Embedded OS Implementation PA1

學號 ：M11407517

姓名 ： 黃志傑

[PART I] Task Control Block Linked List

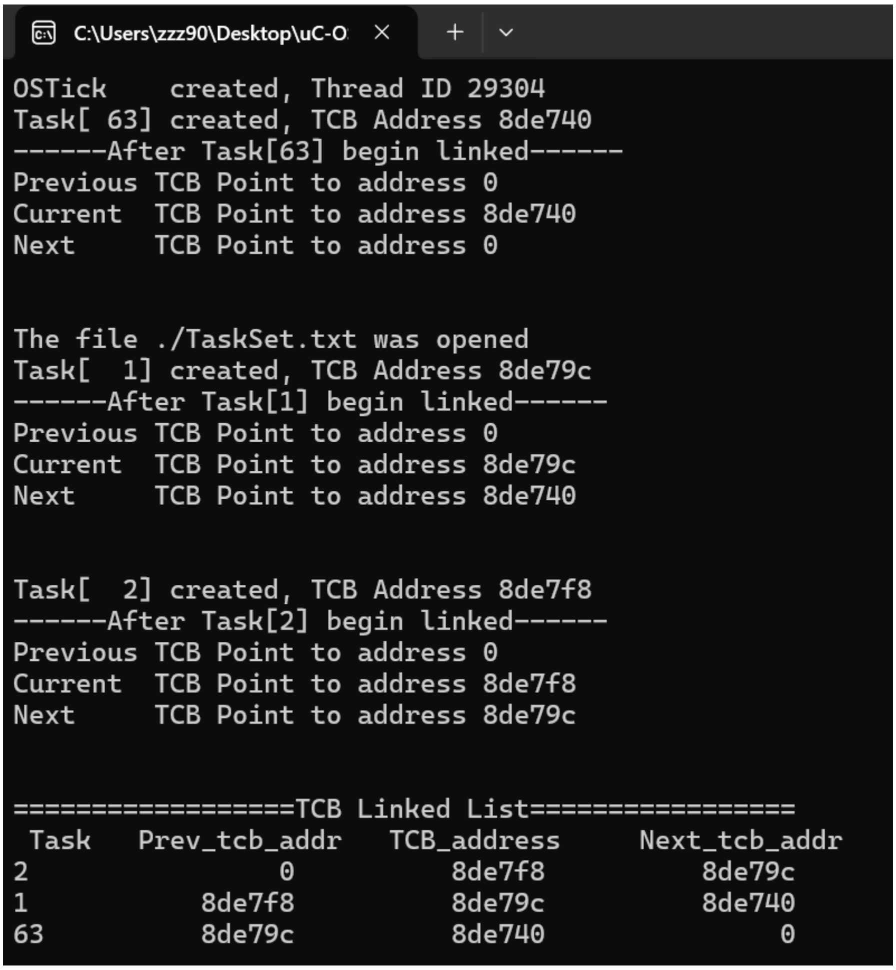
***Objective*:**

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 TCB address is dynamic.

※ This part will be included in the subsequent output and does not require separate code submission.

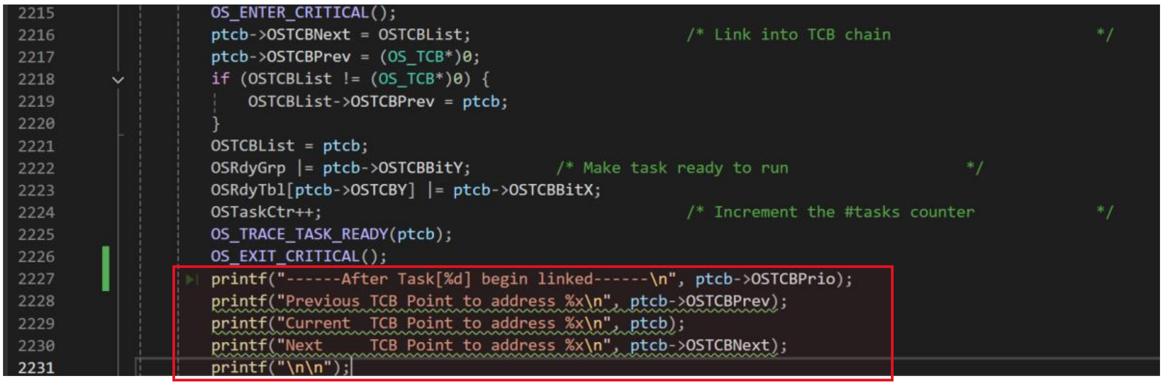
- Output :



```
OSTick    created, Thread ID 29304
Task[ 63] created, TCB Address 8de740
------After Task[63] begin linked------
Previous TCB Point to address 0
Current  TCB Point to address 8de740
Next     TCB Point to address 0


The file ./TaskSet.txt was opened
Task[  1] created, TCB Address 8de79c
------After Task[1] begin linked------
Previous TCB Point to address 0
Current  TCB Point to address 8de79c
Next     TCB Point to address 8de740


Task[  2] created, TCB Address 8de7f8
------After Task[2] begin linked------
Previous TCB Point to address 0
Current  TCB Point to address 8de7f8
Next     TCB Point to address 8de79c


==================TCB Linked List=================
 Task   Prev_tcb_addr   TCB_address     Next_tcb_addr
2                   0        8de7f8           8de79c
1              8de7f8        8de79c           8de740
63             8de79c        8de740                0
```

- Code:

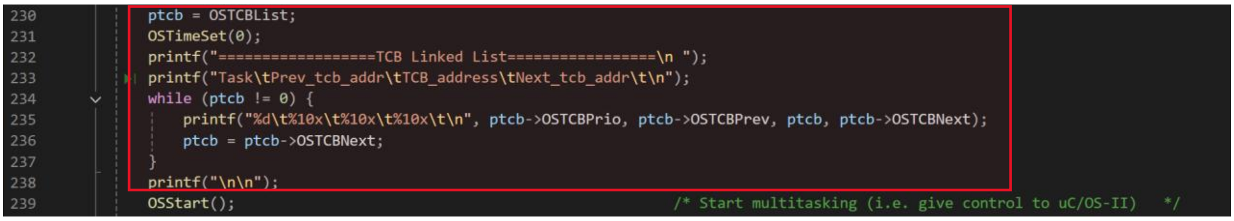
  ➢ os_core.c : OS_TCBInit( )函式中



```c
OS_ENTER_CRITICAL();
ptcb->OSTCBNext = OSTCBList;                    /* Link into TCB chain                      */
ptcb->OSTCBPrev = (OS_TCB*)0;
if (OSTCBList != (OS_TCB*)0) {
    OSTCBList->OSTCBPrev = ptcb;
}
OSTCBList = ptcb;
OSRdyGrp |= ptcb->OSTCBBitY;          /* Make task ready to run                   */
OSRdyTbl[ptcb->OSTCBY] |= ptcb->OSTCBBitX;
OSTaskCtr++;                                    /* Increment the #tasks counter             */
OS_TRACE_TASK_READY(ptcb);
OS_EXIT_CRITICAL();
printf("------After Task[%d] begin linked------\n", ptcb->OSTCBPrio);
printf("Previous TCB Point to address %x\n", ptcb->OSTCBPrev);
printf("Current  TCB Point to address %x\n", ptcb);
printf("Next     TCB Point to address %x\n", ptcb->OSTCBNext);
printf("\n\n");
```

1. TCB 初始化過程中，從 OSTCBFreeList 取出空白 TCB 再由 OSTCBList link 起來，程式部分在 OSTCBList 完後 printf 出 Previous、Current、Next 的地址

  ➢ main.c : main function 中



```c
ptcb = OSTCBList;
OSTimeSet(0);
printf("=================TCB Linked List================\n ");
printf("Task\tPrev_tcb_addr\tTCB_address\tNext_tcb_addr\t\n");
while (ptcb != 0) {
    printf("%d\t%10x\t%10x\t%10x\t\n", ptcb->OSTCBPrio, ptcb->OSTCBPrev, ptcb, ptcb->OSTCBNext);
    ptcb = ptcb->OSTCBNext;
}
printf("\n\n");
OSStart();                                      /* Start multitasking (i.e. give control to uC/OS-II)   */
```

1. 在 OSStart 開始前，使用 ptcb 指標到 OSTCBList，利用 while 迴圈 printf 所有被 OSTCBList Link 起來的 Task TCB address

[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 μC/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.

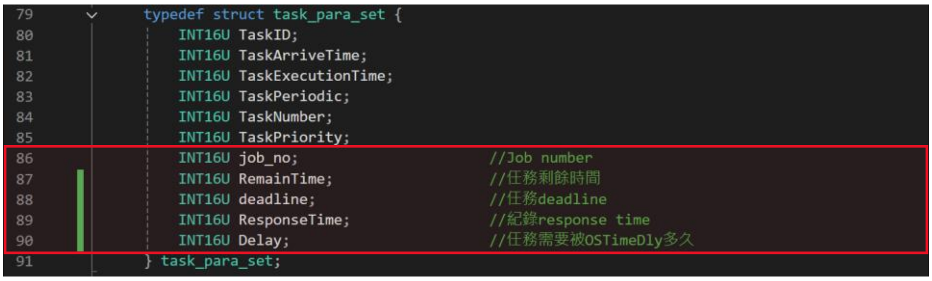
Periodic Task Set = {τ_{ID} (ID, arrival time, execution time, period)}

Example Task Set 1 = {τ_1 (1, 0, 1, 3), τ_2 (2, 0, 3, 5)}

Example Task Set 2 = {τ_1 (1, 0, 1, 3), τ_2 (2, 1, 1, 4), τ_3 (3, 2, 1, 5) }

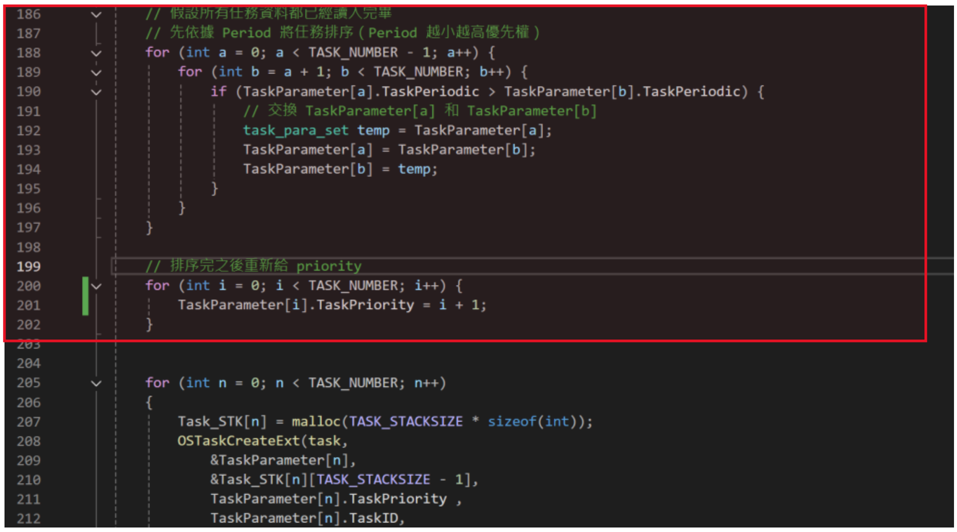
Example Task Set 3 = {τ_1 (1, 0, 3, 8), τ_2 (2, 1, 2, 6), τ_3 (3, 0, 4, 15)}

- Code:
  - ucos.ii.h： 增加 task_para_set member



```c
typedef struct task_para_set {
    INT16U TaskID;
    INT16U TaskArriveTime;
    INT16U TaskExecutionTime;
    INT16U TaskPeriodic;
    INT16U TaskNumber;
    INT16U TaskPriority;
    INT16U job_no;                          //Job number
    INT16U RemainTime;                      //任務剩餘時間
    INT16U deadline;                        //任務deadline
    INT16U ResponseTime;                    //紀錄response time
    INT16U Delay;                           //任務需要被OSTimeDly多久
} task_para_set;
```

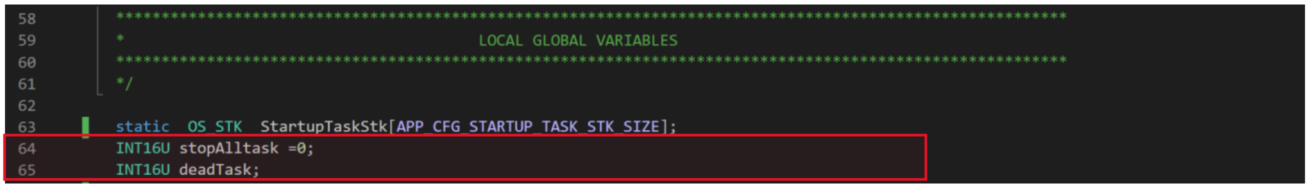
  - main.c : main function



```c
// 假設所有任務資料都已經讀入完畢
// 先依據 Period 將任務排序 ( Period 越小越高優先權 )
for (int a = 0; a < TASK_NUMBER - 1; a++) {
    for (int b = a + 1; b < TASK_NUMBER; b++) {
        if (TaskParameter[a].TaskPeriodic > TaskParameter[b].TaskPeriodic) {
            // 交換 TaskParameter[a] 和 TaskParameter[b]
            task_para_set temp = TaskParameter[a];
            TaskParameter[a] = TaskParameter[b];
            TaskParameter[b] = temp;
        }
    }
}

// 排序完之後重新給 priority
for (int i = 0; i < TASK_NUMBER; i++) {
    TaskParameter[i].TaskPriority = i + 1;
}


for (int n = 0; n < TASK_NUMBER; n++)
{
    Task_STK[n] = malloc(TASK_STACKSIZE * sizeof(int));
    OSTaskCreateExt(task,
        &TaskParameter[n],
        &Task_STK[n][TASK_STACKSIZE - 1],
        TaskParameter[n].TaskPriority ,
        TaskParameter[n].TaskID,
```

1. 在全部任務資料進入 TaskParameter 時，比較任務的 TaskPeriodic 大小，由小往大排列
2. 根據排列順序給 TaskPriority
3. 使 Periodic 較短的任務優先級較高，較長的任務優先級較低，達到 RM 排程的目的，也可以直接和 TaskID 對照

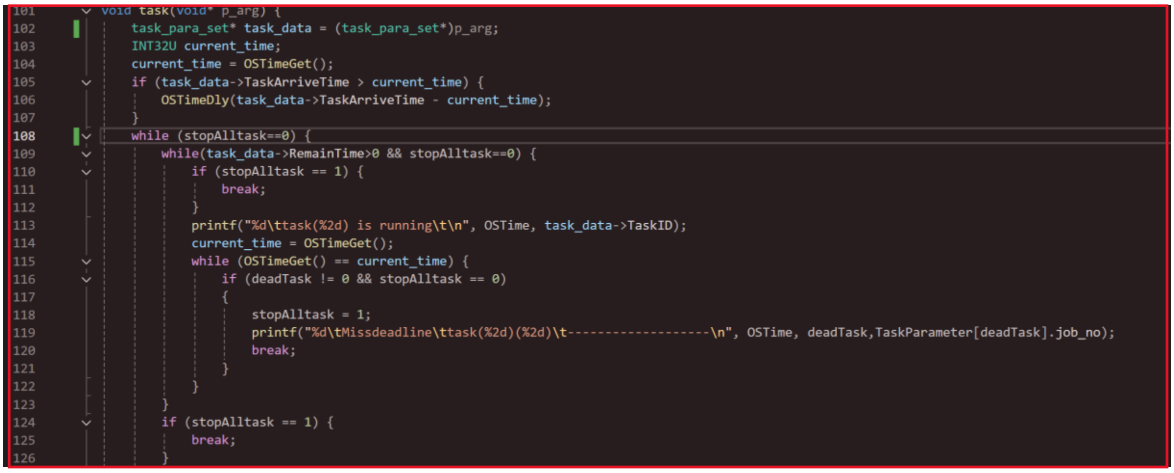
- 設定全域變數 StopAlltask、deadtask



```
*********************************************************************************************************
*                                           LOCAL GLOBAL VARIABLES
*********************************************************************************************************
*/

static  OS_STK  StartupTaskStk[APP_CFG_STARTUP_TASK_STK_SIZE];
INT16U stopAlltask =0;
INT16U deadTask;
```

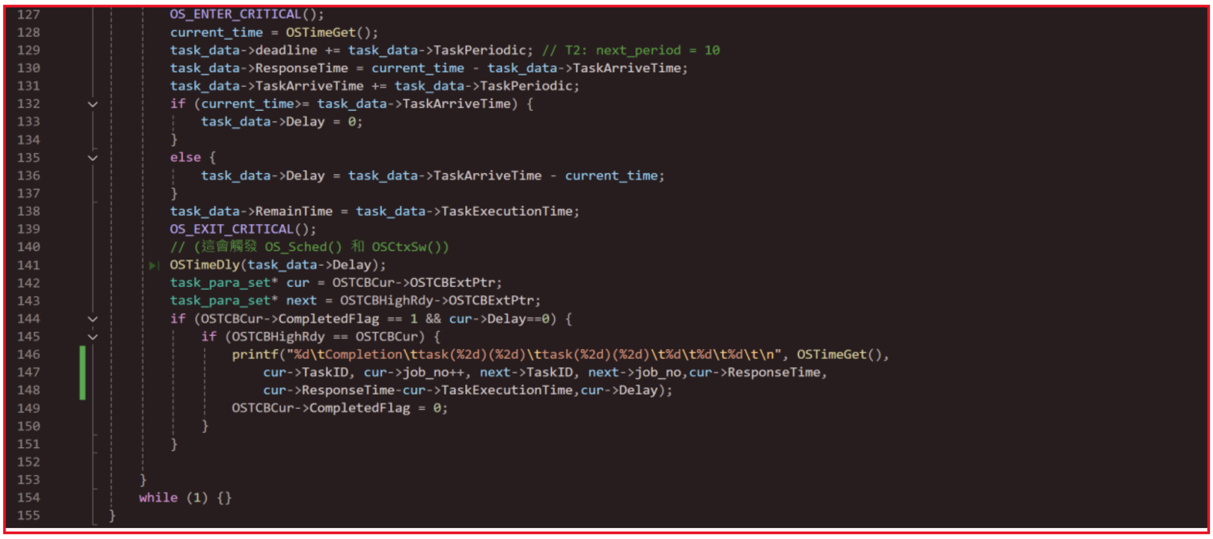
1. StopAlltask 為一個 flag 當有任務 MissDeadline 時觸發為 1，中止所有任務
2. deadTask 紀錄 MissDeadline 發生時的 task

- Task Function :



```
101 void task(void* p_arg) {
102     task_para_set* task_data = (task_para_set*)p_arg;
103     INT32U current_time;
104     current_time = OSTimeGet();
105     if (task_data->TaskArriveTime > current_time) {
106         OSTimeDly(task_data->TaskArriveTime - current_time);
107     }
108     while (stopAlltask==0) {
109         while(task_data->RemainTime>0 && stopAlltask==0) {
110             if (stopAlltask == 1) {
111                 break;
112             }
113             printf("%d\ttask(%2d) is running\t\n", OSTime, task_data->TaskID);
114             current_time = OSTimeGet();
115             while (OSTimeGet() == current_time) {
116                 if (deadTask != 0 && stopAlltask == 0)
117                 {
118                     stopAlltask = 1;
119                     printf("%d\tMissdeadline\ttask(%2d)(%2d)\t------------------\n", OSTime, deadTask,TaskParameter[deadTask].job_no);
120                     break;
121                 }
122             }
123         }
124         if (stopAlltask == 1) {
125             break;
126         }
```

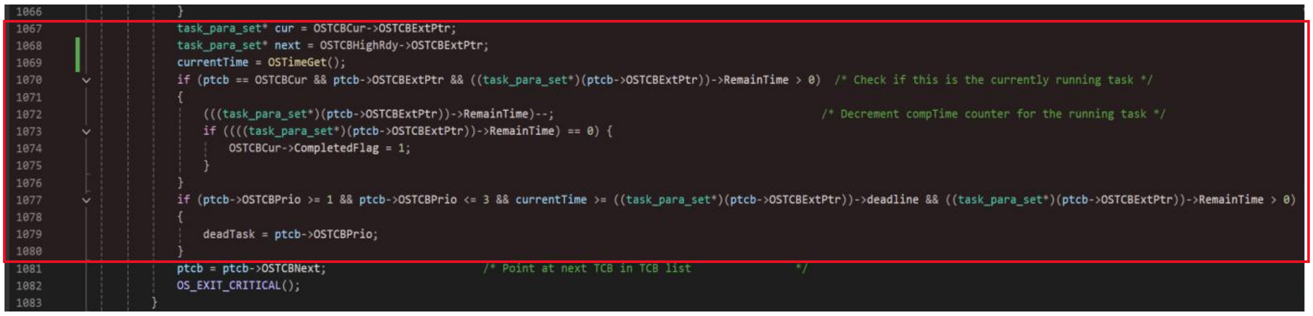
1. (104~107 行)： 進入 Task Function 先檢查目前時間是不是剛好或超過 ArriveTime，如果不是需要 OSTimeDly 等待 ArriveTime 到達
2. stopAlltask 是一個**全域變數**只要為 0，模擬就會繼續進行。一旦它變為 1（發生 Deadline Miss），所有任務都會跳出這個迴圈並終止
3. RemainTime 計數這個任務還有多少 ExecuteTime 的 Tick 需要執行(在 OSTimeTick 進行 RemainTime - -的動作)，當 RemainTime = 0 代表任務做完，跳出 while(task_data->RemainTime>0 && stopAlltask==0)迴圈。
4. 迴圈中做 printf : task is running 的任務，並檢查是否有 MissDeadline 的發生



```c
        OS_ENTER_CRITICAL();
        current_time = OSTimeGet();
        task_data->deadline += task_data->TaskPeriodic; // T2: next_period = 10
        task_data->ResponseTime = current_time - task_data->TaskArriveTime;
        task_data->TaskArriveTime += task_data->TaskPeriodic;
        if (current_time>= task_data->TaskArriveTime) {
            task_data->Delay = 0;
        }
        else {
            task_data->Delay = task_data->TaskArriveTime - current_time;
        }
        task_data->RemainTime = task_data->TaskExecutionTime;
        OS_EXIT_CRITICAL();
        // (這會觸發 OS_Sched() 和 OSCtxSw())
        OSTimeDly(task_data->Delay);
        task_para_set* cur = OSTCBCur->OSTCBExtPtr;
        task_para_set* next = OSTCBHighRdy->OSTCBExtPtr;
        if (OSTCBCur->CompletedFlag == 1 && cur->Delay==0) {
            if (OSTCBHighRdy == OSTCBCur) {
                printf("%d\tCompletion\ttask(%2d)(%2d)\ttask(%2d)(%2d)\t%d\t%d\t%d\t\n", OSTimeGet(),
                    cur->TaskID, cur->job_no++, next->TaskID, next->job_no,cur->ResponseTime,
                    cur->ResponseTime-cur->TaskExecutionTime,cur->Delay);
                OSTCBCur->CompletedFlag = 0;
            }
        }

    }
    while (1) {}
}
```

1. RemainTime = 0 跳出迴圈後，計算需要 Delay 的時間，以及 ResponseTime、下次任務抵達時間、任務 Deadline
2. 抓取當前時間，判斷是否抵達 ArriveTime，根據情況判斷需要 Delay 時間
3. 如果 TimeDly = 0，且沒有更高 priority 任務抵達(如 Taskset1 第五個 tick)，任務不會經過 OS_Sched( )，因此需要補印 complete 訊息
4. While(1)為任務跳出進入無窮迴圈終止狀況

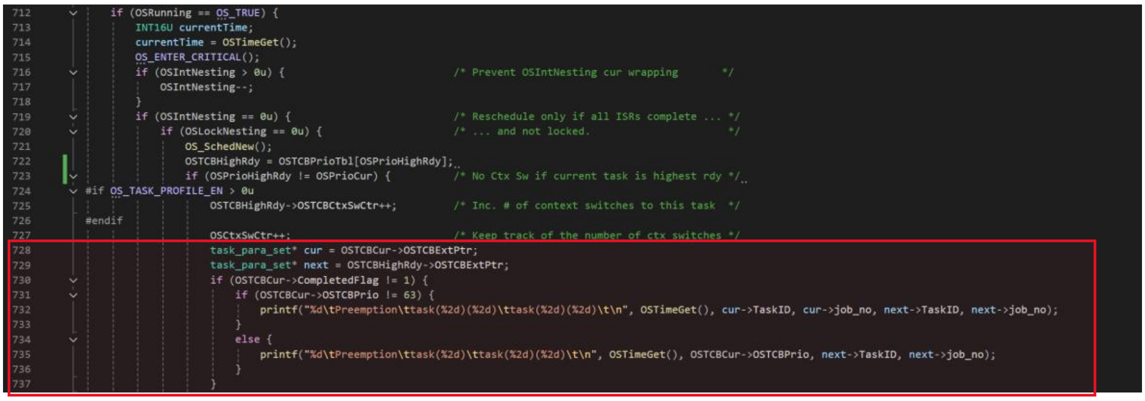
- ➢ OSTimeTick :



```c
            }
            task_para_set* cur = OSTCBCur->OSTCBExtPtr;
            task_para_set* next = OSTCBHighRdy->OSTCBExtPtr;
            currentTime = OSTimeGet();
            if (ptcb == OSTCBCur && ptcb->OSTCBExtPtr && ((task_para_set*)(ptcb->OSTCBExtPtr))->RemainTime > 0)  /* Check if this is the currently running task */
            {
                (((task_para_set*)(ptcb->OSTCBExtPtr))->RemainTime)--;                          /* Decrement compTime counter for the running task */
                if ((((task_para_set*)(ptcb->OSTCBExtPtr))->RemainTime) == 0) {
                    OSTCBCur->CompletedFlag = 1;
                }
            }
            if (ptcb->OSTCBPrio >= 1 && ptcb->OSTCBPrio <= 3 && currentTime >= ((task_para_set*)(ptcb->OSTCBExtPtr))->deadline && ((task_para_set*)(ptcb->OSTCBExtPtr))->RemainTime > 0)
            {
                deadTask = ptcb->OSTCBPrio;
            }
            ptcb = ptcb->OSTCBNext;                        /* Point at next TCB in TCB list              */
            OS_EXIT_CRITICAL();
        }
```

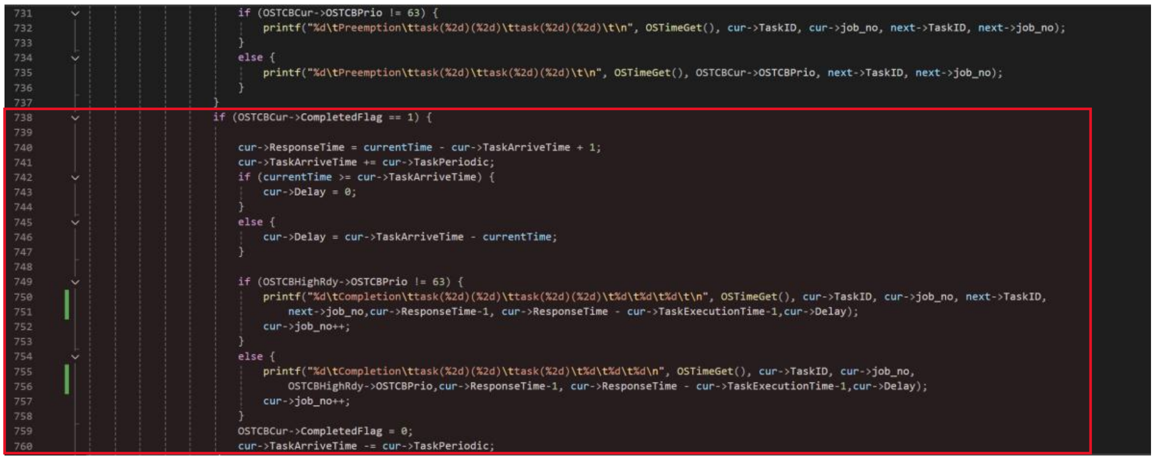
1. 在 OSTimeTick 中每個 Tick 做 RemainTime- -的動作，代表任務 Execution 1tick 時間，並在任務完成時給他一個 completeFlag
2. 巡視 OSTCBList 檢查是否有任務 MissDeadline，如果有就將全域變數 DeadTask 設為此任務

- OSIntExit( ):



```c
    if (OSRunning == OS_TRUE) {
        INT16U currentTime;
        currentTime = OSTimeGet();
        OS_ENTER_CRITICAL();
        if (OSIntNesting > 0u) {                        /* Prevent OSIntNesting cur wrapping       */
            OSIntNesting--;
        }
        if (OSIntNesting == 0u) {                       /* Reschedule only if all ISRs complete ... */
            if (OSLockNesting == 0u) {                  /* ... and not locked.                      */
                OS_SchedNew();
                OSTCBHighRdy = OSTCBPrioTbl[OSPrioHighRdy];
                if (OSPrioHighRdy != OSPrioCur) {       /* No Ctx Sw if current task is highest rdy */
#if OS_TASK_PROFILE_EN > 0u
                    OSTCBHighRdy->OSTCBCtxSwCtr++;      /* Inc. # of context switches to this task  */
#endif
                    OSCtxSwCtr++;                       /* Keep track of the number of ctx switches */
                    task_para_set* cur = OSTCBCur->OSTCBExtPtr;
                    task_para_set* next = OSTCBHighRdy->OSTCBExtPtr;
                    if (OSTCBCur->CompletedFlag != 1) {
                        if (OSTCBCur->OSTCBPrio != 63) {
                            printf("%d\tPreemption\ttask(%2d)(%2d)\ttask(%2d)(%2d)\t\n", OSTimeGet(), cur->TaskID, cur->job_no, next->TaskID, next->job_no);
                        }
                        else {
                            printf("%d\tPreemption\ttask(%2d)\ttask(%2d)(%2d)\t\n", OSTimeGet(), OSTCBCur->OSTCBPrio, next->TaskID, next->job_no);
                        }
                    }
```

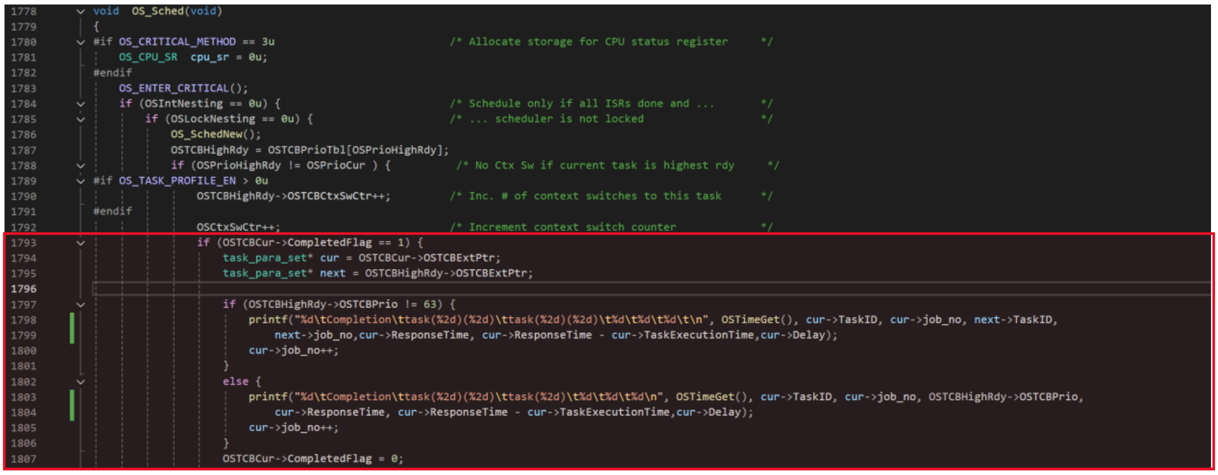
1. OSTimeTick 結束時會跳至 OSIntExit()，OS_SchedNew()更新完 OSTCBHighRdy 的任務後，如果不等於當前任務=>代表有搶斷 Preempt 發生，printf 出 Preempt 訊息



```c
                        if (OSTCBCur->OSTCBPrio != 63) {
                            printf("%d\tPreemption\ttask(%2d)(%2d)\ttask(%2d)(%2d)\t\n", OSTimeGet(), cur->TaskID, cur->job_no, next->TaskID, next->job_no);
                        }
                        else {
                            printf("%d\tPreemption\ttask(%2d)\ttask(%2d)(%2d)\t\n", OSTimeGet(), OSTCBCur->OSTCBPrio, next->TaskID, next->job_no);
                        }
                    }
                    if (OSTCBCur->CompletedFlag == 1) {

                        cur->ResponseTime = currentTime - cur->TaskArriveTime + 1;
                        cur->TaskArriveTime += cur->TaskPeriodic;
                        if (currentTime >= cur->TaskArriveTime) {
                            cur->Delay = 0;
                        }
                        else {
                            cur->Delay = cur->TaskArriveTime - currentTime;
                        }

                        if (OSTCBHighRdy->OSTCBPrio != 63) {
                            printf("%d\tCompletion\ttask(%2d)(%2d)\ttask(%2d)(%2d)\t%d\t%d\t%d\t\n", OSTimeGet(), cur->TaskID, cur->job_no, next->TaskID,
                                next->job_no,cur->ResponseTime-1, cur->ResponseTime - cur->TaskExecutionTime-1,cur->Delay);
                            cur->job_no++;
                        }
                        else {
                            printf("%d\tCompletion\ttask(%2d)(%2d)\ttask(%2d)\t%d\t%d\t%d\n", OSTimeGet(), cur->TaskID, cur->job_no,
                                OSTCBHighRdy->OSTCBPrio,cur->ResponseTime-1, cur->ResponseTime - cur->TaskExecutionTime-1,cur->Delay);
                            cur->job_no++;
                        }
                        OSTCBCur->CompletedFlag = 0;
                        cur->TaskArriveTime -= cur->TaskPeriodic;
```

2. 特殊情況為當 LowPriority 任務完成，且 HighPriority 任務剛好被喚醒，HighPriority 任務會直接 Execute，直到 Execute 結束才 resume 回 LowPrioity 任務，因此需要在 CompleteFlag=1 時補印 LowPrioity 的 Complete 訊息

➢ OS_Sched( ) :

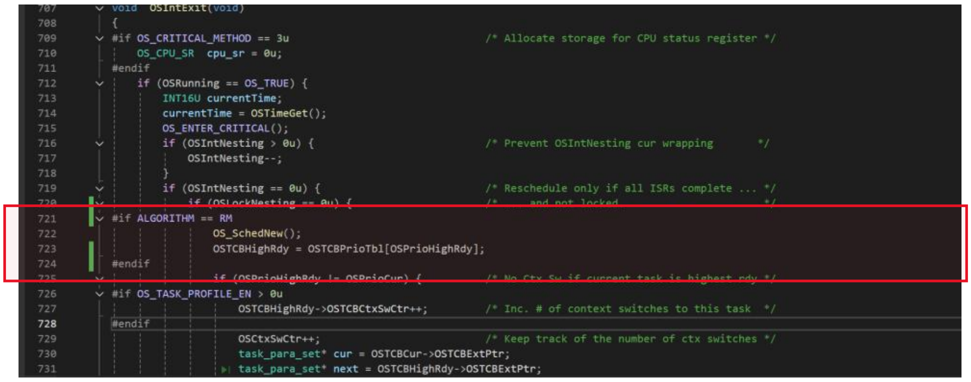


```c
void  OS_Sched(void)
{
#if OS_CRITICAL_METHOD == 3u                           /* Allocate storage for CPU status register     */
    OS_CPU_SR  cpu_sr = 0u;
#endif
    OS_ENTER_CRITICAL();
    if (OSIntNesting == 0u) {                          /* Schedule only if all ISRs done and ...       */
        if (OSLockNesting == 0u) {                     /* ... scheduler is not locked                  */
            OS_SchedNew();
            OSTCBHighRdy = OSTCBPrioTbl[OSPrioHighRdy];
            if (OSPrioHighRdy != OSPrioCur ) {          /* No Ctx Sw if current task is highest rdy     */
#if OS_TASK_PROFILE_EN > 0u
                OSTCBHighRdy->OSTCBCtxSwCtr++;         /* Inc. # of context switches to this task      */
#endif
                OSCtxSwCtr++;                          /* Increment context switch counter             */
                if (OSTCBCur->CompletedFlag == 1) {
                    task_para_set* cur = OSTCBCur->OSTCBExtPtr;
                    task_para_set* next = OSTCBHighRdy->OSTCBExtPtr;

                    if (OSTCBHighRdy->OSTCBPrio != 63) {
                        printf("%d\tCompletion\ttask(%2d)(%2d)\ttask(%2d)(%2d)\t%d\t%d\t%d\t\n", OSTimeGet(), cur->TaskID, cur->job_no, next->TaskID,
                            next->job_no,cur->ResponseTime, cur->ResponseTime - cur->TaskExecutionTime,cur->Delay);
                        cur->job_no++;
                    }
                    else {
                        printf("%d\tCompletion\ttask(%2d)(%2d)\ttask(%2d)\t%d\t%d\t%d\n", OSTimeGet(), cur->TaskID, cur->job_no, OSTCBHighRdy->OSTCBPrio,
                            cur->ResponseTime, cur->ResponseTime - cur->TaskExecutionTime,cur->Delay);
                        cur->job_no++;
                    }
                    OSTCBCur->CompletedFlag = 0;
```

1. 當任務呼叫 OSTimeDly( )：任務主動放棄 cpu 使用權，狀態會被設為 suspend，並進入 OS_Sched()經過 OS_SchedNew()更新完 OSTCBHighRdy 的任務後，找出下個 HighRdy 的任務
2. 任務在**完成**情況下才會主動放棄 cpu，因此在此 printf Complete 訊息

[PART III] FIFO Scheduler Implementation [10%]

- The correctness of schedule results of examples (**Output.txt**). 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%)



```
void  OSIntExit(void)
{
#if OS_CRITICAL_METHOD == 3u                                   /* Allocate storage for CPU status register */
    OS_CPU_SR  cpu_sr = 0u;
#endif
    if (OSRunning == OS_TRUE) {
        INT16U currentTime;
        currentTime = OSTimeGet();
        OS_ENTER_CRITICAL();
        if (OSIntNesting > 0u) {                               /* Prevent OSIntNesting cur wrapping      */
            OSIntNesting--;
        }
        if (OSIntNesting == 0u) {                              /* Reschedule only if all ISRs complete ... */
            if (OSLockNesting == 0u) {                         /*      and not locked                    */
#if ALGORITHM == RM
                OS_SchedNew();
                OSTCBHighRdy = OSTCBPrioTbl[OSPrioHighRdy];
#endif
                if (OSPrioHighRdy != OSPrioCur) {              /* No Ctx Sw if current task is highest rdy */
#if OS_TASK_PROFILE_EN > 0u
                    OSTCBHighRdy->OSTCBCtxSwCtr++;             /* Inc. # of context switches to this task  */
#endif
                    OSCtxSwCtr++;                              /* Keep track of the number of ctx switches */
                    task_para_set* cur = OSTCBCur->OSTCBExtPtr;
                    task_para_set* next = OSTCBHighRdy->OSTCBExtPtr;
```

- OSIntExit() :

  1. FIFO 是 non-preemptive kernel 因此在 OSIntExit()不做 OS_SchedNew()

     - 問題 ：idle task 沒辦法被 preempt，系統一旦進入 idle 就會一直卡在 idle task