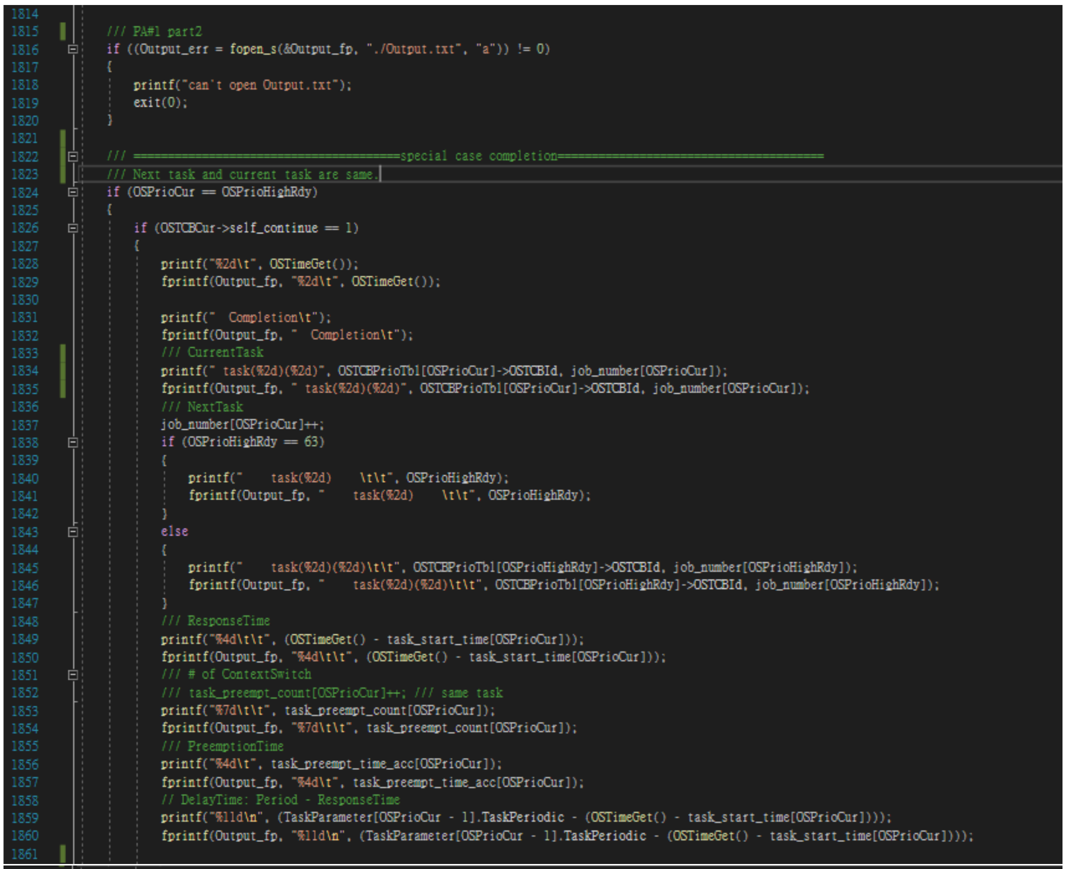
OSTimeTick()



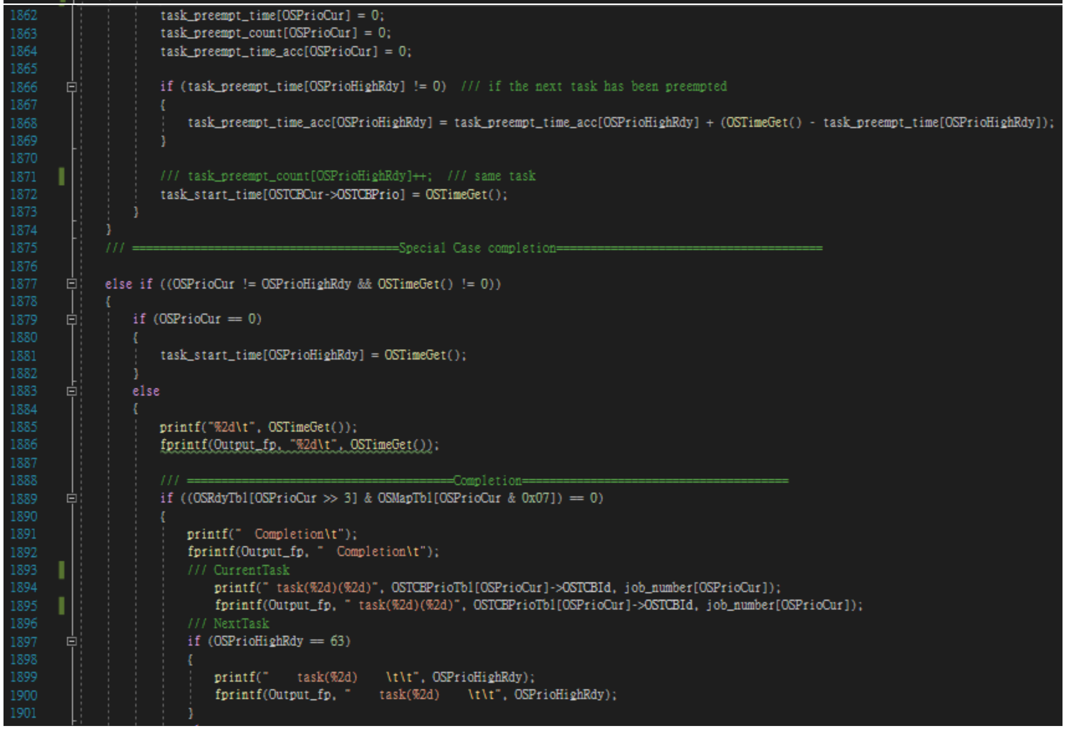
```c
                    }

                    if ((ptcb->OSTCBStat & OS_STAT_SUSPEND) == OS_STAT_RDY) {  /* Is task suspended?       */
                        OSRdyGrp               |= ptcb->OSTCBBitY;              /* No,  Make ready          */
                        OSRdyTbl[ptcb->OSTCBY] |= ptcb->OSTCBBitX;

                        /// PA#1 part2
                        task_start_time[ptcb->OSTCBPrio] = OSTimeGet();
                        /// PA#1 part2

                        OS_TRACE_TASK_READY(ptcb);
                    }
```

在 Time tick 中，設定 task 的啟動時間。

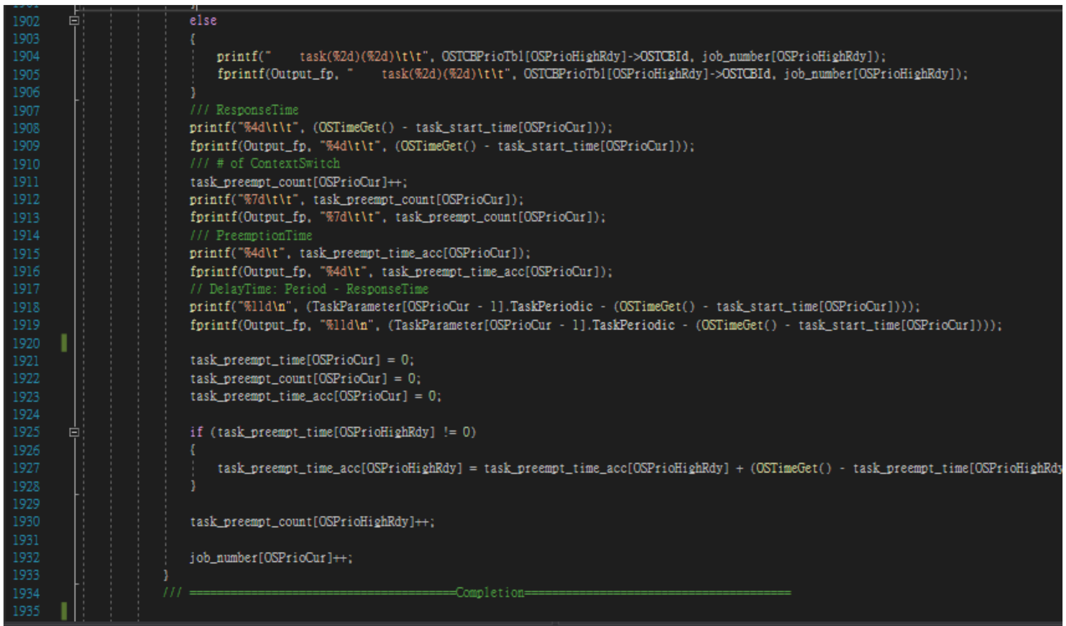
OS_SchedNew()



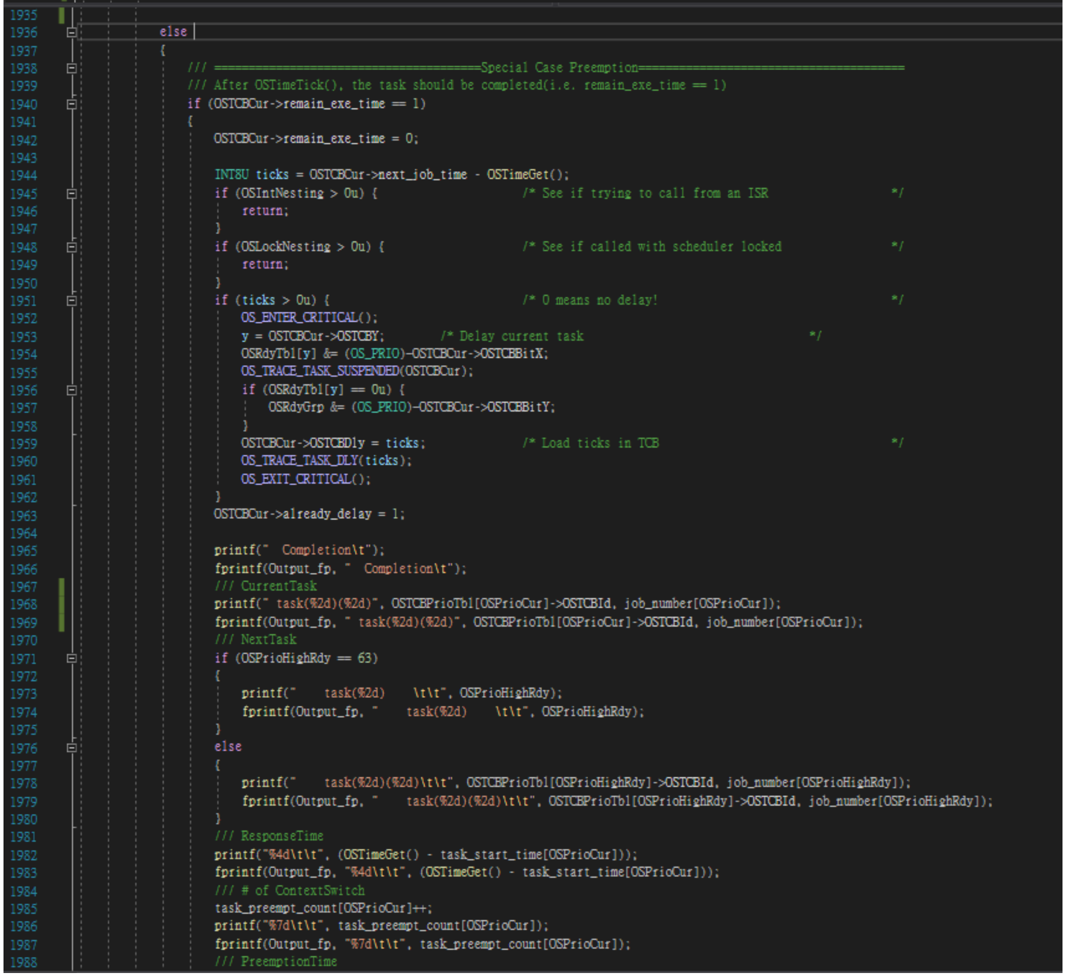
```c

    /// PA#1 part2
    if ((Output_err = fopen_s(&Output_fp, "./Output.txt", "a")) != 0)
    {
        printf("can't open Output.txt");
        exit(0);
    }

    /// =====================================special case completion=====================================
    /// Next task and current task are same.
    if (OSPrioCur == OSPrioHighRdy)
    {
        if (OSTCBCur->self_continue == 1)
        {
            printf("%2d\t", OSTimeGet());
            fprintf(Output_fp, "%2d\t", OSTimeGet());

            printf("  Completion\t");
            fprintf(Output_fp, "  Completion\t");
            /// CurrentTask
            printf(" task(%2d)(%2d)", OSTCBPrioTbl[OSPrioCur]->OSTCBId, job_number[OSPrioCur]);
            fprintf(Output_fp, " task(%2d)(%2d)", OSTCBPrioTbl[OSPrioCur]->OSTCBId, job_number[OSPrioCur]);
            /// NextTask
            job_number[OSPrioCur]++;
            if (OSPrioHighRdy == 63)
            {
                printf("    task(%2d)    \t\t", OSPrioHighRdy);
                fprintf(Output_fp, "    task(%2d)    \t\t", OSPrioHighRdy);
            }
            else
            {
                printf("    task(%2d)(%2d)\t\t", OSTCBPrioTbl[OSPrioHighRdy]->OSTCBId, job_number[OSPrioHighRdy]);
                fprintf(Output_fp, "    task(%2d)(%2d)\t\t", OSTCBPrioTbl[OSPrioHighRdy]->OSTCBId, job_number[OSPrioHighRdy]);
            }
            /// ResponseTime
            printf("%4d\t\t", (OSTimeGet() - task_start_time[OSPrioCur]));
            fprintf(Output_fp, "%4d\t\t", (OSTimeGet() - task_start_time[OSPrioCur]));
            /// # of ContextSwitch
            /// task_preempt_count[OSPrioCur]++; /// same task
            printf("%7d\t\t", task_preempt_count[OSPrioCur]);
            fprintf(Output_fp, "%7d\t\t", task_preempt_count[OSPrioCur]);
            /// PreemptionTime
            printf("%4d\t", task_preempt_time_acc[OSPrioCur]);
            fprintf(Output_fp, "%4d\t", task_preempt_time_acc[OSPrioCur]);
            // DelayTime: Period - ResponseTime
            printf("%lld\n", (TaskParameter[OSPrioCur - 1].TaskPeriodic - (OSTimeGet() - task_start_time[OSPrioCur])));
            fprintf(Output_fp, "%lld\n", (TaskParameter[OSPrioCur - 1].TaskPeriodic - (OSTimeGet() - task_start_time[OSPrioCur])));

```



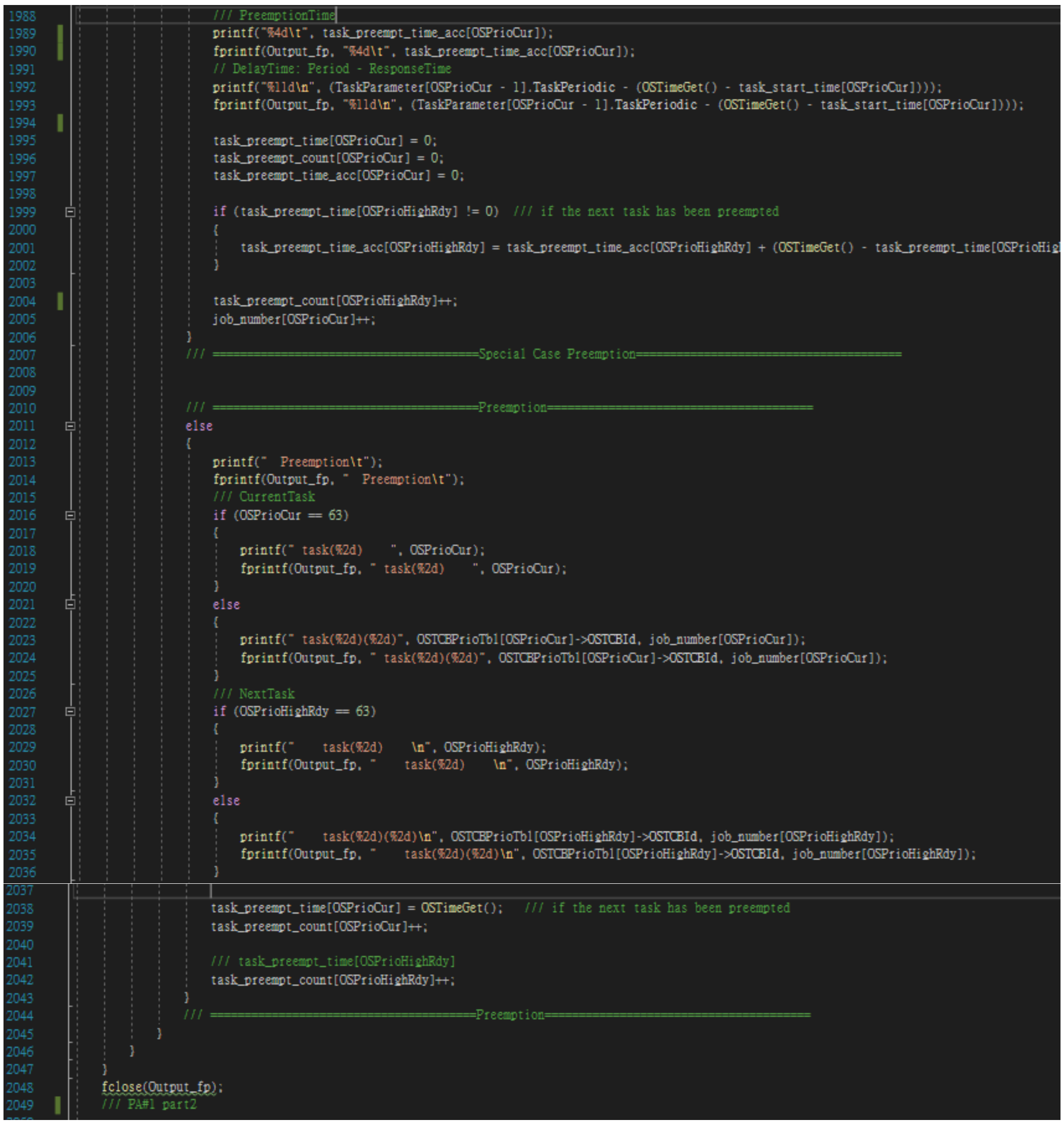
```c
            task_preempt_time[OSPrioCur] = 0;
            task_preempt_count[OSPrioCur] = 0;
            task_preempt_time_acc[OSPrioCur] = 0;

            if (task_preempt_time[OSPrioHighRdy] != 0)  /// if the next task has been preempted
            {
                task_preempt_time_acc[OSPrioHighRdy] = task_preempt_time_acc[OSPrioHighRdy] + (OSTimeGet() - task_preempt_time[OSPrioHighRdy]);
            }

            /// task_preempt_count[OSPrioHighRdy]++;  /// same task
            task_start_time[OSTCBCur->OSTCBPrio] = OSTimeGet();
        }
    }
    /// ======================================Special Case completion======================================

    else if ((OSPrioCur != OSPrioHighRdy && OSTimeGet() != 0))
    {
        if (OSPrioCur == 0)
        {
            task_start_time[OSPrioHighRdy] = OSTimeGet();
        }
        else
        {
            printf("%2d\t", OSTimeGet());
            fprintf(Output_fp, "%2d\t", OSTimeGet());

            /// ======================================Completion======================================
            if ((OSRdyTbl[OSPrioCur >> 3] & OSMapTbl[OSPrioCur & 0x07]) == 0)
            {
                printf("  Completion\t");
                fprintf(Output_fp, "  Completion\t");
                /// CurrentTask
                    printf(" task(%2d)(%2d)", OSTCBPrioTbl[OSPrioCur]->OSTCBId, job_number[OSPrioCur]);
                    fprintf(Output_fp, " task(%2d)(%2d)", OSTCBPrioTbl[OSPrioCur]->OSTCBId, job_number[OSPrioCur]);
                /// NextTask
                if (OSPrioHighRdy == 63)
                {
                    printf("    task(%2d)    \t\t", OSPrioHighRdy);
                    fprintf(Output_fp, "    task(%2d)    \t\t", OSPrioHighRdy);
                }
```



```
        else
        {
            printf("    task(%2d)(%2d)\t\t", OSTCBPrioTbl[OSPrioHighRdy]->OSTCBId, job_number[OSPrioHighRdy]);
            fprintf(Output_fp, "    task(%2d)(%2d)\t\t", OSTCBPrioTbl[OSPrioHighRdy]->OSTCBId, job_number[OSPrioHighRdy]);
        }
        /// ResponseTime
        printf("%4d\t\t", (OSTimeGet() - task_start_time[OSPrioCur]));
        fprintf(Output_fp, "%4d\t\t", (OSTimeGet() - task_start_time[OSPrioCur]));
        /// # of ContextSwitch
        task_preempt_count[OSPrioCur]++;
        printf("%7d\t\t", task_preempt_count[OSPrioCur]);
        fprintf(Output_fp, "%7d\t\t", task_preempt_count[OSPrioCur]);
        /// PreemptionTime
        printf("%4d\t", task_preempt_time_acc[OSPrioCur]);
        fprintf(Output_fp, "%4d\t", task_preempt_time_acc[OSPrioCur]);
        // DelayTime: Period - ResponseTime
        printf("%lld\n", (TaskParameter[OSPrioCur - 1].TaskPeriodic - (OSTimeGet() - task_start_time[OSPrioCur])));
        fprintf(Output_fp, "%lld\n", (TaskParameter[OSPrioCur - 1].TaskPeriodic - (OSTimeGet() - task_start_time[OSPrioCur])));

        task_preempt_time[OSPrioCur] = 0;
        task_preempt_count[OSPrioCur] = 0;
        task_preempt_time_acc[OSPrioCur] = 0;

        if (task_preempt_time[OSPrioHighRdy] != 0)
        {
            task_preempt_time_acc[OSPrioHighRdy] = task_preempt_time_acc[OSPrioHighRdy] + (OSTimeGet() - task_preempt_time[OSPrioHighRdy

        }

        task_preempt_count[OSPrioHighRdy]++;

        job_number[OSPrioCur]++;
    }
    /// ====================================Completion====================================
```



```
    else
    {
        /// ====================================Special Case Preemption====================================
        /// After OSTimeTick(), the task should be completed(i.e. remain_exe_time == 1)
        if (OSTCBCur->remain_exe_time == 1)
        {
            OSTCBCur->remain_exe_time = 0;

            INT8U ticks = OSTCBCur->next_job_time - OSTimeGet();
            if (OSIntNesting > 0u) {                     /* See if trying to call from an ISR                  */
                return;
            }
            if (OSLockNesting > 0u) {                    /* See if called with scheduler locked                */
                return;
            }
            if (ticks > 0u) {                            /* 0 means no delay!                                  */
                OS_ENTER_CRITICAL();
                y = OSTCBCur->OSTCBY;          /* Delay current task                                   */
                OSRdyTbl[y] &= (OS_PRIO)~OSTCBCur->OSTCBBitX;
                OS_TRACE_TASK_SUSPENDED(OSTCBCur);
                if (OSRdyTbl[y] == 0u) {
                    OSRdyGrp &= (OS_PRIO)~OSTCBCur->OSTCBBitY;
                }
                OSTCBCur->OSTCBDly = ticks;              /* Load ticks in TCB                                  */
                OS_TRACE_TASK_DLY(ticks);
                OS_EXIT_CRITICAL();
            }
            OSTCBCur->already_delay = 1;

            printf("  Completion\t");
            fprintf(Output_fp, "  Completion\t");
            /// CurrentTask
            printf(" task(%2d)(%2d)", OSTCBPrioTbl[OSPrioCur]->OSTCBId, job_number[OSPrioCur]);
            fprintf(Output_fp, " task(%2d)(%2d)", OSTCBPrioTbl[OSPrioCur]->OSTCBId, job_number[OSPrioCur]);
            /// NextTask
            if (OSPrioHighRdy == 63)
            {
                printf("    task(%2d)    \t\t", OSPrioHighRdy);
                fprintf(Output_fp, "    task(%2d)    \t\t", OSPrioHighRdy);
            }
            else
            {
                printf("    task(%2d)(%2d)\t\t", OSTCBPrioTbl[OSPrioHighRdy]->OSTCBId, job_number[OSPrioHighRdy]);
                fprintf(Output_fp, "    task(%2d)(%2d)\t\t", OSTCBPrioTbl[OSPrioHighRdy]->OSTCBId, job_number[OSPrioHighRdy]);
            }
            /// ResponseTime
            printf("%4d\t\t", (OSTimeGet() - task_start_time[OSPrioCur]));
            fprintf(Output_fp, "%4d\t\t", (OSTimeGet() - task_start_time[OSPrioCur]));
            /// # of ContextSwitch
            task_preempt_count[OSPrioCur]++;
            printf("%7d\t\t", task_preempt_count[OSPrioCur]);
            fprintf(Output_fp, "%7d\t\t", task_preempt_count[OSPrioCur]);
            /// PreemptionTime
```



```c
                    /// PreemptionTime
                    printf("%4d\t", task_preempt_time_acc[OSPrioCur]);
                    fprintf(Output_fp, "%4d\t", task_preempt_time_acc[OSPrioCur]);
                    // DelayTime: Period - ResponseTime
                    printf("%lld\n", (TaskParameter[OSPrioCur - 1].TaskPeriodic - (OSTimeGet() - task_start_time[OSPrioCur])));
                    fprintf(Output_fp, "%lld\n", (TaskParameter[OSPrioCur - 1].TaskPeriodic - (OSTimeGet() - task_start_time[OSPrioCur])));

                    task_preempt_time[OSPrioCur] = 0;
                    task_preempt_count[OSPrioCur] = 0;
                    task_preempt_time_acc[OSPrioCur] = 0;

                    if (task_preempt_time[OSPrioHighRdy] != 0)  /// if the next task has been preempted
                    {
                        task_preempt_time_acc[OSPrioHighRdy] = task_preempt_time_acc[OSPrioHighRdy] + (OSTimeGet() - task_preempt_time[OSPrioHig
                    }

                    task_preempt_count[OSPrioHighRdy]++;
                    job_number[OSPrioCur]++;
                }
                /// ======================================Special Case Preemption======================================


                /// ======================================Preemption======================================
                else
                {
                    printf("  Preemption\t");
                    fprintf(Output_fp, "  Preemption\t");
                    /// CurrentTask
                    if (OSPrioCur == 63)
                    {
                        printf(" task(%2d)    ", OSPrioCur);
                        fprintf(Output_fp, " task(%2d)    ", OSPrioCur);
                    }
                    else
                    {
                        printf(" task(%2d)(%2d)", OSTCBPrioTbl[OSPrioCur]->OSTCBId, job_number[OSPrioCur]);
                        fprintf(Output_fp, " task(%2d)(%2d)", OSTCBPrioTbl[OSPrioCur]->OSTCBId, job_number[OSPrioCur]);
                    }
                    /// NextTask
                    if (OSPrioHighRdy == 63)
                    {
                        printf("    task(%2d)    \n", OSPrioHighRdy);
                        fprintf(Output_fp, "    task(%2d)    \n", OSPrioHighRdy);
                    }
                    else
                    {
                        printf("    task(%2d)(%2d)\n", OSTCBPrioTbl[OSPrioHighRdy]->OSTCBId, job_number[OSPrioHighRdy]);
                        fprintf(Output_fp, "    task(%2d)(%2d)\n", OSTCBPrioTbl[OSPrioHighRdy]->OSTCBId, job_number[OSPrioHighRdy]);
                    }

                    task_preempt_time[OSPrioCur] = OSTimeGet();   /// if the next task has been preempted
                    task_preempt_count[OSPrioCur]++;

                    /// task_preempt_time[OSPrioHighRdy]
                    task_preempt_count[OSPrioHighRdy]++;
                }
                /// ======================================Preemption======================================
            }
        }
    }
    fclose(Output_fp);
    /// PA#1 part2
```

透過 OSRdyTbl 去確認 task 是否 ready 並進入 waiting，來區分任務為 completion 與 preemption，其中加入 OSPrioCur 與 OSPrioHighRdy，可以協助判斷 task 的切換。

資料	判斷、記錄方式
Event	Task a 切換至 Task b 時,若 Task a 在 OSRdyTbl 上仍處於 ready 狀態(==1)，則此情況為 preemtpion；反之則為 completion。
Response Time	「Completion 的時間點」減掉「task 被設為 ready 的時間點(記錄於 task_start_time)」，即為 response time。
# of ContextSwitch	記錄於 task_preempt_count，每當 task 切換則+1，completion 輸出後歸零。
Preemption Time	發生 preemtpion 時,記錄「task 被 preempt 的時間點」,與「後續返回到該 task 的時間點」之差距,即為 preemption time。再 task complete 之前有多次 preemption 則進行累加。

Report

這次的 PA1 主要為 RM Scheduling 的實作，首先透過熟悉 TCB 的運作過程，再到如何計算 Response Time 等等工作時間，過程挫折感滿重的，需要將紙本的知識應用於實作上，常常因為不夠清楚，導致一直踩坑。

雖然沒有做出完整的功能，但卻更加熟悉 RM 的設計流程，非常有趣，也透過這次作業知道了生活上有許事情要安排，就好像這次作業一樣，要如何評估是非常重要的，才不會因為期中考與作業同時來的時候，導致 Miss Deadline。