TCPIP 网络组件 Lwip 之 telnet 远程命令行交互

RealTouch 评估板 RT-Thread 入门文档

版本号：1.0.0

日期：2012/8/26

修订记录

日期	作者	修订历史
2012/8/26	bloom5	创建文档

实验目的

- 快速了解 Lwip 组件，熟悉使用网络 telnet 的方式使用 finsh，进行内核信息查看等远程命令行交互。
- 帮助读者了解 telnet 服务器端的实现过程以及 telnet 客户端配置和连接方法。

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

- RJ45 接口，作为网络连接的需要，我们需要用网线将 Realtouch 和目标机连接起来，具体请参见《Realtouch 开发板使用手册》
- 串口 3，作为 rt_kprintf 输出，需要连接 JTAG 扩展板

实验原理及程序结构

telnet 协议是 TCP/IP 协议族中的一员，是 Internet 远程登陆服务的标准协议和主要方式。它为用户提供了在本地计算机上完成远程主机工作的能力。在这里，我们利用 telnet 的方式访问 finsh，使得 finsh 操作可通过网络方式进行，这是 RT-Thread 又一个强大而实用的功能。

实验设计

本实验在 RealTouch 端实现并运行了 telnet 服务器端程序，在 PC 上使用 telnet 客户机端程序来访问 telnet 服务器，从而进行 telnet 通信。

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_LWIP

此项必须，本实验使用 LWIP 组件，因此需要开启此项

- #define RT_USING_CONSOLE

此项必须，在系统初始化过程中仍需通过串口进行显示相关的工作

主程序说明

关于 LwIP 在 rtconfig.h 中相关宏的开启、IP 相关设置可参见上一节。在 application.c 中， 建立了一个初始化线程，在其中初始化了 lwip 线程栈，注册以太网设备，并初始化 lwip 系统，这些都包含在了 rt_component_init()中。接着启动 telnet server，具体的实现在在 telnet.c 文件中。

```
void rt_init_thread_entry(void* parameter)
{
#ifdef RT_USING_LWIP
  /* initialize eth interface */
  rt_hw_stm32_eth_init();
#endif

#ifdef RT_USING_COMPONENTS_INIT
    /* initialization RT-Thread Components */
  rt_components_init();
#endif

  rt_platform_init();
    /* do some thing here. */
  telnet_srv();
}

int rt_application_init()
{
    rt_thread_t init_thread;

    init_thread = rt_thread_create("init",
                                rt_init_thread_entry, RT_NULL,
                                2048, 8, 20);


    if (init_thread != RT_NULL)
        rt_thread_startup(init_thread);

    return 0;
}
```

下面是 telnet 服务器线程的创建

```
void telnet_srv()
{
```

```
    rt_thread_t tid;

    if (telnet == RT_NULL)
    {
        rt_uint8_t *ptr;

        telnet = rt_malloc (sizeof(struct telnet_session));
        if (telnet == RT_NULL)
        {
            rt_kprintf("telnet: no memory\n");
            return;
        }

        /* init ringbuffer */
        ptr = rt_malloc (TELNET_RX_BUFFER);
        rb_init(&telnet->rx_ringbuffer, ptr, TELNET_RX_BUFFER);
        /* create rx ringbuffer lock */
        telnet->rx_ringbuffer_lock = rt_sem_create("rxrb", 1,
RT_IPC_FLAG_FIFO);
        ptr = rt_malloc (TELNET_TX_BUFFER);
        rb_init(&telnet->tx_ringbuffer, ptr, TELNET_TX_BUFFER);
        /* create tx ringbuffer lock */
        telnet->tx_ringbuffer_lock = rt_sem_create("txrb", 1,
RT_IPC_FLAG_FIFO);

        /* create network event */
        telnet->nw_event = rt_event_create("telnet",
RT_IPC_FLAG_FIFO);
    }

    tid = rt_thread_create("telnet", telnet_thread, RT_NULL,
                       2048, 25, 5);
    if (tid != RT_NULL)
        rt_thread_startup(tid);
 }
```

接下来，再来看看 finsh 是如何设置成 telnet 形式实现的，具体内容在 telnet_thread()函数中，这也是 telnet 线程的 entry。

```
 /* Process the new connection. */
 /* set console */
 rt_console_set_device("telnet");
 /* set finsh device */
 finsh_set_device("telnet");
```

可以看到，在 telnet 线程中对 finsh 使用的设备做了重定向，从原先默认的串口设备重定向到了 telnet 设备。

到目前为止，需要注意的代码都已经提及，下面就可以编译下载运行了。

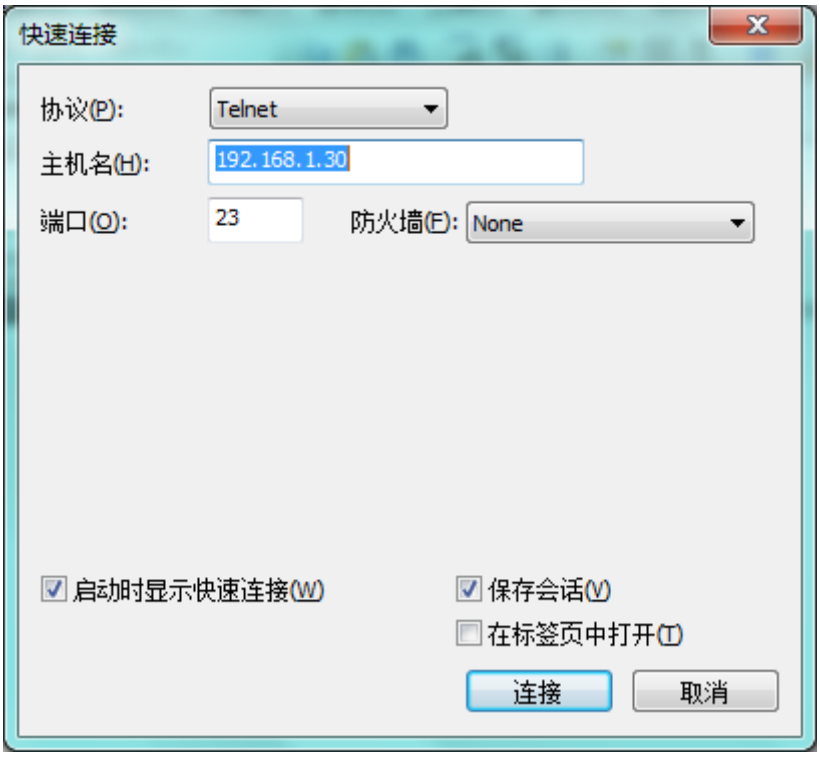
编译调试及观察输出信息

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，连接好串口线，连上网线。

运行后可以看到串口打印如下的信息：

```
 \ | /
- RT -     Thread Operating System
 / | \     1.1.0 build Aug 26 2012
 2006 - 2012 Copyright by rt-thread team
TCP/IP initialized!
finsh>>telnet server waiting for connection
```

此时你需要 telnet 的 server 端已经打开了，正在监听 23 号端口，等待客户端的连接。所以你现在要做的就是打开你的 telnet 客户端，将其连接到主机名为 192.168.1.30(即 realtouch 所设的 IP)，端口为 23 的 telnet server(笔者微软自带的 telnet 客户端此时可以连接成功，但之后无法实现 finsh 功能，所以转而使用了 securecrt 的 telnet 连接)。设置如下：



输入确认后，就可以连上 RealTouch 上的 server 端了。此时串口也会发出最后一条消息：

```
new telnet connection, switch console to telnet...
```

接下来就可以像使用串口线时一样，使用网络方式通过 telnet 客户端进行 finsh 命令行交互了，使起那心爱的 list()。

```
  list()

  --Function List:
  telnet_srv      -- startup telnet server
  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
  set_if          -- set network interface address
  set_dns         -- set DNS server address
  list_if         -- list network interface information
  list_tcps       -- list all of tcp connections
  --Variable List:
  dummy           -- dummy variable for finsh
         0, 0x00000000
  finsh>>list_thread()

   thread  pri  status      sp     stack size max used   left tick
error
  -------- ---- ------- ---------- ---------- ---------- ----------
---
  telnet   0x19 ready   0x00000134 0x00000800 0x000001cc 0x00000004
000
  tcpip    0x0c suspend 0x00000160 0x00000400 0x00000270 0x00000011
000
  etx      0x0f suspend 0x0000011c 0x00000200 0x0000011c 0x00000010
000
  erx      0x0f suspend 0x0000011c 0x00000200 0x0000014c 0x00000010
000
  tshell   0x14 ready   0x00000110 0x00000800 0x000001b0 0x0000000a
000
```

```
  tidle    0x1f ready   0x000000e0 0x00000400 0x000000e0 0x00000009
000
         0, 0x00000000
```

结果分析

通过以上内容的实际操作，就可以实现用 telnet 网络的方式使用 finsh 了。