

嵌入式系统概论实验报告

—— EDF 算法实现

南京大学软件学院

151250048

郭浩滨

一、实验目的

在 ucOS-II 上实现 EDF 调度。

由于 ucOS 默认只提供对固定优先级调度的支持，在此基础上实现 RM 调度较为容易，但效率不及最优调度的 EDF 算法。本实验拟实现 EDF 算法。

二、实验环境

本实验代码修改自 ucOS-II 移植版本，使用 VS2013 编译运行，VSCODE 编写代码。

三、EDF 调度算法实现

3.1. 思路及方法

0. 在看完 PPT 并了解 ucOS 的调度过程和 EDF 算法基本原理以及核心源代码之后，得出总体的过程是在 `os_core.c` 中实现一个 EDF 调度函数 `OS_SchedByEdf` 来覆盖默认的 `OS_SchedNew` 函数，并在 `TimeTick` 的代码后面添加对 `DDL`、`CompTime` 等的维护，这个过程需要先修改在 `ucos_ii.h` 文件中 `TCB` 的结构定义，然后在 `main.c` 文件中增加一些 `task`。

1. 在 `OS_Start()`，`OS_IntExit()`和 `OS_Sched()`均会调用调度算法

2. 关于修改 `TCB` 结构定义：`TCB` 中有一个指向用户自定义结构的拓展指针

`OSTCBExtPtr`，于是可以建立一个结构体存放额外的 `TCB` 属性（具体结构见下文）。

3. 全部修改的文件包括：main.c os_cfg.h ucos_ii.h os_core.c

3.2. 核心数据结构

```
typedef struct edf_task_data{
    INT32U c;
    INT32U p;
    INT32U rc; // the remaining c in per p
    INT32U ddl;
    INT32U start;
    INT32U end;
} EDF_TASK_DATA;
```

1. c 和 p 是对任务的定义，表示在 p 个 tick 里面需要有 c 个始终周期分配给此任务
2. rc 用于存储在当前周期里这个任务剩余的 c（即还需要消耗的 tick 数量）
3. ddl 是该任务的截止时限
4. start 和 end 是计算当前周期内需要 delay 的时间的辅助变量

3.3. 核心算法

```
static void OS_SchedByEdf(void)
{
    int earliestDDL = 1000000; // temporal latest deadline
    int ddl;
    int isAllDelay = 1;

    OSTCB* curTask = OSTCBLst;
    OSTCB* nextTask = OSTCBPrioTbl[OS_IDLE_PRIO];
    OSPrioHighRdy = OS_IDLE_PRIO;

    while(curTask->OSTCBPrio != OS_IDLE_PRIO) {
        if (curTask->OSTCBExtPtr == 0) {
            curTask->OSTCBExtPtr = &idleEdf;
        }
        // linearly find the task with earliest deadline
        if (curTask->OSTCBDly == 0 && ((EDF_TASK_DATA *)curTask->OSTCBExtPtr)->rc>0) {
            ddl = ((EDF_TASK_DATA*) curTask->OSTCBExtPtr)->ddl;
            if (ddl < earliestDDL) {
                earliestDDL = ddl;
                nextTask = curTask;
            }
            isAllDelay = 0;
        }
        curTask = curTask->OSTCBNext;
    }

    int nextTaskID = nextTask->OSTCBId;
    if (OSTCBCur->OSTCBId != nextTaskID) {
        EDF_TASK_DATA* tempPoint = (EDF_TASK_DATA *) OSTCBCur->OSTCBExtPtr;
        if (complete == 1){
            printf("%d\tComplete\t%d\t%d\n", OSTimeGet(), OSTCBCur->OSTCBId, nextTaskID);
        }else{
            printf("%d\tPreempted\t%d\t%d\n", OSTimeGet(), OSTCBCur->OSTCBId, nextTaskID);
        }
    }
    OSPrioHighRdy = nextTask->OSTCBPrio;
    if(isAllDelay == 1) {
        OSPrioHighRdy = OS_IDLE_PRIO;
    }
}
```

3.4. 测试用例

1. TaskSet 1 = { t1(1,3), t2(3,6) }

```
C:\uCOS-II\Debug\uCOSII.exe
1    Complete    1    2
4    Complete    2    1
5    Complete    1    2
6    Preempted   2    1
7    Complete    1    2
9    Complete    2    1
10   Complete    1    2
13   Complete    2    1
14   Complete    1    65535
15   Preempted   65535  1
16   Complete    1    2
19   Complete    2    1
20   Complete    1    2
21   Preempted   2    1
22   Complete    1    2
24   Complete    2    1
25   Complete    1    2
28   Complete    2    1
29   Complete    1    65535
30   Preempted   65535  1
31   Complete    1    2
34   Complete    2    1
35   Complete    1    2
36   Preempted   2    1
37   Complete    1    2
39   Complete    2    1
40   Complete    1    2
43   Complete    2    1
44   Complete    1    65535
45   Preempted   65535  1
```

2. TaskSet 2 = { t1(1,3), t2(3,6), t3(4,9) }

```
C:\uCOS-II\Debug\uCOSII.exe
1    Complete    3    4
3    Complete    4    5
4    Preempted   5    3
5    Complete    3    5
6    Complete    5    4
8    Complete    4    3
9    Complete    3    65535
10   Preempted   65535  4
12   Complete    4    3
13   Complete    3    5
15   Complete    5    4
17   Complete    4    3
18   Complete    3    65535
20   Preempted   65535  3
21   Complete    3    4
23   Complete    4    5
24   Preempted   5    3
25   Complete    3    5
26   Complete    5    4
28   Complete    4    3
29   Complete    3    65535
30   Preempted   65535  4
32   Complete    4    3
33   Complete    3    5
35   Complete    5    4
37   Complete    4    3
38   Complete    3    65535
40   Preempted   65535  3
41   Complete    3    4
43   Complete    4    5
```