Finsh 的基本使用

RealTouch 评估板 RT-Thread 入门文档

版本号：1.0.0

日期：2012/8/17

修订记录

日期	作者	修订历史
2012/8/17	bloom5	创建文档

实验目的

快速熟悉命令行形式的调试和信息查看组件 Finsh

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

对于用户来讲，Finsh 组件有三个主要功能

- 获取系统运行时信息，如各种 RT-Thread 内核对象的动态信息。
- 能够对任意寄存器和内存地址进行读写操作
- 能够直接在 shell 中调用系统函数，访问系统变量

实验设计

本实验的主要设计目的是帮助读者了解 Finsh 组件的基本用法及内置命令。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 Finsh 组件的用法。

源程序说明

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_kprintf 向串口打印按键信息，因此需要开启此项

- #define RT_USING_FINSH

此项必须，只有打开此项，Finsh 组件才会被使用

❑ #define RT_USING_SYMTAB

此项可选，打开此项可以使用宏输出的方式向 Finsh shell 中添加命令。

主程序说明

✧ Finsh 的配置，在 rt_config.h 中使能 Finsh 组件，默认情况是使能的。

```
/* SECTION: Finsh, a C-Express shell */
#define RT_USING_FINSH
/* Using symbol table */
#define FINSH_USING_SYMTAB
#define FINSH_USING_DESCRIPTION
```

✧ Finsh 的初始化

```
#ifdef RT_USING_FINSH
    /* initialize Finsh */
    Finsh_system_init();
    Finsh_set_device(RT_CONSOLE_DEVICE_NAME);
#endif
```

在 application.c 中，首先会运行一个名为 init 的线程，这个线程的工作就是将此次用到的组件全部初始化，所以这个线程中只用到了一条语句 rt_components_init()，我们可以在 components_init.c 中看到这个函数的具体定义，而关于 Finsh 的初始化，则就是上面列出的这一段，如果还有别的组件需要初始化话，其初始化工作也会在这里进行。注意一下的是 RT_CONSOLE_DEVICE_NAME，这个也是在 rt_config.h 中被定义的，因为我们用到的是 UART3，所以

```
#define RT_CONSOLE_DEVICE_NAME    "uart3"
```

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。

烧录程序以后可以看到如下信息：

```
 \ | /
- RT -     Thread Operating System
 / | \     1.1.0 build Aug 17 2012
 2006 - 2012 Copyright by rt-thread team
Finsh>>
```

首先使用 list()命令打印出当前注册到 Finsh 系统的所有 shell 命令

```
 \ | /
- RT -    Thread Operating System
 / | \    1.1.0 build Aug 17 2012
 2006 - 2012 Copyright by rt-thread team
Finsh>>list()
--Function List:
list_mem        -- list memory usage information
version         -- show RT-Thread version information
list_thread     -- list thread
list_sem        -- list semaphone in system
list_event      -- list event in system
list_mutex      -- list mutex in system
list_mailbox    -- list mail box in system
list_msgqueue   -- list message queue in system
list_mempool    -- list memory pool in system
list_timer      -- list timer in system
list_device     -- list device in system
list            -- list all symbol in system
--Variable List:
dummy           -- dummy variable for Finsh
       0, 0x00000000
Finsh>>
```

按照上面命令的描述，先 list_thread() 下，查看当前系统中运行的所有线程

```
\ | /
- RT -    Thread Operating System
 / | \    1.1.0 build Aug 17 2012
 2006 - 2012 Copyright by rt-thread team
thread1 working...
Finsh>>list_thread()
thread  pri  status     sp    stack size max used   left tick
error
-------- ---- ------- ---------- ---------- ---------- ----------
---
tshell   0x14 ready   0x00000110 0x00000800 0x000001e8 0x00000008
-04
tidle    0x1f ready   0x000000e0 0x00000400 0x000000e0 0x00000010
000
thread1  0x0b suspend 0x00000100 0x00000400 0x00000100 0x00000005
000
        0, 0x00000000
Finsh>>
```

可以看到目前只有 shell、idle 以及 thread1 三个线程。标题栏的 thread 表示线程名称， pri 表示线程优先级，status 表示线程当前调度状态，sp 表示当前线程的堆栈指针位置，stack size 表示线程的堆栈大小，max used 表示线程历史用过的最大堆栈大小（肯定会小于堆栈大小），left tick 表示线程剩余执行节拍数。

再输入 list_device() 查看下当前注册到设备管理器中的设备

```
Finsh>>list_device()
device   type
-------- ----------
uart3    Character Device
       0, 0x00000000
```

可以看到当前只有 uart3 一个设备，它的类型是字符型设备。

```
Finsh>>list_sem()
semaphore v   suspend thread
--------  --- --------------
shrx      000 0
heap      001 0
```

通过 list_sem 可以看到当前使用信号量的状况，semaphore 字段表示信号量的名称，v 字段表示信号量的当前值，suspend thread 字段表示等在这个信号量上的线程数目。

```
event    set       suspend thread
-------- ---------- --------------
```

通过 list_event() 则可以查看当前系统中使用的 event，event 字段表示事件的名称，set 字段表示事件的值，suspend thread 字段表示等在这个事件上的线程数目。

```
Finsh>>list_timer()
timer    periodic   timeout    flag
-------- ---------- ---------- -----------
timer1   0x0000000a 0x00008035 activated
tshell   0x00000000 0x00000000 deactivated
tidle    0x00000000 0x00000000 deactivated
current tick:0x00008031
```

通过 list_timer() 命令则可以查看系统当前的定时器使用情况，timer 字段表示定时器的名称，periodic 字段表示定时器是否是周期性的或者是一次性的，timeout 字段表示定时器超时时的节拍数，flag 字段表示定时器的状态，activated 表示活动的，deactivated 表示不活动的，current tick 表示当前系统的节拍。

除了内置命令的操作外，还可以进行对内存或者寄存器进行操作

```
Finsh>>int a                        //定义一个变量
```

```
       0, 0x00000000
Finsh>>int b                        //定义一个变量
       0, 0x00000000
Finsh>>&a                           //对变量 a 取地址
       'l', 536873580, 0x20000a6c
Finsh>>b=&a                         //将 a 的地址赋值给 b
       'l', 536873580, 0x20000a6c
Finsh>>*b=47                        //对 b 中所存的地址处进
                                    //行赋值
       '/', 47, 0x0000002f
Finsh>>a                            //打印出 a
       '/', 47, 0x0000002f
Finsh>>
```

相信看完这一段代码，大家都不会陌生，相当于一段简单的指针操作，而这几乎与 C 语言用法相同。

结果分析

本例程主要想要就是为大家展示了 Finsh 组件的一些已有内置功能，已经有一些惊艳了吧，其实还有更猛的料在后头，你可以在 Finsh 运行自定义函数和查看添加自定义变量，但是这些都请听下回分解啦。距离下篇文档可能会有时间间隔，可以动手先试起来啦！