说明文档

141250034

范佳杰

目录

[1. 实验环境 2](#_Toc20657)

[2. 实验目的 2](#_Toc27819)

[3. 基本EDF调度实现 2](#_Toc17436)

[3.1思路与方法 2](#_Toc18590)

[3.2调度算法 2](#_Toc13257)

[3.3任务函数 2](#_Toc31905)

[4.测试用例 3](#_Toc26047)

1. 实验环境

|  |  |
| --- | --- |
| RTOS | μCOSII\_VC 移植版 |
| VC平台工具集 | Visual Studio 2015 |

1. 实验目的

本次实验是在 VC 移植版本的 μCOSII 上面实现 EDF 调度。 μCOSII 目前只提供对固定优先级调度的支持，但是对于动态优先级调度没有支持，而 EDF 调度是一种理论上可以实现 100%CPU 利用率的调度算法，本实验就是希望对其有所体会和理解。

1. 基本EDF调度实现

3.1思路与方法

1. 由于 EDF 调度采取的是最小截止时限优先的调度策略，需要在 TCB 当中增加变量来存储其当前任务周期内的已经消费的时钟周期数目（compTime）和当前的截止时限(ddl).
2. 仿照app.c中开始任务的创建，创建用于EDF调度的任务（定义栈空间，声明任务函数并添加函数的具体代码，在main函数中用OSTaskCreate创建任务函数）
3. 修改了OSTaskCreate函数，在其中新增了两个INT32U类型的参数，用来存储任务的周期和每周期的执行时间，关于计算compTime和ddl放在task函数里面。
4. 修改了 os\_cfg.h 当中的 OS\_TICKS\_PER\_SEC 变量，设置为 1，即一秒中 1 个 tick
5. 在OSTimeTick()中增加使compTime减一的语句
6. 在OSIntExit()打印preempt信息，在OS\_Sched()中打印complete信息
7. 修改OS\_SchedNew函数，用于挑选出ddl最小的就绪任务

3.2调度算法

static void OS\_SchedNew(void)

{

INT8U prio = OS\_LOWEST\_PRIO;

INT32U deadline = 1000000;

OS\_TCB\* cur = OSTCBList;

while (cur->OSTCBNext) {

if (cur->OSTCBDly == 0&&cur->ddl!=NULL&&cur->ddl < deadline) {

prio = cur->OSTCBPrio;

deadline = cur->ddl;

//printf("p and d=%d,%d", prio, deadline);

}

cur = cur->OSTCBNext;

}

OSPrioHighRdy = prio;

//printf("prio=%d ", prio);

}

使用遍历任务链表的方法，找到ddl最小的就绪任务，给 OSPrioHighRdy赋值

3.3任务函数

static void task1(void \*p\_arg)

{

INT32U start;

INT32U end;

INT32U toDelay;

INT32U cValue = 1;

INT32U pValue = 4;

start = 0;

while (1) {

while (OSTCBCur->compTime>0)

{

//do nothing

}

;

end = OSTimeGet();

toDelay = pValue - (end - start);

start += pValue;

OSTCBCur->compTime = cValue;

OSTCBCur->ddl += pValue;

OSTimeDly(toDelay);

}

}

4.测试用例

Task set1={ t1(1,3) , t2(3,5)}

测试结果

解释一下输出结果:

（1 Comlete 1 2）表示t=1s时，task1完成，接下来运行task2

（6 Preempt 2 1）表示t=6s时，task2被task1抢占

