相同优先级线程轮转调度

RealTouch 评估板 RT-Thread 入门文档

版本号：1.0.0

日期：2012/8/12

修订记录

日期	作者	修订历史
2012/8/12	bloom5	创建文档

实验目的

- 快速了解相同优先级的线程的调度过程

硬件说明

本实验使用 RT-Thread 官方的 Realtouch 开发板作为实验平台。涉及到的硬件主要为

- 串口 3，作为 rt_kprintf 输出，需要连接 JTAG 扩展板

具体请参见《Realtouch 开发板使用手册》

实验原理及程序结构

实验设计

本实验的主要设计目的是帮助读者快速了解两个或多个相同优先级线程的调度过程。在 RT-Thread 中，对于相同优先级的线程，在调度时采用 Round-Robin 算法，也就是设计成一个分时系统，调度器按照轮询的方式依次调度这些线程，每个线程的执行时间为该线程被创建时指定的时间片参数；

源程序说明

本实验对应 1_kernel_thread_sameprio

系统依赖

在 rtconfig.h 中需要开启

- #define RT_USING_HEAP
  此项可选，开启此项可以创建动态线程和动态信号量，如果使用静态线程和静态信号量，则此项不是必要的
- #define RT_USING_CONSOLE
  此项必须，本实验使用 rt_kpriintf 向串口打印按键信息，因此需要开启此项

主程序说明

在 applications/application.c 中的 thread_same_priority_init() 函数中初始化了两个线程 t1、t2，

```
/* 指向线程控制块的指针 */
result = rt_thread_init(&thread1,
            "t1",
            thread1_entry, RT_NULL,
            &thread1_stack[0], sizeof(thread1_stack),
            6, 5);
    if (result == RT_EOK)
            rt_thread_startup(&thread1);


result = rt_thread_init(&thread2,
            "t2",
            thread2_entry, RT_NULL,
            &thread2_stack[0], sizeof(thread2_stack),
            6, 10);
    if (result == RT_EOK)
            rt_thread_startup(&thread2);
```

两个线程具有相同的优先级，均为 6，两者在时间片上有所不同，t1 为 10, t2 为 5.

下面的代码是两个线程的入口程序，两个线程都只执行打印动作

```
static void thread1_entry(void* parameter)
{
    rt_uint8_t i;
    for(i = 0; i < 6; i ++)
    {
            rt_kprintf("Thread1:\n\r");
            rt_kprintf("This is \n");
            rt_kprintf("a\n");
            rt_kprintf("demo\n");
            rt_thread_delay(10);
    }
```

```
}

static void thread2_entry(void* parameter)
{
    rt_uint8_t j;
     for(j = 0; j <60; j ++)
     {
            rt_kprintf("Thread2:\n\r");
            rt_kprintf("This is \n");
            rt_kprintf("a\n");
            rt_kprintf("demo\n");
     }
}
```

编译调试及观察输出信息

编译请参见《RT-Thread 配置开发环境指南》完成编译烧录，参考《Realtouch 开发板使用手册》完成硬件连接，连接扩展板上的串口和 jlink。

运行后可以看到如下信息：

```
 \ | /
- RT -     Thread Operating System
 / | \     1.1.0 build Aug 11 2012
 2006 - 2012 Copyright by rt-thread team
Thread1:
This is
a
demo
Thread2:
This is
a
demo
Thread2:
This is
a
demo
Thread2:
```

```
This is
a
demo
Thread2:
This is
a
demo
Thread2:
This is
a
demo
Thread2:
This isThread1:
This is
a
demo

a
demo
Thread2:
This is
a
demo
Thread2:
This is
a
demo
Thread2:
This is
a
demo
Thread2:
This is
a
Demo
….
```

结果分析

由两个静态线程的初始化可知，两个线程拥有相同的优先级，只是在时间片上略有不同。然后，我们可以观察打印结果，首先是 thread1 执行，它在打印完一次测试语句后，就执行了延时语句，从而 thread2 得到控制权开始执行，它开始一遍遍打印出整个测试语句，当其打印到第六次时，可以发现打印的语句并不完整，因为 thread1 和 threada2 的优先级相同，并不会发生抢占的情况，所以说 thread2 是等到自己的执行时间片到达时，被系统剥离执行权，而将执行权回复给 thread1，从而 thread1 重新获得执行，由此可以看出当两个相同线程间，运行是以时间片为基准的，时间片到达，则交出控制权，交给下一个就绪的同优先级线程执行。