应 用 笔 记

嵌入式 Linux 下双网口 Socket 编程

Rev1.0
2022-07-06

版本记录

版本号	修改说明	修改人	修改日期
V1.0	初始文档	nmy@weathink.com	2022-07-06

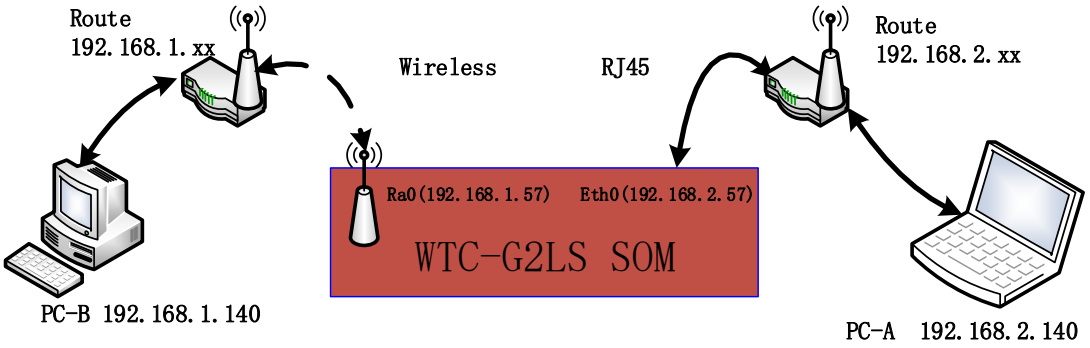
嵌入式Linux下双网口Socket编程

硬件连接：
首先这里有两个网卡，一个是有线网卡 eth0，另外一个是无线网卡 ra0，然后确保有线和无线都已经连接到两个路由器上

```
[root@YuGe-AM1808 /test/rt5370]#ifconfig
eth0      Link encap:Ethernet  HWaddr 00:18:31:E6:3C:15
          inet addr:192.168.2.57  Bcast:192.168.2.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
          Interrupt:40
ra0       Link encap:Ethernet  HWaddr 00:0C:43:53:70:00
          inet addr:192.168.1.57  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:139191 (135.9 KiB)  TX bytes:2038 (1.9 KiB)
```

并且配置成各自网段的 IP 地址
从上面可以看出来双网卡的配置：
开发板 eth0 连接到路由器 A（网段是 192.168.2.xxx），自身 IP 为 192.168.2.57
开发板 ra0 连接到路由器 B（网段是 192.168.1.xxx），自身 IP 为 192.168.1.57
然后需要 2 台 PC：
PC-A 连接到路由器 A（网段是 192.168.2.xxx），自身 IP 为 192.168.2.140
PC-B 连接到路由器 B（网段是 192.168.1.xxx），自身 IP 为 192.168.1.140
此时确保各个系统之间网络是连通的：
路由器和 PC 之间能够通过 ping 命令测试能够连通
路由器和开发板能够通过 ping -I eth0 192.168.2.1 或者 ping -I ra0 192.168.1.1 连通

具体网络拓扑图如下：



Socket 编程

默认的 Socket 是不指定网卡发送接收数据的，因此如果系统中存在双网卡的时候，Socket 会根据路由表选择需要从哪个网卡通讯，但是这种方式不是很好弄，因此我们选择另外一种方式，在 Socket 程序中指定网卡通讯，如下面程序所示，下面的程序是一个 tcp client，指定从 eth0 这个网卡通讯：

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/socket.h>
#include <errno.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <string.h>
#include<errno.h>
#include<netpacket/packet.h>
#include<net/if.h>
#include<net/if_arp.h>
#include<netinet/in.h>
#include<netinet/ip.h>
#include<linux/if_ether.h>
#include<arpa/inet.h>
#include<sys/ioctl.h>

// tcp 客户端连接 ok 标志
int tcp_client_connect_ok_flag = 0;
int send_data_time_cnt = 0;
#define M_DELAY_TIME_MS    300

int main(int argc, char** argv)
{
    int sockfd, recvbytes,res,flags,error,n;
```

```
        socklen_t len;
        fd_set rset,wset;
        struct timeval tval;
        tval.tv_sec = 0;
        tval.tv_usec = 0;
        struct sockaddr_in serv_addr;
        // 发送字符串
        char* sendData = "1234567890";
         // 接收 buffer
        char buf[1024] = "\0";

        printf("eth0-client:Version 1.003\n");
        // 创建 socket 描述符
        sockfd = socket(AF_INET, SOCK_STREAM, 0);
        if( sockfd == -1)
        {
            perror("eth0-client:socket create failed");
            return 1;
        }
        else
        {
            printf("eth0-client:creat sockfd = %d\n",sockfd);
        }
        struct ifreq interface;
        strncpy(interface.ifr_ifrn.ifrn_name, "eth0", IFNAMSIZ);
        if (setsockopt(sockfd, SOL_SOCKET, SO_BINDTODEVICE, (char *)&interface,
sizeof(interface)) < 0)
        {
                perror("eth0-client:SO_BINDTODEVICE failed");
                /* Deal with error... */
            return -1;
        }

        serv_addr.sin_family=AF_INET;
        // 服务器 tcp 端口和 IP 地址设定
        serv_addr.sin_port=htons(5000);
        serv_addr.sin_addr.s_addr = inet_addr("192.168.2.140");
        bzero(&(serv_addr.sin_zero),8);
        printf("eth0-client:connect to ip=192.168.2.140 port=5000\n");

        // 设置为非阻塞,首先获取 flag,设定 noblock,然后设定 flag
        flags = fcntl(sockfd,F_GETFL,0);
        fcntl(sockfd,F_SETFL,flags|O_NONBLOCK);
```

```
        if ( (res = connect(sockfd, (struct sockaddr *)&serv_addr, sizeof(struct sockaddr)) )<
0)
        {
            // 如果返回则表示错误,直接返回
            if(errno != EINPROGRESS)
            {
                printf("eth0-client:connect error\n");
                return 1;
            }
        }

        //如果 server 与 client 在同一主机上，有些环境 socket 设为非阻塞会返回 0
        if(0 == res)
        {
            tcp_client_connect_ok_flag = 1;
            //goto done;
        }

        while(1)
        {
            usleep( M_DELAY_TIME_MS   *1000);
            if(tcp_client_connect_ok_flag == 0)
            {
                FD_ZERO(&rset);
                FD_SET(sockfd,&rset);
                wset = rset;
                if( ( res = select(sockfd+1, NULL, &wset, NULL,&tval) ) <= 0)
                {
                    printf("eth0-client:connect time out\n");
                }
                else
                {
                    len = sizeof(error);
                    getsockopt(sockfd, SOL_SOCKET, SO_ERROR, &error, &len);
                    if (error)
                    {
                        fprintf(stderr, "eth0-client:Error in connection() %d - %s\n", error,
strerror(error));
                        //return 1;
                    }
                    else
                    {
```

```
                        tcp_client_connect_ok_flag = 1;
                        printf("eth0-client:connect ok\n");
                    }
                }
            }
            else if(tcp_client_connect_ok_flag == 1)
            {
                send_data_time_cnt +=   M_DELAY_TIME_MS ;
                if(send_data_time_cnt   >= 1000)
                {
                    send_data_time_cnt   = 0;
                    if ( (n = send(sockfd, sendData, strlen(sendData),0) ) ==-1 )
                    {
                        perror("eth0-client:send error!");
                    }
                }

                FD_ZERO(&rset);
                FD_SET(sockfd,&rset);
                wset = rset;
                if( ( n = select(sockfd+1,&rset,NULL, NULL,&tval)) <= 0 )//rset 没有使用过，
不用重新置为 sockfd
                {
                }
                else if(n > 0)
                {
                    if ((recvbytes=recv(sockfd, buf, 1024, 0)) ==-1)
                    {
                        perror("eth0-client:recv error!");
                        close(sockfd);
                        return 1;
                    }
                    buf[recvbytes] = 0x00;
                    printf("eth0-client:receive num %d\n",recvbytes);
                    printf("%s\n",buf);
                }
            }
        }// while(1)
        close(sockfd);
    }
```

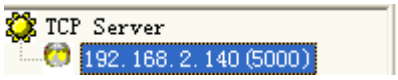
关键代码在于

```
        struct ifreq interface;
```

```
        strncpy(interface.ifr_ifrn.ifrn_name, "eth0", IFNAMSIZ);
        if (setsockopt(sockfd, SOL_SOCKET, SO_BINDTODEVICE, (char *)&interface,
sizeof(interface)) < 0)
        {
                perror("eth0-client:SO_BINDTODEVICE failed");
                /* Deal with error... */
            return -1;
        }
```

这里的意思是这个 Socket 指定到网卡 eth0。

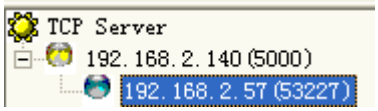
网络测试：
Eth0 上 tcp client 方式
在 PC-B 上创建一个 tcp 服务器，监听 5000 端口，使用 LSD 的网络调试工具，如下



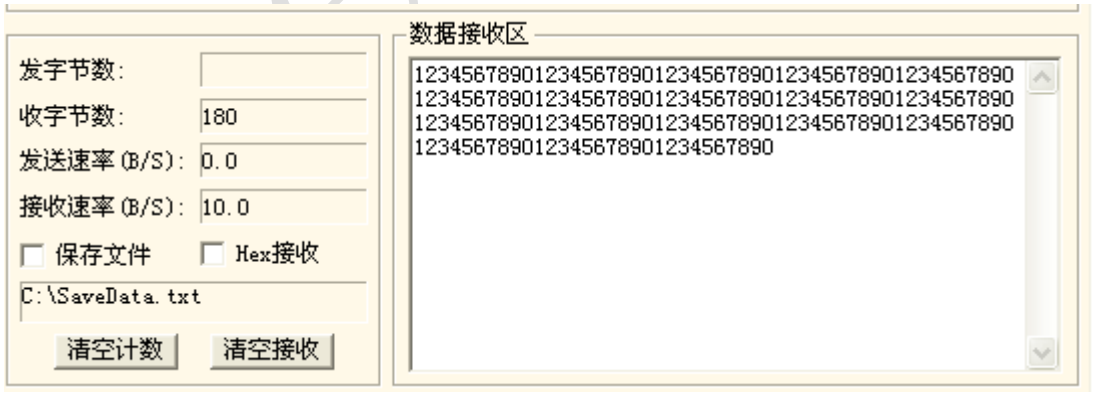
然后在串口终端执行

```
[root@YuGe-AM1808 /]#./tcp-client-noblock-eth0
eth0-client:Version 1.003
eth0-client:creat sockfd = 3
eth0-client:connect to ip=192.168.2.140 port=5000
eth0-client:connect ok
```

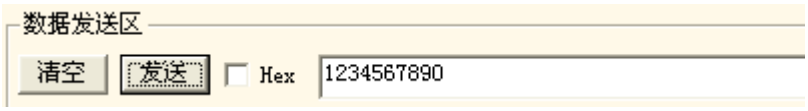
从上面可以看出已经连接上



上图是 PC-B 上显示收到连接



连接建立之后，PC-B 会不断收到数据，另外通过网络调试工具发送数据到 eth0



串口终端显示已经收到数据了

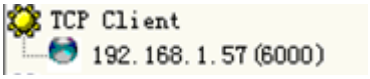
```
eth0-client:receive num 10
1234567890
```

这里通讯 OK 了。

ra0 上 tcp server 方式
首先在串口终端上执行如下命令，监听端口 6000

```
[root@YuGe-AM1808 /]#./tcp-server-noblock-ra0 &
[root@YuGe-AM1808 /]#ra0-server:creat socket ok
ra0-server:SO_BINDTODEVICE ok
ra0-server:setsocslistenJ  kopt ok
ra0-server:bind port=6000 ok
ra0-server:listen ok
ra0-server:listen time out
ra0-server:listen time out
```

这里看到已经在监听了。
然后在 PC-A 上创建一个 tcp client，连接到 192.168.1.57，端口为 6000

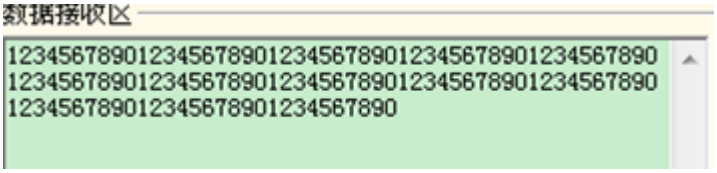


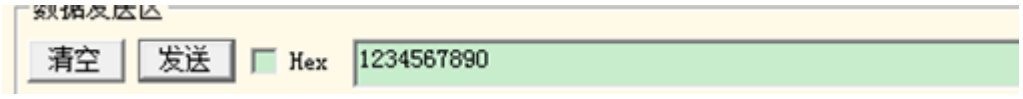
然后单击连接按钮，连接成功后显示如下

此时串口终端显示如下信息

```
ra0-server:listen time out
ra0-server:accept connection ok
ra0-server:Yout got a connection from 192.168.1.140.
```

这里表示已经连接成功了，此时查看 PC-A 的 tcp client，可以看到不断的会收到数据，然后也可以手动发送数据给 ra0。





```
ra0-server:receive num 10
1234567890
```

到这里就表示网络通讯成功了。

注意：基于篇幅问题，没有写四个测试程序的现象，实际我这边测试是按照 eth0 的 client、server，ra0 的 client、server 四种程序做的，同时测试没有出现丢包现象。

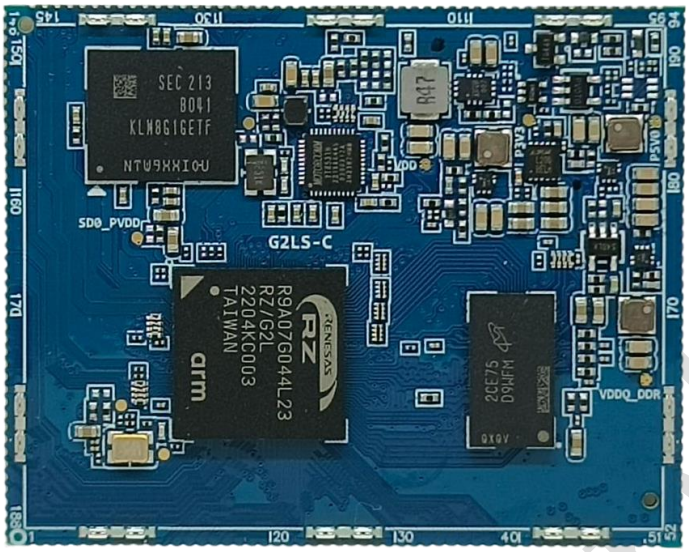
假设 FT3232 的驱动在目录 Test\测试所需软件\FDDI 芯片 USB 驱动\FTDI 下。

```
static void __init am335x_evm_i2c_init(void)
{
    /* Initially assume Low Cost EVM Config */
    am335x_evm_id = LOW_COST_EVM;

    //evm_init_cpld();

    //omap_register_i2c_bus(1, 100, am335x_i2c_boardinfo,
    //            ARRAY_SIZE(am335x_i2c_boardinfo));
    omap_register_i2c_bus(3, 100, am335x_i2c_boardinfo,
                ARRAY_SIZE(am335x_i2c_boardinfo));
}
```

WTC-G2LS 核心板是杭州维芯科电子有限公司推出的一款采用日本瑞萨 Renesas G2L 系列为核心的嵌入式核心板。该系列器件基于 ARM Cortex-A55 内核，具有高性能、低功耗、多接口、低成本等特性，同时提供 3D 图形加速和关键外设的集成，可满足各种应用需要，支持 主流 DDR4 内存，同时提供双路千兆以太网，多路串口来满足工业产品的需求。



销售联系：nmy@weathink.com　18072728558

https://www.weathink.cn/products/hexinban/4.html