

110.1 Embedded OS Implementation

PA#2 Report

四電機四乙 B10707119 陳俊宇
2021/12/14

[PART I] EDF Scheduler Implementation

Screenshot of modified code

(未以紅色括弧標示之截圖表示全部為新程式碼)

typedef struct OS_TCB

```
656 // PA#1
657 //INT8U already_delay;
658 INT8U remain_exe_time;
659 //INT8U next_job_time;
660 //INT8U self_continue;
661 // PA#1
662
663 // PA#2
664 INT8U arrive_time;
665 INT8U exe_time;
666 INT8U period;
667 INT8U deadline;
668 // PA#2
669
670 #if OS_TASK_DEL_EN > 0u
671 INT8U OSTCDBelReq; /* Indicates whether a task needs to delete itself */
672 #endif
673
674 #if OS_TASK_PROFILE_EN > 0u
675 INT32U OSTCBCtxSwCtr; /* Number of time the task was switched in */
676 INT32U OSTCBCyclesTot; /* Total number of clock cycles the task has been running */
677 INT32U OSTCBCyclesStart; /* Snapshot of cycle counter at start of task resumption */
678 OS_STK *OSTCBStkBase; /* Pointer to the beginning of the task stack */
679 INT32U OSTCBStkUsed; /* Number of bytes used from the stack */
680 #endif
```

OS_TCBInit()

```
2291
2292 INT8U OS_TCBInit (INT8U prio,
2293 OS_STK *ptos,
2294 OS_STK *pbos,
2295 INT16U id,
2296 INT32U stk_size,
2297 void *pext,
2298 INT16U opt,
2299 INT8U arrive_time, INT8U exe_time, INT8U period) // PA#2
2300 {
2301
2302 // PA#1 part1
2303
2304 // PA#2 初始化額外TCB變數
2305 if (id == OS_TASK_IDLE_ID)
2306 {
2307     ptcb->arrive_time = 0;
2308     ptcb->exe_time = 255;
2309     ptcb->period = 255;
2310     ptcb->deadline = 255;
2311     ptcb->remain_exe_time = 255;
2312 }
2313 else
2314 {
2315     ptcb->arrive_time = arrive_time;
2316     ptcb->exe_time = exe_time;
2317     ptcb->period = period;
2318     ptcb->deadline = arrive_time + period;
2319     ptcb->remain_exe_time = exe_time;
2320 }
2321
2322 // PA#2 如果arrive time == 0, 設為ready; 反之設定OSTCDBdy = arrive_time
2323 if (arrive_time == 0)
2324 {
2325     OSRdyGrp |= ptcb->OSTCBBdy; /* Make task ready to run */
2326     OSRdyTbl[ptcb->OSTCBY] |= ptcb->OSTCBBdy;
2327 }
2328 else
2329 {
2330     ptcb->OSTCDBdy = arrive_time;
2331 }
2332
2333 OSTaskCtr++; /* Increment the #tasks counter */
2334 OS_TRACE_TASK_READY(ptcb);
2335 OS_EXIT_CRITICAL();
2336 return (OS_ERR_NONE);
2337 }
2338 OS_EXIT_CRITICAL();
2339 return (OS_ERR_TASK_NO_MORE_TCB);
2340 }
```

OSTaskCreateExt()

```
348 #if OS_TASK_CREATE_EXT_EN > 0u
349 INT8U OSTaskCreateExt (void (*task)(void *p_arg),
350                        void *p_arg,
351                        OS_STK *ptos,
352                        INT8U prio,
353                        INT16U id,
354                        OS_STK *pbos,
355                        INT32U stk_size,
356                        void *pext,
357                        INT16U opt,
358                        INT8U arrive_time, INT8U exe_time, INT8U period)
```

main.c (task creation and task function)

```
134 // PA#3, 新增3個argument
135 for (n = 0; n < TASK_NUMBER; n++)
136 {
137     // printf("ID %d, prio %d\n", TaskParameter[n].TaskID, n+1);
138     OSTaskCreateExt(task1, // task function
139                    &TaskParameter[n], // p_arg(給 task function的參數)
140                    &Task_STK[n][TASK_STACKSIZE - 1], // ptos
141                    n+1, // prio, PA#1要求從1開始
142                    TaskParameter[n].TaskID, // id
143                    &Task_STK[n][0], // pbos
144                    TASK_STACKSIZE, // stack size
145                    &TaskParameter[n], // pext(TCB extensions的pointer)
146                    (OS_TASK_OPT_STK_CHK | OS_TASK_OPT_STK_CLR), // opt
147                    TaskParameter[n].TaskArriveTime, // arrive_time
148                    TaskParameter[n].TaskExecutionTime, // exe_time
149                    TaskParameter[n].TaskPeriodic // period
150                );
151 }
```

```
187 void task1(void* p_arg) {
188     task_para_set* task_data;
189     task_data = p_arg;
190
191     while (1)
192     {
193     }
194 }
```

OS_SchedNew()

```
1905 static void OS_SchedNew (void)
1906 {
1907     #if OS_LOWEST_PRIO <= 63u // See if we support up to 64 tasks */
1908     // =====RMS=====
1909     INT8U y;
1910     //y = OSUnMapTbl[OSRdyGrp];
1911     OSPrioHighRdy = (INT8U)((y << 3u) + OSUnMapTbl[OSRdyTbl[y]]);
1912     // =====RMS=====
1913
1914     // =====EDF=====
1915     OSPrioHighRdy = find_ED_prio();
1916     // printf("szd : OSPrioHighRdy = %d \n", OSTimeGet(), OSPrioHighRdy);
1917     // =====EDF=====
```

```
1866 // PA#2, For EDF, 看過OSRdyTbl一遍, 找出deadline最近的task的prio
1867 int find_ED_prio()
1868 {
1869     int ED_prio = OS_LOWEST_PRIO;
1870     int ED = 255;
1871     for (int prio = 0; prio <= OS_LOWEST_PRIO; prio++)
1872     {
1873         int row = prio / 8;
1874         int col_mask = 0b1 << (prio%8);
1875
1876         if ((OSRdyTbl[row] & col_mask) != 0) // 如果某個task是ready的
1877         {
1878             // printf("OSTCBPrioTbl[%d]->deadline = %d \n", prio, OSTCBPrioTbl[prio]->deadline);
1879             if (OSTCBPrioTbl[prio]->deadline < ED) // 如果這個task的deadline是目前看到最小的
1880             {
1881                 ED_prio = prio;
1882                 ED = OSTCBPrioTbl[prio]->deadline;
1883             }
1884         }
1885     }
1886     return ED_prio;
```

OSTimeTick() (包含 miss deadline handling)

```
1045
1046
1047     task_start_time[ptcb->OSTCBPrio] = OSTimeGet(); // 紀錄這個job ready時間，用於計算response time
1048     // PA#1
1049     OS_TRACE_TASK_READY(ptcb);
1050
1051 }
1052
1053
1054
1055 ptcb = ptcb->OSTCBNext; /* Point at next TCB in TCB list */
1056 OS_EXIT_CRITICAL();
1057
1058
1059
1060 // PA#2: 把remain_exe_time處理改到OSTimeTick()處理
1061 if (OSTCBCur->remain_exe_time > 0)
1062 {
1063     OSTCBCur->remain_exe_time--;
1064     // printf("tick %d choose %d to --\n", OSTimeGet(), OSTCBCur->OSTCBPrio);
1065
1066     if (OSTCBCur->remain_exe_time == 0)
1067     {
1068         response_time = OSTimeGet() - task_start_time[OSTCBCur->OSTCBPrio];
1069
1070         task_start_time[OSTCBCur->OSTCBPrio] = OSTimeGet(); // 紀錄這個job ready時間，用於計算response time
1071
1072         OSTCBCur->arrive_time = OSTCBCur->arrive_time + OSTCBCur->period; // 設下個job開始時間
1073         OSTCBCur->deadline = OSTCBCur->arrive_time + OSTCBCur->period; // 設下個job的deadline
1074         OSTCBCur->remain_exe_time = OSTCBCur->exe_time; // 重設剩餘執行時間
1075
1076         INT8U delay_ticks = OSTCBCur->arrive_time - OSTimeGet();
1077         if (delay_ticks > 0u) { /* 0 means no delay! */
1078             INT8U y = OSTCBCur->OSTCBY; /* Delay current task */
1079             OSRdyTbl[y] &= (OS_PRIO)-OSTCBCur->OSTCBBitX;
1080             OS_TRACE_TASK_SUSPENDED(OSTCBCur);
1081             if (OSRdyTbl[y] == 0u) {
1082                 OSRdyGrp &= (OS_PRIO)-OSTCBCur->OSTCBBitY;
1083             }
1084             OSTCBCur->OSTCBBdy = delay_ticks; /* Load ticks in TCB */
1085         }
1086     }
1087 }
1088
1089
1090 // =====miss deadline=====
1091 miss_deadline_prio = check_deadline();
1092 while (miss_deadline_prio != -1)
1093 {
1094     if ((Output_err = fopen_s(&Output_fp, "../Output.txt", "a")) != 0)
1095     {
1096         printf("can't open Output.txt");
1097         exit(0);
1098     }
1099     printf("%2d\t", OSTimeGet());
1100     fprintf(Output_fp, "%2d\t", OSTimeGet());
1101     printf("MissDeadline\t");
1102     fprintf(Output_fp, "MissDeadline\t");
1103
1104     // CurrentTask
1105     printf("Task(%2d)(%2d)\t", OSTCBPrioTbl[miss_deadline_prio]->OSTCBId, job_number[miss_deadline_prio]);
1106     fprintf(Output_fp, "Task(%2d)(%2d)\t", OSTCBPrioTbl[miss_deadline_prio]->OSTCBId, job_number[miss_deadline_prio]);
1107
1108     // NextTask
1109     printf("-----\n");
1110     fprintf(Output_fp, "-----\n");
1111
1112     // =====miss deadline handling=====
1113     printf("===== Drop Task(%2d)(%2d) at tick %2d with remaining execution time %2d. =====\n", \
1114           OSTCBPrioTbl[miss_deadline_prio]->OSTCBId, job_number[miss_deadline_prio], OSTimeGet(), OSTCBPrioTbl[miss_deadline_prio]->remain_exe_time);
1115     fprintf(Output_fp, "===== Drop Task(%2d)(%2d) at tick %2d with remaining execution time %2d. =====\n", \
1116           OSTCBPrioTbl[miss_deadline_prio]->OSTCBId, job_number[miss_deadline_prio], OSTimeGet(), OSTCBPrioTbl[miss_deadline_prio]->remain_exe_time);
1117     fclose(Output_fp);
1118
1119     task_start_time[miss_deadline_prio] = OSTimeGet(); // 紀錄這個job ready時間，用於計算response time
1120
1121     OSTCBPrioTbl[miss_deadline_prio]->arrive_time = OSTimeGet(); // 重設job開始時間
1122     OSTCBPrioTbl[miss_deadline_prio]->deadline = OSTimeGet() + OSTCBPrioTbl[miss_deadline_prio]->period; // 重設job的deadline
1123     OSTCBPrioTbl[miss_deadline_prio]->remain_exe_time = OSTCBPrioTbl[miss_deadline_prio]->exe_time; // 重設剩餘執行時間
1124
1125     missed_deadline = 1;
1126     miss_deadline_prio = check_deadline();
1127 }
1128
1129
1130
1131 // PA#2: For EDF, 看過OSRdyTbl一遍，檢查是否有miss deadline，有則return miss deadline的prio，無則return-1
1132 int check_deadline()
1133 {
1134     int miss_deadline_prio = -1;
1135     for (int prio = 0; prio <= OS_LOWEST_PRIO; prio++)
1136     {
1137         int row = prio / 8;
1138         int col_mask = 0b1 << (prio % 8);
1139
1140         if ((OSRdyTbl[row] & col_mask) != 0) // 如果某個task是ready的
1141         {
1142             if (OSTCBPrioTbl[prio]->deadline <= OSTimeGet())
1143             {
1144                 miss_deadline_prio = prio;
1145                 break;
1146             }
1147         }
1148     }
1149     return miss_deadline_prio;
1150 }
```

Description of implementation

從 PA#1 進行修改，將排程方式由 RMS 改為 EDF，最主要的改變是 OS_SchedNew() 中，原本 OSPrioHighRdy 會設為 priority 最高的 task，而 EDF 則要設為 deadline 最近的 task。

我的方法是在 TCB 中新增變數來紀錄各個 task 的 deadline，在每次排程時，遍歷所有 ready 的 task，從中選出 deadline 最小的 task，將其 priority assign 給 OSPrioHighRdy。

另外，與 PA#1 做法不同的地方還有：我將計算剩餘 execution time 的 code 從 task function 移至 OSTimeTick()，能夠更簡單、統一地處理各種類型的 context switch。

RMS 與 EDF 的共通點是必須要隨時更新、檢查各個 task 是否已 ready，因此保留了原本的 OSRdyTbl。其餘輸出資訊、response time 等等計算，亦與 PA#1 相同。

How to handle the missing deadline situation under EDF

我實作的 miss deadline 處理方式為：任何一個 task 發生 miss deadline 時，**立即中止執行中的 task**，顯示 miss deadline 的時間點、中止執行的 task ID、其剩餘的執行時間等。接下來，**將 miss deadline 的 job 視為不存在**，繼續依照 EDF 規則進行排程。

下圖為 TaskSet Example 2 的執行結果。

20	Completion	Task(1)(4)	Task(3)(6)	4	2	0
21	Completion	Task(3)(6)	Task(1)(5)	2	2	0
22	Completion	Task(1)(5)	Task(2)(3)	2	2	0
24	MissDeadline	Task(2)(3)	-----			
===== Drop Task(2)(3) at tick 24 with remaining execution time 1. =====						
25	Completion	Task(3)(7)	Task(1)(6)	3	1	0
26	Completion	Task(1)(6)	Task(3)(8)	2	2	0
27	Completion	Task(3)(8)	Task(2)(3)	2	2	0
30	Completion	Task(2)(3)	Task(3)(9)	6	3	0
31	Completion	Task(3)(9)	Task(1)(7)	3	2	0
32	Completion	Task(1)(7)	Task(3)(10)	4	2	0
33	Completion	Task(3)(10)	Task(1)(8)	2	2	0
34	Completion	Task(1)(8)	Task(2)(4)	2	2	0
36	MissDeadline	Task(2)(4)	-----			
===== Drop Task(2)(4) at tick 36 with remaining execution time 1. =====						
37	Completion	Task(3)(11)	Task(1)(9)	3	1	0
38	Completion	Task(1)(9)	Task(3)(12)	2	2	0
39	Completion	Task(3)(12)	Task(2)(4)	2	2	0

[PART II] CUS Scheduler Implementation

Screenshot of modified code

(未以紅色括弧標示之截圖表示全部為新程式碼)

Reading input files

```
126 while (ptr != NULL)
127 {
128     TaskInfo[i] = atoi(ptr);
129     ptr = strtok_s(NULL, " ", &pTmp);
130
131     /// printf("Info: %d\n", TaskInfo[i]);
132     if (i == 0) {
133         TASK_NUMBER++;
134         TaskParameter[j].TaskID = TaskInfo[i];
135     }
136     else if (i == 1)
137         TaskParameter[j].TaskArriveTime = TaskInfo[i];
138     else if (i == 2)
139         TaskParameter[j].TaskExecutionTime = TaskInfo[i];
140     else if (i == 3)
141         TaskParameter[j].TaskPeriodic = TaskInfo[i];
142
143     i++;
144 }
145 j++;
146 }
147
148 SERVER_ID = TaskParameter[j - 1].TaskID;
149 SERVER_SIZE = TaskParameter[j - 1].TaskArriveTime;
150 fclose(fp);

160 if ((err = fopen_s(&fp, APERIODIC_FILE_NAME, "r")) == 0)
161 {
162     /// printf("The file 'AperiodicJobs.txt' was opened\n");
163 }
164 else
165 {
166     printf("The file 'AperiodicJobs.txt' was not opened\n");
167 }
168
169 char str[MAX];
170 char* ptr;
171 char* pTmp = NULL;
172 int TaskInfo[INFO], i, j = 0;
173 while (!feof(fp))
174 {
175     i = 0;
176     memset(str, 0, sizeof(str));
177     fgets(str, sizeof(str) - 1, fp);
178     ptr = strtok_s(str, " ", &pTmp);
179
180     while (ptr != NULL)
181     {
182         TaskInfo[i] = atoi(ptr);
183         ptr = strtok_s(NULL, " ", &pTmp);
184
185         /// printf("Info: %d\n", TaskInfo[i]);
186         if (i == 0) {
187             AperiodicJobParameter[j].TaskID = TaskInfo[i];
188         }
189         else if (i == 1)
190             AperiodicJobParameter[j].TaskArriveTime = TaskInfo[i];
191         else if (i == 2)
192             AperiodicJobParameter[j].TaskExecutionTime = TaskInfo[i];
193         else if (i == 3)
194             AperiodicJobParameter[j].AbsoluteDeadline = TaskInfo[i];
195     }
196 }

94 typedef struct aperiodic_job_para_set {
95     INT16U TaskID;
96     INT16U TaskArriveTime;
97     INT16U TaskExecutionTime;
98     INT16U AbsoluteDeadline;
99 } aperiodic_job_para_set;
```


CUS task creation

```
117 // PA#3: OSTaskCreateExt新增3個argument
118 for (n = 0; n < TASK_NUMBER-1; n++)
119 {
120     // printf("ID %d, prio %d\n", TaskParameter[n].TaskID, n+1);
121     OSTaskCreateExt(task1, // task function
122         &TaskParameter[n], // p_arg(給task function的參數)
123         &Task_STK[n][TASK_STACKSIZE - 1], // p_tos
124         n+1, // prio, PA#1要求從1開始
125         TaskParameter[n].TaskID, // id
126         &Task_STK[n][0], // p_bos
127         TASK_STACKSIZE, // stack size
128         &TaskParameter[n], // pext(TCB extension的pointer)
129         (OS_TASK_OPT_STK_CHK | OS_TASK_OPT_STK_CLR), // opt
130         TaskParameter[n].TaskArriveTime, // arrive_time
131         TaskParameter[n].TaskExecutionTime, // exe_time
132         TaskParameter[n].TaskPeriodic, // period
133     );
134 }
135
136 // server task
137 OSTaskCreateExt(task1, // task function
138     &TaskParameter[n], // p_arg(給task function的參數)
139     &Task_STK[n][TASK_STACKSIZE - 1], // p_tos
140     n + 1, // prio, PA#1要求從1開始
141     SERVER_ID, // id
142     &Task_STK[n][0], // p_bos
143     TASK_STACKSIZE, // stack size
144     &TaskParameter[n], // pext(TCB extension的pointer)
145     (OS_TASK_OPT_STK_CHK | OS_TASK_OPT_STK_CLR), // opt
146     0, // arrive_time don't care
147     0, // exe_time don't care
148     255, // period don't care
149 );
150 SERVER_PRIO = n + 1;
```

OSTimeTick()

```
976 int job_index_queue[OS_MAX_TASKS] = { -1 }; // 紀錄server要去執行的job，其在AperiodicJobParameter的index
977 int front = -1; // 指向queue的最後一個job -1的位置
978 int tail = -1; // 指向queue的最後一個job
979 int current_job_id = -1;
980
981 // PA#2: For CUS，看過AperiodicJobParameter一遍，檢查是否有aperiodic job要開始
982 void enqueue_aperiodic_job()
983 {
984     for (int i = 0; i < OS_MAX_TASKS; i++)
985     {
986         if (AperiodicJobParameter[i].TaskArriveTime == OSTimeGet())
987         {
988             tail++;
989             job_index_queue[tail] = i;
990             aperiodic_job_start_time[AperiodicJobParameter[i].TaskID] = OSTimeGet();
991         }
992     }
993 }
994 void dequeue_aperiodic_job()
995 {
996     front++;
997 }
```

```
1158 // ===== CUS =====
1159 enqueue_aperiodic_job();
1160 ptcb = OSTCBPrioTbl[SERVER_PRIO];
1161 if (front != tail && (OSTimeGet() >= ptcb->deadline || ptcb->arrive_time == 0)) // queue裡有job待執行，且目前CUS為空閒(OSTimeGet() > server dead
1162 {
1163     if ((ptcb->OSTCBStat & OS_STAT_SUSPEND) == OS_STAT_RDY) { /* Is task suspended? */
1164         OSRdyGrp |= ptcb->OSTCBItY; /* No, Make ready */
1165         OSRdyTbl[ptcb->OSTCBY] |= ptcb->OSTCBItX;
1166     }
1167
1168     ptcb->arrive_time = OSTimeGet();
1169     ptcb->deadline = OSTimeGet() + AperiodicJobParameter[job_index_queue[front + 1]].TaskExecutionTime * 100 / SERVER_SIZE;
1170     ptcb->remain_exe_time = AperiodicJobParameter[job_index_queue[front + 1]].TaskExecutionTime;
1171     current_job_id = AperiodicJobParameter[job_index_queue[front + 1]].TaskID;
1172
1173     if (AperiodicJobParameter[job_index_queue[front + 1]].TaskArriveTime == OSTimeGet()) // arrive時，剛好CUS可以直接開始
1174     {
1175         if ((Output_err = fopen_s(&Output_fp, "./Output.txt", "a")) != 0)
1176         {
1177             printf("can't open Output.txt");
1178             exit(0);
1179         }
1180         printf("%2d\t", OSTimeGet());
1181         fprintf(Output_fp, "%2d\t", OSTimeGet());
1182         printf("Aperiodic job(%d) arrives and sets CUS server's deadline as %d.\n", current_job_id, ptcb->deadline);
1183         fprintf(Output_fp, "Aperiodic job(%d) arrives and sets CUS server's deadline as %d.\n", current_job_id, ptcb->deadline);
1184         fclose(Output_fp);
1185     }
1186     else // job先等待了前一個CUS deadline，之後才開始
1187     {
1188         if ((Output_err = fopen_s(&Output_fp, "./Output.txt", "a")) != 0)
1189         {
1190             printf("can't open Output.txt");
1191             exit(0);
1192         }
1193         printf("%2d\t", OSTimeGet());
1194 }
```

```

1195         fprintf(Output_fp, "%2d\t", OSTimeGet());
1196         printf("Aperiodic job(%d) sets CUS server's deadline as %d.\n", current_job_id, ptcb->deadline);
1197         fprintf(Output_fp, "Aperiodic job(%d) sets CUS server's deadline as %d.\n", current_job_id, ptcb->deadline);
1198         fclose(Output_fp);
1199     }
1200     dequeue_aperiodic_job(); // 讓CUS開始執行，就先移出queue
1201 }
1202 else if (front != tail && OSTimeGet() < ptcb->deadline) // queue裡有job待執行，且還沒到CUS deadline
1203 {
1204     for (int i = front+1; i <= tail; i++)
1205     {
1206         if (AperiodicJobParameter[job_index_queue[front + 1]].TaskArriveTime == OSTimeGet()) // 還沒被CUS執行的job會一直在queue等待，但只需在arr
1207         {
1208             if ((Output_err = fopen_s(&Output_fp, "./Output.txt", "a")) != 0)
1209             {
1210                 printf("can't open Output.txt");
1211                 exit(0);
1212             }
1213             printf("%2d\t", OSTimeGet());
1214             fprintf(Output_fp, "%2d\t", OSTimeGet());
1215             printf("Aperiodic job(%d) arrives. Do nothing.\n", AperiodicJobParameter[job_index_queue[i]].TaskID);
1216             fprintf(Output_fp, "Aperiodic job(%d) arrives. Do nothing.\n", AperiodicJobParameter[job_index_queue[i]].TaskID);
1217             fclose(Output_fp);
1218         }
1219     }
1220 }
1221 // PA#2
1222 }
1223 }

```

Description of implementation

以part I的EDF排程為基礎，加上了aperiodic jobs、server task的處理。

首先，仿照了處理 TaskSet 的方式進行檔案讀取、並建立了新的 struct 來儲存 aperiodic jobs 的資訊。

Server 方面，在資料結構上，sever 與其他 periodic job 同為 task，但是 sever 需要進行特別的處理，不可混為一談，因此額外建立了全域變數 SERVER_ID, SERVER_PRIO 來紀錄 server 的 task ID, priority，方便後續處理、辨別。

排程的實作上，每次 OSTimeTick()時都會去檢查是否有 aperiodic jobs 的 arrival time == OSTimeGet()，並將所有符合條件者加入 queue。

若 queue 不為空，則再判斷目前 CUS 是否為空閒狀態(初始狀態或是已經過了前一個 deadline)，若為空閒，則根據 server size、queue 中第一個 aperiodic job 之 execution time 來計算、設定 server deadline，最後將此 job 移出 queue；若尚未經過前一個 CUS deadline，則需等待 deadline 到達時再進行上述動作。