

## 1. 執行步驟

```

27 #define TASK_STK_SIZE 512 /* Size of each task's stacks (# of WORDs) */
28
29 #define TASK_START_ID 0 /* Application tasks IDs */
30 #define TASK_CLK_ID 1
31 #define TASK_1_ID 2
32 #define TASK_2_ID 3
33 #define TASK_3_ID 4
34 #define TASK_4_ID 5
35 #define TASK_5_ID 6
36
37 #define TASK_START_PRIO 10 /* Application tasks priorities */
38 #define TASK_CLK_PRIO 11
39 #define TASK_1_PRIO 12
40 #define TASK_2_PRIO 13
41 #define TASK_3_PRIO 14
42 #define TASK_4_PRIO 15
43 #define TASK_5_PRIO 16
44
45
46 #define PERIODIC_TASK_START_ID 20
47 #define PERIODIC_TASK_START_PRIO 20
48
49
50
51 typedef struct {
52     INT32U RemainTime;
53     INT32U ExecutionTime;
54     INT32U Period;
55     INT32U Deadline;
56     INT8U TaskID;
57 } TASK_EXTRA_DATA;

```

```

4
***** VARIABLES *****
*/
05_STK TaskStartStk[TASK_STK_SIZE]; /* Startup task stack */
05_STK TaskClkStk[TASK_STK_SIZE]; /* Clock task stack */
05_STK Task1Stk[TASK_STK_SIZE]; /* Task #1 task stack */
05_STK Task2Stk[TASK_STK_SIZE]; /* Task #2 task stack */
05_STK Task3Stk[TASK_STK_SIZE]; /* Task #3 task stack */
05_STK Task4Stk[TASK_STK_SIZE]; /* Task #4 task stack */
05_STK Task5Stk[TASK_STK_SIZE]; /* Task #5 task stack */

05_EVENT *AckMbox; /* Message mailboxes for Tasks #4 and #5 */
05_EVENT *IoMbox;
05_EVENT *sem;
05_STK TaskStk[8][TASK_STK_SIZE];
TASK_EXTRA_DATA TaskExtraData[8];
INT8U NumberOfTasks;
INT8U ExecutionTime[8];
INT8U PeriodTime[8];

INT8U *TaskList;
INT8U *TempPeriodTime;
// INT8U *Priority;
INT32U MyStartTime;
INT32U task_display_counter = 0;
*****

```

```

***** FUNCTION PROTOTYPES *****
*/
/*
void TaskStart(void *data); /* Function prototypes of tasks */
static void TaskStartCreateTasks(void);
static void TaskStartDispInit(void);
static void TaskStartDisp(void);
void TaskClk(void *data);
void taskTime();

void PeriodicTask(void *data);
void quickSort (INT8U *PeriodTime , INT8U *TaskList,const INT8U left, const INT8U right);
void swap(INT8U* a,INT8U *b);
void selectionSort(INT8U *PeriodTime , INT8U *TaskList, INT8U size);
//void selectionSort(INT8U *exeTime , INT8U *TaskList, INT8U size,INT8U *PeriodTime);
*/

```

基本上沒改動學長給的 source code

前面先做一些宣告變數，function，struct 等等，以及定義值。特別提一下，sem 是用來當作保護的變數，類似 signal,wait 的概念。

## 讀檔

```
129
130     FILE *InputFile;
131     INT8U i;
132     INT8U j;
133     InputFile = fopen("Input1.txt","r");
134     fscanf(InputFile, "%d", &NumberOfTasks);
135     TaskList = (int*)malloc(sizeof(int)*NumberOfTasks);
136     for(i=0; i< NumberOfTasks; i++)
137     {
138         // read file
139         fscanf(InputFile, "%d %d", &PeriodTime[i], &ExecutionTime[i]);
140         TaskExtraData[i].ExecutionTime = ExecutionTime[i] * OS_TICKS_PER_SEC;
141         TaskExtraData[i].Period = PeriodTime[i] * OS_TICKS_PER_SEC;
142         TaskExtraData[i].Deadline = PeriodTime[i] * OS_TICKS_PER_SEC;
143         TaskExtraData[i].RemainTime = ExecutionTime[i] * OS_TICKS_PER_SEC;
144         TaskExtraData[i].TaskID=i+1;
145         TaskList[i] = i;
146     }
147     fclose(InputFile);
148
149     //sorting file data
150     selectionSort(PeriodTime,TaskList,NumberOfTasks);
151
152
```

## 排序

Selectionsort 與 swap 的排序 function，根據週期的大小對 task 進行排程，小的放前面，大的放後面

透過設定 selectionSort function，用 swap 去對週期大小進行排程，小的往前移，大的往後移

```
148
149     //sorting file data
150     selectionSort(PeriodTime,TaskList,NumberOfTasks);
151
152
```

```
188 void swap(INT8U* a,INT8U *b){
189     INT8U temp = *a;
190     *a = *b;
191     *b = temp;
192 }
193
194
195 // toby: selection sort
196 void selectionSort(INT8U *PeriodTime , INT8U *TaskList, INT8U size){
197     INT8U i, j, min_idx, temp;
198     for (i=0; i<size-1; i++) {
199         min_idx = i;
200         for (j=i+1; j<size; j++) {
201             if (PeriodTime[j] < PeriodTime[min_idx]) {
202                 min_idx = j;
203             }
204         }
205         swap(&PeriodTime[min_idx],&PeriodTime[i]);
206         swap(&TaskList[min_idx],&TaskList[i]);
207         //PC_DispStr( 0, 12, PeriodTime[i], DISP_FGND_BLACK + DISP_BGND_LIGHT_GRAY);
208     }
209 }
210
```

顯示面板

把任務名稱 開始時間 結束時間 下次開始時間 週期 執行時間  
週期 執行時間 總共跑幾次資訊列印出來。

```
260 static void TaskStartDispInit(void)
261 {
262     char s[80];
263     INT8U i;
264     INT8U j;
265
266     PC_DisPStr( 0, 0, "                Final Project on uc/OS-II                ", DISP_FGND_WHITE + DISP_BGND_BLACK);
267     PC_DisPStr( 0, 1, "                ", DISP_FGND_WHITE + DISP_BGND_BLACK);
268     PC_DisPStr( 0, 2, "                ", DISP_FGND_CYAN + DISP_BGND_BLUE);
269     PC_DisPStr( 0, 3, "                B0829811                ", DISP_FGND_LIGHT_GRAY + DISP_BGND_BLUE);
270     PC_DisPStr( 0, 4, "                ", DISP_FGND_BLACK + DISP_BGND_CYAN);
271     PC_DisPStr( 0, 5, "                SJF NoPreempt Scheduling Results                ", DISP_FGND_BLACK + DISP_BGND_CYAN);
272     PC_DisPStr( 0, 6, "                ", DISP_FGND_BLACK + DISP_BGND_LIGHT_GRAY);
273     PC_DisPStr( 0, 7, "                ", DISP_FGND_BLACK + DISP_BGND_LIGHT_GRAY);
274     PC_DisPStr( 0, 8, "Task      Start Time  End Time  Deadline  Period  Excution  Run      ", DISP_FGND_BLACK + DISP_BGND_LIGHT_GRAY);
275     PC_DisPStr( 0, 9, "-----", DISP_FGND_BLACK + DISP_BGND_LIGHT_GRAY);
276     for(i=0; i< NumberOfTasks; i++)
277     {
278         INT8U p_time = TaskExtraData[i].Period/OS_TICKS_PER_SEC;
279         INT8U e_time = TaskExtraData[i].ExecutionTime/OS_TICKS_PER_SEC;
280         sprintf(s,"Task%d()", i, TaskExtraData[i].TaskID,p_time , e_time);
281         PC_DisPStr(0, 10+i, s, DISP_FGND_BLACK + DISP_BGND_LIGHT_GRAY);
282     }
283     for(j=(NumberOfTasks+10); j<22; j++)
284     {
285         PC_DisPStr( 0, j, "                ", DISP_FGND_BLACK + DISP_BGND_LIGHT_GRAY);
286     }
287
288     PC_DisPStr( 0, 22, "#Tasks      :      CPU Usage:  %", DISP_FGND_BLACK + DISP_BGND_LIGHT_GRAY);
289     PC_DisPStr( 0, 23, "#Task switch/sec:", DISP_FGND_BLACK + DISP_BGND_LIGHT_GRAY);
290     PC_DisPStr( 0, 24, "                <-PRESS 'ESC' TO QUIT->", DISP_FGND_BLACK + DISP_BGND_LIGHT_GRAY + DISP_BLINK);
291 }
```

在迴圈當中，可以使用 OSSemPend() function 來進行保護被搶斷，當執行時 sem 會被改成 0，阻止其他程式進行搶斷。

```
438 // 顯示任務ID和context switching 的start time
439
440 sprintf(s, "%d->", MyPtr->TaskID);
441 PC_DisPStr( task_display_counter, 18, s, DISP_FGND_BLACK + DISP_BGND_LIGHT_GRAY);
442 totaltime=OSTimeGet()/OS_TICKS_PER_SEC-1;
443 sprintf(s, "%d->", totaltime);
444 PC_DisPStr( task_display_counter, 20, s, DISP_FGND_BLACK + DISP_BGND_LIGHT_GRAY);
445
446 task_display_counter = task_display_counter+4;
447 x++;
448 sprintf(s, "%4d",x);
449 PC_DisPStr(75, 10 + TaskList[OSPrCur - PERIODIC_TASK_START_PRIO], s, DISP_FGND_BLACK + DISP_BGND_LIGHT_GRAY);
450 sprintf(s, "%10d ",OSTimeGet()/OS_TICKS_PER_SEC-1);
451 PC_DisPStr(15, 10 + TaskList[OSPrCur - PERIODIC_TASK_START_PRIO], s, DISP_FGND_BLACK + DISP_BGND_LIGHT_GRAY);
452 sprintf(s, "%10d ", MyPtr->Deadline/OS_TICKS_PER_SEC-1);
453 PC_DisPStr(39, 10 + TaskList[OSPrCur - PERIODIC_TASK_START_PRIO], s, DISP_FGND_BLACK + DISP_BGND_LIGHT_GRAY);
454
455
```

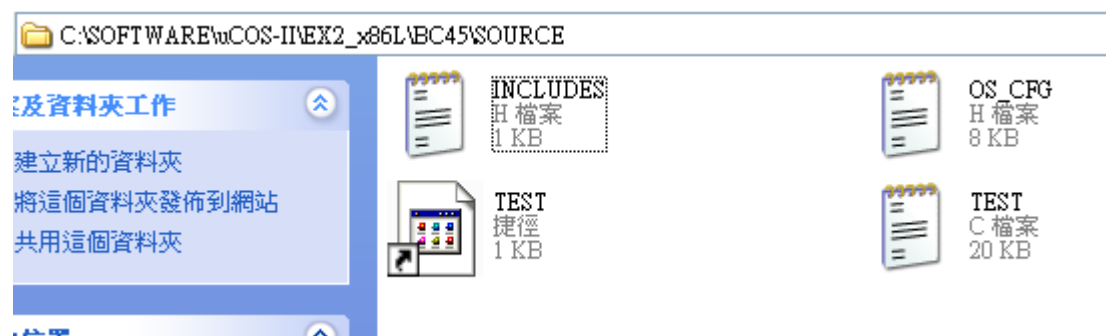
接下來就是把哪件任務在執行印出來，以及跑了多少時間。但是我們要去換算時間，time=200(tick)時，大概等於一秒，所以透過這個 for loop 去抓慢慢嘗試什麼時候等於 200，等等顯示就可以顯示出有幾個一秒鐘。但由於效能不同，所以每台電腦要跑的迴圈數也不同，要多嘗試看看。執行成功後將剛剛的 sem 設為 1，讓其他程式可以執行，最後設定每個工作結束後需要等待的時間，完成一次 task 的執行。

```

455
456 // 執行for loop 去fit 1秒
457 start=OSTimeGet();
458 for(i=0;i<MyPtr->ExecutionTime;i++)
459 {
460     for(j=0;j<3100000;j++)
461     {
462
463     }
464 }
465 end=OSTimeGet();
466 sprintf(s, "%d", (end-start));
467 PC_DisPStr( 0,17, s, DISP_FGND_BLACK + DISP_BGND_LIGHT_GRAY);
468
469 MyPtr->Deadline = MyPtr->Deadline + MyPtr->Period;
470
471 sprintf(s, "%10d ", OSTimeGet()/OS_TICKS_PER_SEC-1);
472 PC_DisPStr(27, 10 + TaskList[OSPrioCur - PERIODIC_TASK_START_PRIO], s, DISP_FGND_BLACK + DISP_BGND_LIGHT_GRAY);
473 OSSemPost(sem);
474 waitTime=MyPtr->Deadline-MyPtr->Period-OSTimeGet(); //想一下每個工作要wait 幾秒 這裡要改哩
475 if(waitTime>0)
476 {
477     OSTimeDly(waitTime);
478 }
479
480 }
481 }
482

```

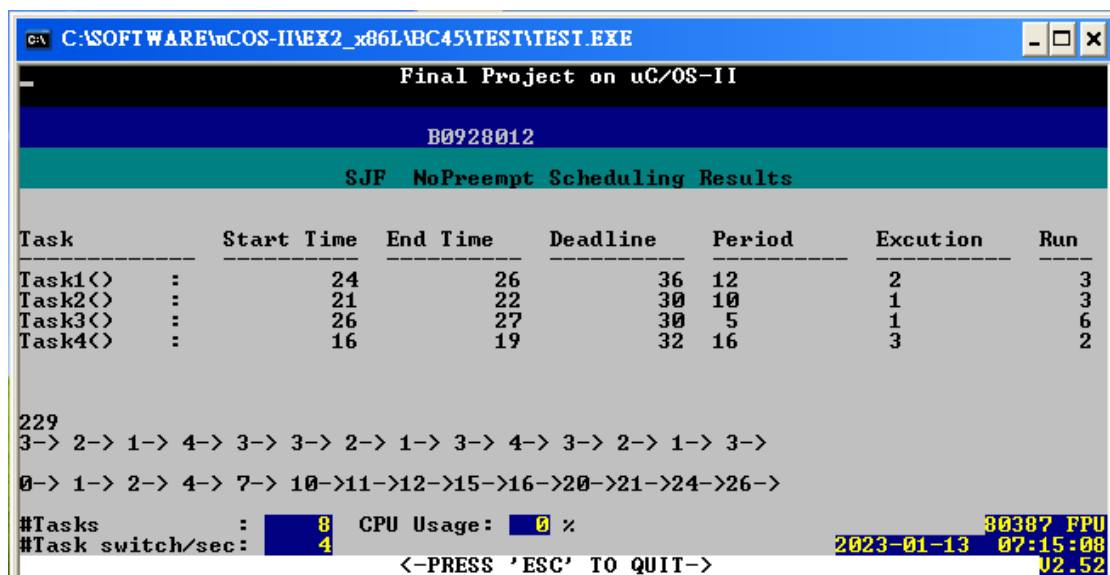
TEST.c 放在這個位置



Input1 輸入老師給的四個 task 任務內容，並放在這個位置。



最後跑出來的樣子



```
C:\SOFTWARE\uCOS-II\EX2_x86\ABC45\TEST\TEST.EXE
Final Project on uC/OS-II
B0928012
SJF NoPreempt Scheduling Results

Task      Start Time  End Time  Deadline  Period  Excution  Run
-----
Task1(<)  :      24      26      36     12        2        3
Task2(<)  :      21      22      30     10        1        3
Task3(<)  :      26      27      30      5        1        6
Task4(<)  :      16      19      32     16        3        2

229
3-> 2-> 1-> 4-> 3-> 3-> 2-> 1-> 3-> 4-> 3-> 2-> 1-> 3->
0-> 1-> 2-> 4-> 7-> 10->11->12->15->16->20->21->24->26->

#Tasks      :      8  CPU Usage:      0 %
#Task switch/sec:      4

                                30387 FPU
                                2023-01-13 07:15:08
                                U2.52

<-PRESS 'ESC' TO QUIT->
```

## 2.遇到的問題：

我去設定一秒時，顯示出來都會離 200 有一大段距離，一開始會在 100 出頭，跑一下後，又發現變成 500 多，根本不可能找到 200。我換了一台電腦做，也還是一樣。後來想到，說不定把核心數調高，可能穩定性會比較好，就不會亂跳，調到四核心後，就可以很接近兩百，且執行時間的內容，也沒有顯示錯誤。

## 3.參考內容：

$\mu$ C/OS-II 範例程式實作

[http://pub.tust.edu.tw/mechanic/microlab/public\\_html/ARM7/item/uCOS-II.htm](http://pub.tust.edu.tw/mechanic/microlab/public_html/ARM7/item/uCOS-II.htm)

$\mu$ C/OS-II 介紹

<https://www.jendow.com.tw/wiki/ucos+ii>