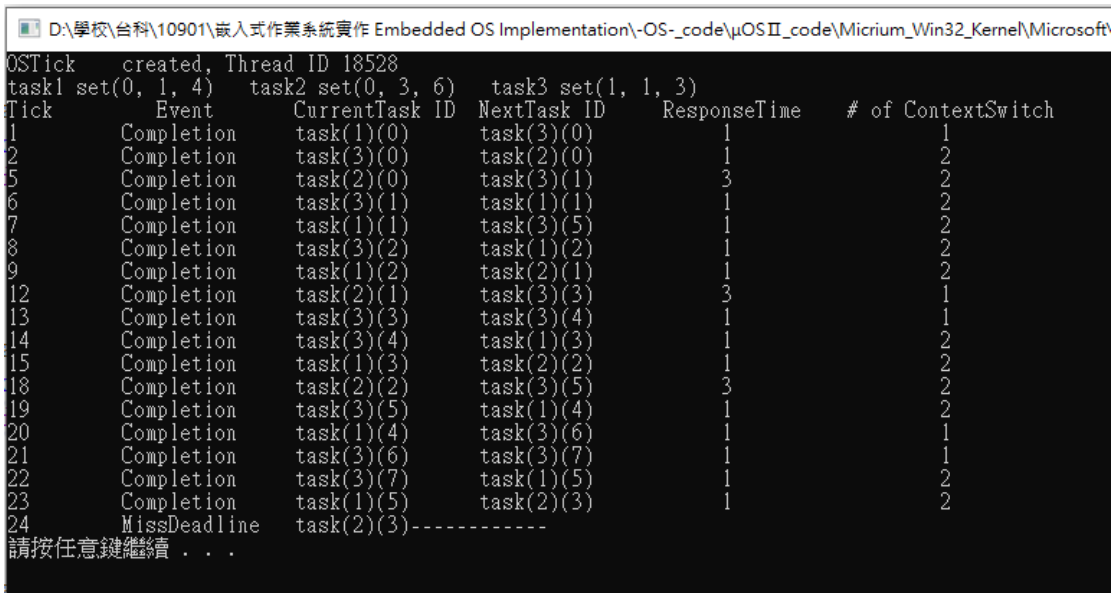
[PART I] EDF Scheduler Implementation.

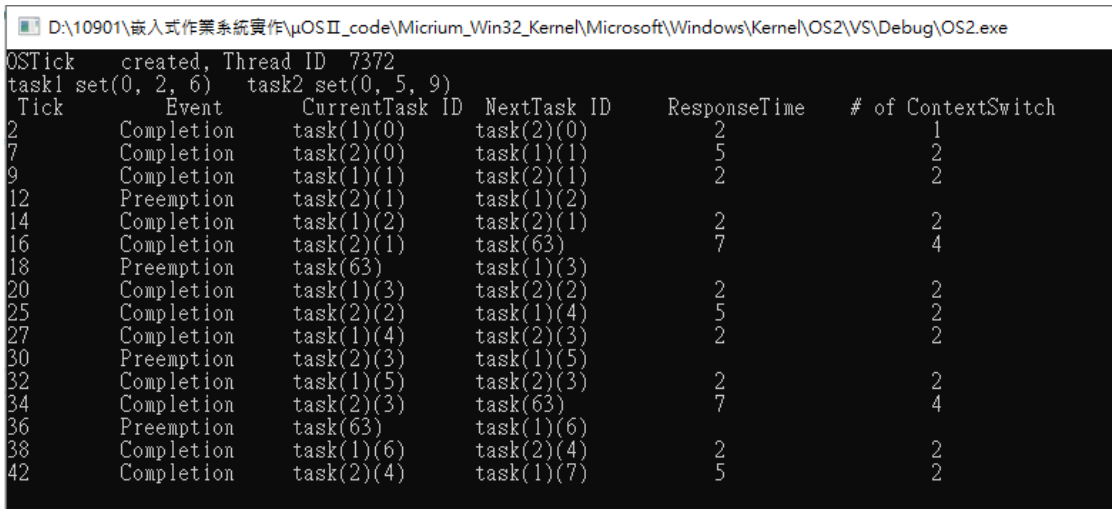
1.The screenshot results (with the given format) of two task sets. (Tick 0 to tick 40 or the tick when a task missing the deadline) (10%)

(1)Task set 1 = {τ1 (0, 2, 6), τ2 (0, 5, 9)}



```
D:\學校\台科\10901\嵌入式作業系統實作 Embedded OS Implementation\-OS-_code\μOSII_code\Micrium_Win32_Kernel\Microsoft\
OSTick    created, Thread ID 18528
task1 set(0, 1, 4)   task2 set(0, 3, 6)   task3 set(1, 1, 3)
Tick         Event       CurrentTask ID  NextTask ID     ResponseTime    # of ContextSwitch
1         Completion     task(1)(0)      task(3)(0)          1                1
2         Completion     task(3)(0)      task(2)(0)          1                2
5         Completion     task(2)(0)      task(3)(1)          3                2
6         Completion     task(3)(1)      task(1)(1)          1                2
7         Completion     task(1)(1)      task(3)(5)          1                2
8         Completion     task(3)(2)      task(1)(2)          1                2
9         Completion     task(1)(2)      task(2)(1)          1                2
12        Completion     task(2)(1)      task(3)(3)          3                1
13        Completion     task(3)(3)      task(3)(4)          1                1
14        Completion     task(3)(4)      task(1)(3)          1                2
15        Completion     task(1)(3)      task(2)(2)          1                2
18        Completion     task(2)(2)      task(3)(5)          3                2
19        Completion     task(3)(5)      task(1)(4)          1                2
20        Completion     task(1)(4)      task(3)(6)          1                1
21        Completion     task(3)(6)      task(3)(7)          1                1
22        Completion     task(3)(7)      task(1)(5)          1                2
23        Completion     task(1)(5)      task(2)(3)          1                2
24        MissDeadline   task(2)(3)------------
請按任意鍵繼續 . . .
```

(2)Task set 2 = {τ1 (0, 1, 4), τ2 (0, 3, 6), τ3 (1, 1, 3)}



```
D:\10901\嵌入式作業系統實作\μOSII_code\Micrium_Win32_Kernel\Microsoft\Windows\Kernel\OS2\VS\Debug\OS2.exe
OSTick    created, Thread ID  7372
task1 set(0, 2, 6)   task2 set(0, 5, 9)
 Tick         Event       CurrentTask ID  NextTask ID     ResponseTime    # of ContextSwitch
2         Completion     task(1)(0)      task(2)(0)          2                1
7         Completion     task(2)(0)      task(1)(1)          5                2
9         Completion     task(1)(1)      task(2)(1)          2                2
12        Preemption     task(2)(1)      task(1)(2)
14        Completion     task(1)(2)      task(2)(1)          2                2
16        Completion     task(2)(1)      task(63)            7                4
18        Preemption     task(63)        task(1)(3)
20        Completion     task(1)(3)      task(2)(2)          2                2
25        Completion     task(2)(2)      task(1)(4)          5                2
27        Completion     task(1)(4)      task(2)(3)          2                2
30        Preemption     task(2)(3)      task(1)(5)
32        Completion     task(1)(5)      task(2)(3)          2                2
34        Completion     task(2)(3)      task(63)            7                4
36        Preemption     task(63)        task(1)(6)
38        Completion     task(1)(6)      task(2)(4)          2                2
42        Completion     task(2)(4)      task(1)(7)          5                2
```

2. Implement and describe how to handle the deadline missing situation under EDF. (10%)

```
//處理missdeadline
ptcb = OSTCBList;
while (ptcb != (OS_TCB*)0) {
    TimeTask* point_t = ptcb->OSTCBExtPtr;
    if (point_t != 0 && OSTimeGet() > point_t->period_time * (point_t->executive_count + 1)) {
        printf("%-6d", OSTimeGet());
        printf("%-14s", "MissDeadline");
        printf("%-5s%d%-2s%d%-9s", "task(", ptcb->OSTCBId, ")(", point_t->executive_count, ")");
        printf("-------------------------\n");
        system("pause");
    }
    ptcb = ptcb->OSTCBNext;
}
```

3. 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%)

(1) Task set 1 = {τ1 (0, 2, 6), τ2 (0, 5, 9)}

```
#define TASK_STACKSIZE       2048
#define TASK1_PRIORITY       1
#define TASK2_PRIORITY       2
#define TASK1_ID             1
#define TASK2_ID             2

static  OS_STK  StartupTaskStk[APP_CFG_STARTUP_TASK_STK_SIZE];
static  OS_STK  Task1_STK[TASK_STACKSIZE];
static  OS_STK  Task2_STK[TASK_STACKSIZE];

static  void  StartupTask(void* p_arg);
static  void  task1(void* p_arg);
static  void  task2(void* p_arg);


TimeTask task1_set = { 0, 2, 6 };      //set : {start time, work time, period time}
TimeTask task2_set = { 0, 5, 9 };      //set : {start time, work time, period time}
```

(2)Task set 2 = {τ1 (0, 1, 4), τ2 (0, 3, 6), τ3 (1, 1, 3)}

```
#define TASK_STACKSIZE       2048
#define TASK1_PRIORITY       2
#define TASK2_PRIORITY       3
#define TASK3_PRIORITY       1
#define TASK1_ID             1
#define TASK2_ID             2
#define TASK3_ID             3

static  OS_STK  StartupTaskStk[APP_CFG_STARTUP_TASK_STK_SIZE];
static  OS_STK  Task1_STK[TASK_STACKSIZE];
static  OS_STK  Task2_STK[TASK_STACKSIZE];
static  OS_STK  Task3_STK[TASK_STACKSIZE];

static  void  StartupTask(void* p_arg);
static  void  task1(void* p_arg);
static  void  task2(void* p_arg);
static  void  task3(void* p_arg);

TimeTask task1_set = { 0, 1, 4 };      //set : {start time, work time, period time}
TimeTask task2_set = { 0, 3, 6 };      //set : {start time, work time, period time}
TimeTask task3_set = { 1, 1, 3 };      //set : {start time, work time, period time}
```

```
/*
*********************************************************************************************************************************************************
*                                                                project2                                                                               *
*********************************************************************************************************************************************************
*/
int responsetime = 0;
int context_switch = 0;
OS_TCB* ptcb;
if (OSPrioHighRdy != 0) {                                                    //程式開始執行
    ptcb = OSTCBPrioTbl[OSPrioHighRdy];
    TimeTask* point_t = ptcb->OSTCBExtPtr;                                   //指向自定義數據以進行TCB擴展的指針
    if (point_t != 0) {                                                      //如果不是idle task
        //如果是completion_task
        if (OSTimeGet() - point_t->current_start_time == point_t->work_time + point_t->preemptive_time) {
            printf("%-6d", OSTimeGet());
            printf("%-14s", "Completion");
            if (OSTimeGet() % point_t->period_time == 0) {
                responsetime = point_t->period_time;                         //計算respondtime
            }
            else {
                responsetime = OSTimeGet() % point_t->period_time - point_t->start_time; //計算respondtime
            }
            printf("%-5s%d%-2s%d%-9s", "task(", ptcb->OSTCBId, ")(", point_t->executive_count, ")");  //print 現在的 task ID

            //Delay current task
            if (point_t->period_time * (point_t->executive_count + 1) - OSTimeGet() + point_t->start_time != 0) {
                OSTimeDly_EDF(ptcb->OSTCBId, ptcb->OSTCBDly, 8, 10, 20, 28, 39);
            }
            point_t->executive_count++;
            if (OSPrioHighRdy != (INT8U)((y << 3u) + OSUnMapTbl[OSRdyTbl[y]]))
                point_t->context_switch++;  //計算conextswitch
            context_switch = point_t->context_switch;//計算conextswitch
            point_t->context_switch = 0;        //contextswitch歸零
            point_t->preemptive_time = 0;       //preemptivetime歸零
        }
        //如果是preemption_task
        if (OSTimeGet() - point_t->current_start_time < point_t->work_time + point_t->preemptive_time &&
            OSPrioHighRdy != (INT8U)((y << 3u) + OSUnMapTbl[OSRdyTbl[y]]) ) {
            printf("%-6d", OSTimeGet());
            printf("%-14s", "Preemption");
            printf("%-5s%d%-2s%d%-9s", "task(", ptcb->OSTCBId, ")(", point_t->executive_count, ")");//print 現在的 task ID
            point_t->context_switch++;  //計算conextswitch
            if (point_t->preemptive_time > 0) {point_t->preemptive_time = OSTimeGet() - point_t->preemptive_time;}//更新preemptivetime
            else {point_t->preemptive_time = OSTimeGet(); }//更新preemptivetime
        }
    }

    //如果是idle task
    else if (point_t == 0) {
        printf("%-6d", OSTimeGet());
        printf("%-14s", "Preemption");
        printf("%-5s%d%-11s", "task(", OSPrioHighRdy, ")");//print idle task
    }

    if (OSPrioHighRdy != (INT8U)((y << 3u) + OSUnMapTbl[OSRdyTbl[y]]) || (point_t->context_switch == 0 && point_t->executive_count != 0)) {
        OSPrioHighRdy = (INT8U)((y << 3u) + OSUnMapTbl[OSRdyTbl[y]]);
        ptcb = OSTCBPrioTbl[OSPrioHighRdy];
        point_t = ptcb->OSTCBExtPtr;
        if (point_t != 0) {
            printf("%-5s%d%-2s%d%-15s", "task(", ptcb->OSTCBId, ")(", point_t->executive_count, ")");//print下一個task ID
            if (OSPrioHighRdy != OSPrioCur) {point_t->context_switch++;}//計算conextswitch
            if (point_t->preemptive_time == 0) {point_t->current_start_time = OSTimeGet();}//儲存task start time
            if (point_t->preemptive_time > 0) {point_t->preemptive_time = OSTimeGet() - point_t->preemptive_time;}//更新preemptive time
        }

        //如果下一個task是idel task
        else if (point_t == 0) {
            printf("%-5s%d%-17s", "task(", OSPrioHighRdy, ")");//print task(63)
        }
        //print responsetime and context switch
        if (responsetime > 0) {
            printf("%-20d%d\n", responsetime, context_switch);
            responsetime = 0;
        }
        else {
            printf("\n");
        }
```

```
//處理missdeadline
ptcb = OSTCBList;
while (ptcb != (OS_TCB*)0) {
    TimeTask* point_t = ptcb->OSTCBExtPtr;
    if (point_t != 0 && OSTimeGet() > point_t->period_time * (point_t->executive_count + 1)) {
        printf("%-6d", OSTimeGet());
        printf("%-14s", "MissDeadline");
        printf("%-5s%d%-2s%d%-9s", "task(", ptcb->OSTCBId, ")(", point_t->executive_count, ")");
        printf("--------------------------\n");
        system("pause");
    }
    ptcb = ptcb->OSTCBNext;
}
```

```
    //initizal setting
    else if (OSPrioHighRdy == 0) {
        OSPrioHighRdy = (INT8U)((y << 3u) + OSUnMapTbl[OSRdyTbl[y]]);

        OS_TCB* ptcb;
        ptcb = OSTCBList;
        while (ptcb != (OS_TCB*)0) {
            TimeTask* point_t = ptcb->OSTCBExtPtr;

            if (point_t != 0 && point_t->start_time > 0u) {                          // 0 means no delay!
                OS_ENTER_CRITICAL();
                y = ptcb->OSTCBY;                                                    //Delay current task
                OSRdyTbl[y] &= (OS_PRIO)~ptcb->OSTCBBitX;
                OS_TRACE_TASK_SUSPENDED(ptcb);
                if (OSRdyTbl[y] == 0u) {
                    OSRdyGrp &= (OS_PRIO)~ptcb->OSTCBBitY;
                }
                ptcb->OSTCBDly = point_t->start_time;                               //Load ticks in TCB
                OS_TRACE_TASK_DLY(point_t->start_time);
                OS_EXIT_CRITICAL();
            }
            ptcb = ptcb->OSTCBNext;
        }
        y = OSUnMapTbl[OSRdyGrp];
        OSPrioHighRdy = (INT8U)((y << 3u) + OSUnMapTbl[OSRdyTbl[y]]);

    }
```

```
void  OSTimeDly_EDF(INT32U ticks, OS_TCB* task_123, INT32U t1, INT32U t2, INT32U t3, INT32U j0, INT32U j1)
{
    INT8U     y;
    int min_task = 0;
#if OS_CRITICAL_METHOD == 3u                 /* Allocate storage for CPU status register           */
    OS_CPU_SR  cpu_sr = 0u;
#endif
    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 = task_123->OSTCBY;       /* Delay current task                                  */
        OSRdyTbl[y] &= (OS_PRIO)~task_123->OSTCBBitX;
        OS_TRACE_TASK_SUSPENDED(task_123);
        if (OSRdyTbl[y] == 0u) {
            OSRdyGrp &= (OS_PRIO)~task_123->OSTCBBitY;
        }
        task_123->OSTCBDly = ticks;          /* Load ticks in TCB                                  */
        OS_TRACE_TASK_DLY(ticks);
        OS_EXIT_CRITICAL();
        int t11 = t1-mod(OSTimeGet(),t1);    //計算task1與deadline的距離
        int t12 = t2 - mod(OSTimeGet(), t2); //計算task2與deadline的距離
        int t13 = t3 - mod(OSTimeGet(), t3); //計算task3與deadline的距離
        int j10 = j0 - mod(OSTimeGet(), j0); //計算job0與deadline的距離
        int j11 = j1 - mod(OSTimeGet(), j1); //計算job1與deadline的距離
        if (t11 < t12 && t11 < t13 && t11 < j10 && t11 < j11) { min_task = 1; }//如果task1最接近deadline
        if (t12 < t11 && t12 < t13 && t12 < j10 && t12 < j11) { min_task = 2; }//如果task2最接近deadline
        if (t13 < t12 && t13 < t11 && t13 < j10 && t13 < j11) { min_task = 3; }//如果task3最接近deadline
        if (j10 < t12 && j10 < t13 && j10 < j11 && j0 < j11) { min_task = 4; }//如果job0最接近deadline
        if (j11 < t12 && j11 < t13 && j1 < j10 && j1 < t11) { min_task = 5; }//如果job1最接近deadline

        if (task_123 != min_task) { OS_Sched(); }//如果現在的task ID 不是離deadline最近的
                                 /* Find next task to run!                                  */
    }
}
```

共用的部分—視情況開啟等量的定義

```
void task1(void *p_arg) {
    (void)p_arg;
    while (1) {
        ;
    }
}

void task2(void* p_arg) {
    (void)p_arg;
    while (1) {
        ;
    }
}

void task3(void* p_arg) {
    (void)p_arg;
    while (1) {
        ;
    }
}
```

```
OSTaskCreateExt(
    task1,
    &task1_set,
    &Task1_STK[TASK_STACKSIZE - 1],
    TASK1_PRIORITY,
    TASK1_ID,
    &Task1_STK[0],
    TASK_STACKSIZE,
    &task1_set,
    (OS_TASK_OPT_STK_CHK | OS_TASK_OPT_STK_CLR));

OSTaskCreateExt(
    task2,
    &task2_set,
    &Task2_STK[TASK_STACKSIZE - 1],
    TASK2_PRIORITY,
    TASK2_ID,
    &Task2_STK[0],
    TASK_STACKSIZE,
    &task2_set,
    (OS_TASK_OPT_STK_CHK | OS_TASK_OPT_STK_CLR));


OSTaskCreateExt(
    task3,
    &task3_set,
    &Task3_STK[TASK_STACKSIZE - 1],
    TASK3_PRIORITY,
    TASK3_ID,
    &Task3_STK[0],
    TASK_STACKSIZE,
    &task3_set,
    (OS_TASK_OPT_STK_CHK | OS_TASK_OPT_STK_CLR));
```

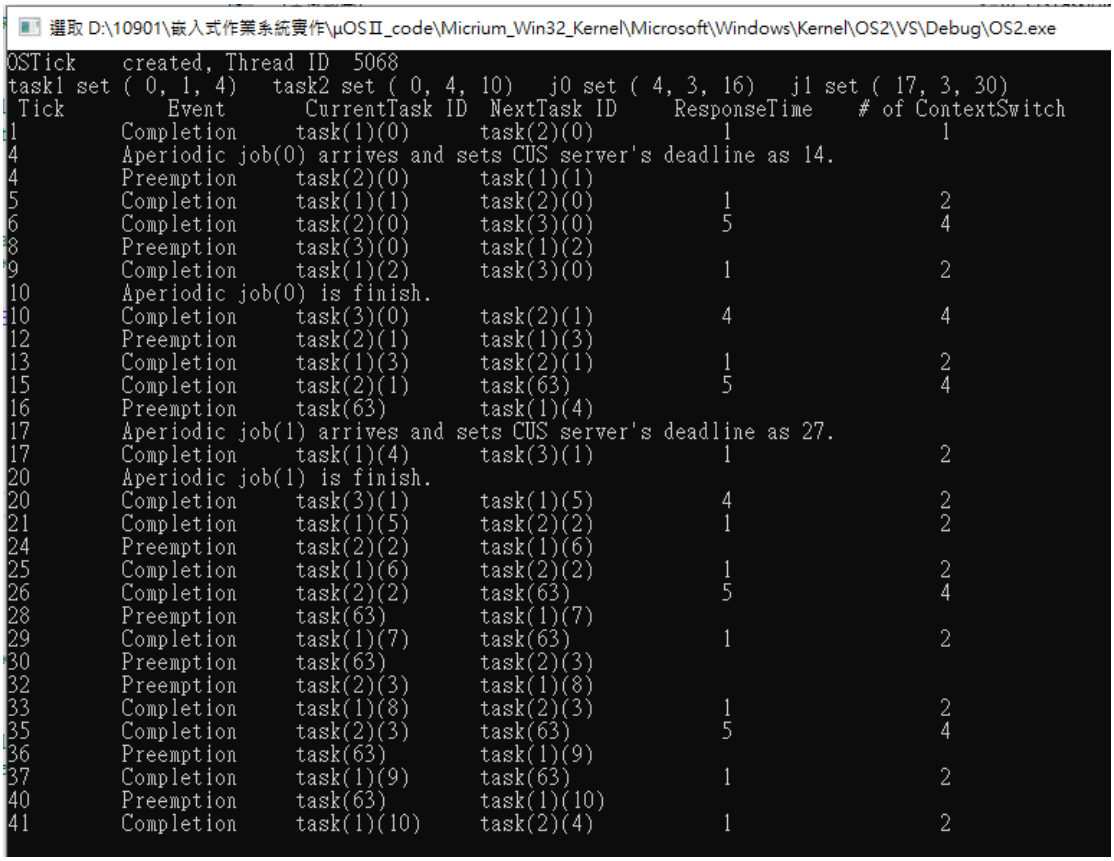
[PART II] CUS Scheduler Implementation [40%]

1. The screenshot results (with the given format) of two task sets. (Tick 0 to tick 40 or the tick when a task missing the deadline). (10%)

========== Task Set 1 ==========

Periodic Task Set1 = {τ1 (0, 1, 4), τ2 (0, 4, 10), τ3_ServerSize (0.3)}

Aperiodic Jobs Set1 = {j0 (4, 3, 16), j1 (17, 3, 30)}



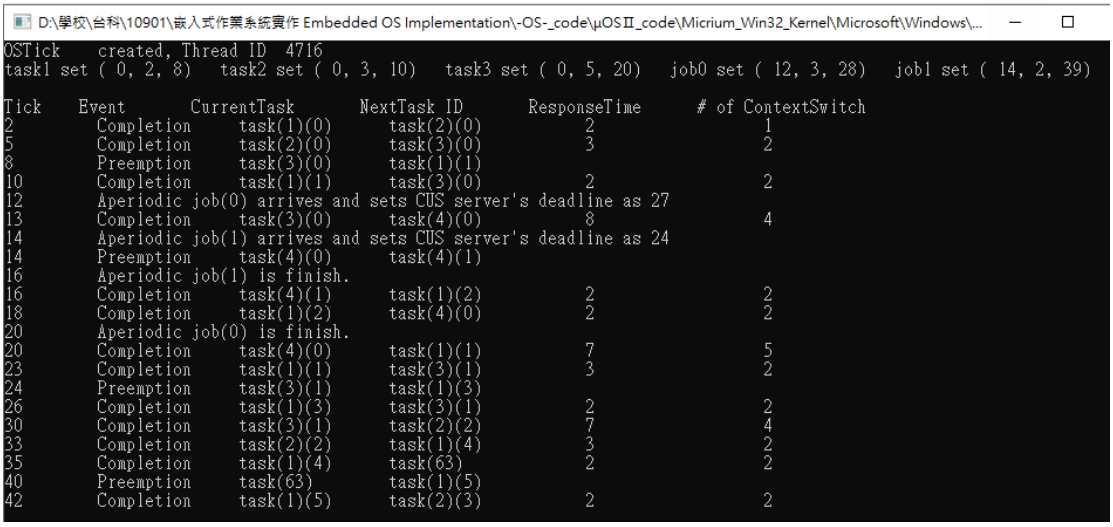
選取 D:\10901\嵌入式作業系統實作\μOSII_code\Micrium_Win32_Kernel\Microsoft\Windows\Kernel\OS2\VS\Debug\OS2.exe

```
OSTick    created, Thread ID  5068
task1 set ( 0, 1, 4)   task2 set ( 0, 4, 10)   j0 set ( 4, 3, 16)   j1 set ( 17, 3, 30)
 Tick         Event       CurrentTask ID  NextTask ID     ResponseTime    # of ContextSwitch
1         Completion     task(1)(0)      task(2)(0)          1                 1
4         Aperiodic job(0) arrives and sets CUS server's deadline as 14.
4         Preemption     task(2)(0)      task(1)(1)
5         Completion     task(1)(1)      task(2)(0)          1                 2
6         Completion     task(2)(0)      task(3)(0)          5                 4
8         Preemption     task(3)(0)      task(1)(2)
9         Completion     task(1)(2)      task(3)(0)          1                 2
10        Aperiodic job(0) is finish.
10        Completion     task(3)(0)      task(2)(1)          4                 4
12        Preemption     task(2)(1)      task(1)(3)
13        Completion     task(1)(3)      task(2)(1)          1                 2
15        Completion     task(2)(1)      task(63)            5                 4
16        Preemption     task(63)        task(1)(4)
17        Aperiodic job(1) arrives and sets CUS server's deadline as 27.
17        Completion     task(1)(4)      task(3)(1)          1                 2
20        Aperiodic job(1) is finish.
20        Completion     task(3)(1)      task(1)(5)          4                 2
21        Completion     task(1)(5)      task(2)(2)          1                 2
24        Preemption     task(2)(2)      task(1)(6)
25        Completion     task(1)(6)      task(2)(2)          1                 2
26        Completion     task(2)(2)      task(63)            5                 4
28        Preemption     task(63)        task(1)(7)
29        Completion     task(1)(7)      task(63)            1                 2
30        Preemption     task(63)        task(2)(3)
32        Preemption     task(2)(3)      task(1)(8)
33        Completion     task(1)(8)      task(2)(3)          1                 2
35        Completion     task(2)(3)      task(63)            5                 4
36        Preemption     task(63)        task(1)(9)
37        Completion     task(1)(9)      task(63)            1                 2
40        Preemption     task(63)        task(1)(10)
41        Completion     task(1)(10)     task(2)(4)          1                 2
```

========== Task Set 2 ==========

Periodic Task Set2 = {τ1 (0, 2, 8), τ2 (0, 3, 10), τ3 (0, 5, 20), τ4_ServerSize (0.2)}

Aperiodic Jobs Set2 = {j0 (12, 3, 28), j1 (14, 2, 39)}



2. 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%)

========== Task Set 1 ==========

Periodic Task Set1 = {τ1 (0, 1, 4), τ2 (0, 4, 10), τ3_ServerSize (0.3)}

Aperiodic Jobs Set1 = {j0 (4, 3, 16), j1 (17, 3, 30)}

```
#define TASK_STACKSIZE       2048
#define TASK1_PRIORITY       1
#define TASK2_PRIORITY       3
#define TASK3_PRIORITY       2
#define TASK4_PRIORITY       4
#define TASK1_ID             1
#define TASK2_ID             2
#define TASK3_ID             3
#define TASK4_ID             4
static  OS_STK  StartupTaskStk[APP_CFG_STARTUP_TASK_STK_SIZE];
static  OS_STK  Task1_STK[TASK_STACKSIZE];
static  OS_STK  Task2_STK[TASK_STACKSIZE];
static  OS_STK  Task3_STK[TASK_STACKSIZE];
static  OS_STK  Task4_STK[TASK_STACKSIZE];
static  void  StartupTask(void* p_arg);
static  void  task1(void* p_arg);
static  void  task2(void* p_arg);
static  void  task3(void* p_arg);
static  void  task4(void* p_arg);
TimeTask task1_set = { 0, 1, 4 };      //set : {start time, work time, period time}
TimeTask task2_set = { 0, 4, 10 };      //set : {start time, work time, period time}
TimeTask task3_set = { 4, 3, 6 };      //set : {start time, work time, period time}
TimeTask task4_set = { 17, 3, 30 };      //set : {start time, work time, period time}
```

========== Task Set 2 ==========

Periodic Task Set2 = {τ1 (0, 2, 8), τ2 (0, 3, 10), τ3 (0, 5, 20), τ4_ServerSize (0.2)}

Aperiodic Jobs Set2 = {j0 (12, 3, 28), j1 (14, 2, 39)}

```
#define TASK_STACKSIZE       2048
#define TASK1_PRIORITY       1
#define TASK2_PRIORITY       2
#define TASK3_PRIORITY       3
#define TASK4_PRIORITY       5
#define TASK5_PRIORITY       4
#define TASK1_ID             1
#define TASK2_ID             2
#define TASK3_ID             3
#define TASK4_ID             4
#define TASK5_ID             5
static  OS_STK  StartupTaskStk[APP_CFG_STARTUP_TASK_STK_SIZE];
static  OS_STK  Task1_STK[TASK_STACKSIZE];
static  OS_STK  Task2_STK[TASK_STACKSIZE];
static  OS_STK  Task3_STK[TASK_STACKSIZE];
static  OS_STK  Task4_STK[TASK_STACKSIZE];
static  OS_STK  Task5_STK[TASK_STACKSIZE];
static  void  StartupTask(void* p_arg);
static  void  task1(void* p_arg);
static  void  task2(void* p_arg);
static  void  task3(void* p_arg);
static  void  task4(void* p_arg);
static  void  task5(void* p_arg);
TimeTask task1_set = { 0, 2, 8 };      //set : {start time, work time, period time}
TimeTask task2_set = { 0, 3, 10 };      //set : {start time, work time, period time}
TimeTask task3_set = { 0, 5, 20 };      //set : {start time, work time, period time}
TimeTask task4_set = { 12, 3, 28 };      //set : {start time, work time, period time}
TimeTask task5_set = { 14, 2, 39 };      //set : {start time, work time, period time}
```

```
/*
*********************************************************************************************************************************************
*                                                                project2                                                                   *
*********************************************************************************************************************************************
*/

int jo_1[3] = { 12,3,28 };
int j1_1[3] = { 14,2,39 };
int t4 = 2; int d1 = 0; int d0 = 0;
d0 = jo_1[0] + (jo_1[1]*10 / t4);
d1 = j1_1[0] + (j1_1[1]*10 / t4);
int j0_in = 0; int j1_in = 0;

if (OSTimeGet() == jo_1[0]) {
    printf("%-5d Aperiodic job(0) arrives and sets CUS server's deadline as %d\n",OSTimeGet(),d0);
}
else if (OSTimeGet() == j1_1[0]) {
    printf("%-5d Aperiodic job(1) arrives and sets CUS server's deadline as %d\n", OSTimeGet(), d1);
}
int responsetime = 0;
int context_switch = 0;

OS_TCB* ptcb;

if (OSPrioHighRdy != 0) {                                                   //程式開始執行
    ptcb = OSTCBPrioTbl[OSPrioHighRdy];
    TimeTask* point_t = ptcb->OSTCBExtPtr;                                  //指向自定義數據以進行TCB擴展的指針

    if (point_t != 0) {                                                     //如果不是idle task
        //如果是completion_task
        if (OSTimeGet() - point_t->current_start_time == point_t->work_time + point_t->preemptive_time) {
            printf("%-6d", OSTimeGet());
            printf("%-14s", "Completion");
            if (OSTimeGet() % point_t->period_time == 0) {
                responsetime = point_t->period_time;                        //計算respondtime
            }
            else {
                responsetime = OSTimeGet() % point_t->period_time - point_t->start_time; //計算respondtime
            }
            if (ptcb->OSTCBId == 5 && j1_in != 0) { j1_in = 1; printf("%-5s%d%-2s%d%-9s", "task(", 4, ")(", 1, ")"); } //如果task ID 為5的時候 ，更改print task ID為4
            if (ptcb->OSTCBId == 4 && j0_in != 0) { j0_in = 1; printf("%-5s%d%-2s%d%-9s", "task(", ptcb->OSTCBId, ")(", point_t->executive_count, ")");}
            else { printf("%-5s%d%-2s%d%-9s", "task(", ptcb->OSTCBId, ")(", point_t->executive_count, ")"); } //print 現在的 task ID

            //Delay current task
            if (point_t->period_time * (point_t->executive_count + 1) - OSTimeGet() + point_t->start_time != 0) {
                OSTimeDly_EDF(ptcb->OSTCBId, ptcb->OSTCBDly, 8, 10, 20, 28, 39);
            }
            point_t->executive_count++;
            if (OSPrioHighRdy != (INT8U)((y << 3u) + OSUnMapTbl[OSRdyTbl[y]]))
                point_t->context_switch++;  //計算conextswitch
            context_switch = point_t->context_switch;//計算conextswitch
            point_t->context_switch = 0;        //contextswitch歸零
            point_t->preemptive_time = 0;       //preemptivetime歸零
        }
        //如果是preemption_task
        if (OSTimeGet() - point_t->current_start_time < point_t->work_time + point_t->preemptive_time &&
            OSPrioHighRdy != (INT8U)((y << 3u) + OSUnMapTbl[OSRdyTbl[y]]) ) {
            printf("%-6d", OSTimeGet());
            printf("%-14s", "Preemption");
            printf("%-5s%d%-2s%d%-9s", "task(", ptcb->OSTCBId, ")(", point_t->executive_count, ")");//print 現在的 task ID
            point_t->context_switch++;  //計算conextswitch


            if (point_t->preemptive_time > 0) {point_t->preemptive_time = OSTimeGet() - point_t->preemptive_time;}//更新preemptivetime
            else {point_t->preemptive_time = OSTimeGet(); }//更新preemptivetime
        }
    }
    //如果是idle task
    else if (point_t == 0) {
        printf("%-6d", OSTimeGet());
        printf("%-14s", "Preemption");
        printf("%-5s%d%-11s", "task(", OSPrioHighRdy, ")");//print idle task
    }

    if (OSPrioHighRdy != (INT8U)((y << 3u) + OSUnMapTbl[OSRdyTbl[y]]) || (point_t->context_switch == 0 && point_t->executive_count != 0)) {
        OSPrioHighRdy = (INT8U)((y << 3u) + OSUnMapTbl[OSRdyTbl[y]]);
        ptcb = OSTCBPrioTbl[OSPrioHighRdy];
        point_t = ptcb->OSTCBExtPtr;
        if (point_t != 0) {
            if (ptcb->OSTCBId == 4) { printf("%-5d Aperiodic job(0) is finish.\n", OSTimeGet()); }
            if (ptcb->OSTCBId == 5) { printf("%-5d Aperiodic job(1) is finish.\n", OSTimeGet()); }
            printf("%-5s%d%-2s%d%-15s", "task(", ptcb->OSTCBId, ")(", point_t->executive_count, ")");//print下一個task ID
            if (OSPrioHighRdy != OSPrioCur) {point_t->context_switch++;}//計算conextswitch
            if (point_t->preemptive_time == 0) {point_t->current_start_time = OSTimeGet();}//儲存task start time
            if (point_t->preemptive_time > 0) {point_t->preemptive_time = OSTimeGet() - point_t->preemptive_time;}//更新preemptive time
        }

        //如果下一個task是idel task
        else if (point_t == 0) {
            printf("%-5s%d%-17s", "task(", OSPrioHighRdy, ")");//print task(63)
        }
        //print responsetime and context switch
```

```
        //print responsetime and context switch
        if (responsetime > 0) {
            printf("%-20d%d\n", responsetime, context_switch);
            responsetime = 0;
        }
        else {
            printf("\n");
        }
        //處理missdeadline
        ptcb = OSTCBList;
        while (ptcb != (OS_TCB*)0) {
            TimeTask* point_t = ptcb->OSTCBExtPtr;
            if (point_t != 0 && OSTimeGet() > point_t->period_time * (point_t->executive_count + 1)) {
                printf("%-6d", OSTimeGet());
                printf("%-14s", "MissDeadline");
                printf("%-5s%d%-2s%d%-9s", "task(", ptcb->OSTCBId, ")(", point_t->executive_count, ")");
                printf("-------------------------\n");
                system("pause");
            }
            ptcb = ptcb->OSTCBNext;
        }

    }
}
```

```
    //initizal setting
    else if (OSPrioHighRdy == 0) {
        OSPrioHighRdy = (INT8U)((y << 3u) + OSUnMapTbl[OSRdyTbl[y]]);

        OS_TCB* ptcb;
        ptcb = OSTCBList;
        while (ptcb != (OS_TCB*)0) {
            TimeTask* point_t = ptcb->OSTCBExtPtr;

            if (point_t != 0 && point_t->start_time > 0u) {                          // 0 means no delay!
                OS_ENTER_CRITICAL();
                y = ptcb->OSTCBY;                                                    //Delay current task
                OSRdyTbl[y] &= (OS_PRIO)~ptcb->OSTCBBitX;
                OS_TRACE_TASK_SUSPENDED(ptcb);
                if (OSRdyTbl[y] == 0u) {
                    OSRdyGrp &= (OS_PRIO)~ptcb->OSTCBBitY;
                }
                ptcb->OSTCBDly = point_t->start_time;                               //Load ticks in TCB
                OS_TRACE_TASK_DLY(point_t->start_time);
                OS_EXIT_CRITICAL();
            }
            ptcb = ptcb->OSTCBNext;
        }
        y = OSUnMapTbl[OSRdyGrp];
        OSPrioHighRdy = (INT8U)((y << 3u) + OSUnMapTbl[OSRdyTbl[y]]);

    }
    /*
    ********************************************************************************************************************************************
    *                                                                project2                                                                  *
    ********************************************************************************************************************************************
    */
```

```
void  OSTimeDly_EDF(INT32U ticks, OS_TCB* task_123, INT32U t1, INT32U t2, INT32U t3, INT32U j0, INT32U j1)
{
    INT8U     y;
    int min_task = 0;
#if OS_CRITICAL_METHOD == 3u                    /* Allocate storage for CPU status register           */
    OS_CPU_SR  cpu_sr = 0u;
#endif
    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 = task_123->OSTCBY;       /* Delay current task                                 */
        OSRdyTbl[y] &= (OS_PRIO)~task_123->OSTCBBitX;
        OS_TRACE_TASK_SUSPENDED(task_123);
        if (OSRdyTbl[y] == 0u) {
            OSRdyGrp &= (OS_PRIO)~task_123->OSTCBBitY;
        }
        task_123->OSTCBDly = ticks;             /* Load ticks in TCB                                  */
        OS_TRACE_TASK_DLY(ticks);
        OS_EXIT_CRITICAL();
        int t11 = t1-mod(OSTimeGet(),t1);    //計算task1與deadline的距離
        int t12 = t2 - mod(OSTimeGet(), t2); //計算task2與deadline的距離
        int t13 = t3 - mod(OSTimeGet(), t3); //計算task3與deadline的距離
        int j10 = j0 - mod(OSTimeGet(), j0); //計算job0與deadline的距離
        int j11 = j1 - mod(OSTimeGet(), j1); //計算job1與deadline的距離
        if (t11 < t12 && t11 < t13 && t11 < j10 && t11 < j11) { min_task = 1; }//如果task1最接近deadline
        if (t12 < t11 && t12 < t13 && t12 < j10 && t12 < j11) { min_task = 2; }//如果task2最接近deadline
        if (t13 < t12 && t13 < t11 && t13 < j10 && t13 < j11) { min_task = 3; }//如果task3最接近deadline
        if (j10 < t12 && j10 < t13 && j0 < j11 && j0 < j11) { min_task = 4; }//如果job0最接近deadline
        if (j11 < t12 && j11 < t13 && j1 < j10 && j1 < t11) { min_task = 5; }//如果job1最接近deadline

        if (task_123 != min_task) { OS_Sched(); }//如果現在的task ID 不是離deadline最近的
                                /* Find next task to run!                             */
    }
}
```

共用的部分--視情況開啟等量的定義

```
void task1(void *p_arg) {
    (void)p_arg;
    while (1) {
        ;
    }
}
```

```
void task2(void* p_arg) {
    (void)p_arg;
    while (1) {
        ;
    }
}
```

```
void task3(void* p_arg) {
```

```
void task4(void* p_arg) {
    (void)p_arg;
    while (1) {
        ;
    }
}
```

```
void task5(void* p_arg) {
    (void)p_arg;
    while (1) {
        ;
    }
}
```

```
OSTaskCreateExt(
    task1,
    &task1_set,
    &Task1_STK[TASK_STACKSIZE - 1],
    TASK1_PRIORITY,
    TASK1_ID,
    &Task1_STK[0],
    TASK_STACKSIZE,
    &task1_set,
    (OS_TASK_OPT_STK_CHK | OS_TASK_OPT_STK_CLR));
```

```
OSTaskCreateExt(
    task2,
    &task2_set,
    &Task2_STK[TASK_STACKSIZE - 1],
    TASK2_PRIORITY,
    TASK2_ID,
    &Task2_STK[0],
    TASK_STACKSIZE,
    &task2_set,
    (OS_TASK_OPT_STK_CHK | OS_TASK_OPT_STK_CLR));
```

```
OSTaskCreateExt(
    task3,
    &task3_set,
    &Task3_STK[TASK_STACKSIZE - 1],
    TASK3_PRIORITY,
    TASK3_ID,
    &Task3_STK[0],
    TASK_STACKSIZE,
    &task3_set,
    (OS_TASK_OPT_STK_CHK | OS_TASK_OPT_STK_CLR));
```

```
OSTaskCreateExt(
    task4,
    &task4_set,
    &Task4_STK[TASK_STACKSIZE - 1],
    TASK4_PRIORITY,
    TASK4_ID,
    &Task4_STK[0],
    TASK_STACKSIZE,
    &task4_set,
    (OS_TASK_OPT_STK_CHK | OS_TASK_OPT_STK_CLR));
```

```
OSTaskCreateExt(
    task5,
    &task5_set,
    &Task5_STK[TASK_STACKSIZE - 1],
    TASK5_PRIORITY,
    TASK5_ID,
    &Task5_STK[0],
    TASK_STACKSIZE,
    &task5_set,
    (OS_TASK_OPT_STK_CHK | OS_TASK_OPT_STK_CLR));
```

```
/*
*************************************************************************************************************************
*                                                    project2                                                           *
*************************************************************************************************************************
*/
typedef struct TimeTask {
    int start_time;             //作業開始時間
    int work_time;              //工作工作時間
    int period_time;            //任務期
    int context_switch;         //任務完成後，清除上下文開關
    int executive_count;        //任務執行時間
    int current_start_time;     //作業的實際開始時間
    int preemptive_time;        //搶占任務時間
}TimeTask;
/*
*************************************************************************************************************************
*                                                    project2                                                           *
*************************************************************************************************************************
*/
```