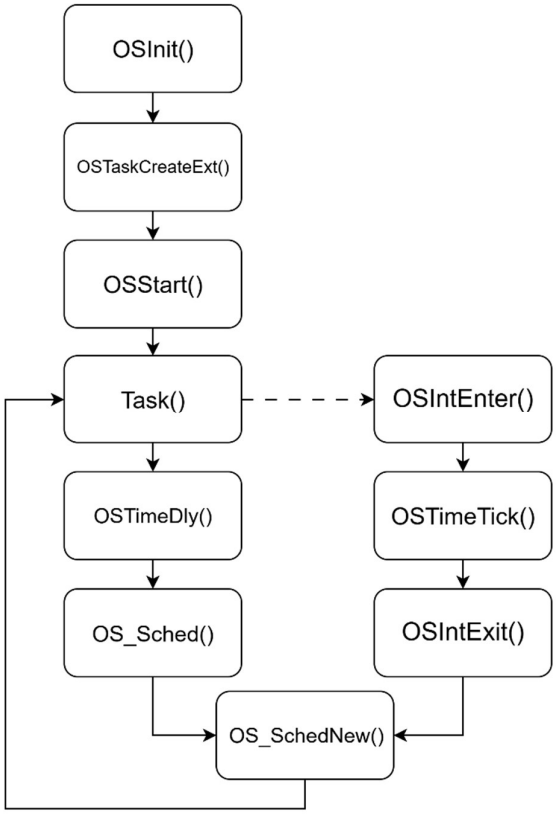
- OSInit()：初始化系統，負責設定核心資源、數據結構，如 ReadyList、TCB、FLAG、MEM 等。
- OSTaskCreateExt()：建立任務，為每個任務分配必要的資源，並將任務加入 ReadyList。
- OSStart()：啟動排程器，在系統完成初始化後，讓 OS 接管 CPU 控制權，並開始執行最優先任務中。
- Task()：開始執行任務，會持續執行直到它 OSTimeDly、中斷、被更高優先權的任務搶佔，或執行完畢。
- OSTimeDly()：任務主動呼叫延遲休眠。
- OS_Sched()：啟動排程器。由於當前任務進入阻塞狀態，它不再需要 CPU。OSTimeDly()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。



- 在 "Micrium/Software/uCOS-II/Source/ucos_ii.h" 的 task_para_set 中新增 TaskCount 用於計算 Job number。

```
typedef struct task_para_set {
    INT16U TaskID;
    INT16U TaskArriveTime;
    INT16U TaskExecutionTIme;
    INT16U TaskPeriodic;
    INT16U TaskNumber;
    INT16U TaskPriority;
    INT16U TaskCount;
} task_para_set;
```

- 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。

```
void  App_TaskSwHook (void)
{
#if (APP_CFG_PROBE_OS_PLUGIN_EN > 0) && (OS_PROBE_HOOKS_EN > 0)
    OSProbe_TaskSwHook();
#endif
    if (OSTCBCur != NULL && OSTCBCur->OSTCBPrio == OS_TASK_IDLE_PRIO) {
        printf("%2d  task(%2d)     \ttask(%2d)(%2d)\t%2d\n", OSTimeGet(), 63, OSPrioHighRdy, TaskParameter[OSPrioHighRdy - 1].TaskCount, OSCtxSwCtr - 1);
        if ((Output_err = fopen_s(&Output_fp, "./Output.txt", "a")) == 0)
        {
            fprintf(Output_fp, "%2d  task(%2d)     \ttask(%2d)(%2d)\t%2d\n", OSTimeGet(), 63, OSPrioHighRdy, TaskParameter[OSPrioHighRdy - 1].TaskCount, OSCtxSwCtr - 1);
            fclose(Output_fp);
        }
    }
}
```

- 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。

```
    for (int n = 0; n < TASK_NUMBER; n++) {
        Task_STK[n] = malloc(TASK_STACKSIZE * sizeof(int));
        OSTaskCreateExt(task,
            &TaskParameter[n],
            &Task_STK[n][TASK_STACKSIZE - 1],
            TaskParameter[n].TaskID, //modefied
            TaskParameter[n].TaskID,
            &Task_STK[n][0],
            TASK_STACKSIZE,
            &TaskParameter[n],
            (OS_TASK_OPT_STK_CHK | OS_TASK_OPT_STK_CLR));
    }

    printf("%2d\t**********\ttask(%2d)(%2d)\t%2d\n", OSTimeGet(), TaskParameter[0].TaskID, TaskParameter[0].TaskCount, OSCtxSwCtr);
    if ((Output_err = fopen_s(&Output_fp, "./Output.txt", "a")) == 0)
    {
        fprintf(Output_fp, "%2d\t**********\ttask(%2d)(%2d)\t%2d\n", OSTimeGet(), TaskParameter[0].TaskID, TaskParameter[0].TaskCount, OSCtxSwCtr);
        fclose(Output_fp);
    }
```

- ✧ 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。

```
87      void task(void* p_arg) {
88          task_para_set* task_data;
89          task_data = p_arg;
90
91          while (1) {
92              printf("%2d  task(%2d) is running\n", OSTimeGet(), task_data->TaskID);
93              if ((Output_err = fopen_s(&Output_fp, "./Output.txt", "a")) == 0)
94              {
95                  fprintf(Output_fp, "%2d  task(%2d) is running\n", OSTimeGet(), task_data->TaskID);
96                  fclose(Output_fp);
97              }
98              task_data->TaskCount += 1;
99              OSTimeDly(task_data->TaskPeriodic);
100         }
101     }
```

- ➢ 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。

```
1707    void  OS_Sched (void)
1708    {
1709    #if OS_CRITICAL_METHOD == 3u                       /* Allocate storage for CPU status register     */
1710        OS_CPU_SR  cpu_sr = 0u;
1711    #endif
1712        printf("%2d  task(%2d)(%2d)\t", OSTimeGet(), ((task_para_set*)(OSTCBHighRdy->OSTCBExtPtr))->TaskID, ((task_para_set*)(OSTCBHighRdy->OSTCBExtPtr))->TaskCount-1);
1713        if ((Output_err = fopen_s(&Output_fp, "./Output.txt", "a")) == 0)
1714        {
1715            fprintf(Output_fp, "%2d  task(%2d)(%2d)\t", OSTimeGet(), ((task_para_set*)(OSTCBHighRdy->OSTCBExtPtr))->TaskID, ((task_para_set*)(OSTCBHighRdy->OSTCBExtPtr))->TaskCount-1);
1716            fclose(Output_fp);
1717        }
```

  - ✧ 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。

```
if (OSTCBHighRdy != NULL && OSTCBHighRdy->OSTCBExtPtr != NULL)  printf("task(%2d)(%2d)\t%2d\n", ((task_para_set*)(OSTCBHighRdy->OSTCBExtPtr))->TaskID, ((task_para_set*)(OSTCBHighRdy->OSTCBExtPtr))->TaskCount, OSCtxSwCtr-1);
else printf("task(%2d)\t%2d\n",63, OSCtxSwCtr-1);
if ((Output_err = fopen_s(&Output_fp, "./Output.txt", "a")) == 0)
{
    if (OSTCBHighRdy != NULL && OSTCBHighRdy->OSTCBExtPtr != NULL)  fprintf(Output_fp, "task(%2d)(%2d)\t%2d\n", ((task_para_set*)(OSTCBHighRdy->OSTCBExtPtr))->TaskID, ((task_para_set*)(OSTCBHighRdy->OSTCBExtPtr))->TaskCount, OSCtxSwCtr - 1);
    else fprintf(Output_fp, "task(%2d)\t%2d\n", 63, OSCtxSwCtr - 1);
    fclose(Output_fp);
}
OS_TASK_SW();                      /* Perform a context switch                 */
```