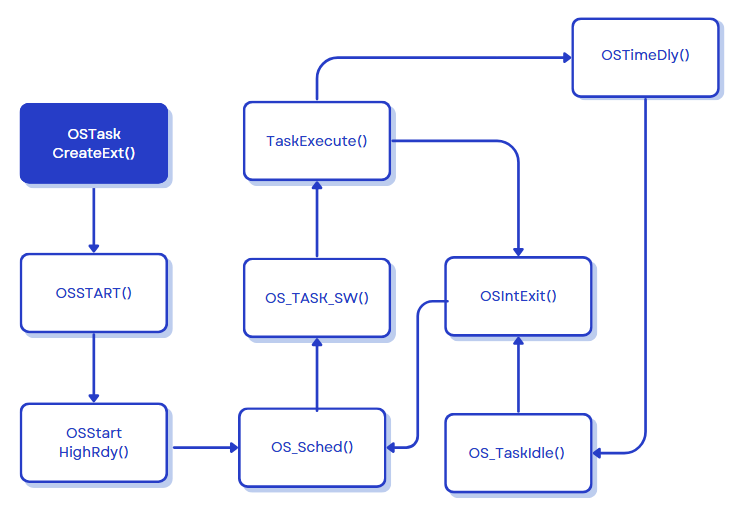
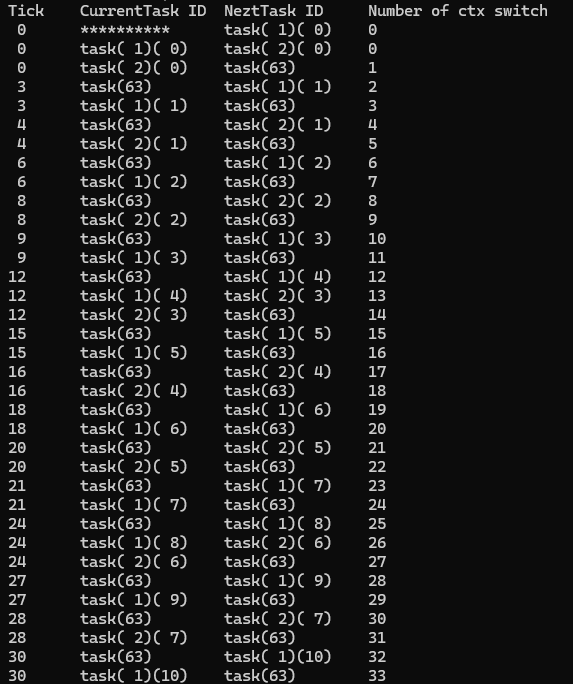
流程圖:



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:



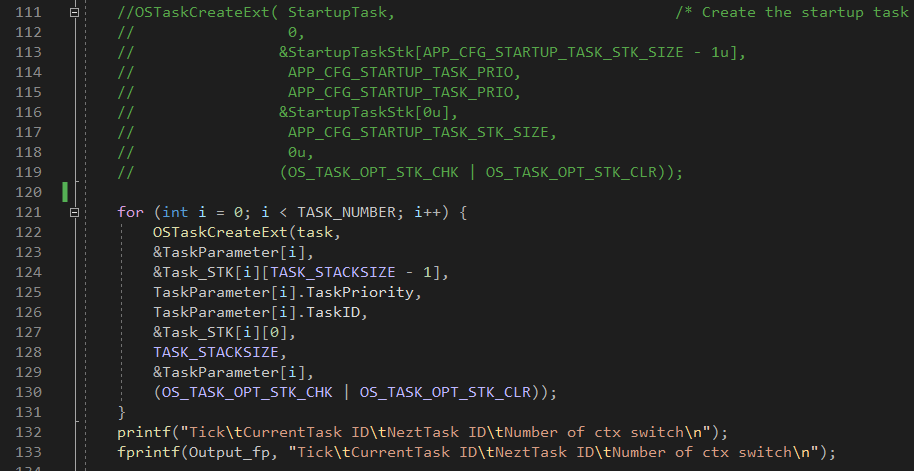
程式修改:

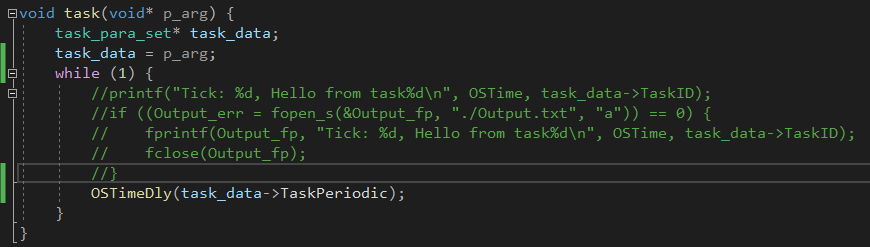
app\_hooks.c中的InputFile()函數



我將初始priority改為TaskID以方便後續輸出

main.c





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



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。