动态线程创建与删除

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

- 快速熟悉动态线程创建和删除的接口
- 可以使用线程实现简单任务

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线程相关 API，包括动态线程的创建/删除相关 API，为了简化起见，我们将这些 API 放在同一个线程中调用。请读者注意，本实验本身不具有实际的工程参考价值，只是帮助读者快速了解线程 API 的用法。

源程序说明

本实验对应 1_1_thread_dynamic

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
- #define RT_TICK_PER_SECOND 100

此项定义了 1 秒= RTTICKPERSECOND 个 tick = 100 tick，一个 tick 为 10ms

主程序说明

在 applications/application.c 中的 thread_dynamic_init()函数中定义了两个线程 tid1、tid2，

```
/* 指向线程控制块的指针 */
rt_thread_t tid1,tid2;
```

两个线程的创建过程如下

```
tid1 = rt_thread_create("thread1",
            thread1_entry,
            RT_NULL,
            512, 6, 10);
    if (tid1 != RT_NULL)
            rt_thread_startup(tid1);

    tid2 = rt_thread_create("thread2",
            thread2_entry, RT_NULL,
            512, 6, 10);
    if (tid2 != RT_NULL)
            rt_thread_startup(tid2);
```

下面的代码是两个线程的入口程序，在入口程序中 thread2 将会去删除 thread1，也就是调用了线程删除的 API

```
/*线程 1 运行一直在打印计数,由于线程 1 和 2 优先级相同均为 THREAD_PRIORITY，所以将会采用时间片轮转的方式轮换执行*/
static void thread1_entry(void* parameter)
{
    rt_uint32_t count = 0;
    rt_kprintf("thread1 dynamicly created ok\n");
    while(1)
    {
            rt_kprintf("thread1 count: %d\n",count++);
            rt_thread_delay(RT_TICK_PER_SECOND);
    }
}
```

在线程 1 时间片用尽后得到执行，在打印完成后线程 2 delay，于是回到线程 1 执行，在 delay 结束后回到线程 2 执行，线程 2 将删除线程 1，并且在执行结束后也将自行删除

```
static void thread2_entry(void* parameter)
{
    rt_kprintf("thread2 dynamicly created ok\n");

    rt_thread_delay(RT_TICK_PER_SECOND * 4);

    rt_thread_delete(tid1);
    rt_kprintf("thread1 deleted ok\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 10 2012
 2006 - 2012 Copyright by rt-thread team
thread1 dynamicly created ok
thread1 count: 0
thread2 dynamicly created ok
thread1 count: 1
thread1 count: 2
thread1 count: 3
thread1 count: 4
thread1 deleted ok
```

结果分析

从打印信息可以看到首先看到 thread1 和 thread2 先后建立成功，然后 thread2

```
rt_thread_delay(RT_TICK_PER_SECOND * 4);
```

系统调度到 thread1 执行可以看到继续执行计数，因为 thread1 的延时为 1 个 tick，所以可以看到它打印了四次计数，当它第四次延时时，thread2 的延时也到了，系统调度到 thread2 继续运行，它调用了

```
rt_thread_delete(tid1);
```

将 thread1 删除，删除成功后打印信息

```
thread1 deleted ok
```

此时可以看到 thread1 也不再出现计数信息，可以确定 thread1 被成功删除了。