* OSInit()：初始化系統，負責設定核心資源、數據結構，如ReadyList、TCB、FLAG、MEM等。
* OSTaskCreateExt()：建立任務，為每個任務分配必要的資源，並將任務加入ReadyList。
* OSStart()：啟動排程器，在系統完成初始化後，讓OS接管CPU控制權，並開始執行最優先任務中。
* Task()：開始執行任務，會持續執行直到它OSTimeDly、中斷、被更高優先權的任務搶佔，或執行完畢。
* OSTimeDly()：任務主動呼叫延遲休眠。
* OS\_Sched()：啟動排程器。由於當前任務進入阻塞狀態，它不再需要 CPU。OSTimeDly()會間接呼叫排程器，排程器會找出目前所有就緒中優先權最高的那個。
* OSIntEnter()：中斷發生時呼叫，進入中斷模式以避免系統誤判中斷層級。
* OSTimeTick()：執行系統時間更新，並檢查所有被延遲的任務是否到期。
* OSIntExit()：中斷後檢查，系統檢查是否有任何任務變ready，並且優先權高於目前正在被中斷的任務。
* OS\_SchedNew()：比較所有ready任務的優先權，比較決定下一個要執行的任務。

一張含有 文字, 螢幕擷取畫面, 字型, 數字 的圖片

AI 產生的內容可能不正確。

* 在"Micrium/Software/uCOS-II/Source/ucos\_ii.h"的task\_para\_set中新增TaskCount用於計算Job number。

一張含有 文字, 螢幕擷取畫面, 字型, 數字 的圖片

AI 產生的內容可能不正確。

* 在"Microsoft/Windows/Kernel/OS2/app\_hooks.c"的App\_TaskSwHook中新增if用於偵測當前任務是否為idle task，是就輸出idle 跟下一個任務的ID與Job number。

一張含有 文字, 螢幕擷取畫面, 字型, 行 的圖片

AI 產生的內容可能不正確。

* 在"Microsoft/Windows/Kernel/OS2/main.c"
  + main() :
    - 將建任務prio從TaskPeriodic換成TaskID
    - 在初始時印出時間、第一個任務ID、執行次數跟context switch次數並同時將結果寫入檔案。

一張含有 文字, 軟體, 網頁, 網站 的圖片

AI 產生的內容可能不正確。

* + task():任務一開始會接收結構指標 task\_para\_set\* task\_data，其中記錄任務的ID、週期與執行次數。無限迴圈每次會輸出目前系統時間、任務ID is running，同時將結果寫入檔案，完成後Job number遞增，之後依照period進行delay。

一張含有 文字, 螢幕擷取畫面, 字型 的圖片

AI 產生的內容可能不正確。

* 在"Micrium/Software/uCOS-II/Source/os\_core.c"的OS\_Sched
  + 在排程前將系統時間跟原本的任務資訊印出。

一張含有 文字, 螢幕擷取畫面, 字型, 行 的圖片

AI 產生的內容可能不正確。

* 排程完後，switch之前，當OSTCBHighRdy有指向有效任務，將要進行的任務資訊印出，反之則輸出idle task。

