110.1 Embedded OS Implementation
PA#1 Report

四電機四乙 B10707119 陳俊宇

2021/11/10

[PART I] Task Control Block Linked List

Screenshot of result

```
OSTick    created, Thread ID 53644
Task[ 63] created, TCB Address  0068EAC0
------After TCB[63] being linked------
Previous TCB pint to address  00000000
Current  TCB pint to address  0068EAC0
Next     TCB pint to address  00000000

Task[  1] created, TCB Address  0068EB18
------After TCB[1] being linked------
Previous TCB pint to address  00000000
Current  TCB pint to address  0068EB18
Next     TCB pint to address  0068EAC0

Task[  2] created, TCB Address  0068EB70
------After TCB[2] being linked------
Previous TCB pint to address  00000000
Current  TCB pint to address  0068EB70
Next     TCB pint to address  0068EB18

============ TCB linked list ============
Task    Prev_TCB_addr   TCB_addr        Next_TCB_addr
2       00000000        0068EB70        0068EB18
1       0068EB70        0068EB18        0068EAC0
63      0068EB18        0068EAC0        00000000
```

Screenshot of modified code

OS_TCBInit()

```
        OS_ENTER_CRITICAL();
        ptcb->OSTCBNext = OSTCBList;                    /* Link into TCB chain                     */
        ptcb->OSTCBPrev = (OS_TCB *)0;
        if (OSTCBList != (OS_TCB *)0) {
            OSTCBList->OSTCBPrev = ptcb;
        }
        OSTCBList               = ptcb;


        /// PA#1 part1
        printf("Task[%3.0d] created, TCB Address %9.0p\n", ptcb->OSTCBPrio, ptcb);
        printf("------After TCB[%d] being linked------\n", ptcb->OSTCBPrio);
        printf("Previous TCB pint to address %9.0p\n", ptcb->OSTCBPrev);   /// (OS_TCB *)0為pointer type，用%p
        printf("Current  TCB pint to address %9.0p\n", ptcb);
        printf("Next     TCB pint to address %9.0p\n\n", ptcb->OSTCBNext);
        /// PA#1 part1


        OSRdyGrp               |= ptcb->OSTCBBitY;      /* Make task ready to run                  */
        OSRdyTbl[ptcb->OSTCBY] |= ptcb->OSTCBBitX;
        OSTaskCtr++;                                    /* Increment the #tasks counter            */
        OS_TRACE_TASK_READY(ptcb);
        OS_EXIT_CRITICAL();
        return (OS_ERR_NONE);
    }
    OS_EXIT_CRITICAL();
    return (OS_ERR_TASK_NO_MORE_TCB);
}
```

main()

```c
/// RM scheduling rules : task with smallest time period will have highest priority
/// insertion sort by TaskPeriodic => TaskParameter[0]為period最小的Task，升序排列
int i, j, key;
task_para_set tmp_TaskParameter;
for (j = 1;j < TASK_NUMBER;j++)
{
    key = TaskParameter[j].TaskPeriodic;
    tmp_TaskParameter = TaskParameter[j];
    i = j - 1;
    while (i >= 0 && TaskParameter[i].TaskPeriodic > key)
    {
        TaskParameter[i + 1] = TaskParameter[i];
        i = i - 1;
    }
    TaskParameter[i + 1] = tmp_TaskParameter;
}

/// PA#1，建立Task
for (n = 0;n < TASK_NUMBER;n++)
{
    /// printf("ID %d, prio %d\n", TaskParameter[n].TaskID, n+1);
    OSTaskCreateExt(task1,                          /// task function
        &TaskParameter[n],                          /// p_arg(給task function的參數)
        &Task_STK[n][TASK_STACKSIZE - 1],           /// ptos
        n+1,                                        /// prio, PA#1要求從1開始
        TaskParameter[n].TaskID,                    /// id
        &Task_STK[n][0],                            /// pbos
        TASK_STACKSIZE,                             /// stack size
        &TaskParameter[n],                          /// pext(TCB extentsion的pointer)
        (OS_TASK_OPT_STK_CHK | OS_TASK_OPT_STK_CLR));   /// opt
}

/// PA#1 part1
printf("=========== TCB linked list ===========\n");
printf("Task\tPrev_TCB_addr\tTCB_addr\tNext_TCB_addr\n");
for (OS_TCB* ptcb = OSTCBList; ptcb != (OS_TCB*)0; ptcb = ptcb->OSTCBNext)
{
    printf("%d\t%p\t%p\t%p\n", ptcb->OSTCBPrio, ptcb->OSTCBPrev, ptcb, ptcb->OSTCBNext);
}
/// PA#1 part1
```

Description of implementation

從 HW#1 進行修改，InputFile()讀取 TaskSet.txt 後，使用 for 迴圈建立 task，確保程式能夠應對數量不定的 TaskSet。

在 OS_TCBInit() 中，完成 TCB 各項資料(OSTCBPrio, OSTCBPrev, OSTCBNext……)初始化後，輸出 TCB linked list 的 pointer。所有 task 建立完成後，輸出完整的 TCB linked list。

此外，為了符合 PART II 的 RM scheduling，task 會先依照 TaskPeriod 進行排序，period 最短者有最高的 priority。

[PART II] RM Scheduler Implementation

(未以紅色括弧標示之截圖表示全部為新程式碼)

task function

```c
void task1(void* p_arg) {
    task_para_set* task_data;
    task_data = p_arg;

    int job_ready_time = task_data->TaskArriveTime;  /// 第一個job什麼時候被排在schedule上
    int next_job_ready_time;
    int exe_time;
    int job_start_time;

    if (OSTimeGet() < task_data->TaskArriveTime)
    {
        OSTimeDly( task_data->TaskArriveTime - OSTimeGet());
    }
    while (1)
    {
        OSTCBCur->self_continue = 0;
        next_job_ready_time = job_ready_time + task_data->TaskPeriodic;
        OSTCBCur->next_job_time = next_job_ready_time;
        exe_time = task_data->TaskExecutionTime;
        OSTCBCur->remain_exe_time = exe_time;
        job_start_time = OSTimeGet();

        while (1)
        {
            if (OSTimeGet() != job_start_time)
            {
                exe_time--;
                OSTCBCur->remain_exe_time = exe_time;
                ///printf("%2d\tTaskID %2d remain exe_time : %2d\n", OSTimeGet(), task_data->TaskID, exe_time);
                if (exe_time == 0)
                {
                    break;
                }
                job_start_time = OSTimeGet();
            }
        }

        job_ready_time = next_job_ready_time;  /// 預先更新

        if (OSTCBCur->already_delay == 1)
        {
            OSTCBCur->already_delay = 0;
            continue;
        }

        int delay_time = next_job_ready_time - OSTimeGet();
        if (delay_time == 0)
        {

            OSTCBCur->self_continue = 1;
            OS_Sched();
        }
        else if (delay_time < 0)
        {
            printf("TaskID %2d miss deadline at tick %2d !\n", task_data->TaskID, OSTimeGet());
        }
        /// printf("TaskID : %2d\tend exe at tick %2d, next job start at tick %2d\n", task_data->TaskID, OSTimeGet(), OSTi
        OSTimeDly(delay_time);
    }
}
```

根據給定的 arrive time, execution time, period 進行運作的 task funtion。execution 期間卡在 while 迴圈中不斷檢查執行時間是否結束，結束後則使用 OSTimeDly()進入 waiting 狀態，直到下個周期開始。

ucos_ii.h

OS_TCB 額外建立的變數

```c
INT32U           OSTCBDly;             /* Nbr ticks to delay task or, timeout waiting for event   */
INT8U            OSTCBStat;            /* Task      status                                         */
INT8U            OSTCBStatPend;        /* Task PEND status                                         */
INT8U            OSTCBPrio;            /* Task priority (0 == highest)                             */

INT8U            OSTCBX;               /* Bit position in group  corresponding to task priority   */
INT8U            OSTCBY;               /* Index into ready table corresponding to task priority   */
OS_PRIO          OSTCBBitX;            /* Bit mask to access bit position in ready table           */
OS_PRIO          OSTCBBitY;            /* Bit mask to access bit position in ready group           */

/// PA#1
INT8U already_delay;
INT8U remain_exe_time;
INT8U next_job_time;
INT8U self_continue;
/// PA#1
```

Task function 執行時的某些資訊(如：task 剩餘的 execution time)需要儲存到 TCB 中，才有辦法在 kernel level 讀取。

os_core.c

額外建立的變數

```c
/// PA#1，紀錄每個task已執行幾次
int job_number[64] = { 0 };

/// PA#1，用於確認OSRdyTbl中的某個priority是否為1
INT8U OSMapTbl[8] = { 0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80 };

/// PA#1，用於計算各個Task的response time、紀錄被preempt的次數、時間(# of ContextSwitch, PreemptionTime)
int task_start_time[64] = { 0 };  /// 在OSTimeDly()把task設為ready時紀錄
int task_preempt_count[64] = { 0 };
int task_preempt_time[64] = { 0 };
int task_preempt_time_acc[64] = { 0 };
```

OSMapTbl 用於查詢 OSRdyTbl，確認某個 priority 的 task 是否為 ready 狀態，其餘 array 則用於記錄需要輸出的資料。

OSTimeTick()

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

    /// PA#1
    task_start_time[ptcb->OSTCBPrio] = OSTimeGet();
    /// PA#1

    OS_TRACE_TASK_READY(ptcb);
}
```

記錄 task 被設為 ready 的時間，用於後續計算 response time。

OS_SchedNew()

```c
/// PA#1，控制輸出第一列
int first_line = 1;
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]]);

    /// PA#1
    if ((Output_err = fopen_s(&Output_fp, "./Output.txt", "a")) != 0)
    {
        printf("can't open Output.txt");
        exit(0);
    }

    if (first_line == 1)  /// PA#1，輸出第一列
    {
        printf("\nTick\tEvent\t\tCurrentTask ID\tNextTask ID\tResponseTime\t# of ContextSwitch\tPreemptionTime\n");
        first_line = 0;
    }

    /// =======================================特殊Completion=======================================
    /// task的job結束後，緊接者又是同個task的下個job，如Task Set 3 tick 15
    if (OSPrioCur == OSPrioHighRdy)
    {
        if (OSTCBCur->self_continue == 1)
        {
            printf("%2d\t", OSTimeGet());
            fprintf(Output_fp, "%2d\t", OSTimeGet());

            printf("Completion\t");
            fprintf(Output_fp, "Completion\t");
            /// CurrentTask
            if (OSPrioCur == 63)  /// 其實不會發生這種情況
            {
                printf("Task(%2d)\t", OSPrioCur);
                fprintf(Output_fp, "Task(%2d)\t", OSPrioCur);
            }
            else
            {
                printf("Task(%2d)(%2d)\t", OSTCBPrioTbl[OSPrioCur]->OSTCBId, job_number[OSPrioCur]);
                fprintf(Output_fp, "Task(%2d)(%2d)\t", OSTCBPrioTbl[OSPrioCur]->OSTCBId, job_number[OSPrioCur]);
            }
            /// NextTask
            job_number[OSPrioCur]++;
            if (OSPrioHighRdy == 63)
            {
                printf("Task(%2d)\t", OSPrioHighRdy);
                fprintf(Output_fp, "Task(%2d)\t", OSPrioHighRdy);
            }
            else
            {
                printf("Task(%2d)(%2d)\t", OSTCBPrioTbl[OSPrioHighRdy]->OSTCBId, job_number[OSPrioHighRdy]);
                fprintf(Output_fp, "Task(%2d)(%2d)\t", OSTCBPrioTbl[OSPrioHighRdy]->OSTCBId, job_number[OSPrioHighRdy]);
            }
            /// ResponseTime
            printf("%5d\t\t", (OSTimeGet() - task_start_time[OSPrioCur]));
            fprintf(Output_fp, "%5d\t\t", (OSTimeGet() - task_start_time[OSPrioCur]));
            /// # of ContextSwitch
            /// task_preempt_count[OSPrioCur]++; /// same task
            printf("%5d\t\t\t", task_preempt_count[OSPrioCur]);
            fprintf(Output_fp, "%5d\t\t\t", task_preempt_count[OSPrioCur]);
            /// PreemptionTime
            printf("%5d\n", task_preempt_time_acc[OSPrioCur]);
            fprintf(Output_fp, "%5d\n", task_preempt_time_acc[OSPrioCur]);

            task_preempt_time[OSPrioCur] = 0;
            task_preempt_count[OSPrioCur] = 0;
            task_preempt_time_acc[OSPrioCur] = 0;

```

```
        if (task_preempt_time[OSPrioHighRdy] != 0)  /// 有被preempt的紀錄
        {
            task_preempt_time_acc[OSPrioHighRdy] = task_preempt_time_acc[OSPrioHighRdy] + (OSTimeGet() - task_preempt_
        }

        /// task_preempt_count[OSPrioHighRdy]++;  /// same task

        task_start_time[OSTCBCur->OSTCBPrio] = OSTimeGet();
    }
}
/// ========================================特殊Completion========================================

/// Task變了 => completed or preempted。 tick = 0時會有arrive time != 0的task completion，不需要輸出
else if ((OSPrioCur != OSPrioHighRdy && OSTimeGet() != 0))
{
    if (OSPrioCur == 0)
    {
        task_start_time[OSPrioHighRdy] = OSTimeGet();
    }
    else
    {
        printf("%2d\t", OSTimeGet());
        fprintf(Output_fp, "%2d\t", OSTimeGet());

        /// ========================================一般Completion========================================
        if ((OSRdyTbl[OSPrioCur >> 3] & OSMapTbl[OSPrioCur & 0x07]) == 0)  /// 表示原本的Task已完成，進入等待狀態
        {
            printf("Completion\t");
            fprintf(Output_fp, "Completion\t");
            /// CurrentTask
            if (OSPrioCur == 63)  /// 其實不會發生這種情況
            {
                printf("Task(%2d)\t", OSPrioCur);
                fprintf(Output_fp, "Task(%2d)\t", OSPrioCur);
            }
            else
            {
                printf("Task(%2d)(%2d)\t", OSTCBPrioTbl[OSPrioCur]->OSTCBId, job_number[OSPrioCur]);
                fprintf(Output_fp, "Task(%2d)(%2d)\t", OSTCBPrioTbl[OSPrioCur]->OSTCBId, job_number[OSPrioCur]);
            }
            /// NextTask
            if (OSPrioHighRdy == 63)
            {
                printf("Task(%2d)\t", OSPrioHighRdy);
                fprintf(Output_fp, "Task(%2d)\t", OSPrioHighRdy);
            }
            else
            {
                printf("Task(%2d)(%2d)\t", OSTCBPrioTbl[OSPrioHighRdy]->OSTCBId, job_number[OSPrioHighRdy]);
                fprintf(Output_fp, "Task(%2d)(%2d)\t", OSTCBPrioTbl[OSPrioHighRdy]->OSTCBId, job_number[OSPrioHighRdy]
            }
            /// ResponseTime
            printf("%5d\t\t", (OSTimeGet() - task_start_time[OSPrioCur]));
            fprintf(Output_fp, "%5d\t\t", (OSTimeGet() - task_start_time[OSPrioCur]));
            /// # of ContextSwitch
            task_preempt_count[OSPrioCur]++;
            printf("%5d\t\t\t", task_preempt_count[OSPrioCur]);
            fprintf(Output_fp, "%5d\t\t\t", task_preempt_count[OSPrioCur]);
            /// PreemptionTime
            printf("%5d\n", task_preempt_time_acc[OSPrioCur]);
            fprintf(Output_fp, "%5d\n", task_preempt_time_acc[OSPrioCur]);

            task_preempt_time[OSPrioCur] = 0;
            task_preempt_count[OSPrioCur] = 0;
            task_preempt_time_acc[OSPrioCur] = 0;

            if (task_preempt_time[OSPrioHighRdy] != 0)  /// 有被preempt的紀錄
            {
                task_preempt_time_acc[OSPrioHighRdy] = task_preempt_time_acc[OSPrioHighRdy] + (OSTimeGet() - task_pree
            }

            task_preempt_count[OSPrioHighRdy]++;  /// 換過去第一次也算，才會符合PA#1範例
```

```c

        job_number[OSPrioCur]++;
    }
    /// ======================================一般Completion======================================


    else  /// 表示原本的Task尚未完成，preempt(必定是低prio換到高prio)
    {
        /// ======================================特殊Preemption======================================
        /// task在OSTimeTick()之後應complete(i.e. remain_exe_time == 1)，但先被preempt掉了，如Task Set 2 tick 12
        if (OSTCBCur->remain_exe_time == 1)
        {
            OSTCBCur->remain_exe_time = 0;

            INT8U ticks = OSTCBCur->next_job_time - OSTimeGet();
            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 = 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();
            }
            OSTCBCur->already_delay = 1;

            printf("Completion\t");
            fprintf(Output_fp, "Completion\t");
            /// CurrentTask
            if (OSPrioCur == 63)  /// 其實不會發生這種情況
            {
                printf("Task(%2d)\t", OSPrioCur);
                fprintf(Output_fp, "Task(%2d)\t", OSPrioCur);
            }
            else
            {
                printf("Task(%2d)(%2d)\t", OSTCBPrioTbl[OSPrioCur]->OSTCBId, job_number[OSPrioCur]);
                fprintf(Output_fp, "Task(%2d)(%2d)\t", OSTCBPrioTbl[OSPrioCur]->OSTCBId, job_number[OSPrioCur]);
            }
            /// NextTask
            if (OSPrioHighRdy == 63)
            {
                printf("Task(%2d)\t", OSPrioHighRdy);
                fprintf(Output_fp, "Task(%2d)\t", OSPrioHighRdy);
            }
            else
            {
                printf("Task(%2d)(%2d)\t", OSTCBPrioTbl[OSPrioHighRdy]->OSTCBId, job_number[OSPrioHighRdy]);
                fprintf(Output_fp, "Task(%2d)(%2d)\t", OSTCBPrioTbl[OSPrioHighRdy]->OSTCBId, job_number[OSPrioHigh
            }
            /// ResponseTime
            printf("%5d\t\t", (OSTimeGet() - task_start_time[OSPrioCur]));
            fprintf(Output_fp, "%5d\t\t", (OSTimeGet() - task_start_time[OSPrioCur]));
            /// # of ContextSwitch
            task_preempt_count[OSPrioCur]++;
            printf("%5d\t\t\t", task_preempt_count[OSPrioCur]);
            fprintf(Output_fp, "%5d\t\t\t", task_preempt_count[OSPrioCur]);
            /// PreemptionTime
            printf("%5d\n", task_preempt_time_acc[OSPrioCur]);
            fprintf(Output_fp, "%5d\n", task_preempt_time_acc[OSPrioCur]);

            task_preempt_time[OSPrioCur] = 0;
            task_preempt_count[OSPrioCur] = 0;
```

```c
                task_preempt_time_acc[OSPrioCur] = 0;

                if (task_preempt_time[OSPrioHighRdy] != 0)  /// 有被preempt的紀錄
                {
                    task_preempt_time_acc[OSPrioHighRdy] = task_preempt_time_acc[OSPrioHighRdy] + (OSTimeGet() - task_
                }

                task_preempt_count[OSPrioHighRdy]++;  /// 換過去第一次也算，才會符合PA#1範例

                job_number[OSPrioCur]++;
            }
            /// ========================================特殊Preemption========================================


            /// ========================================一般Preemption========================================
            else
            {
                printf("Preemption\t");
                fprintf(Output_fp, "Preemption\t");
                /// CurrentTask
                if (OSPrioCur == 63)
                {
                    printf("Task(%2d)\t", OSPrioCur);
                    fprintf(Output_fp, "Task(%2d)\t", OSPrioCur);
                }
                else
                {
                    printf("Task(%2d)(%2d)\t", OSTCBPrioTbl[OSPrioCur]->OSTCBId, job_number[OSPrioCur]);
                    fprintf(Output_fp, "Task(%2d)(%2d)\t", OSTCBPrioTbl[OSPrioCur]->OSTCBId, job_number[OSPrioCur]);
                }
                /// NextTask
                if (OSPrioHighRdy == 63)
                {
                    printf("Task(%2d)\t", OSPrioHighRdy);
                    fprintf(Output_fp, "Task(%2d)\t", OSPrioHighRdy);
                }
                else
                {
                    printf("Task(%2d)(%2d)\n", OSTCBPrioTbl[OSPrioHighRdy]->OSTCBId, job_number[OSPrioHighRdy]);
                    fprintf(Output_fp, "Task(%2d)(%2d)\n", OSTCBPrioTbl[OSPrioHighRdy]->OSTCBId, job_number[OSPrioHigh
                }

                task_preempt_time[OSPrioCur] = OSTimeGet();   /// 先記錄何時被preempt
                task_preempt_count[OSPrioCur]++;

                /// task_preempt_time[OSPrioHighRdy] 不動
                task_preempt_count[OSPrioHighRdy]++;
            }
            /// ========================================一般Preemption========================================
        }
    }
}
fclose(Output_fp);
/// PA#1
```

從 OS_SchedNew()中，觀察 task 之間的切換情形，取得需要的輸出資訊。

資料	判斷、記錄方式
Event	Task a 切換至 Task b 時，若 Task a 在 OSRdyTbl 上仍處於 ready 狀態(==1)，則此情況為 preemtpion；反之則為 completion。
Response Time	「Completion 的時間點」減掉「task 被設為 ready 的時間點(記錄於 task_start_time)」，即為 response time。
# of ContextSwitch	記錄於 task_preempt_count，每當 task 切換則+1，completion 輸出後歸零。
Preemption Time	發生 preemtpion 時，記錄「task 被 preempt 的時間點」，與「後續返回到該 task 的時間點」之差距，即為 preemption time。再 task complete 之前有多次 preemption 則進行累加。