Embedded OS Implementation, Spring 2025

PA2

學號:M11407517

姓名:黃志傑

[PART I] EDF Scheduler Implementation

Objective:

To implement the Earliest-Deadline-First (EDF) scheduler for periodic tasks and to observe the scheduling behaviors.

Problem Definition:

uC/OS-II supports priority-driven scheduling. However, it lacks deadline-driven scheduling. In this assignment, you are going to implement the EDF scheduler in uC/OS-II. To accomplish this assignment, you must know about the scheduler of uC/OS-II. It can be implemented based on the existing data structures of uC/OS-II. The objectives of this assignment are the following:

(1) To add some functional data structures for your EDF scheduler.

(2) To cooperate with existing data structures/mechanisms in uC/OS-II.

Implement the following examples. Add necessary code to the µC/OS-II scheduler **in the kernel level** to observe how the task incurs the schedule delay.

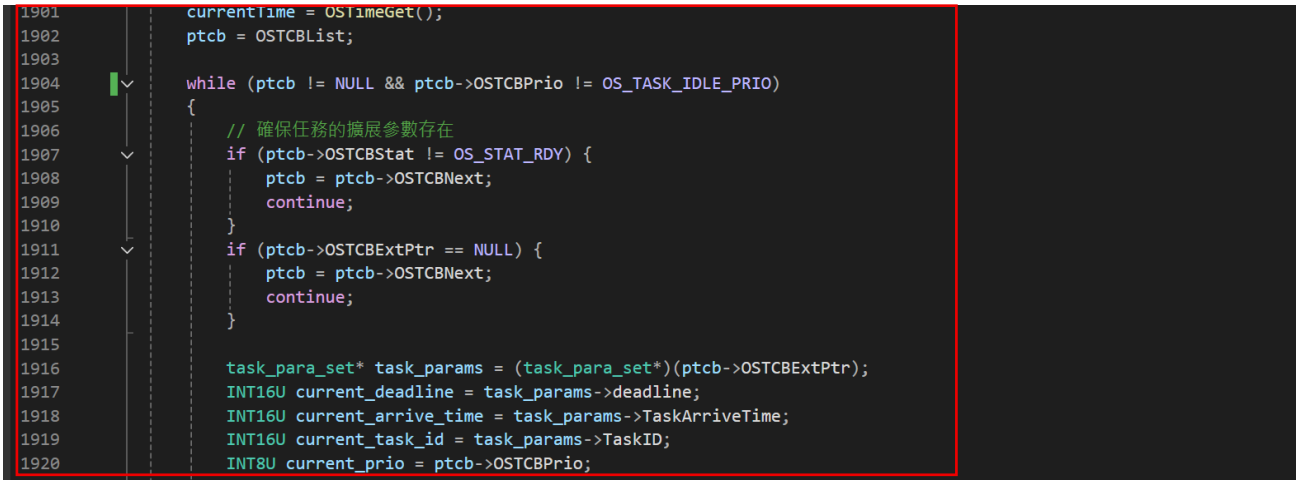
Code :

1. os_core.c :

- **OS_SchedNew()**

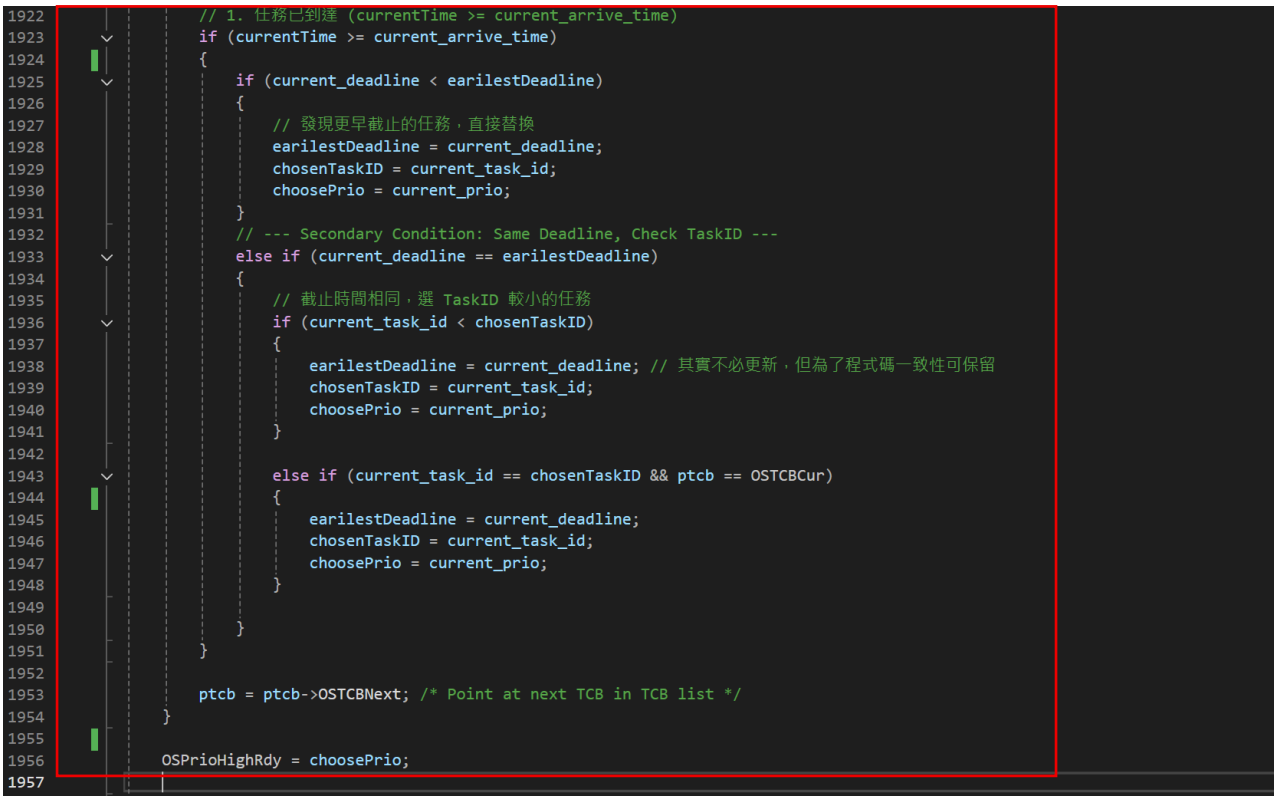
```
1860    static  void  OS_SchedNew(void)
1861    {
1862    #if ALGORITHM == RM
1863    #if OS_LOWEST_PRIO <= 63u                    /* See if we support up to 64 tasks                 */
1864        INT8U   y;
1865
1866
1867        y = OSUnMapTbl[OSRdyGrp];
1868        OSPrioHighRdy = (INT8U)((y << 3u) + OSUnMapTbl[OSRdyTbl[y]]);
1869    #else                                        /* We support up to 256 tasks                       */
1870        INT8U     y;
1871        OS_PRIO* ptbl;
1872
1873
1874        if ((OSRdyGrp & 0xFFu) != 0u) {
1875            y = OSUnMapTbl[OSRdyGrp & 0xFFu];
1876        }
1877        else {
1878            y = OSUnMapTbl[(OS_PRIO)(OSRdyGrp >> 8u) & 0xFFu] + 8u;
1879        }
1880        ptbl = &OSRdyTbl[y];
1881        if ((*ptbl & 0xFFu) != 0u) {
1882            OSPrioHighRdy = (INT8U)((y << 4u) + OSUnMapTbl[(*ptbl & 0xFFu)]);
1883        }
1884        else {
1885            OSPrioHighRdy = (INT8U)((y << 4u) + OSUnMapTbl[(OS_PRIO)(*ptbl >> 8u) & 0xFFu] + 8u);
1886        }
1887    #endif
1888    #endif
1889
1890    #if ALGORITHM == EDF
1891        OS_TCB* ptcb;
1892        INT16U earilestDeadline;
1893        INT16U currentTime;
1894        INT16U choosePrio;
1895        INT16U chosenTaskID;
1896
1897        choosePrio = OS_TASK_IDLE_PRIO;
1898        earilestDeadline = 65535; // 最大值
1899        chosenTaskID = 63;
```

- 當 ALGORITHM ==EDF 時，初始化 earilestDeadline = 65535(最大值)，並先選擇 IDLE_TASK 為 choosePrio



```
1901        currentTime = OSTimeGet();
1902        ptcb = OSTCBList;
1903
1904        while (ptcb != NULL && ptcb->OSTCBPrio != OS_TASK_IDLE_PRIO)
1905        {
1906            // 確保任務的擴展參數存在
1907            if (ptcb->OSTCBStat != OS_STAT_RDY) {
1908                ptcb = ptcb->OSTCBNext;
1909                continue;
1910            }
1911            if (ptcb->OSTCBExtPtr == NULL) {
1912                ptcb = ptcb->OSTCBNext;
1913                continue;
1914            }
1915
1916            task_para_set* task_params = (task_para_set*)(ptcb->OSTCBExtPtr);
1917            INT16U current_deadline = task_params->deadline;
1918            INT16U current_arrive_time = task_params->TaskArriveTime;
1919            INT16U current_task_id = task_params->TaskID;
1920            INT8U current_prio = ptcb->OSTCBPrio;
```

- 檢查 ptcb->OSTCBStat 是否為 OS_STAT_RDY，排除被掛起 (Suspended) 或延遲中的任務，確保僅針對「就緒狀態」的任務進行排程判斷
- 驗證 OSTCBExtPtr 是否為空，防止存取無效記憶體
- 從 TCB 的 OSTCBExtPtr 中讀取 EDF 所需的 deadline、TaskArriveTime 與 TaskID，以供後續演算法進行比較



```
1922            // 1. 任務已到達 (currentTime >= current_arrive_time)
1923            if (currentTime >= current_arrive_time)
1924            {
1925                if (current_deadline < earilestDeadline)
1926                {
1927                    // 發現更早截止的任務，直接替換
1928                    earilestDeadline = current_deadline;
1929                    chosenTaskID = current_task_id;
1930                    choosePrio = current_prio;
1931                }
1932                // --- Secondary Condition: Same Deadline, Check TaskID ---
1933                else if (current_deadline == earilestDeadline)
1934                {
1935                    // 截止時間相同，選 TaskID 較小的任務
1936                    if (current_task_id < chosenTaskID)
1937                    {
1938                        earilestDeadline = current_deadline; // 其實不必更新，但為了程式碼一致性可保留
1939                        chosenTaskID = current_task_id;
1940                        choosePrio = current_prio;
1941                    }
1942
1943                    else if (current_task_id == chosenTaskID && ptcb == OSTCBCur)
1944                    {
1945                        earilestDeadline = current_deadline;
1946                        chosenTaskID = current_task_id;
1947                        choosePrio = current_prio;
1948                    }
1949
1950                }
1951            }
1952
1953            ptcb = ptcb->OSTCBNext; /* Point at next TCB in TCB list */
1954        }
1955
1956        OSPrioHighRdy = choosePrio;
1957
```

- 判斷當前時間已到達 (currentTime >= TaskArriveTime) 的任務
- EDF 比較 (Deadline Comparis**on)：** 尋找並記錄目前為止 Deadline 最早的任務
- 當多個任務 Deadline 相同時，依據作業規範 ，優先選擇 Task ID 較小者
- 利用 ptcb 指標搜尋整個 TCBList

[Part II] CUS Scheduler Implementation

Objective:

To implement Constant Utilization Servers (CUS) for serving aperiodic tasks and to observe the scheduling behaviors.

Problem Definition:

As you did in Part I, uC/OS-II supports the EDF scheduling algorithm. Based on your EDF scheduler, you are going to implement Constant Utilization Servers (CUS) to serve aperiodic tasks.

Implement the following two task sets. Add necessary code to the μC/OS-II scheduler **in the kernel level** to observe how the task suffers the schedule delay.

1. ucos_ii.h :

```
#define FIFO 1     //沒成功完成
#define EDF 2
#define ALGORITHM EDF

#define MAX_PRINTF_BUFFER 16

    typedef struct task_para_set {
        INT16U TaskID;
        INT16U TaskArriveTime;
        INT16U TaskExecutionTime;
        INT16U TaskPeriodic;
        INT16U TaskNumber;
        INT16U TaskPriority;
        INT16U job_no;                  //Job number
        INT16U RemainTime;              //任務剩餘時間
        INT16U deadline;                //任務deadline
        INT16U ResponseTime;            //紀錄response time
        INT16U Delay;                   //任務需要被OSTimeDly多久
    } task_para_set;

    typedef struct {
        INT16U time[MAX_PRINTF_BUFFER];      //紀錄時間
        INT8U  prev[MAX_PRINTF_BUFFER];      //context switch時紀錄前任務ID
        INT8U  cur[MAX_PRINTF_BUFFER];       //context switch時記錄目前任務ID
        INT16U rear;
        INT16U front;
    }BUFFER;

    typedef struct {
        INT32U JobId;
        INT32U ArrivalTime;
        INT32U ExecutionTime;
        INT32U AbsoluteDeadline; // Job deadline
        INT32U FinishedTime;
        INT16U RemainTime;
        INT8U  IsArrived;       // 1: 時間到了，已抵達
        INT8U  IsFinished;      // 1: 做完了 (或被 Reject 了)
    } AperiodicJob;

    int TASK_NUMBER;
    int AperTaskNumber;
    OS_STK** Task_STK;
    task_para_set TaskParameter[OS_MAX_TASKS];
    AperiodicJob AperTaskParameter[OS_MAX_TASKS];
```

- 建立一個新的 structure 存取 aperiodic job 的資料
- 設定全域變數 AperTaskNumber 紀錄 aperiodic job 的數量

2. app_hooks.c :

- InputFile():

```
93  void InputFile() {
94      errno_t err;
95
96      if ((err = fopen_s(&fp, INPUT_FILE_NAME, "r")) == 0)
97          printf("The file %s was opened\n", INPUT_FILE_NAME);
98      else
99          printf("The file %s was not opened\n", INPUT_FILE_NAME);
100
101     char str[MAX];
102     char* ptr;
103     char* pTmp = NULL;
104     int TaskInfo[INFO], i, j = 0;
105     TASK_NUMBER = 0;
106
107     while (!feof(fp)) {
108         i = 0;
109         memset(str, 0, sizeof(str));
110         fgets(str, sizeof(str) - 1, fp);
111         ptr = strtok_s(str, " ", &pTmp);
112
113         while (ptr != NULL) {
114             TaskInfo[i] = atoi(ptr);
115             ptr = strtok_s(NULL, " ", &pTmp);
116             if (i == 0) {
117                 TASK_NUMBER++;
118                 TaskParameter[j].TaskID = TASK_NUMBER;
119             }
120             else if (i == 1) {
121                 TaskParameter[j].TaskArriveTime = TaskInfo[i];
122
123                 if (ptr == NULL) {
124                     CUS_SERVER_SIZE = (float)TaskInfo[i] / 100.0;
125                     TaskParameter[j].TaskArriveTime = 0;
126                     TaskParameter[j].TaskPriority = 0;
127                     TaskParameter[j].deadline = 65535;
128
129                 }
130             }
131             else if (i == 2) {
132                 TaskParameter[j].TaskExecutionTime = TaskInfo[i];
133                 TaskParameter[j].RemainTime = TaskInfo[i];
134             }
135             else if (i == 3) {
136                 TaskParameter[j].TaskPeriodic = TaskInfo[i];
137             }
138             TaskParameter[j].deadline = TaskParameter[j].TaskArriveTime + TaskParameter[j].TaskPeriodic;
139             i++;
```

  - 開啟檔案時檢查 taskset.txt 是否有只有 2 項的內有，如果有就代表有 CUS server 的設定
  - 讀取 CUS_SEVER_SIZE 例如:4 25 ->Server size = 0.25
  - 初始化 server_task 的 arriveTime、deadline 等

- InputAperiodicjobsFile()

```c
void InputAperiodicjobsFile() {
    errno_t err;

    if ((err = fopen_s(&fp, APERIODIC_FILE_NAME, "r")) == 0)
        printf("The file %s was opened\n", APERIODIC_FILE_NAME);
    else {
        printf("The file %s was not opened\n", APERIODIC_FILE_NAME);
        return;
    }

    char str[MAX];
    char* ptr;
    char* pTmp = NULL;
    int TaskInfo[INFO], i;
    AperTaskNumber = 0;
    int j = 0;
    while (fgets(str, sizeof(str) - 1, fp) != NULL) { // 使用 fgets 當條件比較安全
        i = 0;
        ptr = strtok_s(str, " ", &pTmp);

        while (ptr != NULL) {
            TaskInfo[i] = atoi(ptr);
            ptr = strtok_s(NULL, " ", &pTmp);

            if (i == 0) {
                // Job ID
                AperTaskParameter[j].JobId = TaskInfo[0]; // 修正邏輯
                AperTaskParameter[j].IsArrived = 0;
                AperTaskParameter[j].IsFinished = 0;
            }
            else if (i == 1) {
                AperTaskParameter[j].ArrivalTime = TaskInfo[i];
            }
            else if (i == 2) {
                AperTaskParameter[j].ExecutionTime = TaskInfo[i];
                AperTaskParameter[j].RemainTime = TaskInfo[i];
            }
            else if (i == 3) {
                AperTaskParameter[j].AbsoluteDeadline = TaskInfo[i];
            }
            i++;
        }
        if (i > 0) {
            AperTaskNumber++; // 更新全域變數
            j++;
        }
    }
    fclose(fp);
```

  - ➢ 從./Aperiodicjobs.txt 讀取資料，並使用前面創立的 structure AperTaskParameter 儲存 aperiodic job 的資料

- App_TimeTickHook():

```c
void  App_TimeTickHook (void)
{
#if (APP_CFG_PROBE_OS_PLUGIN_EN == DEF_ENABLED) && (OS_PROBE_HOOKS_EN > 0)
    OSProbe_TickHook();
#endif
    if (CUS_SERVER_SIZE > 0) {
        INT32U current_time = OSTimeGet() + 1;
        int i;
        for (i = 0; i < AperTaskNumber; i++) {
            // 檢查預判的時間是否等於到達時間
            if (AperTaskParameter[i].ArrivalTime == current_time) {
                AperTaskParameter[i].IsArrived = 1;

                // Server 忙碌中或 Deadline 未到 -> 不做事
                if (current_time < ServerDeadline && CurrentJobIndex != -1) {
                    LOG_print(3, "./Output.txt", "%d\tAperiodic job (%d) arrives. Do nothing.\n",
                        current_time, AperTaskParameter[i].JobId);
                }
                else if (CurrentJobIndex == -1) {
                    INT32U new_deadline = current_time + (INT32U)(AperTaskParameter[i].ExecutionTime / CUS_SERVER_SIZE);

                    if (AperTaskParameter[i].AbsoluteDeadline >= new_deadline) {
                        LOG_print(3, "./Output.txt", "%d\tAperiodic job (%d) arrives and sets CUS's deadline as %d.\n",
                            current_time, AperTaskParameter[i].JobId, new_deadline);

                        OS_ENTER_CRITICAL();
                        ServerDeadline = new_deadline;
                        CurrentJobIndex = i;

                        if (CusTCB != NULL) {
                            task_para_set* cus_ext = (task_para_set*)CusTCB->OSTCBExtPtr;
                            if (cus_ext != NULL) {
                                cus_ext->deadline = ServerDeadline;
                            }
                            OSTaskResume(CusTCB->OSTCBPrio);
                        }
                        OS_EXIT_CRITICAL();
                    }
                    else {
                        LOG_print(3, "./Output.txt", "%d\tAperiodic job (%d) rejects scheduling.\n",
                            current_time, AperTaskParameter[i].JobId);
                        AperTaskParameter[i].IsFinished = 1;
                    }
                }
            }
        }
```

- App_TimeTickHook 程式碼負責處理 **非週期性任務 (Aperiodic Job) 的到達事件與控制**
- 使用 OSTimeGet() + 1 預判下一個 Tick 的時間，以補償 Hook 函式在系統 Tick 更新前執行的時間差，確保任務到達的判斷與系統時鐘同步
- 伺服器狀態檢查：
  - 忙碌狀態：若 Server 尚未到達 Deadline 且正在執行任務 (CurrentJobIndex != -1)，則僅記錄新任務到達，暫不搶佔，維持當前執行流程 (Do nothing)
  - 閒置狀態：若 Server 目前無任務執行 (CurrentJobIndex == -1)，則立即計算該任務所需的 Server Deadline (d_k = t + e_k / U_s) 準備進行排程 。
- 針對閒置狀態下的新任務進行檢查：
  - 接受：若任務的 AbsoluteDeadline 晚於計算出的 Server Deadline，則更新 Server 全域參數與 TCB 擴展屬性，並喚醒 (OSTaskResume) CUS Task 進行執行 。
  - 拒絕：若無法在 Deadline 前完成，則拒絕該任務的排程要求並標記結束 。

```
if (current_time == ServerDeadline) {
    int next_job_idx = -1;
    for (i = 0; i < AperTaskNumber; i++) {
        if (AperTaskParameter[i].IsArrived && !AperTaskParameter[i].IsFinished && i != CurrentJobIndex) {
            next_job_idx = i;
            break;
        }
    }

    if (next_job_idx != -1) {
        INT32U new_deadline = current_time + (INT32U)(AperTaskParameter[next_job_idx].ExecutionTime / CUS_SERVER_SIZE);

        if (AperTaskParameter[next_job_idx].AbsoluteDeadline >= new_deadline) {
            LOG_print(3, "./Output.txt", "%d\tAperiodic job (%d) sets CUS's deadline as %d.\n",
                current_time, AperTaskParameter[next_job_idx].JobId, new_deadline);

            OS_ENTER_CRITICAL();
            ServerDeadline = new_deadline;
            CurrentJobIndex = next_job_idx;

            if (CusTCB != NULL) {
                task_para_set* cus_ext = (task_para_set*)CusTCB->OSTCBExtPtr;
                if (cus_ext != NULL) {
                    cus_ext->deadline = ServerDeadline;
                }
                OSTaskResume(CusTCB->OSTCBPrio);
            }
            OS_EXIT_CRITICAL();
        }
        else {
            LOG_print(3, "./Output.txt", "%d\tAperiodic job (%d) rejects scheduling.\n",
                current_time, AperTaskParameter[next_job_idx].JobId);
            AperTaskParameter[next_job_idx].IsFinished = 1;
        }
    }
}
}
```

- 當系統時間到達伺服器當前的 deadline (current_time == ServerDeadline) 時執行，此時伺服器預算已用盡或過期，需檢查是否有其他的任務等待處理。
- 若存在非週期任務，依據 CUS 公式計算新的伺服器 Deadline
- 比較任務本身的絕對截止期限與新計算的伺服器截止時間：
  - **接受排程**：若任務期限足夠 (AbsoluteDeadline >= NewDeadline)，則更新全域變數與 CUS TCB 的 Deadline，並喚醒 (OSTaskResume) CUS Task 進行處理。
  - **拒絕排程**：若任務無法在新的截止時間前完成，則拒絕該次排程並將任務標記為結束。

3. main.c:

- main function:

```
int  main(void)
{
#if OS_TASK_NAME_EN > 0u
    CPU_INT08U  os_err;
#endif

    OS_TCB* ptcb;
    CPU_IntInit();

    Mem_Init();                                 /* Initialize Memory Managment Module                   */
    CPU_IntDis();                               /* Disable all Interrupts                               */
    CPU_Init();                                 /* Initialize the uC/CPU services                       */

    OSInit();                                   /* Initialize uC/OS-II                                  */


    OutFileInit();
    InputFile();
    InputAperiodicjobsFile();
```

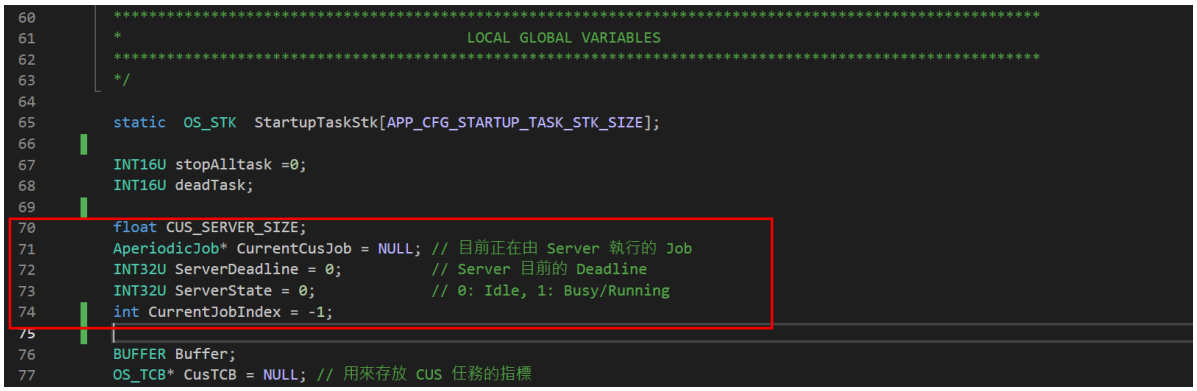
  - 開啟 aperiodicjob.txt 並讀取檔案

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

    }

    // 2. 單獨建立 CUS Task (第 TASK_NUMBER - 1 個)
    int cus_index = TASK_NUMBER - 1;
    Task_STK[cus_index] = malloc(TASK_STACKSIZE * sizeof(int));

    OSTaskCreateExt(CUS_Task,      //傳入 CUS 專用的函式
        &TaskParameter[cus_index],      // 參數結構
        &Task_STK[cus_index][TASK_STACKSIZE - 1],
        TaskParameter[cus_index].TaskPriority,
        TaskParameter[cus_index].TaskID,
        &Task_STK[cus_index][0],
        TASK_STACKSIZE,
        &TaskParameter[cus_index],
        (OS_TASK_OPT_STK_CHK | OS_TASK_OPT_STK_CLR));

    CusTCB = OSTCBPrioTbl[TaskParameter[cus_index].TaskPriority];

    if (CusTCB == NULL) {
        printf("Error: CUS Task TCB not found!\n");
    }
}
```

```
else {
    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].TaskPriority,
            TaskParameter[n].TaskID,
            &Task_STK[n][0],
            TASK_STACKSIZE,
            &TaskParameter[n],
            (OS_TASK_OPT_STK_CHK | OS_TASK_OPT_STK_CLR));

    }
}
```

  - 如果 CUS_SERVER_SIZE > 0，代表有 CUS server，需要額外創一個 CUS_task function 去處理所有的 Aperiodic job 任務

- Global variable:



```
/*********************************************************************************************************
*                                      LOCAL GLOBAL VARIABLES
*********************************************************************************************************
*/

static  OS_STK  StartupTaskStk[APP_CFG_STARTUP_TASK_STK_SIZE];

INT16U stopAlltask =0;
INT16U deadTask;

float CUS_SERVER_SIZE;
AperiodicJob* CurrentCusJob = NULL; // 目前正在由 Server 執行的 Job
INT32U ServerDeadline = 0;          // Server 目前的 Deadline
INT32U ServerState = 0;             // 0: Idle, 1: Busy/Running
int CurrentJobIndex = -1;

BUFFER Buffer;
OS_TCB* CusTCB = NULL; // 用來存放 CUS 任務的指標
```

  - 宣告全域變數 ：ServerDeadline、job_Index 等等(OSTimeTick 使用

- CUS_Task function

```
void CUS_Task(void* parg) {

    while (1) {
        OS_ENTER_CRITICAL();
        if (CurrentJobIndex == -1) {
            OS_EXIT_CRITICAL();
            OSTaskSuspend(OS_PRIO_SELF);

            OS_ENTER_CRITICAL();
            if (CurrentJobIndex == -1) {
                OS_EXIT_CRITICAL();
                continue;
            }
        }
        OS_EXIT_CRITICAL();

        if (CurrentJobIndex < 0 || CurrentJobIndex >= AperTaskNumber) continue;

        AperiodicJob* myJob = &AperTaskParameter[CurrentJobIndex];

        while (myJob->RemainTime > 0) {
            LOG_print(3, "./Output.txt", "% d\tAperiodic job(% d) is running\n", OSTimeGet(), myJob->JobId);

            int current_time = OSTimeGet();
            while (OSTimeGet() == current_time) {
                // Busy wait
            }
            myJob->RemainTime--;
        }
```

  - CUS_Task 程式碼負責**執行被分配的非週期性任務**
  - 任務掛起與等待 (Task Suspension & Waiting)： 若當前無指派任務 (CurrentJobIndex == -1)，Task 主動掛起 (OSTaskSuspend) 以釋放 CPU 資源，等待 TimeTickHook 分派任務並喚醒
  - 執行時間： 獲取任務後，透過 Busy Wait 迴圈 (while (OSTimeGet() == current_time)) 鎖定 CPU 直到系統 Tick 增加，藉此精確模擬任務在每一個 Tick 的執行消耗。
  - 剩餘時間更新： 在每個 Tick 執行完畢後，遞減任務的 RemainTime，並持續輸出 "Aperiodic job is running" 狀態，直到任務執行時間歸零。

```
OS_ENTER_CRITICAL();
if (myJob->RemainTime == 0) {

    INT32U current_time = OSTimeGet();

    // 計算時間數據
    int response_time = current_time - myJob->ArrivalTime;
    int preemption_time = response_time - myJob->ExecutionTime;

    // 尋找 "Next Task"
    OS_TCB* next_tcb = NULL;
    INT32U min_deadline = 0xFFFFFFFF; // 初始設為最大值
    INT16U min_task_id = 0xFFFF;

    OS_TCB* ptcb = OSTCBList; // 從 TCB List 頭開始掃

    while (ptcb != NULL) {
        if (ptcb->OSTCBStat == OS_STAT_RDY &&
            ptcb->OSTCBDly == 0 &&
            ptcb != OSTCBCur &&
            ptcb->OSTCBPrio != OS_TASK_IDLE_PRIO)
        {
            // 取得該任務的 Deadline
            if (ptcb->OSTCBExtPtr != NULL) {
                task_para_set* ext = (task_para_set*)ptcb->OSTCBExtPtr;
                INT32U task_dl = ext->deadline;
                INT16U task_id = ext->TaskID;

                // EDF 比較邏輯
                if (task_dl < min_deadline) {
                    min_deadline = task_dl;
                    min_task_id = task_id;
                    next_tcb = ptcb;
                }
                // Tie-Breaker: Deadline 相同，選 ID 小的 [cite: 18]
                else if (task_dl == min_deadline) {
                    if (task_id < min_task_id) {
                        min_task_id = task_id;
                        next_tcb = ptcb;
                    }
                }
            }
        }
        ptcb = ptcb->OSTCBNext; // 下一個 TCB
    }
```

- 依據當前時間 (current_time) 計算該任務的 Response Time 與 Preemption Time
- 搜尋所有 OSTCBList 尋找下一個任務(EDF 排程:同 OSSchedNew())

```
    int next_id = 0;
    int next_job_no = 0;

    if (next_tcb != NULL) {
        task_para_set* next_ext = (task_para_set*)next_tcb->OSTCBExtPtr;
        next_id = next_ext->TaskID;
        next_job_no = next_ext->job_no;
    }
    else {
        // 沒找到任何任務 下一個是 Idle Task (Task 63)
        next_id = 63;
        next_job_no = 0;
    }

    // 4. 印出 Completion 訊息
    int cus_task_id = TaskParameter[TASK_NUMBER - 1].TaskID;
    LOG_print(3, "./Output.txt", "%d\tAperiodic job (%d) is finished.\n", current_time, myJob->JobId);
    LOG_print(3, "./Output.txt", "%d\tCompletion\ttask(%2d)(%2d)\ttask(%2d)(%2d)\t%d\t%d\tN/A\n",
        current_time,
        cus_task_id,      // Current: CUS ID
        myJob->JobId,     // Current: Job Number
        next_id,          // Next: ID
        next_job_no,      // Next: Job Number
        response_time,
        preemption_time);

    myJob->IsFinished = 1;
    CurrentJobIndex = -1;
}
OS_EXIT_CRITICAL();
}
}
```

- 印出 Aperiodic job finish 訊息
- 印出 Aperiodic job Complete 訊息