1. [PART I] EDF Scheduler Implementation

- 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)

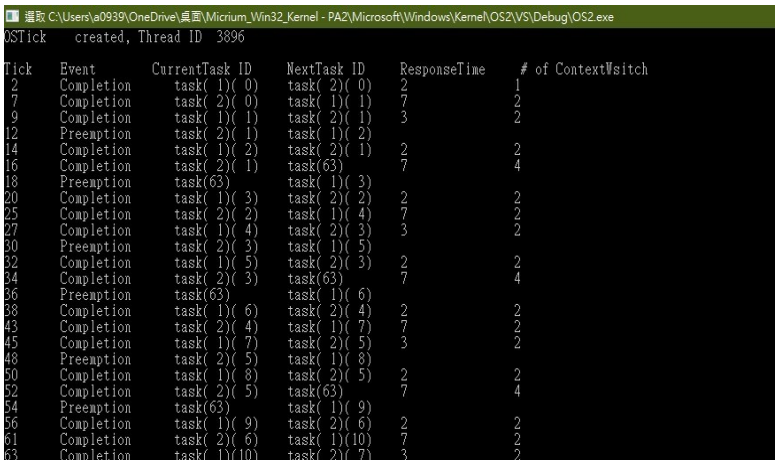


Fig 1. Result of Task{ $\tau_1(0,2,6)$, $\tau_2(0,5,9)$}

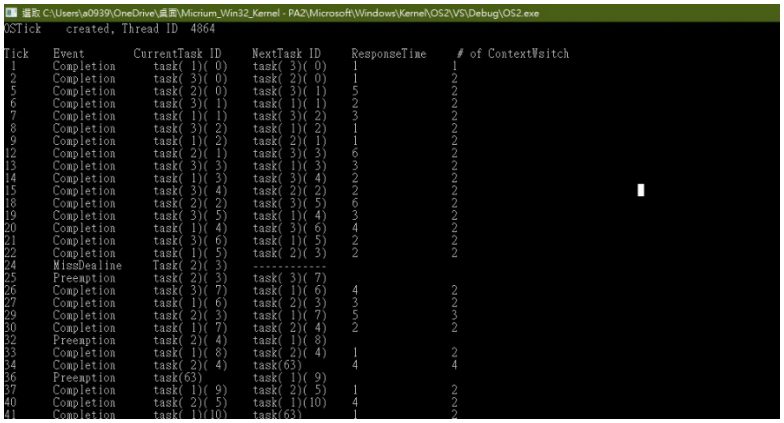


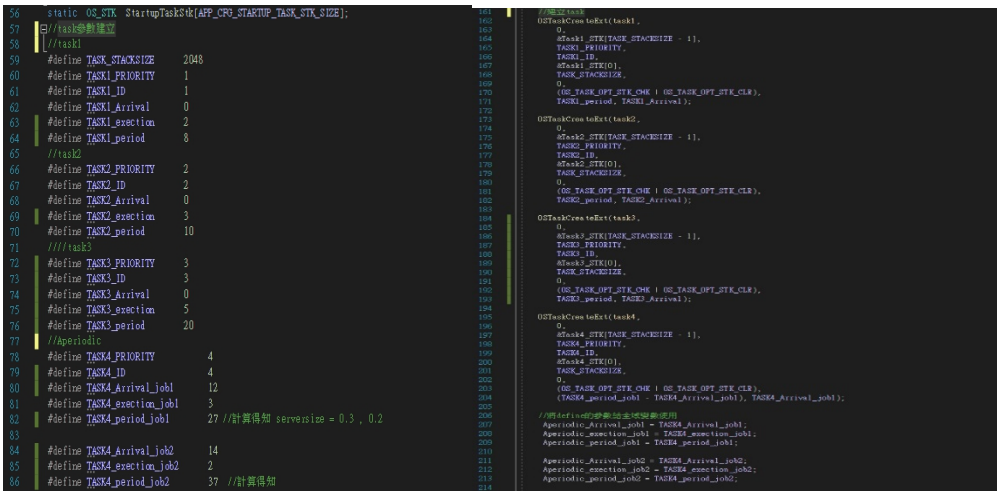
Fig 2. Result of Task{ $\tau_1(0,1,4)$, $\tau_2(0,3,6)$, $\tau_3(1,1,3)$}

- Implement and describe how to handle the deadline missing situation under EDF.

我的做法是當我的MissDeadline發生時，我設置一個break_flag，讓task直接break，強制跳掉發在MissDeadline的task，並將此task所有參數清為0，如Fig 2.所示，在TimeTick 25他會繼續執行，如Fig 4. 與 Fig 11.所示。

- 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)

因為此程式的EDF與CUS是同一個架構，有些圖片會包含CUS的方式，在下面CUS敘述時，也會用到這邊有共用的圖片說明。



```
static  OS_STK  StartupTaskStk[APP_CFG_STARTUP_TASK_STK_SIZE];
//task參數建立
//task1
#define TASK_STACKSIZE      2048
#define TASK1_PRIORITY      1
#define TASK1_ID            1
#define TASK1_Arrival       0
#define TASK1_exection      2
#define TASK1_period        8
//task2
#define TASK2_PRIORITY      2
#define TASK2_ID            2
#define TASK2_Arrival       0
#define TASK2_exection      3
#define TASK2_period        10
////task3
#define TASK3_PRIORITY      3
#define TASK3_ID            3
#define TASK3_Arrival       0
#define TASK3_exection      5
#define TASK3_period        20
//Aperiodic
#define TASK4_PRIORITY          4
#define TASK4_ID                4
#define TASK4_Arrival_job1      12
#define TASK4_exection_job1     3
#define TASK4_period_job1       27 //計算得知 serversize = 0.3 , 0.2

#define TASK4_Arrival_job2      14
#define TASK4_exection_job2     2
#define TASK4_period_job2       37 //計算得知
```

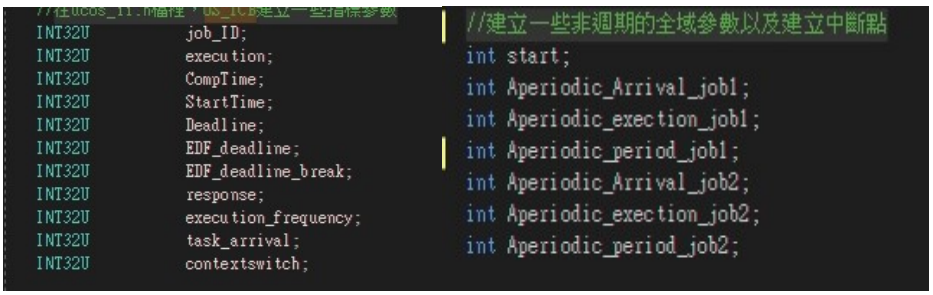
```
//建立task
OSTaskCreateExt(task1,
    0,
    &Task1_STK[TASK_STACKSIZE - 1],
    TASK1_PRIORITY,
    TASK1_ID,
    &Task1_STK[0],
    TASK_STACKSIZE,
    0,
    (OS_TASK_OPT_STK_CHK | OS_TASK_OPT_STK_CLR),
    TASK1_period, TASK1_Arrival);

OSTaskCreateExt(task2,
    0,
    &Task2_STK[TASK_STACKSIZE - 1],
    TASK2_PRIORITY,
    TASK2_ID,
    &Task2_STK[0],
    TASK_STACKSIZE,
    0,
    (OS_TASK_OPT_STK_CHK | OS_TASK_OPT_STK_CLR),
    TASK2_period, TASK2_Arrival);

OSTaskCreateExt(task3,
    0,
    &Task3_STK[TASK_STACKSIZE - 1],
    TASK3_PRIORITY,
    TASK3_ID,
    &Task3_STK[0],
    TASK_STACKSIZE,
    0,
    (OS_TASK_OPT_STK_CHK | OS_TASK_OPT_STK_CLR),
    TASK3_period, TASK3_Arrival);

OSTaskCreateExt(task4,
    0,
    &Task4_STK[TASK_STACKSIZE - 1],
    TASK4_PRIORITY,
    TASK4_ID,
    &Task4_STK[0],
    TASK_STACKSIZE,
    0,
    (OS_TASK_OPT_STK_CHK | OS_TASK_OPT_STK_CLR),
    (TASK4_period_job1 - TASK4_Arrival_job1), TASK4_Arrival_job1);

//將define的參數給全域變數使用
Aperiodic_Arrival_job1 = TASK4_Arrival_job1;
Aperiodic_exection_job1 = TASK4_exection_job1;
Aperiodic_period_job1 = TASK4_period_job1;

Aperiodic_Arrival_job2 = TASK4_Arrival_job2;
Aperiodic_exection_job2 = TASK4_exection_job2;
Aperiodic_period_job2 = TASK4_period_job2;
```

Fig 3. 左圖為task的參數設定(包含Aperiodic)，右圖為task建立與非週期性參數使用建立題目所需求的task預設參數，以及在OSTaskCreateExt()初始所需要的參數建立。

```
//週期性task內部結構
void task1(void* p_arg) {
    (void)p_arg;
    // OSTCBCur->task_arrival = TASK1_Arrival;
    OSTCBCur->execution = TASK1_exection;
    if (OSTimeGet() < TASK1_Arrival) {
        OSTimeDly(OSTCBCur->task_arrival - OSTimeGet());
    }
    while (1) {
        OSTCBCur->EDF_deadline_break = 0;
        OSTCBCur->StartTime = OSTimeGet();
        while (OSTCBCur->execution_frequency  < TASK1_exection)
        {
            if (OSTCBCur->EDF_deadline_break == 1) {
                break;
            }
            //Do something
        }
        // printf("time %d task1\n",OSTimeGet());
        if (TASK1_period - OSTCBCur->response == 0) {
            //printf("back\n");
            OS_Sched();
        }
        else {
            OSTimeDly(TASK1_period - OSTCBCur->response);
        }
    }
}
```

Fig 4. 週期性task的內部結構

期性task內部結構，在task沒到達arrival time會先讓task進到OSTimeDly，切換至目前ready的task，並在while()裡面除了判斷執行次數還有MissDeadline中斷點判斷。



```
//在ucos_ii.h檔裡，從OS_TCB建立一些指標參數
INT32U          job_ID;
INT32U          execution;
INT32U          CompTime;
INT32U          StartTime;
INT32U          Deadline;
INT32U          EDF_deadline;
INT32U          EDF_deadline_break;
INT32U          response;
INT32U          execution_frequency;
INT32U          task_arrival;
INT32U          contextswitch;
```

```
//建立一些非週期的全域參數以及建立中斷點
int start;
int Aperiodic_Arrival_job1;
int Aperiodic_exection_job1;
int Aperiodic_period_job1;
int Aperiodic_Arrival_job2;
int Aperiodic_exection_job2;
int Aperiodic_period_job2;
```

Fig 5. 右圖為在ucos_ii.h檔裡從OS_TCB建立數據結構，
右圖為非週期全域變數與中斷點參數

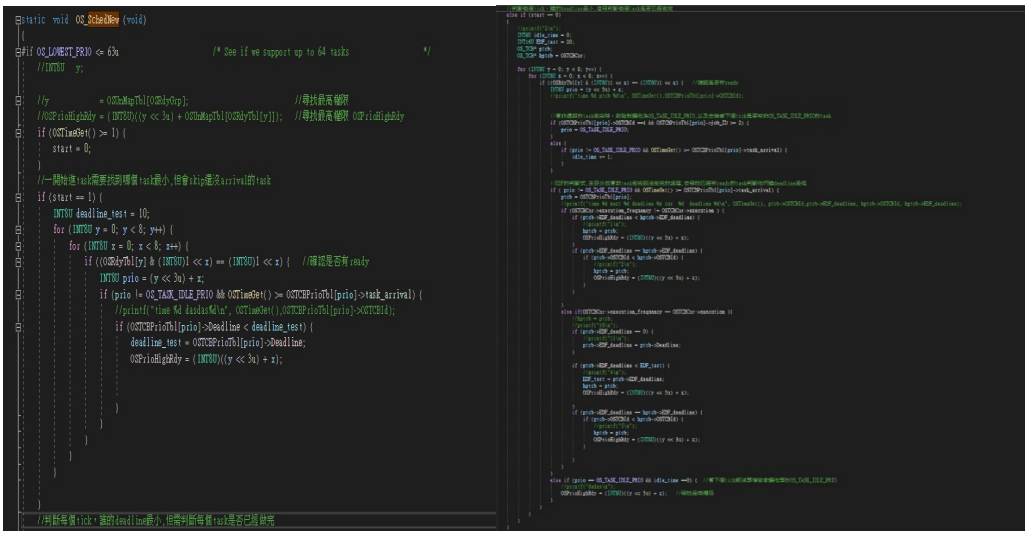


Fig 6. OS_SchedNew()修改的code

在OS_SchedNew()裡原先是尋找最高優先權，但因為EDF的判斷是找最小deadline，因此在裡面設計所要判斷的EDF程式，一開始OSStart()也會進到OS_SchedNew()找，因此左圖先處理一開始進來的問題，右圖部分除了找最小deadline，還需判斷目前task是否已經做完了，然後讓其他ready的task去比較，如果還沒做完就一起比較。

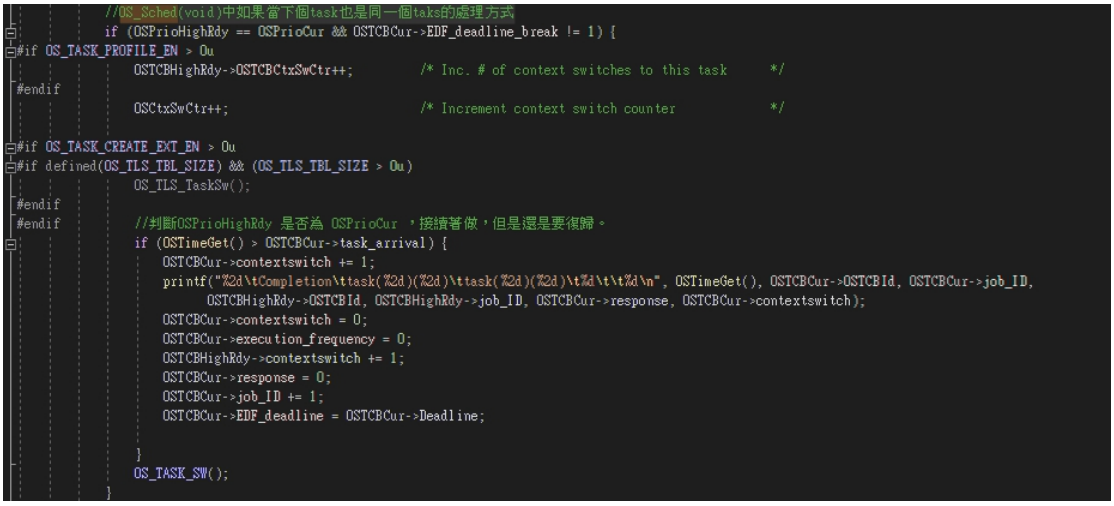
Fig 7. OSTimeTick()修改的code-1

在OSTimeTick()裡，分成兩部分去判斷，在已經ready但沒有執行的task，會去增加他的respones time與調整task的EDF_deadline，並判斷是否有MissDeadline的，並給予此task break flag，輪到此task動作會觸發MissDeadline後續的處理。在OSTCBCur中，除了增加respones time與調整task的EDF_deadline，還有會讓他增加工作完成次數，以及目前task MissDeadline的判斷。

```
                    // OSIntExit (void)中計算每個被中斷的task contextswitch
                    if (OSTimeGet() > OSTCBCur->task_arrival) {
                        if (OSPrioCur == 63) {
                            printf("%2d\tPreemption\ttask(%2d)     \ttask(%2d)(%2d)\n", OSTimeGet(),
                                OSPrioCur, OSTCBHighRdy->OSTCBId, OSTCBHighRdy->job_ID);
                        }
                        else
                        {
                            printf("%2d\tPreemption\ttask(%2d)(%2d)\ttask(%2d)(%2d)\n", OSTimeGet(),
                                OSTCBCur->OSTCBId, OSTCBCur->job_ID, OSTCBHighRdy->OSTCBId, OSTCBHighRdy->job_ID);
                        }
                        OSTCBHighRdy->contextswitch += 1;
                        OSTCBCur->contextswitch += 1;

                    }
                    OSIntCtxSw();                       /* Perform interrupt level ctx switch      */
```

Fig 8. OSIntExit 修改的code

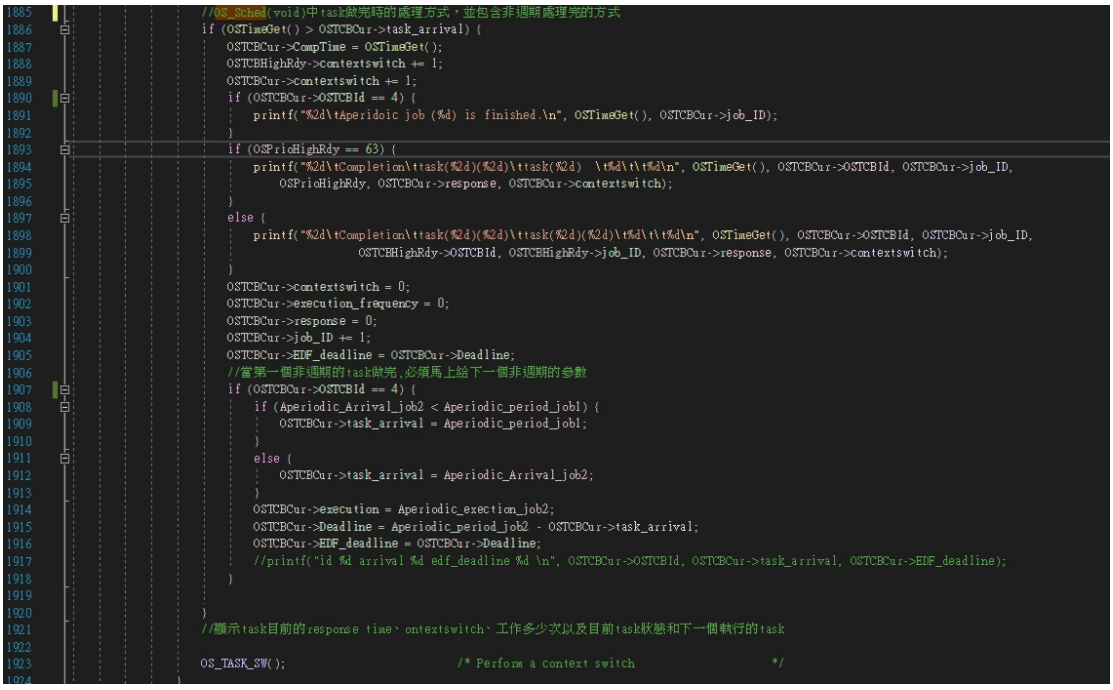
在OSIntExit()中，顯示目前的資訊以及增加contexswitch次數。



```
            //OS_Sched(void)中如果當下個task也是同一個taks的處理方式
            if (OSPrioHighRdy == OSPrioCur && OSTCBCur->EDF_deadline_break != 1) {
#if OS_TASK_PROFILE_EN > 0u
                OSTCBHighRdy->OSTCBCtxSwCtr++;         /* Inc. # of context switches to this task      */
#endif

                OSCtxSwCtr++;                          /* Increment context switch counter            */

#if OS_TASK_CREATE_EXT_EN > 0u
#if defined(OS_TLS_TBL_SIZE) && (OS_TLS_TBL_SIZE > 0u)
                OS_TLS_TaskSw();
#endif
#endif          //判斷OSPrioHighRdy 是否為 OSPrioCur ，接續著做，但是還是要復歸。
                if (OSTimeGet() > OSTCBCur->task_arrival) {
                    OSTCBCur->contextswitch += 1;
                    printf("%2d\tCompletion\ttask(%2d)(%2d)\ttask(%2d)(%2d)\t%d\t\t%d\n", OSTimeGet(), OSTCBCur->OSTCBId, OSTCBCur->job_ID,
                          OSTCBHighRdy->OSTCBId, OSTCBHighRdy->job_ID, OSTCBCur->response, OSTCBCur->contextswitch);
                    OSTCBCur->contextswitch = 0;
                    OSTCBCur->execution_frequency = 0;
                    OSTCBHighRdy->contextswitch += 1;
                    OSTCBCur->response = 0;
                    OSTCBCur->job_ID += 1;
                    OSTCBCur->EDF_deadline = OSTCBCur->Deadline;

                }
                OS_TASK_SW();
            }
```

Fig 9. OS_Sched()修改的code-1

OS_Sched()要判斷說，如果目前執行的task跟經過EDF排好之後的task如果都是同一個task，要去做一些處理判斷，並且切割他們是不同的事件。



```
1885            //OS_Sched(void)中task做完時的處理方式，並包含非週期處理完的方式
1886            if (OSTimeGet() > OSTCBCur->task_arrival) {
1887                OSTCBCur->CompTime = OSTimeGet();
1888                OSTCBHighRdy->contextswitch += 1;
1889                OSTCBCur->contextswitch += 1;
1890                if (OSTCBCur->OSTCBId == 4) {
1891                    printf("%2d\tAperidoic job (%d) is finished.\n", OSTimeGet(), OSTCBCur->job_ID);
1892                }
1893                if (OSPrioHighRdy == 63) {
1894                    printf("%2d\tCompletion\ttask(%2d)(%2d)\ttask(%2d)  \t%d\t\t%d\n", OSTimeGet(), OSTCBCur->OSTCBId, OSTCBCur->job_ID,
1895                        OSPrioHighRdy, OSTCBCur->response, OSTCBCur->contextswitch);
1896                }
1897                else {
1898                    printf("%2d\tCompletion\ttask(%2d)(%2d)\ttask(%2d)(%2d)\t%d\t\t%d\n", OSTimeGet(), OSTCBCur->OSTCBId, OSTCBCur->job_ID,
1899                                OSTCBHighRdy->OSTCBId, OSTCBHighRdy->job_ID, OSTCBCur->response, OSTCBCur->contextswitch);
1900                }
1901                OSTCBCur->contextswitch = 0;
1902                OSTCBCur->execution_frequency = 0;
1903                OSTCBCur->response = 0;
1904                OSTCBCur->job_ID += 1;
1905                OSTCBCur->EDF_deadline = OSTCBCur->Deadline;
1906                //當第一個非週期的task做完,必須馬上給下一個非週期的參數
1907                if (OSTCBCur->OSTCBId == 4) {
1908                    if (Aperiodic_Arrival_job2 < Aperiodic_period_job1) {
1909                        OSTCBCur->task_arrival = Aperiodic_period_job1;
1910                    }
1911                    else {
1912                        OSTCBCur->task_arrival = Aperiodic_Arrival_job2;
1913                    }
1914                    OSTCBCur->execution = Aperiodic_exection_job2;
1915                    OSTCBCur->Deadline = Aperiodic_period_job2 - OSTCBCur->task_arrival;
1916                    OSTCBCur->EDF_deadline = OSTCBCur->Deadline;
1917                    //printf("id %d arrival %d edf_deadline %d \n", OSTCBCur->OSTCBId, OSTCBCur->task_arrival, OSTCBCur->EDF_deadline);
1918                }
1919
1920            }
1921            //顯示task目前的response time、ontextswitch、工作多少次以及目前task狀態和下一個執行的task
1922
1923            OS_TASK_SW();                       /* Perform a context switch                    */
1924        }
```

Fig 10. OS_Sched()修改的code-2

OS_Sched()第二部分判斷目前task跟排完後的task不同的處理方式，除了顯示目前task的資訊，以及做相對應的參數初始化、工作完成次數。

```
1927        //OS_Sched(void)中如果此task missdeadline的後續處理
1928        else {
1929            OSTCBCur->contextswitch = 0;
1930            OSTCBCur->execution_frequency = 0;
1931            OSTCBCur->response = 0;
1932            OSTCBCur->EDF_deadline = OSTCBCur->Deadline;
1933
1934        }
```

Fig 11. OS_Sched()修改的code-3

OS_Sched()第三個判斷此task是否有MissDeadline，就之前講到的處理方式，task裡面會跳break到OSTimeDly，最後會進到這裡判斷是否有MissDeadline，再去做參數初始化。

2. [PART II] CUS Scheduler Implementation [40%]

- 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).

```
選取 C:\Users\a0939\OneDrive\桌面\Micrium_Win32_Kernel - PA2\Microsoft\Windows\Kernel\OS2\VS\Debug\OS2.exe
OSTick    created, Thread ID 15192

Tick    Event        CurrentTask ID     NextTask ID     ResponseTime     # of ContextWsitch
 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)    6                4
 8      Preemption       task( 3)( 0)    task( 1)( 2)
 9      Completion       task( 1)( 2)    task( 3)( 0)    1                2
10      Aperidoic job (0) is finished.
10      Completion       task( 3)( 0)    task( 2)( 1)    6                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      Aperidoic job (1) is finished.
20      Completion       task( 3)( 1)    task( 1)( 5)    3                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)        6                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)
```

Fig 12. Result of Task{ $\tau_1(0,1,4)$, $\tau_2(0,4,10)$, $\tau_3 ServerSize(0.3)$} and Aperiodic{$j_0(4,3,16), j_1(17,3,30)$}.

```
選取 C:\Users\a0939\OneDrive\桌面\Micrium_Win32_Kernel - PA2\Microsoft\Windows\Kernel\OS2\VS\Debug\OS2.exe

Tick    Event        CurrentTask ID     NextTask ID     ResponseTime     # of ContextWsitch
 2      Completion       task( 1)( 0)    task( 2)( 0)    2                1
 5      Completion       task( 2)( 0)    task( 3)( 0)    5                2
 8      Preemption       task( 3)( 0)    task( 1)( 1)
10      Completion       task( 1)( 1)    task( 2)( 1)    2                2
12      Aperiodic job (0) arrives and sets CUS server's deadline as 27.
13      Completion       task( 2)( 1)    task( 3)( 0)    3                2
14      Aperiodic job (1) arrives. Do nothing.
14      Aperiodic job (1) sets CUS server's deadline as 37.
15      Completion       task( 3)( 0)    task( 4)( 0)    15               4
16      Preemption       task( 4)( 0)    task( 1)( 2)
18      Completion       task( 1)( 2)    task( 4)( 0)    2                2
20      Aperidoic job (0) is finished.
20      Completion       task( 4)( 0)    task( 2)( 2)    8                4
23      Completion       task( 2)( 2)    task( 3)( 1)    3                2
24      Preemption       task( 3)( 1)    task( 1)( 3)
26      Completion       task( 1)( 3)    task( 3)( 1)    2                2
27      Preemption       task( 3)( 1)    task( 4)( 1)
29      Aperidoic job (1) is finished.
29      Completion       task( 4)( 1)    task( 3)( 1)    2                2
30      Preemption       task( 3)( 1)    task( 2)( 3)
32      Preemption       task( 2)( 3)    task( 1)( 4)
34      Completion       task( 1)( 4)    task( 2)( 3)    2                2
35      Completion       task( 2)( 3)    task( 3)( 1)    5                4
37      Completion       task( 3)( 1)    task(63)        17               8
40      Preemption       task(63)        task( 1)( 5)
```

Fig 13. Result of Task{ $\tau_1(0,2,8)$, $\tau_2(0,3,10)$, $\tau_3(0,5,20)$, $\tau_4 ServerSize(0.2)$} and Aperiodic{$j_0(12,3,28), j_1(14,2,39)$}.

- 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).

因為架構是從EDF改過來的，所以大部分的功能都與第一部分一樣，只是多增加非週期性task的判斷，如下圖 Fig14.所示，而在參數設定的部分，如Fig 3. 與 Fig 5. 所示，雖然非週期性task其實跟週期性task一樣的判斷寫法，但是在非週期性task都做完的時候，為了防止他被喚醒，我寫一個判斷說他超過幾個job之後都會變成是idle task狀態，如Fig 6.(右圖)所示，然後再顯示非週期性task狀態，如Fig 7. 與 Fig 15.所示。

```
351     //Aperiodic job 此task一共放了兩個Aperiodic job
352   void task4(void* p_arg) {
353       (void)p_arg;
354       //first job
355
356       OSTCBCur->execution = TASK4_exection_job1;
357       //OSTCBCur->Deadline = TASK4_period_job1 - TASK4_Arrival_job1;
358       aperiodic(p_arg);
359       //second job
360       aperiodic(p_arg);
361   }
362   void aperiodic(void* p_arg) {
363
364       if (OSTimeGet() < OSTCBCur->task_arrival) {
365           OSTimeDly(OSTCBCur->task_arrival - OSTimeGet());
366       }
367       OSTCBCur->EDF_deadline_break = 0;
368       OSTCBCur->StartTime = OSTimeGet();
369
370       while (OSTCBCur->execution_frequency < OSTCBCur->execution)
371       {
372           if (OSTCBCur->EDF_deadline_break == 1) {
373               break;
374           }
375           //Do something
376       }
377       if (OSTCBCur->Deadline - OSTCBCur->response == 0) {
378
379           OS_Sched();
380       }
381       else {
382           OSTimeDly(OSTCBCur->Deadline - OSTCBCur->response);
383       }
384   }
```

Fig 14. 非週期性task的內部結構

```
//此程式在OSTimeTick()執行，判斷非週期到達arrival time,或小於上個deadline會做一些處理
if (OSTimeGet() == Aperiodic_Arrival_job1) {
    printf("%2d\tAperiodic job (%d) arrives and sets CUS server's deadline as %2d.\n",OSTimeGet(), OSTCBPrioTbl[4]->job_ID,Aperiodic_period_job1);
}
else if (OSTimeGet() == Aperiodic_Arrival_job2) {
    if (Aperiodic_Arrival_job2 < Aperiodic_period_job1) {
        printf("%2d\tAperiodic job (1) arrives. Do nothing.\n", OSTimeGet());
        printf("%2d\tAperiodic job (1) sets CUS server's deadline as %2d.\n", OSTimeGet(), Aperiodic_period_job2);
    }
    else {
        printf("%2d\tAperiodic job (1) arrives and sets CUS server's deadline as %2d.\n", OSTimeGet(), Aperiodic_period_job2);
    }

}
```

Fig 15. OSTimeTick()顯示目前非週期性task的狀態