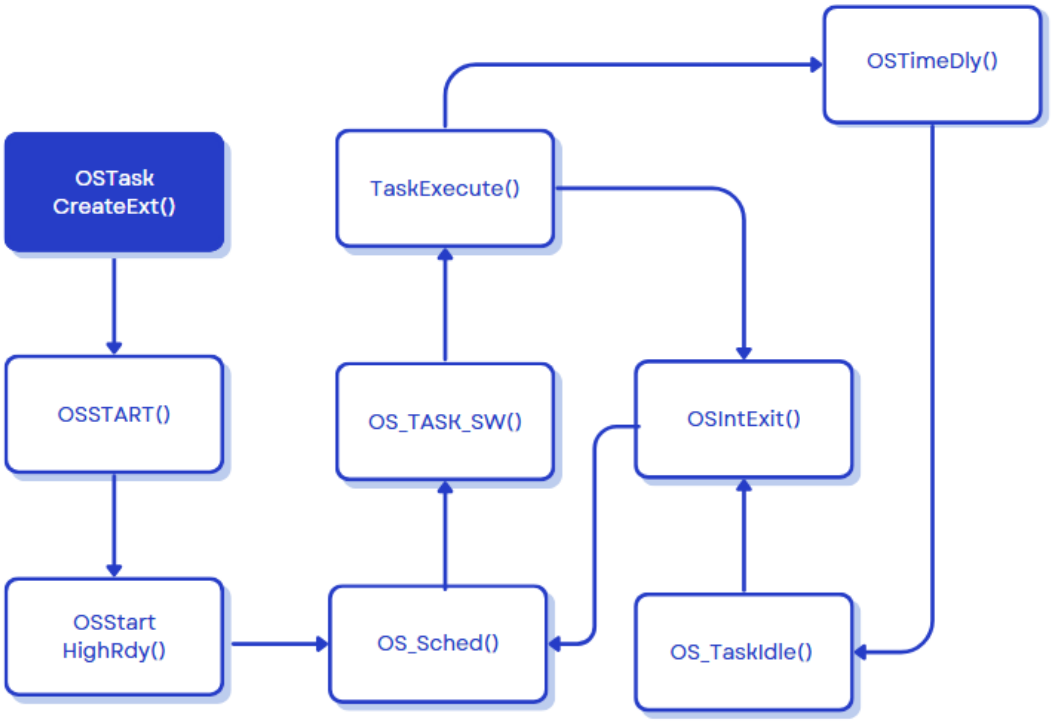
流程圖:



執行程序:

在 main function 中，會分別對所有我們想創建的 task 執行 OSTaskCreatExt()來設定 priority 等參數，接著進入 OSSTART()開始 RTOS 的工作流程，首先會執行 OSStartHighRdy()將最高 Priority 的 task 進行 OS_TASK_SW()以排入 cpu 執行工作 TaskExecute()，結束後會用 OSTimeDly()來等待下一個週期的 JOB 進行，這時會啟動 OS_TaskIdle()來在這段閒置時間 Run Idle task，如果有其他 task 預備好可以工作而產生 interrupt，則觸發 OSInitExit()切換並回到 OS_Sched()的步驟，在 TaskExecution()期間，如果有更高 priority()的 task 準備好，也可能觸發 interrupt 而被迫終止。

執行結果:

Tick	CurrentTask ID	NeztTask ID	Number of ctx switch
0	**********	task(1)(0)	0
0	task(1)(0)	task(2)(0)	0
0	task(2)(0)	task(63)	1
3	task(63)	task(1)(1)	2
3	task(1)(1)	task(63)	3
4	task(63)	task(2)(1)	4
4	task(2)(1)	task(63)	5
6	task(63)	task(1)(2)	6
6	task(1)(2)	task(63)	7
8	task(63)	task(2)(2)	8
8	task(2)(2)	task(63)	9
9	task(63)	task(1)(3)	10
9	task(1)(3)	task(63)	11
12	task(63)	task(1)(4)	12
12	task(1)(4)	task(2)(3)	13
12	task(2)(3)	task(63)	14
15	task(63)	task(1)(5)	15
15	task(1)(5)	task(63)	16
16	task(63)	task(2)(4)	17
16	task(2)(4)	task(63)	18
18	task(63)	task(1)(6)	19
18	task(1)(6)	task(63)	20
20	task(63)	task(2)(5)	21
20	task(2)(5)	task(63)	22
21	task(63)	task(1)(7)	23
21	task(1)(7)	task(63)	24
24	task(63)	task(1)(8)	25
24	task(1)(8)	task(2)(6)	26
24	task(2)(6)	task(63)	27
27	task(63)	task(1)(9)	28
27	task(1)(9)	task(63)	29
28	task(63)	task(2)(7)	30
28	task(2)(7)	task(63)	31
30	task(63)	task(1)(10)	32
30	task(1)(10)	task(63)	33

程式修改:

app_hooks.c 中的 InputFile()函數

```
147         /* Initial Priority */
148         TaskParameter[j].TaskPriority = TaskParameter[j].TaskID; // just an example
```

我將初始 priority 改為 TaskID 以方便後續輸出

main.c

```
111     //OSTaskCreateExt( StartupTask,                                /* Create the startup task
112     //                 0,
113     //                &StartupTaskStk[APP_CFG_STARTUP_TASK_STK_SIZE - 1u],
114     //                 APP_CFG_STARTUP_TASK_PRIO,
115     //                 APP_CFG_STARTUP_TASK_PRIO,
116     //                &StartupTaskStk[0u],
117     //                 APP_CFG_STARTUP_TASK_STK_SIZE,
118     //                 0u,
119     //                (OS_TASK_OPT_STK_CHK | OS_TASK_OPT_STK_CLR));
120
121     for (int i = 0; i < TASK_NUMBER; i++) {
122         OSTaskCreateExt(task,
123         &TaskParameter[i],
124         &Task_STK[i][TASK_STACKSIZE - 1],
125         TaskParameter[i].TaskPriority,
126         TaskParameter[i].TaskID,
127         &Task_STK[i][0],
128         TASK_STACKSIZE,
129         &TaskParameter[i],
130         (OS_TASK_OPT_STK_CHK | OS_TASK_OPT_STK_CLR));
131     }
132     printf("Tick\tCurrentTask ID\tNeztTask ID\tNumber of ctx switch\n");
133     fprintf(Output_fp, "Tick\tCurrentTask ID\tNeztTask ID\tNumber of ctx switch\n");
```

```
void task(void* p_arg) {
    task_para_set* task_data;
    task_data = p_arg;
    while (1) {
        //printf("Tick: %d, Hello from task%d\n", OSTime, task_data->TaskID);
        //if ((Output_err = fopen_s(&Output_fp, "./Output.txt", "a")) == 0) {
        //    fprintf(Output_fp, "Tick: %d, Hello from task%d\n", OSTime, task_data->TaskID);
        //    fclose(Output_fp);
        //}
        OSTimeDly(task_data->TaskPeriodic);
    }
}
```

將 task 創建的程式設為更彈性的寫法來適應各種輸入測試檔案的長度，在輸出文件第一航寫上格式名稱

os_cpu_c.c

```
554 void  OSTaskSwHook(void)
555 {
556     static INT32U CtxSwCount = 0;
557     static BOOLEAN firstenter = TRUE;
558     INT8U oldtask, newtask;
559     INT32U oldjobcount, newjobcount;
560 
561     oldtask = OSTCBCur->OSTCBPrio;
562     oldjobcount = OSTCBCur->OSTCBCtxSwCtr - 1;
563     newtask = OSTCBHighRdy->OSTCBPrio;
564     newjobcount = OSTCBHighRdy->OSTCBCtxSwCtr - 1;
565     if (firstenter) {
566     printf("%2d\t**********\ttask(%2d)(%2d)\t%d\n", OSTime, newtask, newjobcount, CtxSwCount);
567     fprintf(Output_fp, "%2d\t**********\ttask(%2d)(%2d)\t%d\n", OSTime, newtask, newjobcount, CtxSwCount);
568     CtxSwCount--;
569     firstenter = FALSE;
570     }
571     else {
572         if (oldtask == 63) {
573             printf("%2d\ttask(%2d)\ttask(%2d)(%2d)\t%d\n", OSTime, oldtask, newtask, newjobcount, CtxSwCount);
574             fprintf(Output_fp, "%2d\ttask(%2d)\ttask(%2d)(%2d)\t%d\n", OSTime, oldtask, newtask, newjobcount, CtxSwCount);
575         }
576         else if (newtask == 63) {
577             printf("%2d\ttask(%2d)(%2d)\ttask(%2d)\t%d\n", OSTime, oldtask, oldjobcount, newtask, CtxSwCount);
578             fprintf(Output_fp, "%2d\ttask(%2d)(%2d)\ttask(%2d)\t%d\n", OSTime, oldtask, oldjobcount, newtask, CtxSwCount);
579         }
580         else {
581             printf("%2d\ttask(%2d)(%2d)\ttask(%2d)(%2d)\t%d\n", OSTime, oldtask, oldjobcount, newtask, newjobcount, CtxSwCount);
582             fprintf(Output_fp, "%2d\ttask(%2d)(%2d)\ttask(%2d)(%2d)\t%d\n", OSTime, oldtask, oldjobcount, newtask, newjobcount, CtxSwCount);
583         }
584 
585     }
586     CtxSwCount++;
587 #if (OS_APP_HOOKS_EN > 0u)
588     App_TaskSwHook();
```

本次作業需將所有 ctxsw 出現的地方列印出現，我觀察到每次執行此動作時都會 call OSTaskSwHook()這個 function，因此選擇在這裡實作，首先創建 CtxSwCount 來記錄 ctxsw 次數，接著使用了 oldtask 跟 newtask 兩類的變數來記錄切換前後的 task 資訊，包含各自的 Priority 跟 ctxsw 的次數，再分別輸出這兩者的資訊，為了符合作業範例的需求，我觀察到只有第一個 task 執行時 ctxsw 次數是沒有加 1 過的，因此會先判斷這函式是不是第一次執行來決定要變化他和舊 task 的輸出，此外也判斷輸出的是不是 Idle_task 資訊來不顯示 job 執行次數(雖然它本身的值是取的到的)，這份作業原先我是嘗試在 os_core.c 中透過 OSCtxSwCtr 變數前後來做操作的，但發先某些程序的執行順序不太合才改為此方法，另外我有將 SYSTEM_END_TIME 改為 60 來方便觀察前面的輸出因為執行結束後會跳掉畫面。