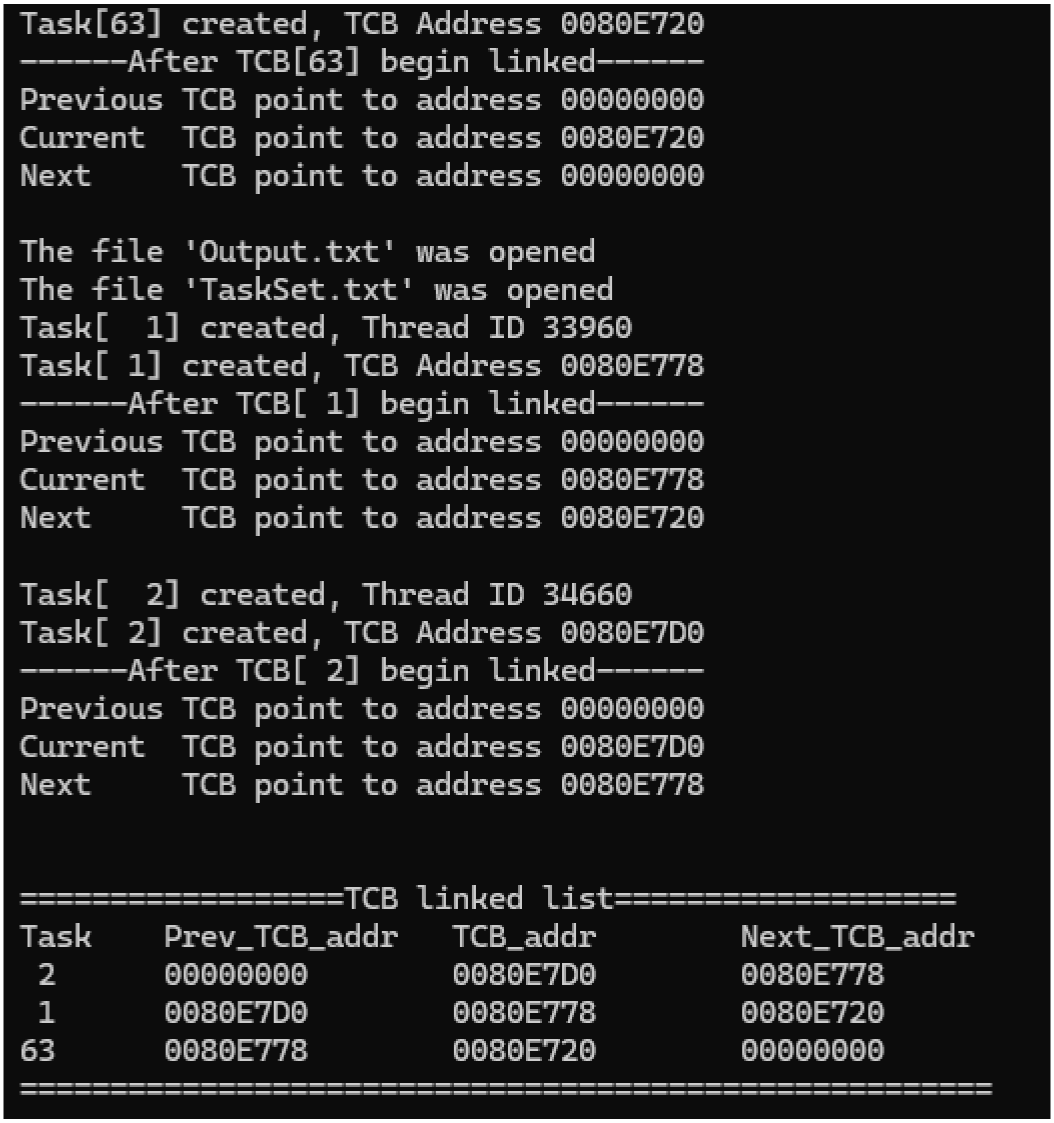
PART I

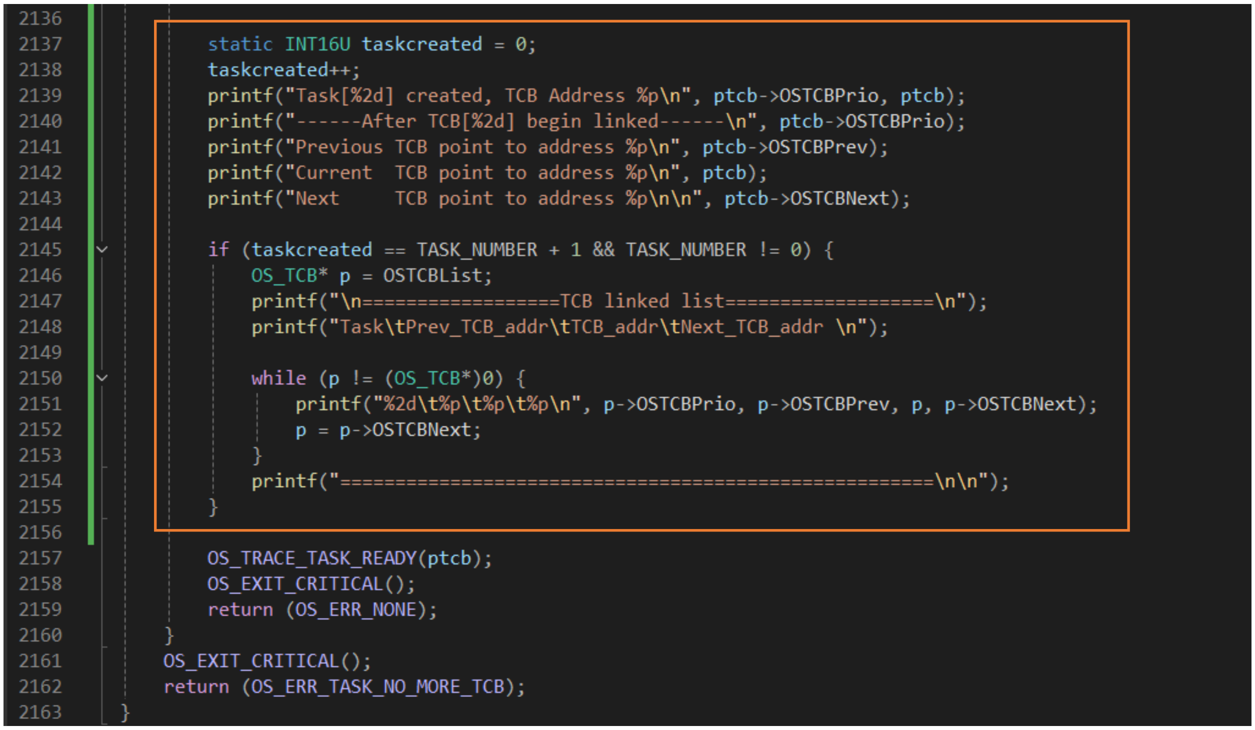
TCB 創建時記憶體資訊



```
Task[63] created, TCB Address 0080E720
------After TCB[63] begin linked------
Previous TCB point to address 00000000
Current  TCB point to address 0080E720
Next     TCB point to address 00000000

The file 'Output.txt' was opened
The file 'TaskSet.txt' was opened
Task[  1] created, Thread ID 33960
Task[ 1] created, TCB Address 0080E778
------After TCB[ 1] begin linked------
Previous TCB point to address 00000000
Current  TCB point to address 0080E778
Next     TCB point to address 0080E720

Task[  2] created, Thread ID 34660
Task[ 2] created, TCB Address 0080E7D0
------After TCB[ 2] begin linked------
Previous TCB point to address 00000000
Current  TCB point to address 0080E7D0
Next     TCB point to address 0080E778


==================TCB linked list===================
Task    Prev_TCB_addr   TCB_addr        Next_TCB_addr
 2      00000000        0080E7D0        0080E778
 1      0080E7D0        0080E778        0080E720
63      0080E778        0080E720        00000000
=====================================================
```

修改部分

函式 OS_TCBInit()的末端:

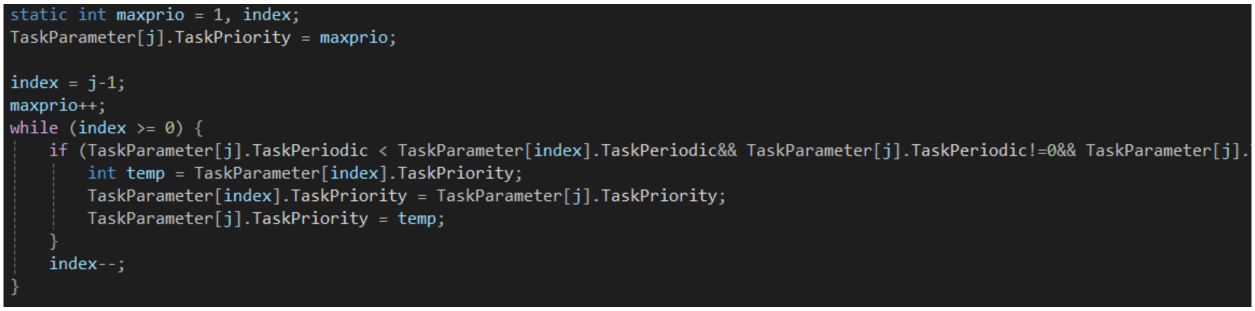


```c
        static INT16U taskcreated = 0;
        taskcreated++;
        printf("Task[%2d] created, TCB Address %p\n", ptcb->OSTCBPrio, ptcb);
        printf("------After TCB[%2d] begin linked------\n", ptcb->OSTCBPrio);
        printf("Previous TCB point to address %p\n", ptcb->OSTCBPrev);
        printf("Current  TCB point to address %p\n", ptcb);
        printf("Next     TCB point to address %p\n\n", ptcb->OSTCBNext);

        if (taskcreated == TASK_NUMBER + 1 && TASK_NUMBER != 0) {
            OS_TCB* p = OSTCBList;
            printf("\n=================TCB linked list==================\n");
            printf("Task\tPrev_TCB_addr\tTCB_addr\tNext_TCB_addr \n");

            while (p != (OS_TCB*)0) {
                printf("%2d\t%p\t%p\t%p\n", p->OSTCBPrio, p->OSTCBPrev, p, p->OSTCBNext);
                p = p->OSTCBNext;
            }
            printf("=====================================================\n\n");
        }

        OS_TRACE_TASK_READY(ptcb);
        OS_EXIT_CRITICAL();
        return (OS_ERR_NONE);
    }
    OS_EXIT_CRITICAL();
    return (OS_ERR_TASK_NO_MORE_TCB);
}
```

這裡我使用 task 創建函式 OS_TCBInit()中所使用的 ptcb 來擷取資訊，因為它包含了所需的 TCB 資料，能指出前後 TCB 與自身位址，再用 taskcreated 變數來計算是否創立完所有 TCB，最後輸出所有 TCB 的 Linked list

PART II

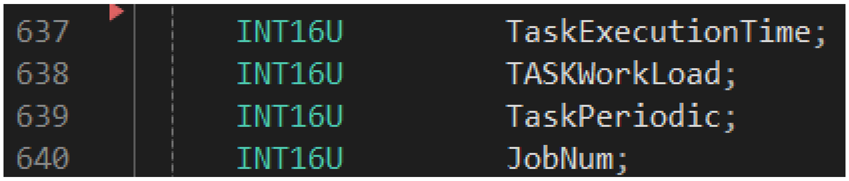
App_hooks.c:



```c
static int maxprio = 1, index;
TaskParameter[j].TaskPriority = maxprio;

index = j-1;
maxprio++;
while (index >= 0) {
    if (TaskParameter[j].TaskPeriodic < TaskParameter[index].TaskPeriodic&& TaskParameter[j].TaskPeriodic!=0&& TaskParameter[j].
        int temp = TaskParameter[index].TaskPriority;
        TaskParameter[index].TaskPriority = TaskParameter[j].TaskPriority;
        TaskParameter[j].TaskPriority = temp;
    }
    index--;
}
```

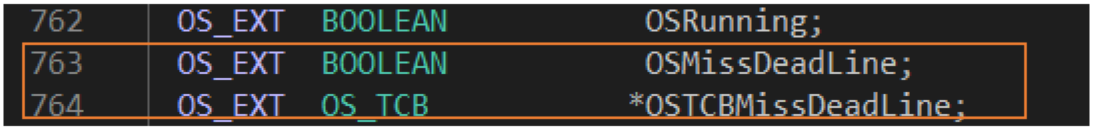
在上次 Lab 的 InputFile()中新增此段程式來重新依據 Periodic 排列 Priority

Ucos_ii.h:



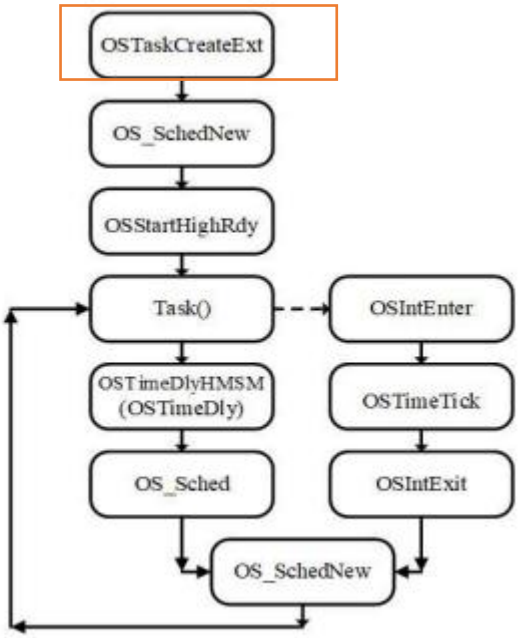
```
    INT16U          TaskExecutionTime;
    INT16U          TASKWorkLoad;
    INT16U          TaskPeriodic;
    INT16U          JobNum;
```

在 OS_TCB 的 STRUCT 宣告中新增參數，TASKWorkLoad 和 JobNum 分別是該 Job 剩餘工作量和第幾個 Job

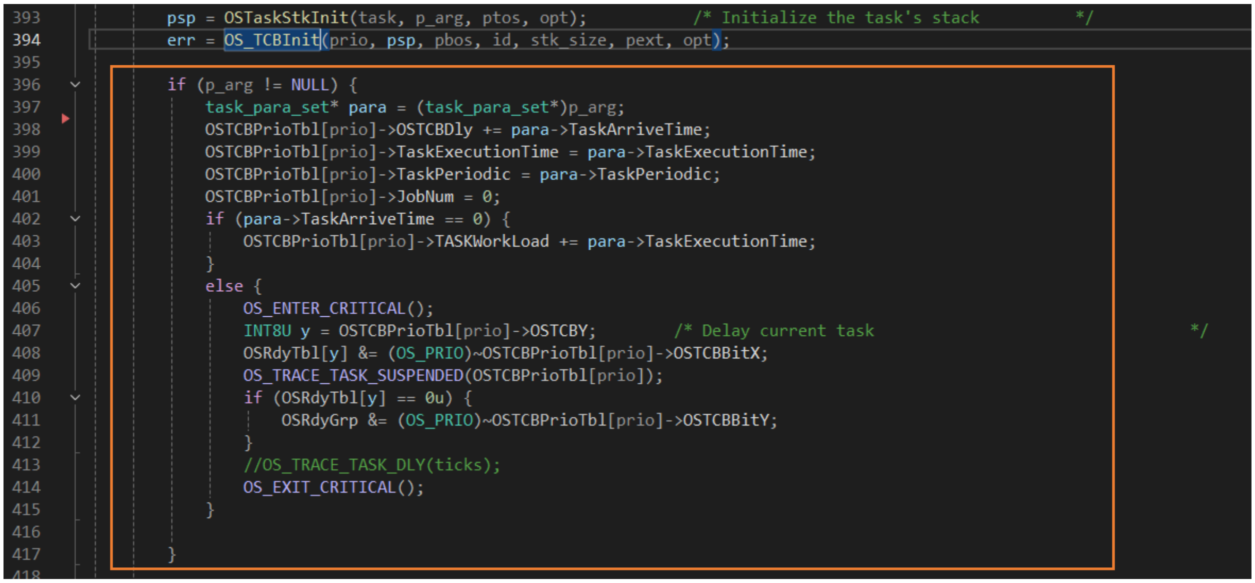


```
OS_EXT  BOOLEAN          OSRunning;
OS_EXT  BOOLEAN          OSMissDeadLine;
OS_EXT  OS_TCB          *OSTCBMissDeadLine;
```

新增 GLOBAL VARIABLES，是否 MissDeadLine 和 Miss 的 TCB

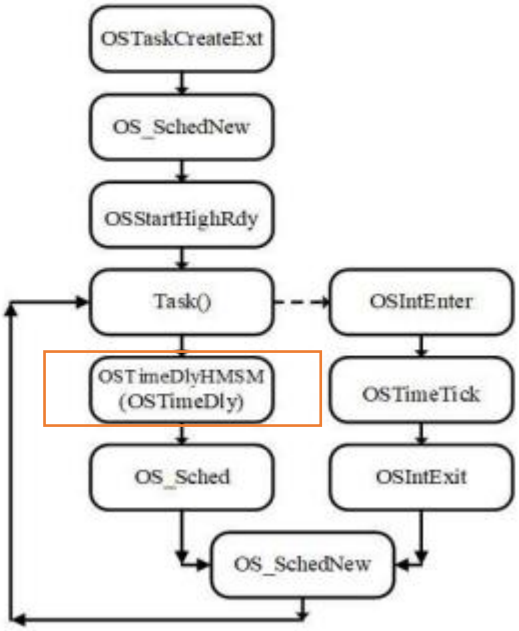


Example Flowchart

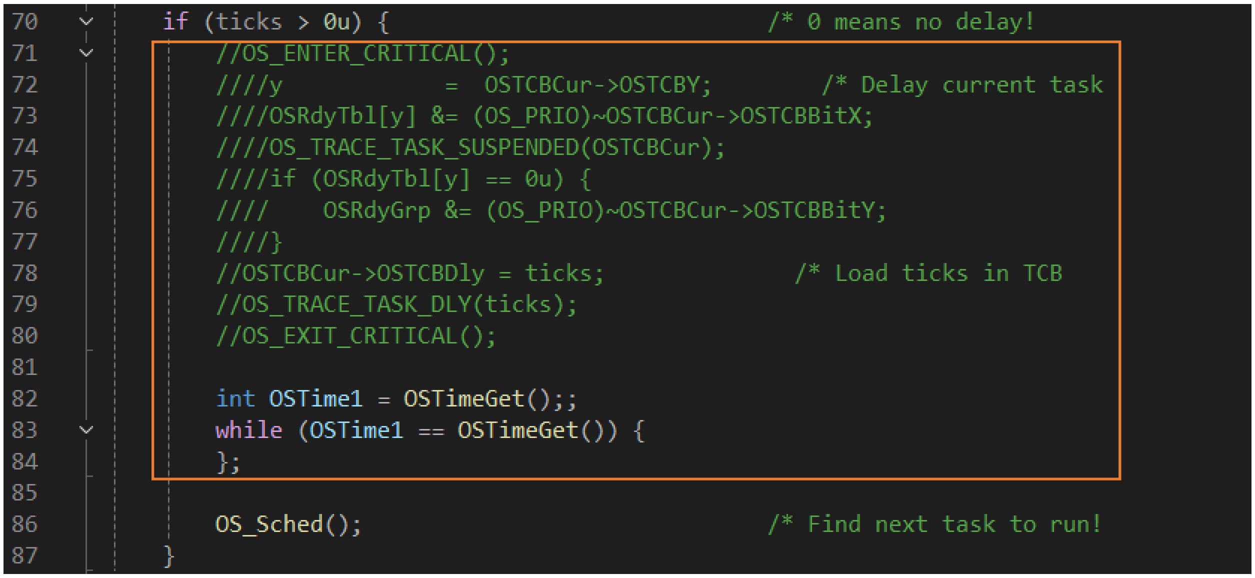


```
        psp = OSTaskStkInit(task, p_arg, ptos, opt);            /* Initialize the task's stack          */
        err = OS_TCBInit(prio, psp, pbos, id, stk_size, pext, opt);

        if (p_arg != NULL) {
            task_para_set* para = (task_para_set*)p_arg;
            OSTCBPrioTbl[prio]->OSTCBDly += para->TaskArriveTime;
            OSTCBPrioTbl[prio]->TaskExecutionTime = para->TaskExecutionTime;
            OSTCBPrioTbl[prio]->TaskPeriodic = para->TaskPeriodic;
            OSTCBPrioTbl[prio]->JobNum = 0;
            if (para->TaskArriveTime == 0) {
                OSTCBPrioTbl[prio]->TASKWorkLoad += para->TaskExecutionTime;
            }
            else {
                OS_ENTER_CRITICAL();
                INT8U y = OSTCBPrioTbl[prio]->OSTCBY;        /* Delay current task                                  */
                OSRdyTbl[y] &= (OS_PRIO)~OSTCBPrioTbl[prio]->OSTCBBitX;
                OS_TRACE_TASK_SUSPENDED(OSTCBPrioTbl[prio]);
                if (OSRdyTbl[y] == 0u) {
                    OSRdyGrp &= (OS_PRIO)~OSTCBPrioTbl[prio]->OSTCBBitY;
                }
                //OS_TRACE_TASK_DLY(ticks);
                OS_EXIT_CRITICAL();
            }

        }
```

OSTaskCreateExt 中，為 TASK 初始化一些多的參數，主要是運用 TASKSET.TXT 的資料，OSTCBDly 會加上抵達時間的延遲，若有延遲，會將 RdyTbl 該 TASK 設為未就緒，沒有的話會直些賦予工作量。

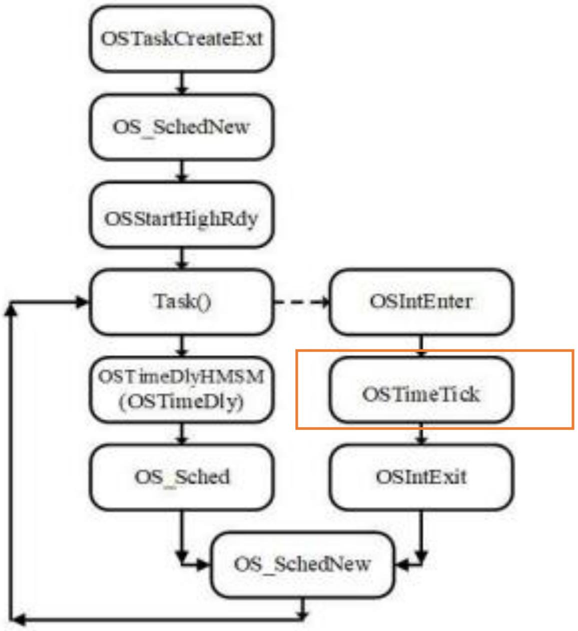


Example Flowchart

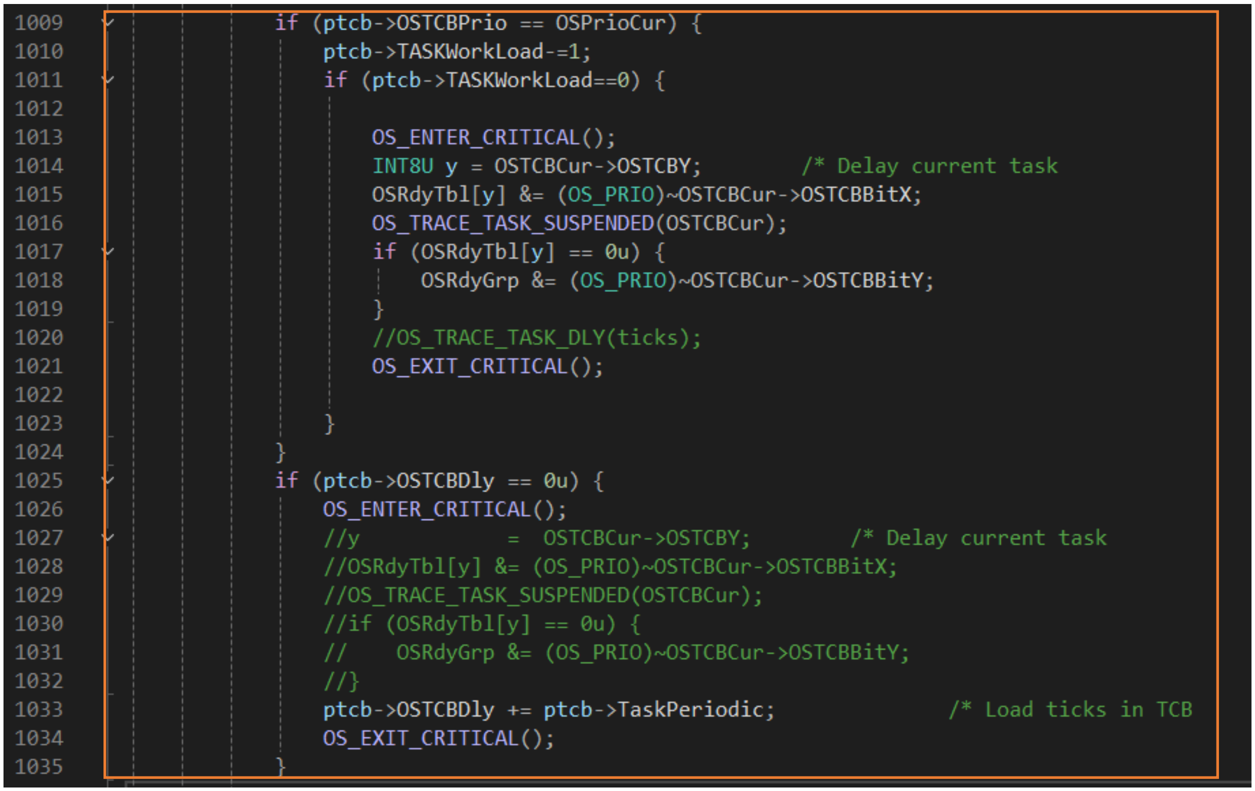


```
    if (ticks > 0u) {                            /* 0 means no delay!
        //OS_ENTER_CRITICAL();
        ////y            =  OSTCBCur->OSTCBY;        /* Delay current task
        ////OSRdyTbl[y] &= (OS_PRIO)~OSTCBCur->OSTCBBitX;
        ////OS_TRACE_TASK_SUSPENDED(OSTCBCur);
        ////if (OSRdyTbl[y] == 0u) {
        ////    OSRdyGrp &= (OS_PRIO)~OSTCBCur->OSTCBBitY;
        ////}
        //OSTCBCur->OSTCBDly = ticks;              /* Load ticks in TCB
        //OS_TRACE_TASK_DLY(ticks);
        //OS_EXIT_CRITICAL();

        int OSTime1 = OSTimeGet();;
        while (OSTime1 == OSTimeGet()) {
        };

        OS_Sched();                              /* Find next task to run!
    }
```

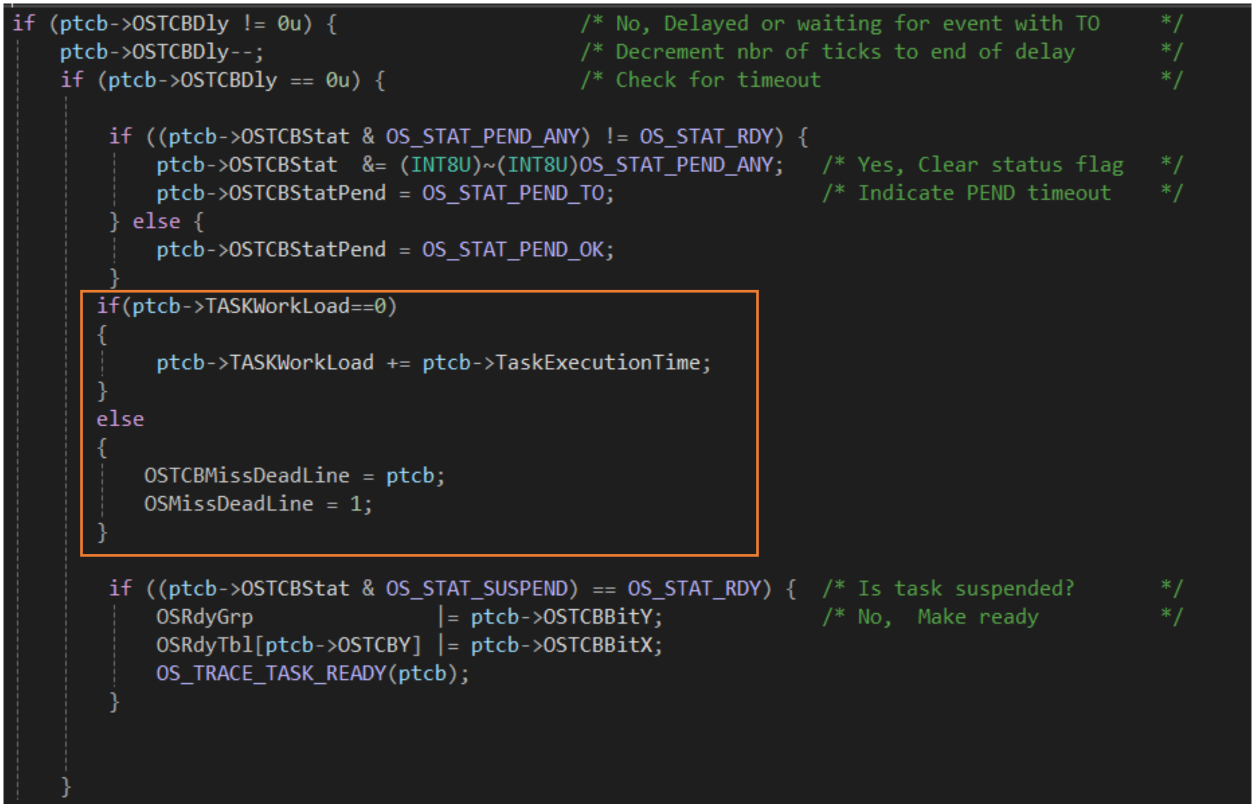
OSTimeDly 本來被 TASK 使用後會讓其進入非預備狀態並 OS_sched()，這裡我用 OSTimeGet()來讓他卡著直到下一 Tick 到來，並將相關排程事宜移到 OSTimeTick()。



Example Flowchart

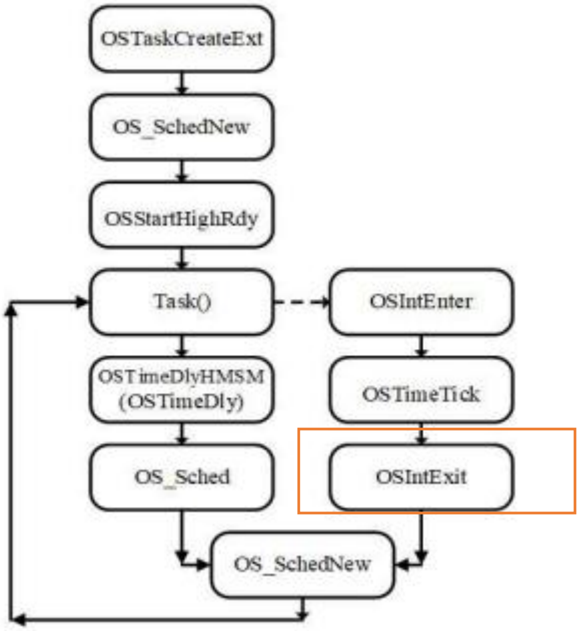


```
if (ptcb->OSTCBPrio == OSPrioCur) {
    ptcb->TASKWorkLoad-=1;
    if (ptcb->TASKWorkLoad==0) {

        OS_ENTER_CRITICAL();
        INT8U 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;
        }
        //OS_TRACE_TASK_DLY(ticks);
        OS_EXIT_CRITICAL();

    }
}
if (ptcb->OSTCBDly == 0u) {
    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;
    //}
    ptcb->OSTCBDly += ptcb->TaskPeriodic;              /* Load ticks in TCB
    OS_EXIT_CRITICAL();
}
```

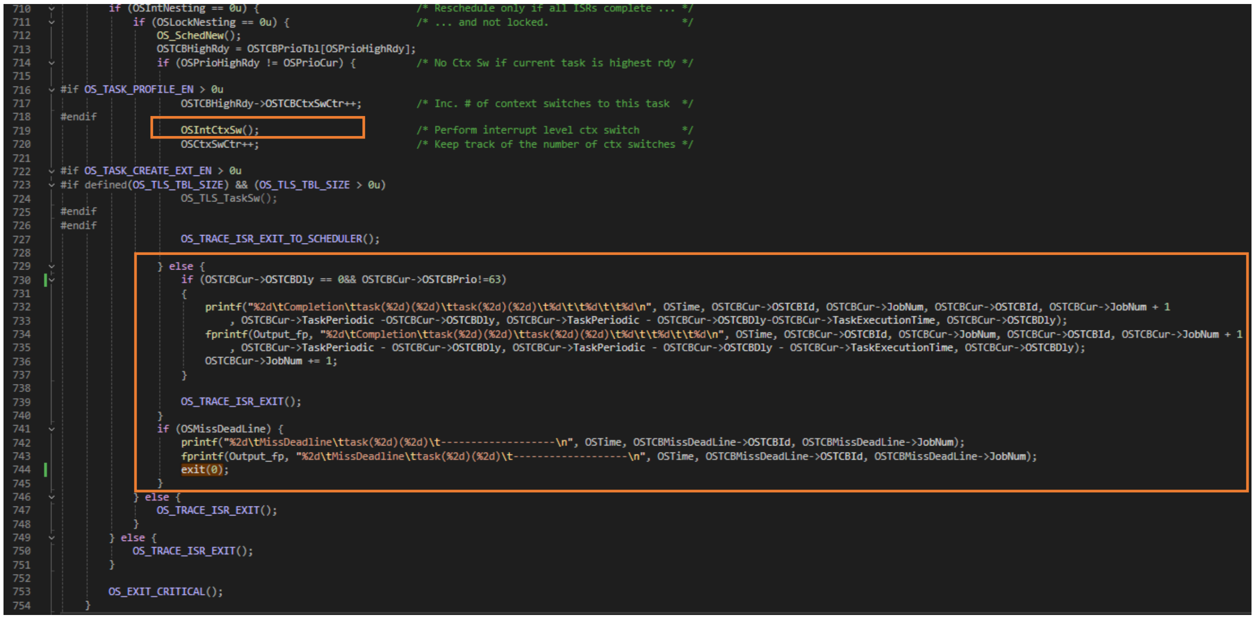


```
if (ptcb->OSTCBDly != 0u) {                    /* No, Delayed or waiting for event with TO      */
    ptcb->OSTCBDly--;                          /* Decrement nbr of ticks to end of delay        */
    if (ptcb->OSTCBDly == 0u) {                /* Check for timeout                             */

        if ((ptcb->OSTCBStat & OS_STAT_PEND_ANY) != OS_STAT_RDY) {
            ptcb->OSTCBStat  &= (INT8U)~(INT8U)OS_STAT_PEND_ANY;   /* Yes, Clear status flag    */
            ptcb->OSTCBStatPend = OS_STAT_PEND_TO;                  /* Indicate PEND timeout     */
        } else {
            ptcb->OSTCBStatPend = OS_STAT_PEND_OK;
        }
       if(ptcb->TASKWorkLoad==0)
       {
            ptcb->TASKWorkLoad += ptcb->TaskExecutionTime;
       }
       else
       {
           OSTCBMissDeadLine = ptcb;
           OSMissDeadLine = 1;
       }

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


    }
```

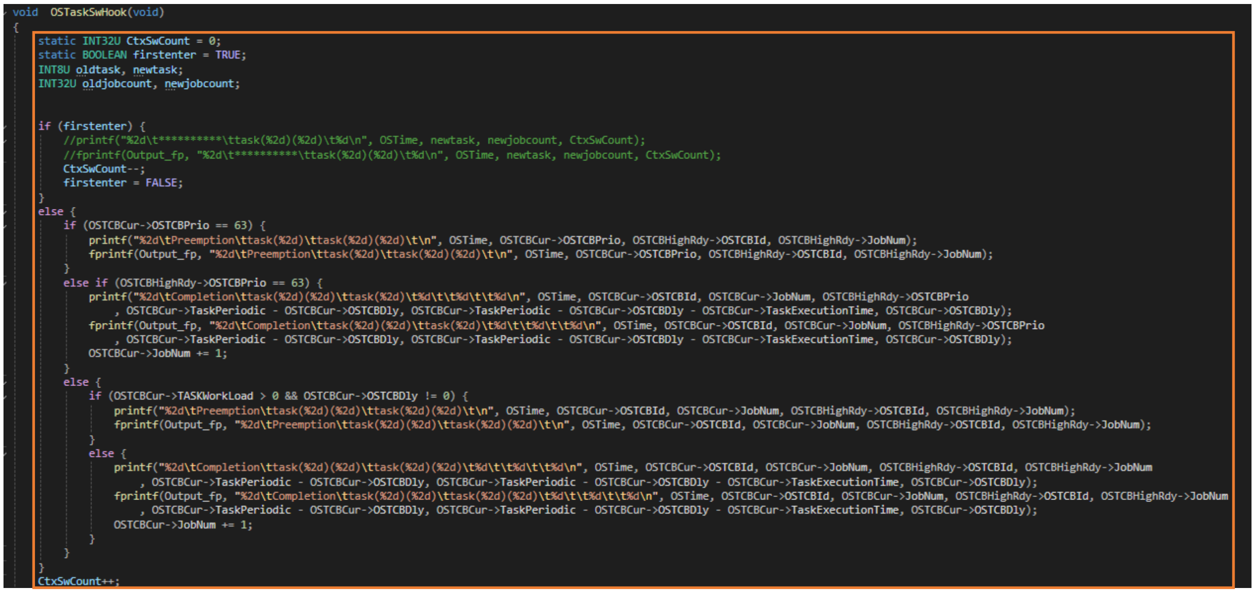
這裡我用 TASKWorkLoad 來判斷該 JOB 工作是否完成，如果變 0 則將其設為非預備狀態，OSTCBDLY 則移到歸零時累加週期時間當 DEADLINE 並新增工作，若時間到但工作量未完成則視為 MissDeadLine。



Example Flowchart



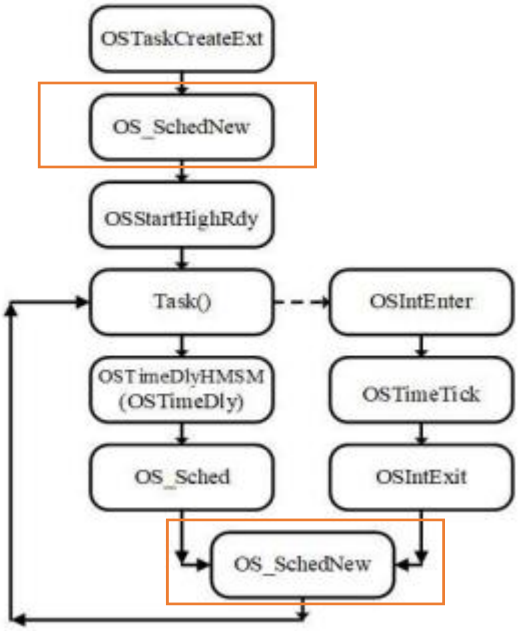
```
        if (OSIntNesting == 0u) {                          /* Reschedule only if all ISRs complete ... */
            if (OSLockNesting == 0u) {                     /* ... and not locked.                      */
                OS_SchedNew();
                OSTCBHighRdy = OSTCBPrioTbl[OSPrioHighRdy];
                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
                    OSIntCtxSw();                          /* Perform interrupt level ctx switch       */
                    OSCtxSwCtr++;                          /* Keep track of the number of ctx switches */

#if OS_TASK_CREATE_EXT_EN > 0u
#if defined(OS_TLS_TBL_SIZE) && (OS_TLS_TBL_SIZE > 0u)
                    OS_TLS_TaskSw();
#endif
#endif

                    OS_TRACE_ISR_EXIT_TO_SCHEDULER();

                } else {
                    if (OSTCBCur->OSTCBDly == 0&& OSTCBCur->OSTCBPrio!=63)
                    {
                        printf("%2d\tCompletion\ttask(%2d)(%2d)\ttask(%2d)(%2d)\t%d\t\t%d\t\t%d\n", OSTime, OSTCBCur->OSTCBId, OSTCBCur->JobNum, OSTCBCur->OSTCBId, OSTCBCur->JobNum + 1
                            , OSTCBCur->TaskPeriodic -OSTCBCur->OSTCBDly, OSTCBCur->TaskPeriodic - OSTCBCur->OSTCBDly-OSTCBCur->TaskExecutionTime, OSTCBCur->OSTCBDly);
                        fprintf(Output_fp, "%2d\tCompletion\ttask(%2d)(%2d)\ttask(%2d)(%2d)\t%d\t\t%d\t\t%d\n", OSTime, OSTCBCur->OSTCBId, OSTCBCur->JobNum, OSTCBCur->OSTCBId, OSTCBCur->JobNum + 1
                            , OSTCBCur->TaskPeriodic - OSTCBCur->OSTCBDly, OSTCBCur->TaskPeriodic - OSTCBCur->OSTCBDly - OSTCBCur->TaskExecutionTime, OSTCBCur->OSTCBDly);
                        OSTCBCur->JobNum += 1;
                    }

                    OS_TRACE_ISR_EXIT();
                }
                if (OSMissDeadLine) {
                    printf("%2d\tMissDeadline\ttask(%2d)(%2d)\t------------------\n", OSTime, OSTCBMissDeadLine->OSTCBId, OSTCBMissDeadLine->JobNum);
                    fprintf(Output_fp, "%2d\tMissDeadline\ttask(%2d)(%2d)\t------------------\n", OSTime, OSTCBMissDeadLine->OSTCBId, OSTCBMissDeadLine->JobNum);
                    exit(0);
                }
            } else {
                OS_TRACE_ISR_EXIT();
            }
        } else {
            OS_TRACE_ISR_EXIT();
        }

        OS_EXIT_CRITICAL();
    }
```



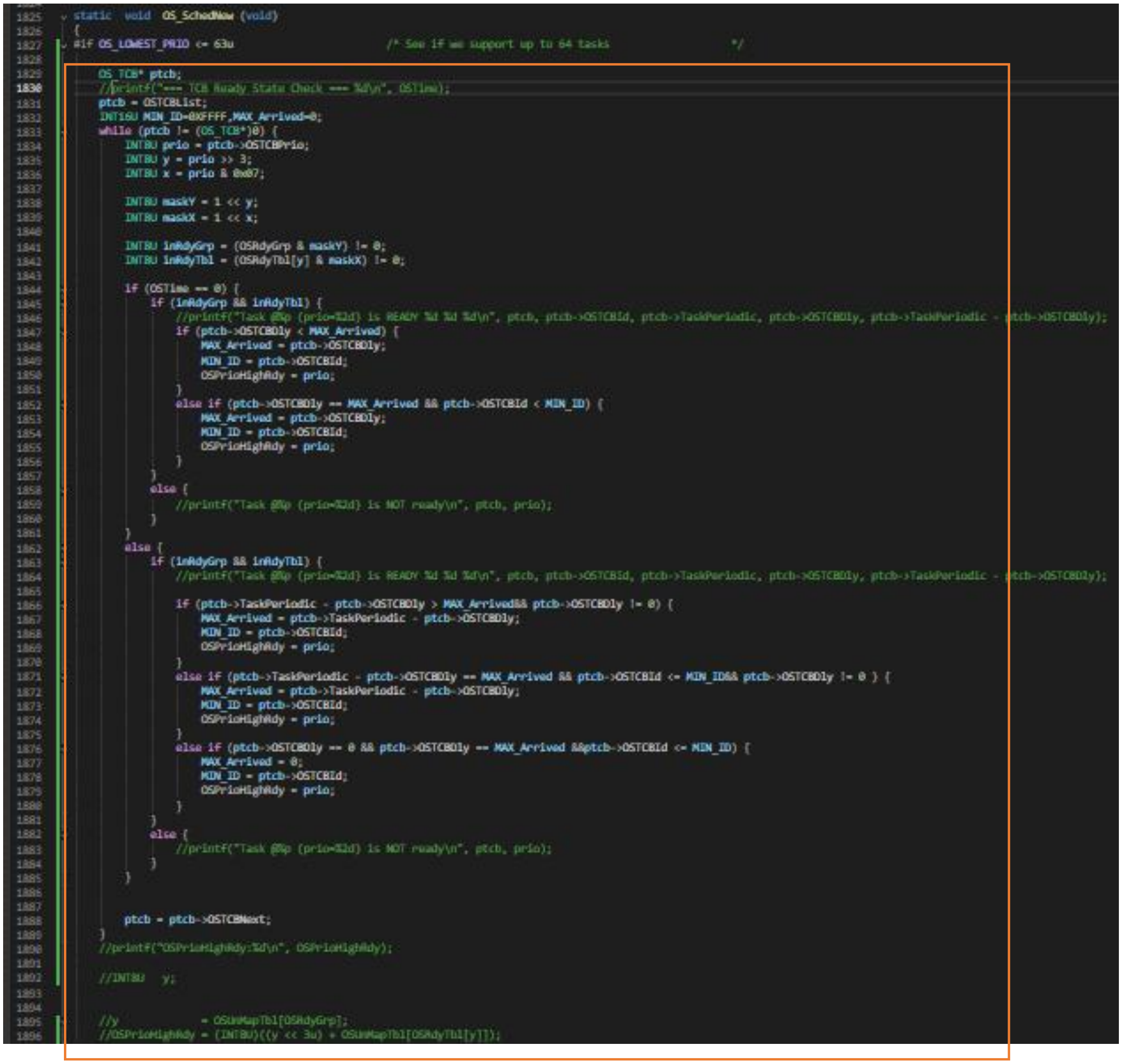
```
void  OSTaskSwHook(void)
{
    static INT32U CtxSwCount = 0;
    static BOOLEAN firstenter = TRUE;
    INT8U oldtask, newtask;
    INT32U oldjobcount, newjobcount;


    if (firstenter) {
        //printf("%2d\t**********\ttask(%2d)(%2d)\t%d\n", OSTime, newtask, newjobcount, CtxSwCount);
        //fprintf(Output_fp, "%2d\t**********\ttask(%2d)(%2d)\t%d\n", OSTime, newtask, newjobcount, CtxSwCount);
        CtxSwCount--;
        firstenter = FALSE;
    }
    else {
        if (OSTCBCur->OSTCBPrio == 63) {
            printf("%2d\tPreemption\ttask(%2d)\ttask(%2d)(%2d)\t\n", OSTime, OSTCBCur->OSTCBPrio, OSTCBHighRdy->OSTCBId, OSTCBHighRdy->JobNum);
            fprintf(Output_fp, "%2d\tPreemption\ttask(%2d)\ttask(%2d)(%2d)\t\n", OSTime, OSTCBCur->OSTCBPrio, OSTCBHighRdy->OSTCBId, OSTCBHighRdy->JobNum);
        }
        else if (OSTCBHighRdy->OSTCBPrio == 63) {
            printf("%2d\tCompletion\ttask(%2d)(%2d)\ttask(%2d)\t%d\t\t%d\t\t%d\n", OSTime, OSTCBCur->OSTCBId, OSTCBCur->JobNum, OSTCBHighRdy->OSTCBPrio
                , OSTCBCur->TaskPeriodic - OSTCBCur->OSTCBDly, OSTCBCur->TaskPeriodic - OSTCBCur->OSTCBDly - OSTCBCur->TaskExecutionTime, OSTCBCur->OSTCBDly);
            fprintf(Output_fp, "%2d\tCompletion\ttask(%2d)(%2d)\ttask(%2d)\t%d\t\t%d\t\t%d\n", OSTime, OSTCBCur->OSTCBId, OSTCBCur->JobNum, OSTCBHighRdy->OSTCBPrio
                , OSTCBCur->TaskPeriodic - OSTCBCur->OSTCBDly, OSTCBCur->TaskPeriodic - OSTCBCur->OSTCBDly - OSTCBCur->TaskExecutionTime, OSTCBCur->OSTCBDly);
            OSTCBCur->JobNum += 1;
        }
        else {
            if (OSTCBCur->TASKWorkLoad > 0 && OSTCBCur->OSTCBDly != 0) {
                printf("%2d\tPreemption\ttask(%2d)(%2d)\ttask(%2d)(%2d)\t\n", OSTime, OSTCBCur->OSTCBId, OSTCBCur->JobNum, OSTCBHighRdy->OSTCBId, OSTCBHighRdy->JobNum);
                fprintf(Output_fp, "%2d\tPreemption\ttask(%2d)(%2d)\ttask(%2d)(%2d)\t\n", OSTime, OSTCBCur->OSTCBId, OSTCBCur->JobNum, OSTCBHighRdy->OSTCBId, OSTCBHighRdy->JobNum);
            }
            else {
                printf("%2d\tCompletion\ttask(%2d)(%2d)\ttask(%2d)(%2d)\t%d\t\t%d\t\t%d\n", OSTime, OSTCBCur->OSTCBId, OSTCBCur->JobNum, OSTCBHighRdy->OSTCBId, OSTCBHighRdy->JobNum
                    , OSTCBCur->TaskPeriodic - OSTCBCur->OSTCBDly, OSTCBCur->TaskPeriodic - OSTCBCur->OSTCBDly - OSTCBCur->TaskExecutionTime, OSTCBCur->OSTCBDly);
                fprintf(Output_fp, "%2d\tCompletion\ttask(%2d)(%2d)\ttask(%2d)(%2d)\t%d\t\t%d\t\t%d\n", OSTime, OSTCBCur->OSTCBId, OSTCBCur->JobNum, OSTCBHighRdy->OSTCBId, OSTCBHighRdy->JobNum
                    , OSTCBCur->TaskPeriodic - OSTCBCur->OSTCBDly, OSTCBCur->TaskPeriodic - OSTCBCur->OSTCBDly - OSTCBCur->TaskExecutionTime, OSTCBCur->OSTCBDly);
                OSTCBCur->JobNum += 1;
            }
        }
    }
    CtxSwCount++;
```

OSTimetick 執行完後，查看新舊 TASK 是不是相同且剛好舊的週期歸零，是的話輸出 TASK 完成並執行新 JOB，如果新舊不同則進入 OSIntCtxSw()，裡面會 Call OSTaskSwHook()，這裡會依據切換 TASK 時的狀況顯示是 Completion 還是被 Preemption，最後查看有沒有 MissDeadLine，有的話暫停程序並退出。

PART III



Example Flowchart

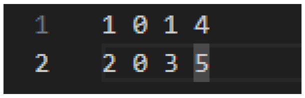


```
static  void  OS_SchedNew (void)
{
#if OS_LOWEST_PRIO <= 63u                        /* See if we support up to 64 tasks              */

    OS_TCB* ptcb;
    //printf("=== TCB Ready State Check === %d\n", OSTime);
    ptcb = OSTCBList;
    INT16U MIN_ID=0XFFFF,MAX_Arrived=0;
    while (ptcb != (OS_TCB*)0) {
        INT8U prio = ptcb->OSTCBPrio;
        INT8U y = prio >> 3;
        INT8U x = prio & 0x07;

        INT8U maskY = 1 << y;
        INT8U maskX = 1 << x;

        INT8U inRdyGrp = (OSRdyGrp & maskY) != 0;
        INT8U inRdyTbl = (OSRdyTbl[y] & maskX) != 0;

        if (OSTime == 0) {
            if (inRdyGrp && inRdyTbl) {
                //printf("Task @%p (prio=%2d) is READY %d %d %d\n", ptcb, ptcb->OSTCBId, ptcb->TaskPeriodic, ptcb->OSTCBDly, ptcb->TaskPeriodic - ptcb->OSTCBDly);
                if (ptcb->OSTCBDly < MAX_Arrived) {
                    MAX_Arrived = ptcb->OSTCBDly;
                    MIN_ID = ptcb->OSTCBId;
                    OSPrioHighRdy = prio;
                }
                else if (ptcb->OSTCBDly == MAX_Arrived && ptcb->OSTCBId < MIN_ID) {
                    MAX_Arrived = ptcb->OSTCBDly;
                    MIN_ID = ptcb->OSTCBId;
                    OSPrioHighRdy = prio;
                }
            }
            else {
                //printf("Task @%p (prio=%2d) is NOT ready\n", ptcb, prio);
            }
        }
        else {
            if (inRdyGrp && inRdyTbl) {
                //printf("Task @%p (prio=%2d) is READY %d %d %d\n", ptcb, ptcb->OSTCBId, ptcb->TaskPeriodic, ptcb->OSTCBDly, ptcb->TaskPeriodic - ptcb->OSTCBDly);

                if (ptcb->TaskPeriodic - ptcb->OSTCBDly > MAX_Arrived&& ptcb->OSTCBDly != 0) {
                    MAX_Arrived = ptcb->TaskPeriodic - ptcb->OSTCBDly;
                    MIN_ID = ptcb->OSTCBId;
                    OSPrioHighRdy = prio;
                }
                else if (ptcb->TaskPeriodic - ptcb->OSTCBDly == MAX_Arrived && ptcb->OSTCBId <= MIN_ID&& ptcb->OSTCBDly != 0 ) {
                    MAX_Arrived = ptcb->TaskPeriodic - ptcb->OSTCBDly;
                    MIN_ID = ptcb->OSTCBId;
                    OSPrioHighRdy = prio;
                }
                else if (ptcb->OSTCBDly == 0 && ptcb->OSTCBDly == MAX_Arrived &&ptcb->OSTCBId <= MIN_ID) {
                    MAX_Arrived = 0;
                    MIN_ID = ptcb->OSTCBId;
                    OSPrioHighRdy = prio;
                }
            }
            else {
                //printf("Task @%p (prio=%2d) is NOT ready\n", ptcb, prio);
            }
        }


        ptcb = ptcb->OSTCBNext;
    }
    //printf("OSPrioHighRdy:%d\n", OSPrioHighRdy);

    //INT8U   y;


    //y             = OSUnMapTbl[OSRdyGrp];
    //OSPrioHighRdy = (INT8U)((y << 3u) + OSUnMapTbl[OSRdyTbl[y]]);
```

FIFO 我主要用剛剛 RM 的專案稍微修改 OS_SchedNew 的部分，我將選擇最高優先度 TASK 的方式改為檢查哪個 TASK 已抵達的時間最長，並會在 TASK 剛進入新周期和系統剛啟動時做特殊判斷

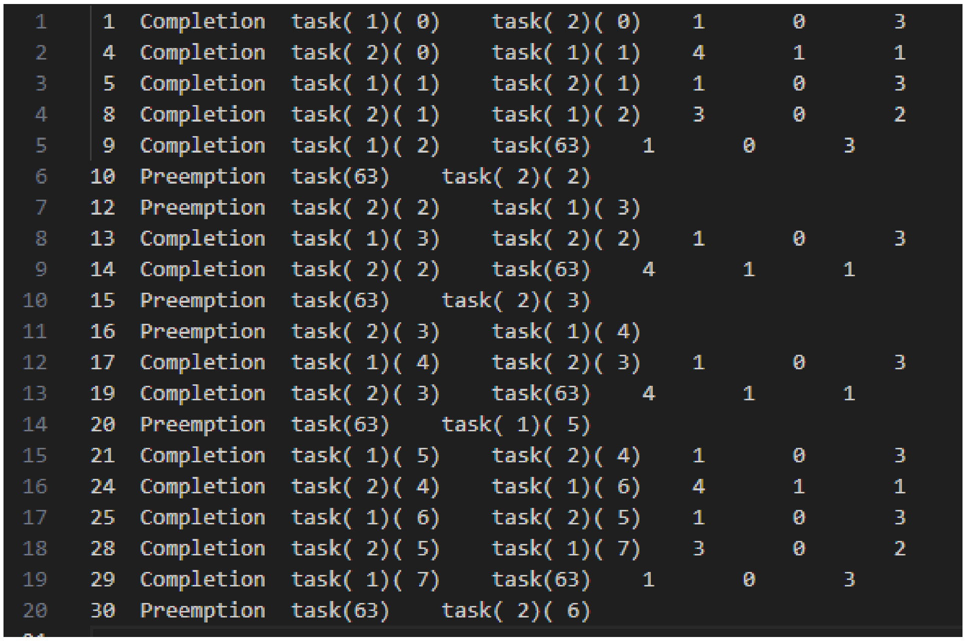
RM VS FIFO

測資



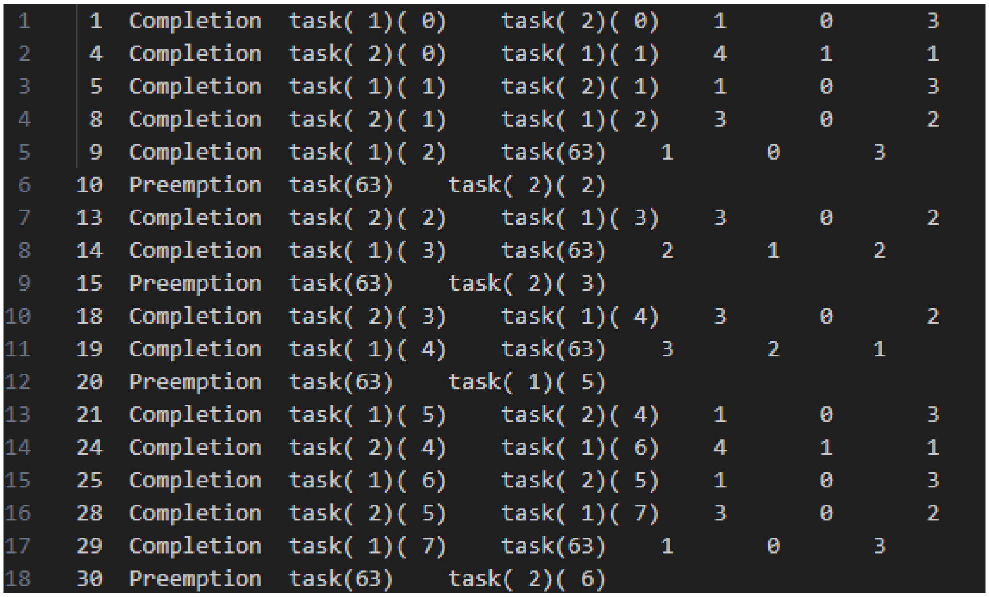
```
1 0 1 4
2 0 3 5
```

RM:



```
 1  Completion  task( 1)( 0)    task( 2)( 0)    1       0       3
 4  Completion  task( 2)( 0)    task( 1)( 1)    4       1       1
 5  Completion  task( 1)( 1)    task( 2)( 1)    1       0       3
 8  Completion  task( 2)( 1)    task( 1)( 2)    3       0       2
 9  Completion  task( 1)( 2)    task(63)    1       0       3
10  Preemption  task(63)    task( 2)( 2)
12  Preemption  task( 2)( 2)    task( 1)( 3)
13  Completion  task( 1)( 3)    task( 2)( 2)    1       0       3
14  Completion  task( 2)( 2)    task(63)    4       1       1
15  Preemption  task(63)    task( 2)( 3)
16  Preemption  task( 2)( 3)    task( 1)( 4)
17  Completion  task( 1)( 4)    task( 2)( 3)    1       0       3
19  Completion  task( 2)( 3)    task(63)    4       1       1
20  Preemption  task(63)    task( 1)( 5)
21  Completion  task( 1)( 5)    task( 2)( 4)    1       0       3
24  Completion  task( 2)( 4)    task( 1)( 6)    4       1       1
25  Completion  task( 1)( 6)    task( 2)( 5)    1       0       3
28  Completion  task( 2)( 5)    task( 1)( 7)    3       0       2
29  Completion  task( 1)( 7)    task(63)    1       0       3
30  Preemption  task(63)    task( 2)( 6)
```

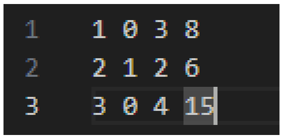
FIFO:



```
 1  Completion  task( 1)( 0)    task( 2)( 0)    1       0       3
 4  Completion  task( 2)( 0)    task( 1)( 1)    4       1       1
 5  Completion  task( 1)( 1)    task( 2)( 1)    1       0       3
 8  Completion  task( 2)( 1)    task( 1)( 2)    3       0       2
 9  Completion  task( 1)( 2)    task(63)    1       0       3
10  Preemption  task(63)    task( 2)( 2)
13  Completion  task( 2)( 2)    task( 1)( 3)    3       0       2
14  Completion  task( 1)( 3)    task(63)    2       1       2
15  Preemption  task(63)    task( 2)( 3)
18  Completion  task( 2)( 3)    task( 1)( 4)    3       0       2
19  Completion  task( 1)( 4)    task(63)    3       2       1
20  Preemption  task(63)    task( 1)( 5)
21  Completion  task( 1)( 5)    task( 2)( 4)    1       0       3
24  Completion  task( 2)( 4)    task( 1)( 6)    4       1       1
25  Completion  task( 1)( 6)    task( 2)( 5)    1       0       3
28  Completion  task( 2)( 5)    task( 1)( 7)    3       0       2
29  Completion  task( 1)( 7)    task(63)    1       0       3
30  Preemption  task(63)    task( 2)( 6)
```

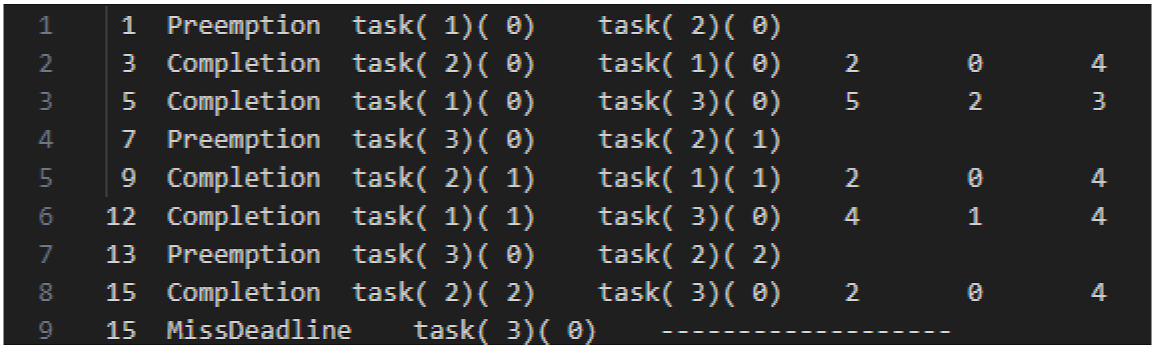
兩者都成功

測資



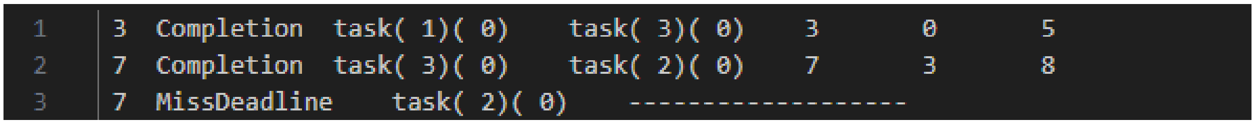
```
1 0 3 8
2 1 2 6
3 0 4 15
```

RM:



```
 1  Preemption  task( 1)( 0)    task( 2)( 0)
 3  Completion  task( 2)( 0)    task( 1)( 0)    2       0       4
 5  Completion  task( 1)( 0)    task( 3)( 0)    5       2       3
 7  Preemption  task( 3)( 0)    task( 2)( 1)
 9  Completion  task( 2)( 1)    task( 1)( 1)    2       0       4
12  Completion  task( 1)( 1)    task( 3)( 0)    4       1       4
13  Preemption  task( 3)( 0)    task( 2)( 2)
15  Completion  task( 2)( 2)    task( 3)( 0)    2       0       4
15  MissDeadline    task( 3)( 0)    -------------------
```

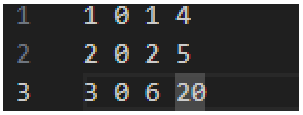
FIFO



```
3  Completion  task( 1)( 0)    task( 3)( 0)    3       0       5
7  Completion  task( 3)( 0)    task( 2)( 0)    7       3       8
7  MissDeadline    task( 2)( 0)    ------------------
```

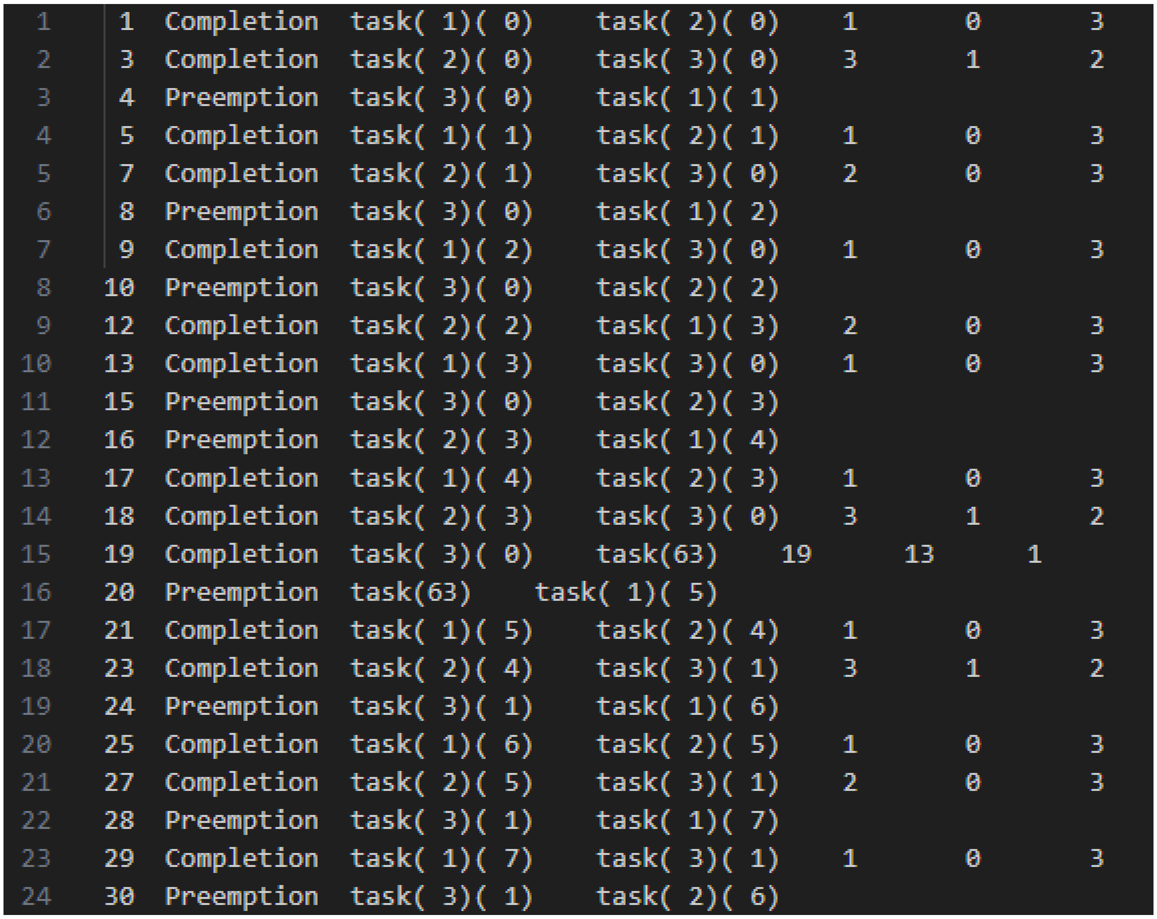
兩者都失敗

測資



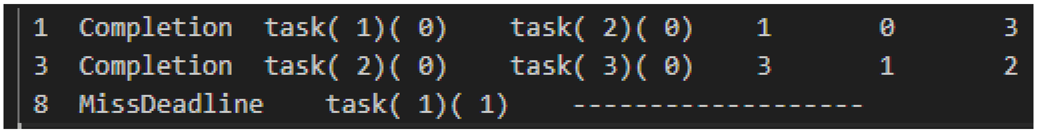
```
1 0 1 4
2 0 2 5
3 0 6 20
```

RM:



```
 1  Completion  task( 1)( 0)    task( 2)( 0)    1       0       3
 3  Completion  task( 2)( 0)    task( 3)( 0)    3       1       2
 4  Preemption  task( 3)( 0)    task( 1)( 1)
 5  Completion  task( 1)( 1)    task( 2)( 1)    1       0       3
 7  Completion  task( 2)( 1)    task( 3)( 0)    2       0       3
 8  Preemption  task( 3)( 0)    task( 1)( 2)
 9  Completion  task( 1)( 2)    task( 3)( 0)    1       0       3
10  Preemption  task( 3)( 0)    task( 2)( 2)
12  Completion  task( 2)( 2)    task( 1)( 3)    2       0       3
13  Completion  task( 1)( 3)    task( 3)( 0)    1       0       3
15  Preemption  task( 3)( 0)    task( 2)( 3)
16  Preemption  task( 2)( 3)    task( 1)( 4)
17  Completion  task( 1)( 4)    task( 2)( 3)    1       0       3
18  Completion  task( 2)( 3)    task( 3)( 0)    3       1       2
19  Completion  task( 3)( 0)    task(63)    19      13      1
20  Preemption  task(63)    task( 1)( 5)
21  Completion  task( 1)( 5)    task( 2)( 4)    1       0       3
23  Completion  task( 2)( 4)    task( 3)( 1)    3       1       2
24  Preemption  task( 3)( 1)    task( 1)( 6)
25  Completion  task( 1)( 6)    task( 2)( 5)    1       0       3
27  Completion  task( 2)( 5)    task( 3)( 1)    2       0       3
28  Preemption  task( 3)( 1)    task( 1)( 7)
29  Completion  task( 1)( 7)    task( 3)( 1)    1       0       3
30  Preemption  task( 3)( 1)    task( 2)( 6)
```

FIFO



```
1  Completion  task( 1)( 0)    task( 2)( 0)    1       0       3
3  Completion  task( 2)( 0)    task( 3)( 0)    3       1       2
8  MissDeadline    task( 1)( 1)    -------------------
```

RM 成功，FIFO 失敗

從排程過程發現由於 FIFO 是 JOB 開始後就無法變更執行任務的，因此會出現過長執行時間的 TASK 佔領 CPU 導致 MissDeadLine 的問題