國立台灣科技大學
National Taiwan University of Science and Technology

Embedded OS Implementation, Fall 2020
Project #1 (due November 11, 2020 (Wednesday) 13:00)

老師：陳雅淑

學生：邱暐凱

一、 Task Control Block Linked List

1. screenshot results

```
C:\Users\user\Documents\Git\嵌入式作業系統實作HW\20201028_project1\
OSTick    created, Thread ID 16028
Task[ 63] created, TCB Address 00EF3260
-------After TCB[63] being linked-------
Previous TCB point to address 00000000
Current  TCB point to address 00EF3260
Next     TCB point to address 00000000

Task[  1] created, TCB Address 00EF32B8
-------After TCB[1] being linked-------
Previous TCB point to address 00000000
Current  TCB point to address 00EF32B8
Next     TCB point to address 00EF3260

Task[  2] created, TCB Address 00EF3310
-------After TCB[2] being linked-------
Previous TCB point to address 00000000
Current  TCB point to address 00EF3310
Next     TCB point to address 00EF32B8

================== TCB linked list ==================
Task     Prev_TCB_addr   TCB_addr        Next_TCB_addr
2        00000000        00EF3310        00EF32B8
1        00EF3310        00EF32B8        00EF3260
63       00EF32B8        00EF3260        00000000
```

2. implementation

利用一些內部變數(ptcb, ostcbpriotbl)來印出 linked list

os_cpu_c.c 列印 created 的部分

```
README.md   os_cpu_c.c   main.c   os_core.c   os_cpu_c.c (HEAD) ↔ os_cpu_c.c (b8e381c)
c: > Users > user > Documents > Git > 嵌入式作業系統實作HW > 20201028_project1 > Micrium_Win32_Kernel > Micrium_Win32_
    }

    p_stk->ThreadHandle = CreateThread(NULL, 0, OSTaskW32, p_tcb, CREATE_SUSPENDED, &p_stk
    if (p_stk->ThreadHandle == NULL) {
        CloseHandle(p_stk->InitSignalPtr);
        CloseHandle(p_stk->SignalPtr);

        p_stk->InitSignalPtr = NULL;
        p_stk->SignalPtr     = NULL;
#if (OS_MSG_TRACE > 0u)
        OS_Printf("Task[%3.1d] failed to be created.\n", p_tcb->OSTCBPrio);
#endif
        return;
    }

#if (OS_MSG_TRACE > 0u)
    // OS_Printf("Task[%3.1d] created, Thread ID %5.0d\n", p_tcb->OSTCBPrio, p_stk->Thread
    OS_Printf("Task[%3.1d] created, TCB Address %08X\n", p_tcb->OSTCBPrio, p_tcb);
    // printf("%08x %08x\n", p_tcb, p_stk);
#endif

    p_stk->TaskState = STATE_CREATED;
    p_stk->OSTCBPtr  = p_tcb;
}
#endif


/*$PAGE*/
```

os_core.c 列印創建當下 previous current next

```
README.md   os_cpu_c.c   main.c   os_core.c   os_core.c (HEAD) ↔ os_core.c (b8e381c)
c: > Users > user > Documents > Git > 嵌入式作業系統實作HW > 20201028_project1 > Micrium_Win32_Kernel > Micrium_Win32_
            OSTCBList->OSTCBPrev = ptcb;
        }
        OSTCBList               = ptcb;
        OSRdyGrp               |= ptcb->OSTCBBitY;         /* Make task ready to run
        OSRdyTbl[ptcb->OSTCBY] |= ptcb->OSTCBBitX;
        OSTaskCtr++;                                       /* Increment the #tasks counte
        OS_TRACE_TASK_READY(ptcb);
        OS_EXIT_CRITICAL();

        ////////////////////////// kevin ////////////////////////////////
        printf("-------After TCB[%d] being linked-------\n",ptcb->OSTCBPrio);
        printf("Previous TCB point to address %08X\n", ptcb->OSTCBPrev);
        printf("Current  TCB point to address %08X\n", ptcb);
        printf("Next     TCB point to address %08X\n\n", ptcb->OSTCBNext);
        ////////////////////////////////////////////////////////////////

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

os_core.c 最後在 start 列印所有創建的 task

```
void  OSStart (void)
{
    if (OSRunning == OS_FALSE) {
        OS_SchedNew();                              /* Find highest priority's task priority number   */
        OSPrioCur     = OSPrioHighRdy;
        OSTCBHighRdy  = OSTCBPrioTbl[OSPrioHighRdy]; /* Point to highest priority task ready to run    */
        OSTCBCur      = OSTCBHighRdy;
        Kevin_StartContextSwitches();
        OSStartHighRdy();                           /* Execute target specific code to start task     */
    }
}
```

os_core.c

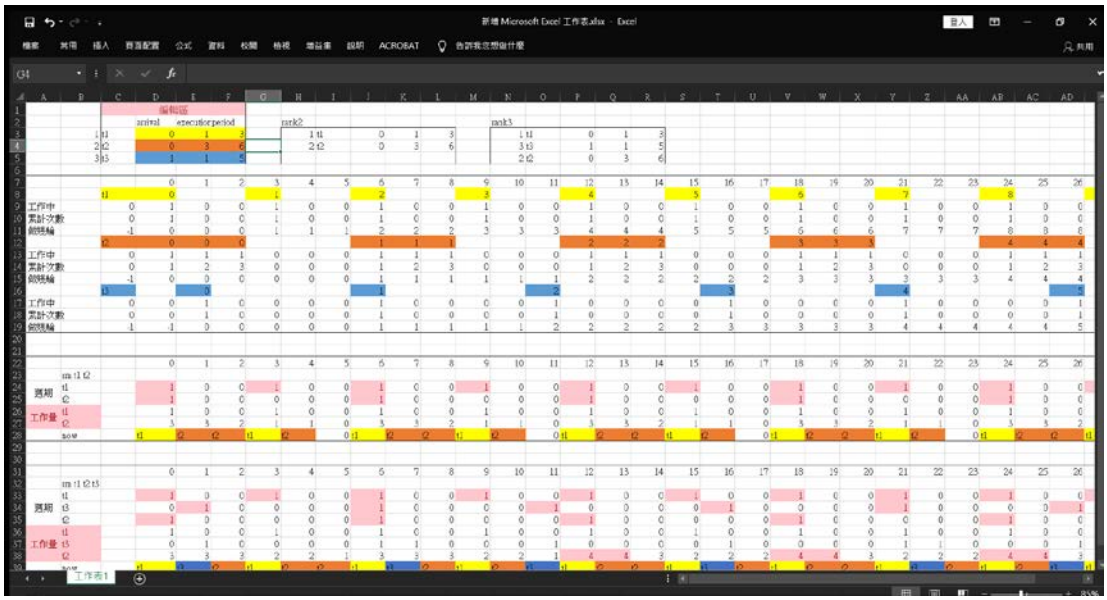
```
void Kevin_StartContextSwitches(void) {
    // kevin linked list
    printf("================== TCB linked list ==================\n");
    printf("Task \t Prev_TCB_addr \t TCB_addr \t Next_TCB_addr\n");
    for(int i = kevin_task_num; i > 0; i--)
        printf("%d \t %08X \t %08X \t %08X\n",i, OSTCBPrioTbl[i]->OSTCBPrev, OSTCBPrioTbl[i], OSTCBPrioTbl[i]->OSTCBNext);
    printf("%d \t %08X \t %08X \t %08X\n\n",63, OSTCBPrioTbl[63]->OSTCBPrev, OSTCBPrioTbl[63], OSTCBPrioTbl[63]->OSTCBNext);

    // kevin rm title
    printf("\nTick \t Event \t \t CurrentTask ID \t NextTask ID \t ResponseTime \t # of ContextSwitch \n"); // kevin title
}
```

二、 RM Scheduler Implementation

1. screenshot results

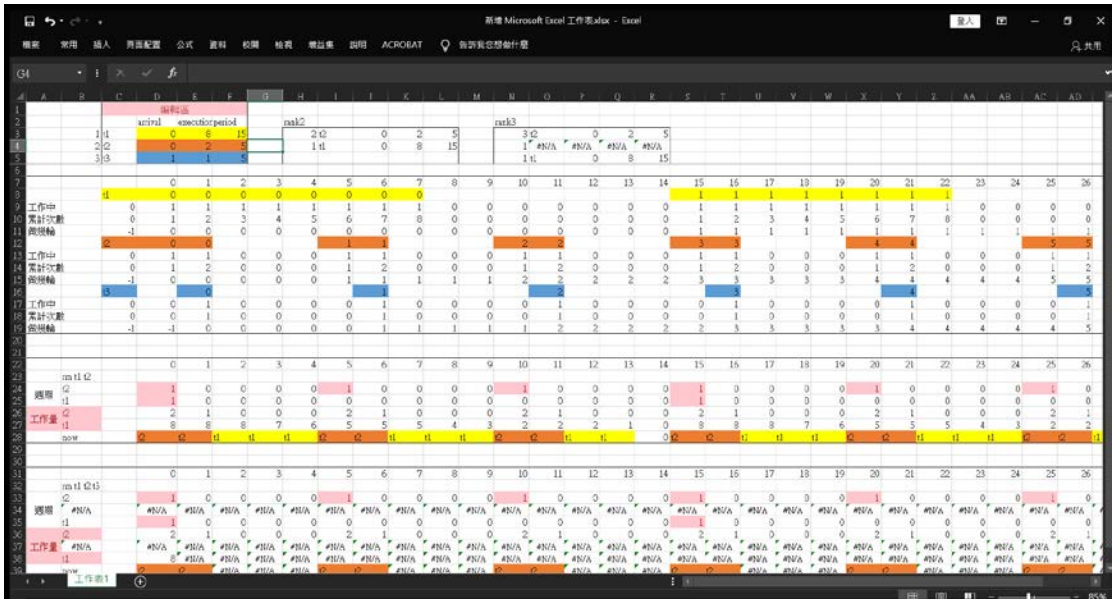
Task set 1 = {τ1 (0, 1, 3), τ2 (0, 3, 6)}

```
================= TCB linked list =================
Task      Prev_TCB_addr   TCB_addr        Next_TCB_addr
2         00DE3368        00DE3310        00DE32B8
1         00DE3310        00DE32B8        00DE3260
63        00DE32B8        00DE3260        00000000

Tick      Event           CurrentTask ID          NextTask ID     ResponseTime    # of ContextSwitch
1         Completion      task(1)(0)              task(2)(0)              1               1
3         Preemption      task(2)(0)              task(1)(1)
4         Completion      task(1)(1)              task(2)(0)              1               2
5         Completion      task(2)(0)              task(63)                5               4
6         Preemption      task(63)                task(1)(2)
7         Completion      task(1)(2)              task(2)(1)              1               2
9         Preemption      task(2)(1)              task(1)(3)
10        Completion      task(1)(3)              task(2)(1)              1               2
11        Completion      task(2)(1)              task(63)                5               4
12        Preemption      task(63)                task(1)(4)
13        Completion      task(1)(4)              task(2)(2)              1               2
15        Preemption      task(2)(2)              task(1)(5)
16        Completion      task(1)(5)              task(2)(2)              1               2
17        Completion      task(2)(2)              task(63)                5               4
18        Preemption      task(63)                task(1)(6)
19        Completion      task(1)(6)              task(2)(3)              1               2
21        Preemption      task(2)(3)              task(1)(7)
22        Completion      task(1)(7)              task(2)(3)              1               2
23        Completion      task(2)(3)              task(63)                5               4
24        Preemption      task(63)                task(1)(8)
25        Completion      task(1)(8)              task(2)(4)              1               2
27        Preemption      task(2)(4)              task(1)(9)
28        Completion      task(1)(9)              task(2)(4)              1               2
29        Completion      task(2)(4)              task(63)                5               4
30        Preemption      task(63)                task(1)(10)
```



Task set 2 = {τ1 (0, 8, 15), τ2 (0, 2, 5)}

```
================ TCB linked list =================
Task      Prev_TCB_addr   TCB_addr        Next_TCB_addr
2         00103368        00103310        001032B8
1         00103310        001032B8        00103260
63        001032B8        00103260        00000000

Tick      Event           CurrentTask ID          NextTask ID     ResponseTime    # of ContextSwitch
2         Completion      task(2)(0)              task(1)(0)              2               1
5         Preemption      task(1)(0)              task(2)(1)
7         Completion      task(2)(1)              task(1)(0)              2               2
10        Preemption      task(1)(0)              task(2)(2)
12        Completion      task(2)(2)              task(1)(0)              2               2
14        Completion      task(1)(0)              task(63)                14              6
15        Preemption      task(63)                task(2)(3)
17        Completion      task(2)(3)              task(1)(1)              2               2
20        Preemption      task(1)(1)              task(2)(4)
22        Completion      task(2)(4)              task(1)(1)              2               2
25        Preemption      task(1)(1)              task(2)(5)
27        Completion      task(2)(5)              task(1)(1)              2               2
29        Completion      task(1)(1)              task(63)                14              6
30        Preemption      task(63)                task(2)(6)
32        Completion      task(2)(6)              task(1)(2)              2               2
35        Preemption      task(1)(2)              task(2)(7)
```



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

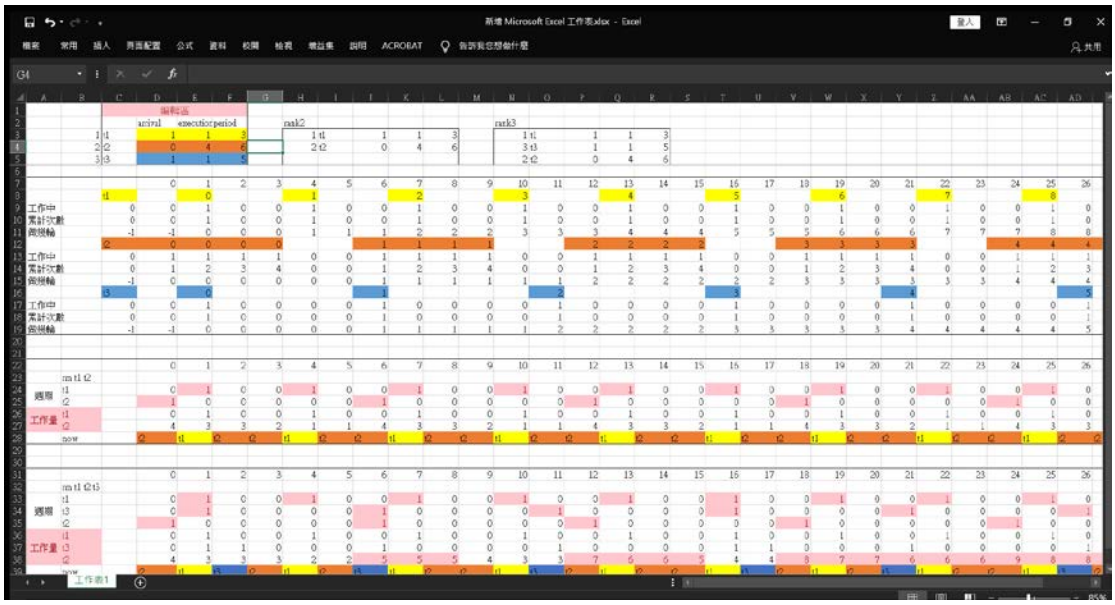
版本一：雖然 task2 做完繼續 task2 沒有經過 context switch，

但還是顯示方便觀測就變成 task(2)(0) task(2)(1)

```
C:\Users\user\Documents\Git\嵌入式作業系統實作HW\20201028_project1\Micrium_Win32_Kernel\Micrium_Win32_Kernel\Microsoft\Windows\...
Current  TCB point to address 00983260
Next     TCB point to address 00000000

Task[  1] created, TCB Address 009832B8
-------After TCB[1] being linked-------
Previous TCB point to address 00000000
Current  TCB point to address 009832B8
Next     TCB point to address 00983260

Task[  2] created, TCB Address 00983310
-------After TCB[2] being linked-------
Previous TCB point to address 00000000
Current  TCB point to address 00983310
Next     TCB point to address 009832B8

================= TCB linked list =================
Task     Prev_TCB_addr   TCB_addr        Next_TCB_addr
2        00000000        00983310        009832B8
1        00983310        009832B8        00983260
63       009832B8        00983260        00000000


Tick     Event           CurrentTask ID          NextTask ID     ResponseTime    # of ContextSwitch
1        Preemption      task(2)(0)              task(1)(0)
2        Completion      task(1)(0)              task(2)(0)              1               2
4        Preemption      task(2)(0)              task(1)(1)
5        Completion      task(1)(1)              task(2)(0)              1               2
6        Completion      task(2)(0)              task(2)(1)              6               4
7        Preemption      task(2)(1)              task(1)(2)
8        Completion      task(1)(2)              task(2)(1)              1               2
10       Preemption      task(2)(1)              task(1)(3)
11       Completion      task(1)(3)              task(2)(1)              1               2
12       Completion      task(2)(1)              task(2)(2)              6               4
13       Preemption      task(2)(2)              task(1)(4)
14       Completion      task(1)(4)              task(2)(2)              1               2
16       Preemption      task(2)(2)              task(1)(5)
17       Completion      task(1)(5)              task(2)(2)              1               2
18       Completion      task(2)(2)              task(2)(3)              6               4
19       Preemption      task(2)(3)              task(1)(6)
20       Completion      task(1)(6)              task(2)(3)              1               2
22       Preemption      task(2)(3)              task(1)(7)
23       Completion      task(1)(7)              task(2)(3)              1               2
24       Completion      task(2)(3)              task(2)(4)              6               4
25       Preemption      task(2)(4)              task(1)(8)
26       Completion      task(1)(8)              task(2)(4)              1               2
28       Preemption      task(2)(4)              task(1)(9)
29       Completion      task(1)(9)              task(2)(4)              1               2
30       Completion      task(2)(4)              task(2)(5)              6               4
```

版本二：就假設題目希望 context switch 時才列印，就會變這樣看不到 task2 completion，但他的 job 其實是有再增加

```
C:\Users\user\Documents\Git\嵌入式作業系統實作HW\20201028_project1\Micrium_Win32_Kernel\Micrium_Win32_Kernel\Microsoft\Windows\...

Task[  1] created, TCB Address 005A42B8
-------After TCB[1] being linked-------
Previous TCB point to address 00000000
Current  TCB point to address 005A42B8
Next     TCB point to address 005A4260

Task[  2] created, TCB Address 005A4310
-------After TCB[2] being linked-------
Previous TCB point to address 00000000
Current  TCB point to address 005A4310
Next     TCB point to address 005A42B8

================= TCB linked list =================
Task     Prev_TCB_addr   TCB_addr        Next_TCB_addr
2        00000000        005A4310        005A42B8
1        005A4310        005A42B8        005A4260
63       005A42B8        005A4260        00000000


Tick     Event           CurrentTask ID          NextTask ID     ResponseTime    # of ContextSwitch
1        Preemption      task(2)(0)              task(1)(0)
2        Completion      task(1)(0)              task(2)(0)              1               2
4        Preemption      task(2)(0)              task(1)(1)
5        Completion      task(1)(1)              task(2)(0)              1               2
7        Preemption      task(2)(1)              task(1)(2)
8        Completion      task(1)(2)              task(2)(1)              1               2
10       Preemption      task(2)(1)              task(1)(3)
11       Completion      task(1)(3)              task(2)(1)              1               2
13       Preemption      task(2)(2)              task(1)(4)
14       Completion      task(1)(4)              task(2)(2)              1               2
16       Preemption      task(2)(2)              task(1)(5)
17       Completion      task(1)(5)              task(2)(2)              1               2
19       Preemption      task(2)(3)              task(1)(6)
20       Completion      task(1)(6)              task(2)(3)              1               2
22       Preemption      task(2)(3)              task(1)(7)
23       Completion      task(1)(7)              task(2)(3)              1               2
25       Preemption      task(2)(4)              task(1)(8)
26       Completion      task(1)(8)              task(2)(4)              1               2
28       Preemption      task(2)(4)              task(1)(9)
29       Completion      task(1)(9)              task(2)(4)              1               2
31       Preemption      task(2)(5)              task(1)(10)
32       Completion      task(1)(10)             task(2)(5)              1               2
```



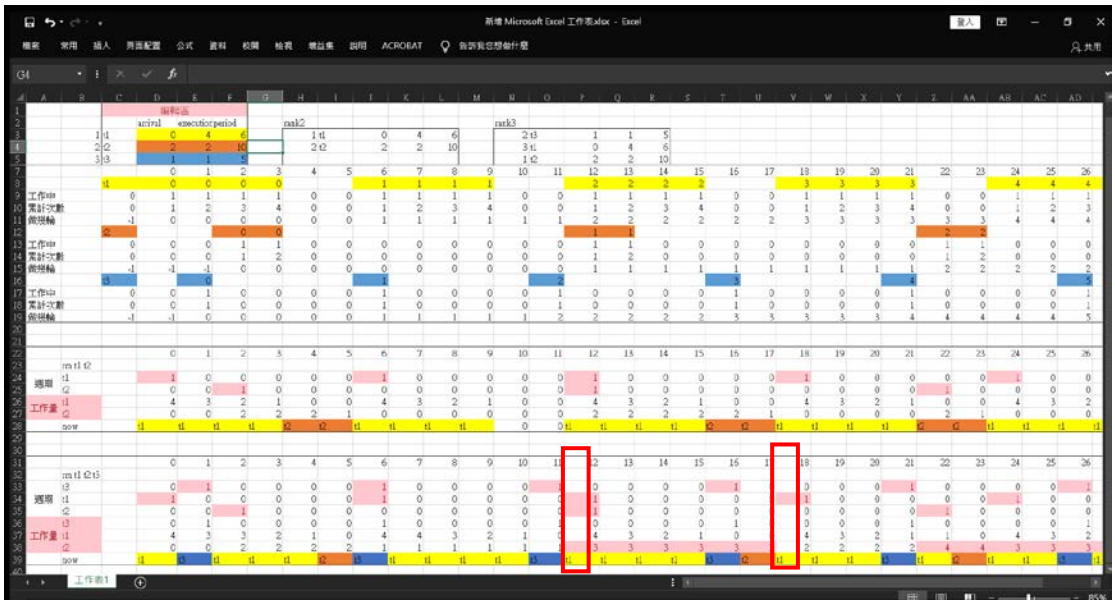
Task set 4 = {τ1 (0, 4, 6), τ2 (2, 2, 10), τ3 (1, 1, 5)}

版本一：在 context 才檢查自己 miss 為了符合標題 currenTaskID

```
C:\Users\user\Documents\Git\嵌入式作業系統實作HW\20201028_project1\Micrium_Win32_Kernel\Micrium_Win32_Kernel...
OSTick    created, Thread ID  2860
Task[ 63] created, TCB Address 00A63260
-------After TCB[63] being linked-------
Previous TCB point to address 00000000
Current  TCB point to address 00A63260
Next     TCB point to address 00000000

Task[  1] created, TCB Address 00A632B8
-------After TCB[1] being linked-------
Previous TCB point to address 00000000
Current  TCB point to address 00A632B8
Next     TCB point to address 00A63260

Task[  2] created, TCB Address 00A63310
-------After TCB[2] being linked-------
Previous TCB point to address 00000000
Current  TCB point to address 00A63310
Next     TCB point to address 00A632B8

Task[  3] created, TCB Address 00A63368
-------After TCB[3] being linked-------
Previous TCB point to address 00000000
Current  TCB point to address 00A63368
Next     TCB point to address 00A63310

================= TCB linked list =================
Task     Prev_TCB_addr   TCB_addr        Next_TCB_addr
3        00000000        00A63368        00A63310
2        00A63368        00A63310        00A632B8
1        00A63310        00A632B8        00A63260
63       00A632B8        00A63260        00000000

Tick     Event           CurrentTask ID          NextTask ID     ResponseTime    # of ContextSwitch
1        Preemption      task(1)(0)              task(3)(0)
2        Completion      task(3)(0)              task(1)(0)              1               2
5        Completion      task(1)(0)              task(2)(0)              5               3
6        Preemption      task(2)(0)              task(3)(1)
7        Completion      task(3)(1)              task(1)(1)              1               2
11       Completion      task(1)(1)              task(3)(2)              5               2
12       Completion      task(3)(2)              task(1)(2)              1               2
16       Completion      task(1)(2)              task(3)(3)              4               2
17       Completion      task(3)(3)              task(2)(0)              1               2
18       MissDeadline    task(2)(0)              ------------------
```

版本二：但其實在週期時檢查，工作量就已經

```
C:\Users\user\Documents\Git\嵌入式作業系統實作HW\20201028_project1\Micrium_Win32_Kernel\Micrium_Win32_Kernel\Microsoft\Windows\...
OSTick    created, Thread ID  6080
Task[ 63] created, TCB Address 00BF5CA0
-------After TCB[63] being linked-------
Previous TCB point to address 00000000
Current  TCB point to address 00BF5CA0
Next     TCB point to address 00000000

Task[  1] created, TCB Address 00BF5CF8
-------After TCB[1] being linked-------
Previous TCB point to address 00000000
Current  TCB point to address 00BF5CF8
Next     TCB point to address 00BF5CA0

Task[  2] created, TCB Address 00BF5D50
-------After TCB[2] being linked-------
Previous TCB point to address 00000000
Current  TCB point to address 00BF5D50
Next     TCB point to address 00BF5CF8

Task[  3] created, TCB Address 00BF5DA8
-------After TCB[3] being linked-------
Previous TCB point to address 00000000
Current  TCB point to address 00BF5DA8
Next     TCB point to address 00BF5D50

================= TCB linked list =================
Task     Prev_TCB_addr   TCB_addr        Next_TCB_addr
3        00000000        00BF5DA8        00BF5D50
2        00BF5DA8        00BF5D50        00BF5CF8
1        00BF5D50        00BF5CF8        00BF5CA0
63       00BF5CF8        00BF5CA0        00000000


Tick     Event           CurrentTask ID          NextTask ID     ResponseTime    # of ContextSwitch
1        Preemption      task(1)(0)              task(3)(0)
2        Completion      task(3)(0)              task(1)(0)              1               2
5        Completion      task(1)(0)              task(2)(0)              5               3
6        Preemption      task(2)(0)              task(3)(1)
7        Completion      task(3)(1)              task(1)(1)              1               2
11       Completion      task(1)(1)              task(3)(2)              5               2
12       MissDeadline    task(2)(0)              -------------------
```



2. implementation

取代了原本 OS_SchedNew 找 OSPrioHighRdy 的方式，

改由自己建立的 RMS 找到當下的 OSPrioHighRdy ，

全部都在 kernal 完成，main 只負責卡著任務不斷重複。

ucos_ii.h

```
OS_EXT  OS_TMR_WHEEL      OSTmrWheelTbl[OS_TMR_CFG_WHEEL_SIZE];
#endif

extern  INT8U   const     OSUnMapTbl[256];          /* Priority->Index    lookup table                */


////////////////////////////////////////////// kevin ////////////////////////////////////////////////

struct KevinPeriodic {
    INT8U arrival;      // 開始時間
    INT8U execution;    // 每次工作量
    INT8U period;       // 週期期限
    INT8U job;          // 紀錄工作機輪
    INT8U response;     // 紀錄所花時間
    INT8U context;      // 紀錄切換次數
    INT8U work;         // 紀錄當前工作
};

struct KevinPeriodic kevin_arr_task_periodic[4];

struct KevinPeriodic *kevin_task1_periodic;
struct KevinPeriodic *kevin_task2_periodic;
struct KevinPeriodic *kevin_task3_periodic;

INT8U kevin_task_num;
INT8U kevin_arr_short[4]; // 優先順序

////////////////////////////////////////////////////////////////////////////////////////////////////

/*
*********************************************************************************************************
*                                                FUNCTION PROTOTYPES
```

os_cfg.h

```
#define OS_TASK_DEL_EN          1u   /*    Include code for OSTaskDel()                          */
#define OS_TASK_NAME_EN         1u   /*    Enable task names                                     */
#define OS_TASK_PROFILE_EN      1u   /*    Include variables in OS_TCB for profiling             */
#define OS_TASK_QUERY_EN        1u   /*    Include code for OSTaskQuery()                        */
#define OS_TASK_REG_TBL_SIZE    1u   /*    Size of task variables array (#of INT32U entries)     */
#define OS_TASK_STAT_EN         0u   /*    Enable (1) or Disable(0) the statistics task          */ // kevin
#define OS_TASK_STAT_STK_CHK_EN 1u   /*    Check task stacks from statistic task                 */
```

```
                                     /* -------------------- TIME MANAGEMENT --------------------- */
#define OS_TIME_DLY_HMSM_EN     1u   /*    Include code for OSTimeDlyHMSM()                      */
#define OS_TIME_DLY_RESUME_EN   1u   /*    Include code for OSTimeDlyResume()                    */
#define OS_TIME_GET_SET_EN      1u   /*    Include code for OSTimeGet() and OSTimeSet()          */
#define OS_TIME_TICK_HOOK_EN    1u   /*    Include code for OSTimeTickHook()                     */


                                     /* -------------------- TIMER MANAGEMENT -------------------- */
#define OS_TMR_EN               0u   /* Enable (1) or Disable (0) code generation for TIMERS     */ // kevin
```

main.c 基本上沒什麼改

Terminal Help main.c (HEAD) ↔ main.c (b8e381c) - 嵌入式作業系統實作HW - Visual Studio Code

README.md | os_cpu_c.c | main.c | os_core.c | ucos_ii.h | main.c (HEAD) ↔ main.c (b8e381c)

C > Users > user > Documents > Git > 嵌入式作業系統實作HW > 20201028_project1 > Micrium_Win32_Kernel > Micrium_Win32_Kernel > Microsoft

```c


/*
*********************************************************************************************************
*                                                LOCAL DEFINES
*********************************************************************************************************
*/

#define TASK_STACKSIZE   2048
#define TASK1_PRIORITY   1      // kevin 設多少都沒差後面會改
#define TASK2_PRIORITY   2      // kevin 設多少都沒差後面會改
#define TASK3_PRIORITY   3      // kevin 設多少都沒差後面會改
#define TASK1_ID         1
#define TASK2_ID         2
#define TASK3_ID         3
```

Terminal Help main.c (HEAD) ↔ main.c (b8e381c) - 嵌入式作業系統實作HW - Visual Studio Code

README.md | os_cpu_c.c | main.c | os_core.c | ucos_ii.h | main.c (HEAD) ↔ main.c (b8e381c)

C > Users > user > Documents > Git > 嵌入式作業系統實作HW > 20201028_project1 > Micrium_Win32_Kernel > Micrium_Win32_Kernel > Microsoft > Wi

```c
static  OS_STK  Task1_STK[TASK_STACKSIZE];
static  OS_STK  Task2_STK[TASK_STACKSIZE];
static  OS_STK  Task3_STK[TASK_STACKSIZE];

/*
*********************************************************************************************************
*                                             FUNCTION PROTOTYPES
*********************************************************************************************************
*/

static  void  StartupTask (void  *p_arg);

static void task1(void* p_arg);
static void task2(void* p_arg);
static void task3(void* p_arg); // kevin

```

Terminal Help main.c (HEAD) ↔ main.c (b8e381c) - 嵌入式作業系統實作HW - Visual Studio Code

README.md | os_cpu_c.c | main.c | os_core.c | ucos_ii.h | main.c (HEAD) ↔ main.c (b8e381c)

C > Users > user > Documents > Git > 嵌入式作業系統實作HW > 20201028_project1 > Micrium_Win32_Kernel > Micrium_Win32_Kernel > Microsoft >

```c
        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));

    if(kevin_task_num >= 3) // kevin
        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));

    OSTimeSet(0);    // kevin reset time
    OSStart();                                          /* Start multitasking (i.e. give control to u
```

Terminal Help main.c (HEAD) ↔ main.c (b8e381c) - 嵌入式作業系統實作HW - Visual Studio Code

README.md | os_cpu_c.c | main.c | os_core.c | ucos_ii.h | main.c (HEAD) ↔ main.c (b8e381c)

c: > Users > user > Documents > Git > 嵌入式作業系統實作HW > 20201028_project1 > Micrium_Win32_Kernel > Micrium_Win32_Kernel > Microsoft > Win

```c
    while (DEF_TRUE) {                                          /* Task body, always written as an infinite loo
        OSTimeDlyHMSM(0u, 0u, 1u, 0u);
        APP_TRACE_DBG(("Time: %d\n\r", OSTimeGet()));
    }
}

void task1(void* p_arg) {
    (void)p_arg;
    while (1) {
        //printf("Hello from task1\n");
        while (1); // kevin 讓他一直卡在裡面 靠OSintexit來切
    }
}

void task2(void* p_arg) {
    (void)p_arg;
    while (1) {
        //printf("Hello from task2\n");
        while (1); // kevin 讓他一直卡在裡面 靠OSintexit來切
    }
}

void task3(void* p_arg) {
    (void)p_arg;
    while (1) {
        //printf("Hello from task3\n");
        while (1); // kevin 讓他一直卡在裡面 靠OSintexit來切
```

os_core.c functions 大部分都寫在這

os_core.c (HEAD) ↔ os_core.c (b8e381c) - 嵌入式作業系統實作HW - Visual Studio Code

c: > Users > user > Documents > Git > 嵌入式作業系統實作HW > 20201028_project1 > Micrium_Win32_Kernel > Micrium_Win32_Kernel > Micrium > Software > uCOS-II > Source

```c
////////////////////////////////////// kevin print function ////////////////////////////////////////////////
void Kevin_StartContextSwitches(void) {
    // kevin linked list
    printf("================= TCB linked list ==================\n");
    printf("Task \t Prev_TCB_addr \t TCB_addr \t Next_TCB_addr\n");
    for(int i = kevin_task_num; i > 0; i--)
        printf("%d \t %08X \t %08X \t %08X\n",i, OSTCBPrioTbl[i]->OSTCBPrev, OSTCBPrioTbl[i], OSTCBPrioTbl[i]->OSTCBNext);
    printf("%d \t %08X \t %08X \t %08X\n\n",63, OSTCBPrioTbl[63]->OSTCBPrev, OSTCBPrioTbl[63], OSTCBPrioTbl[63]->OSTCBNext);

    // kevin rm title
    printf("\nTick \t Event \t \t CurrentTask ID \t NextTask ID \t ResponseTime \t # of ContextSwitch \n"); // kevin title
}
```

os_core.c (HEAD) ↔ os_core.c (b8e381c) - 嵌入式作業系統實作HW - Visual Studio Code

c: > Users > user > Documents > Git > 嵌入式作業系統實作HW > 20201028_project1 > Micrium_Win32_Kernel > Micrium_Win32_Kernel > Micrium > Software > uCOS-II > S

```c
void Kevin_ContextSwitches(void) {
    // context counting
    for(int i = 1; i <= kevin_task_num; i++) // 每個 task
    {
        if(kevin_arr_task_periodic[i].work < kevin_arr_task_periodic[i].execution || OSPrioHighRdy == i) // 當這task開始做
        {
            kevin_arr_task_periodic[i].context++; // counter
            // printf("task:%d work:%d\n", i, kevin_arr_task_periodic[i].work);
        }
    }

    // print
    Kevin_print();
}
```

os_core.c - 嵌入式作業系統實作HW - Visual Studio Code

20201028_project1 > Micrium_Win32_Kernel > Micrium_Win32_Kernel > Micrium > Software > uCOS-II > Source > os_core.c > Kevin_OS_SchedNew(void)

```c
    Kevin_print();
}

void Kevin_OS_SchedNew(void) {

    // counter
    if (OSPrioHighRdy <= kevin_task_num && OSTime != 0) // 排除 idle 跟開始
        kevin_arr_task_periodic[OSPrioHighRdy].work--; // 計算execution被做掉了幾次

    // periodic plus work
    for(int i = 1; i <= kevin_task_num; i++) // 每個 task
    {
        kevin_arr_task_periodic[i].response++; // counter 紀錄從週期開始響應時間

        // periodic plus work
        int kevin_OSTime_arrival = (OSTime - kevin_arr_task_periodic[i].arrival); // arrival
        if(kevin_OSTime_arrival >= 0 && kevin_OSTime_arrival % kevin_arr_task_periodic[i].period  == 0) // 排除0取餘數 找到週期
        {
            if(OSPrioHighRdy == i && kevin_arr_task_periodic[i].work == 0){ // 做完了但接著做 不會進context 所以在這job++ 印出來
                printf("%d \t Completion \t task(%d)(%d) \t \t task(%d)(%d) \t \t %d \t \t %d \n"
                , OSTime, OSPrioHighRdy,kevin_arr_task_periodic[OSPrioHighRdy].job++, OSPrioHighRdy, kevin_arr_task_periodic[OSPrioHighRdy].job, kevi
                // kevin_arr_task_periodic[OSPrioHighRdy].job++; // 或是不列印純job++
            }
            if(kevin_arr_task_periodic[i].work != 0){ // 發工作時發現miss
                printf("%d \t MissDeadline \t task(%d)(%d) \t \t -------------------\n", OSTime, i, kevin_arr_task_periodic[i].job);
                while (1);
            }
            kevin_arr_task_periodic[i].work += kevin_arr_task_periodic[i].execution; // 發工作
            kevin_arr_task_periodic[i].response = 0;
            kevin_arr_task_periodic[i].context = 0;
        }
        // printf("task:%d OSTime:%d %d work:%d\n", i,(OSTime - kevin_arr_task_periodic[i].arrival), (OSTime - kevin_arr_task_periodic[i].arrival) %
    }

    // find OSPrioHighRdy
    OSPrioHighRdy = 63; // find OSPrioHighRdy 先假設沒人要做事
    for(int i = 1; i <= kevin_task_num; i++) // 每個 task
    {
        if (kevin_arr_task_periodic[kevin_arr_short[i]].work != 0) // 從 週期短 高優先 開始找有事做的
        {
            OSPrioHighRdy = kevin_arr_short[i];
            break;
        }
    }

    // printf("OS_SchedNew OSTime:%d OSPrioHighRdy:%d work:%d \n", OSTime, OSPrioHighRdy, kevin_arr_task_periodic[OSPrioHighRdy].work); // kevin
}

void Kevin_OSInit(void){
```

os_core.c (HEAD) ↔ os_core.c (b8e381c) - 嵌入式作業系統實作HW - Visual Studio Code

c: > Users > user > Documents > Git > 嵌入式作業系統實作HW > 20201028_project1 > Micrium_Win32_Kernel > Micrium_Win32_Kernel > Micrium > Software > uCOS-II > Source > os_core.c

```c
void Kevin_OSInit(void){
    // define
    kevin_task1_periodic = &kevin_arr_task_periodic[1];
    kevin_task2_periodic = &kevin_arr_task_periodic[2];
    kevin_task3_periodic = &kevin_arr_task_periodic[3];

    kevin_task1_periodic->arrival = 0;
    kevin_task1_periodic->execution = 1;
    kevin_task1_periodic->period = 3;

    kevin_task2_periodic->arrival = 0;
    kevin_task2_periodic->execution = 3;
    kevin_task2_periodic->period = 6;

    kevin_task3_periodic->arrival = 1;      // 1
    kevin_task3_periodic->execution = 1;    // 1
    kevin_task3_periodic->period = 5;       // 5

    kevin_task_num = 2;

    // kevin sort
    for(int i = 1; i <= kevin_task_num; i++)
    {
        int k = 1;
        for(int j = 1; j <= kevin_task_num; j++)
            if(kevin_arr_task_periodic[i].period > kevin_arr_task_periodic[j].period)
                k++;
        kevin_arr_short[k] = i;
        // printf("=>task%d short:%d short:%d\n", i,kevin_arr_task_periodic[i].sort, kevin_arr_short[k]);
    }

    // kevin print task rm show
    for(int i = 1; i <= kevin_task_num; i++)
        printf("task:%d (%d,%d,%d),", i, kevin_arr_task_periodic[i].arrival,kevin_arr_task_periodic[i].execution,kevin_arr_task_periodic[i].period);
    printf("\n\n");
}

void Kevin_print(void){
    // tick
```

os_core.c (HEAD) ↔ os_core.c (b8e381c) - 嵌入式作業系統實作HW - Visual Studio Code

c: > Users > user > Documents > Git > 嵌入式作業系統實作HW > 20201028_project1 > Micrium_Win32_Kernel > Micrium_Win32_Kernel > Micrium > Software > uCOS-II > Source > os_core.c

```c
    printf("\n\n");
}

void Kevin_print(void){
    // tick
    printf("%d \t ",OSTime);

    // Event
    if(kevin_arr_task_periodic[OSTCBCur->OSTCBPrio].work == 0 && OSTCBCur->OSTCBPrio != 63)
        printf("Completion \t ");
    else if (kevin_arr_task_periodic[OSTCBCur->OSTCBPrio].work >= kevin_arr_task_periodic[OSTCBCur->OSTCBPrio].execution && OSTCBCur->OSTCBPrio != 63)
    {
        printf("MissDeadline \t task(%d)(%d) \t \t ------------------\n ", OSTCBCur->OSTCBPrio, kevin_arr_task_periodic[OSTCBCur->OSTCBPrio].job);
        while (1);
    }
    else
        printf("Preemption \t ");

    // CurrentTask ID
    printf("task(%d)",OSTCBCur->OSTCBPrio);
    if(OSTCBCur->OSTCBPrio != 63)
        printf("(%d) ", kevin_arr_task_periodic[OSTCBCur->OSTCBPrio].job);
    printf("\t \t ");

    // NextTask ID
    printf("task(%d)",OSTCBHighRdy->OSTCBPrio);
    if(OSTCBHighRdy->OSTCBPrio != 63)
        printf("(%d) ", kevin_arr_task_periodic[OSTCBHighRdy->OSTCBPrio].job);

    // ResponseTime ContextSwitch
    if(kevin_arr_task_periodic[OSTCBCur->OSTCBPrio].work == 0 && OSTCBCur->OSTCBPrio != 63)
    {
        printf("\t \t %d \t \t %d ", kevin_arr_task_periodic[OSTCBCur->OSTCBPrio].response, kevin_arr_task_periodic[OSTCBCur->OSTCBPrio].context);
        kevin_arr_task_periodic[OSTCBCur->OSTCBPrio].job++;
    }
    printf("\n");
}
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

os_core.c function 對應位置

os_core.c (HEAD) ↔ os_core.c (b8e381c) - 嵌入式作業系統實作HW - Visual Studio Code

```c

static void OS_SchedNew(void);

//////////////////////////////// kevin function ////////////////////////////////
void Kevin_StartContextSwitches(void);
void Kevin_ContextSwitches(void);
void Kevin_OS_SchedNew(void);
void Kevin_OSInit(void);
void Kevin_print(void);
////////////////////////////////////////////////////////////////////////////////
```

os_core.c (HEAD) ↔ os_core.c (b8e381c) - 嵌入式作業系統實作HW - Visual Studio Code

```c

void OSInit (void)
{
    // kevin init
    Kevin_OSInit();

#if OS_TASK_CREATE_EXT_EN > 0u
```

os_core.c (HEAD) ↔ os_core.c (b8e381c) - 嵌入式作業系統實作HW - Visual Studio Code

```c

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) {
        OS_ENTER_CRITICAL();
        if (OSIntNesting > 0u) {                      /* Prevent OSIntNesting from 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 */

#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();
                    Kevin_ContextSwitches(); // Kevin ContextSwitches
```

os_core.c (HEAD) ↔ os_core.c (b8e381c) - 嵌入式作業系統實作HW - Visual Studio Code

```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 OS_TASK_CREATE_EXT_EN > 0u
#if defined(OS_TLS_TBL_SIZE) && (OS_TLS_TBL_SIZE > 0u)
                OS_TLS_TaskSw();
#endif
#endif
                Kevin_ContextSwitches(); // Kevin ContextSwitches
                OS_TASK_SW();                         /* Perform a context switch                     */
```

os_core.c (HEAD) ↔ os_core.c (b8e381c) - 嵌入式作業系統實作HW - Visual Studio Code

```c
*/

static  void  OS_SchedNew (void)
{
#if OS_LOWEST_PRIO <= 63u                         /* See if we support up to 64 tasks                 */
    INT8U   y;

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

    Kevin_OS_SchedNew(); // kevin find OSPrioHighRdy
```

os_core.c (HEAD) ↔ os_core.c (b8e381c) - 嵌入式作業系統實作HW - Visual Studio Code

Kevin_StartContextSwitches 2 of 3

```c
*/

void  OSStart (void)
{
    if (OSRunning == OS_FALSE) {
        OS_SchedNew();                            /* Find highest priority's task priority number   */
        OSPrioCur     = OSPrioHighRdy;
        OSTCBHighRdy  = OSTCBPrioTbl[OSPrioHighRdy]; /* Point to highest priority task ready to run    */
        OSTCBCur      = OSTCBHighRdy;
        Kevin_StartContextSwitches();
        OSStartHighRdy();                         /* Execute target specific code to start task     */
    }
}
```