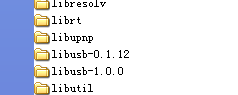
硬件：3g usb模块（华为E353s网卡）+一张sim卡联通WCDMA)，R603W

系统：Linux2.6.21

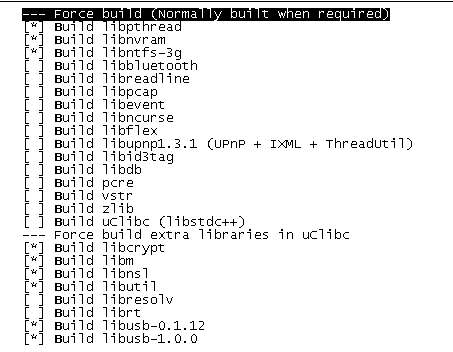
1. 准备工作，内核配置，工具编译
2. 工具libusb-1.0.18 交叉编译需要的库文件
3. Usb-modeswitch-2.2.0 3G网卡模式转换工具
4. ppp-2.4.4 用于拨号的pppd，和发送拨号命令的chat程序

具备以上三个工具，就可以开始网卡驱动的移植了，所幸，这三个工具在R603W中都是自带的，只要打开相应配置选项就可以了，但是有几点是需要注意的，不然会编译不过，或者编译不出。



在目录\source\lib下可以找到，有两个版本的libusb，这两个版本内容是有区别的，其中很多函数名称都是不一样的，在usb-modeswitch-2.0版本以上用的是libusb1.0不需要修改，如果用libusb0.1版本就需要修改其源代码了，我们用户态用的是usb-modeswitch-0.9.5，所以我们选libusb-0.1.12库文件。

打开相应的配置选项

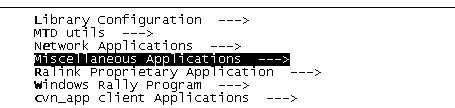


这里可以看到，这里同时选上了两个版本的，这是他自动选上的，就算现在取消了，退出再进来，他还是会自己选上，并且会使编译不过。

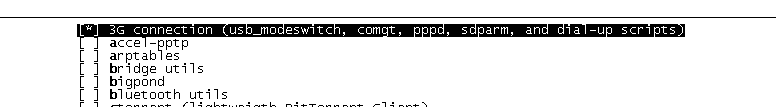
解决方法：

进入RT288x\_SDK\source\lib\lib找到libusb.a文件，默认这里用的是1.0版本的，自己手动将0.1版本的复制到这里替换掉就可以了。上面的选项两个都不要选，为什么后面会讲到。顺便一提，每次推出之前Build uClibc选项一定要取消，它也会自己选上取消编译就可以通过了。

现在打开usb-modeswtich，在



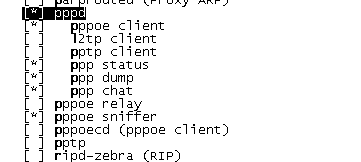
打开



这里只要打开3G就行了，下面的usb-modeswitch就会自己选上，包括前面的libusb也一样都会自己选上，所以就算取消也会自己选上。唯一要做的就是取消掉Build uClibc选项



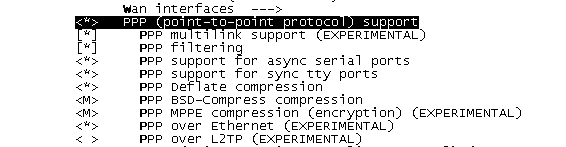
接下来是pppd拨号工具配置



好了，以上都是用户态的一些配置，配置好了之后我们再来配置内核相关选项

打开usb串口相关配置：  
[\*] USB support  --->   
    <\*>   USB Serial Converter support  --->  
        [\*]   USB Serial Console device support  
        [\*]   USB Generic Serial Driver  
        <\*>   USB driver for GSM and CDMA modems

打开ppp相关配置：  
Device Drivers  --->  
    [\*] Network device support  --->



这几个配置比较重要，之前忘记打开

 <\*>     PPP support for async serial ports  
 <\*>     PPP support for sync tty ports

从而导致拨号一直没有成功。

总结：好了，到这里，配置选项都已经配置好了，看似简单的几个步骤。但是在我不了解他们的结构的时候，去配置还是吃了不少苦头的，网上的一些案例也是五花八门，比较不靠谱，环境也不一样。再加上我们内核代码中本身需要做一些修改，所以前段时间就算配置成功了也得不到成功的效果。期间还自己网上去下载了libusb和usb-modeswitch的最新版本来编译，编译过程中也吃了不少苦头，编译环境的设置以及交叉编译时库文件找不到等问题，最后在查看Makefile文件.sh脚本的时候找到了解决并编译完成。其实说白了就是Makefile中库文件的路径还有一个看似不起眼的关键参数的问题（-lusb-1.0和-lpthread后面这个是查找多线程库文件的关键词，提供多线程入口的，前面那个我至今还不知道，网上也查不到）。虽然编译成功了，但是受其他原因也没有模式转换成功，其实这些都是无用功，主要还是不了解内核对USB设备驱动过程导致的。

1. 修改内核代码，添加相关设备

这次的移植花的最多的时间的还是第一步，而导致一直卡在第一步的原因就是这个，内核对USB设备的识别。

由于我们的内核版本比较老，3G网卡的型号比较新，内核代码不能识别这些设备，需要自己添加。添加代码很简单，但是内核对USB设备的处理过程还是非常复杂的，幸亏在网上看到了一篇这部分的文章，才解决了这个问题。先来介绍怎么修改：

linux-2.6.21.x\drivers\usb\serial\generic.c

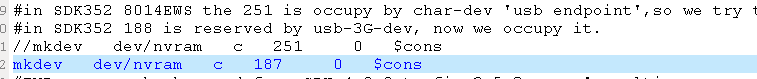


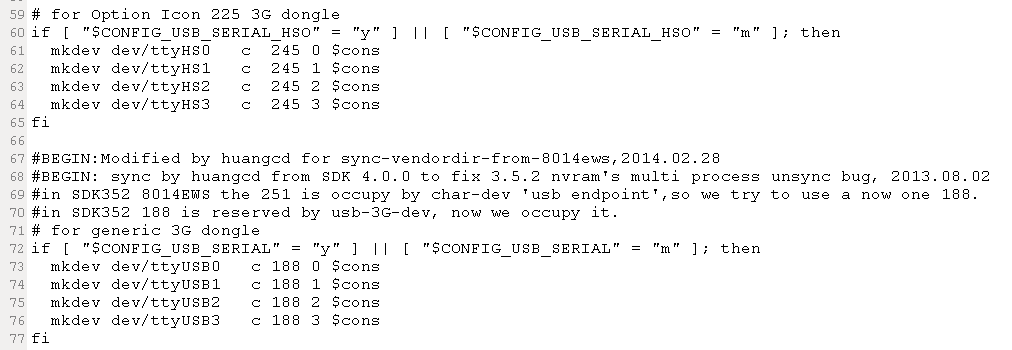
这里修改成3G网卡转化成功后的产品号。可以在USB-MODESWITCH的配置文件中看到。

在linux-2.6.21.x\arch\mips\rt2880\nvram.c中修改如下代码



这个是产品R603W之前做的修改，导致内核设备号188被占用了，而我们3G网卡也是需要用到这个设备号的，因此导致3G网卡插上之后设备一直创建不出来。修改成187就好了，这里做了修改。那我们的链接文件也需要做相应修改，都是一一对应的：RT288x\_SDK\source\vendors\Ralink\R603W\makedevlinks





最后，添加我们的设备

linux-2.6.21.x\drivers\usb\serial\option.c

添加内容：这次移植了两块3G网卡华为E353s和中兴MF667这里拿华为举例，操作都是一样的  
...  
#define HUAWEI\_VENDOR\_ID                0x12D1  
...  
#define HUAWEI\_PRODUCT\_E353s           0x14fe  
...  
static struct usb\_device\_id option\_ids[] = {  
...  
{ USB\_DEVICE(HUAWEI\_VENDOR\_ID, HUAWEI\_PRODUCT\_ E353s) },  
...  
};  
...  
static struct usb\_device\_id option\_ids1[] = {  
...  
{ USB\_DEVICE(HUAWEI\_VENDOR\_ID, HUAWEI\_PRODUCT\_ E353s) },  
...  
};

OK，完成到这里，就可以成功转化，创建设备号了，至于其中内核是怎么处理的，比较复杂，后面我简单总结一下。下载后，加载模块

#insmod usbserial.ko  
#insmod option.ko

#usb-modeswitch –c /etc/usb-modeswitch.conf

这里usb-modeswitch.conf是自己写的一个文件，里面存放的转化信息，网上都有下载的需要在文件中加入MessageEndpoint=0x01这个很重要，指定信息发送出口，否则会失败。中兴的网卡稍有不同，它的MessageContent有三条，具体用哪一条成功需要自己去试。MessageEndpoint是我在虚拟机上调试的时候看到的。

1. 添加拨号脚本，实现拨号

前面已经打开相关选项，版本中应该已经有PPPD拨号工具了，现在只需要写两个拨号脚本就好了。具体有wcdma，wcdma-chat-connect，wcdma-chat-disconnect

[root@localhost peers]# cat wcdma

nodetach

lock

/dev/ttyUSB0

115200

user "card"

password "card"

crtscts

show-password

usepeerdns

noauth

noipdefault

novj

novjccomp

noccp

defaultroute

ipcp-accept-local

ipcp-accept-remote

connect '/bin/chat -s -v -f /etc/ wcdma-chat-connect'

disconnect '/ bin/chat -s -v -f /etc/ wcdma-chat-disconnect'

[root@localhost peers]# cat wcdma-chat-connect

ABORT 'NO CARRIER'

ABORT 'ERROR'

ABORT 'NO DIALTONE'

ABORT 'BUSY'

ABORT 'NO ANSWER'

'' \rAT

OK \rATZ

OK \rAT+CGDCONT=1,"IP","3gnet",,0,0

OK-AT-OK ATDT\*99#

CONNECT \d\c

[root@localhost peers]# cat wcdma-chat-disconnect

ABORT "ERROR"

ABORT "NO DIALTONE"

SAY "\nSending break to the modem\n"

'' "\K"

'' "+++ATH"

SAY "\nGoodbay\n"

从脚本中看到，有个目录，那个可以自己修改，你的chat脚本放在哪，就做相应的修改。编译，下载到板子上

Mkdir /var/lock（这个是pppd脚本中需要的，手工创建）

pppd 1 file /etc/wcdma –d (命令就是这样，自己做对应修改)

到这里就完成了，正常应该可以上网了，看附带的文本，其中记录各部分成功之后的效果

总结：总的来说，这次移植主要还是因为知识面不够广，经验不足。总结一下[USB-SERIAL 驱动](http://blog.csdn.net/istone107/article/details/8123482)。

在 linux 内核中，有一个叫 usbserial 的模块可用作 usb modem 的驱动并能良好的工作, 所以详细  
的分析下 usbserial 模块的源码(2.6.31)。对于那些仅仅是用 USB 来通信, 在上层可看作 tty 设备, 不属  
于任何 USB 设备类型, 没有什么流控等的普通 USB 设备来说都可以使用这个 generic 驱动来作为设  
备驱动程序。  
该驱动中，默认的 idVendor 和 idProduct 如下：  
static \_\_u16 vendor  = 0x05f9;  
static \_\_u16 product = 0xffff;  
为了使该驱动支持自己的设备，可在内核代码中更改该值，亦可通过模块参数来改变 id。  
sudo modeprobe usbserial vendor=0x12d1 product=0x1001  
首先, 当然是要知道 usbserial 模块由哪些文件编译而成，这样才能有目的性的去分析其代码，  
而要知道其组成当然是去其目录下看 Makefile，它位于内核源码目录下的./drivers/usb/serial/下  
obj-$(CONFIG\_USB\_SERIAL) += usbserial.o  
usbserial-obj-$(CONFIG\_USB\_SERIAL\_CONSOLE)  += console.o  
usbserial-obj-$(CONFIG\_USB\_EZUSB)        += ezusb.o  
usbserial-objs    := usb-serial.o generic.o bus.o $(usbserial-obj-y)  
obj-$(CONFIG\_USB\_SERIAL\_AIRCABLE)        += aircable.o  
obj-$(CONFIG\_USB\_SERIAL\_ARK3116) += ark3116.o  
obj-$(CONFIG\_USB\_SERIAL\_BELKIN) += belkin\_sa.o  
obj-$(CONFIG\_USB\_SERIAL\_CH341) += ch341.o  
……  
内核配置中有：  
CONFIG\_USB\_SERIAL=y  
CONFIG\_USB\_SERIAL\_GENERIC=y  
CONFIG\_USB\_SERIAL\_OPTION=y (linux 2.6中usb modem 驱 动 程 序 ， 因 为 usb-serial generic驱动程序读写速度不能满足 3G Modem 的要求)  
重点的几个文件有：usb-serial.c、generic.c、usb.c 和 option.c  
该模块的原理及整体结构：  
跟应用层交互的是一个 tty 设备，也就是说该模块把 USB 设备映射成一个 tty 设备(即在/dev/目录下为该 USB 设备创建一个tty 设备文件)，然后用于可以用 minicom 之类的串口工具来打开。

对于发送过程：tty 设备文件在获取了用户要求发送的数据之后传递到下层 usbserial 模块的核心层，而该核心层就是将数据打包成 USB 格式的数据并由 USB 通信发送到设备端去    。

对于接收过程：usbserial 模块会在该设备打开时就启动一个 urb 在那等待设备端发数据过来，收到数据后就 push 到上层 tty 设备的缓冲中去，而 tty 设备在收到数据后就会给用户，或直接显示在 minicom 之类的工具上。

详细参考：<http://blog.csdn.net/istone107/article/details/8123482。或者文档usbmodeswtich之后usbserial驱动加载.docx>

DefaultVendor= 0x19d2

DefaultProduct=0x2000

TargetVendor= 0x19d2

TargetProductList="0001,0002,0015,0016,0017,0031,0037,0052,0055,0063,0064,0066,0091,0108,0117,0128,0157,1402,2002,2003,2004"

MessageEndpoint=0x01

MessageContent="5553424312345679000000000000061b000000020000000000000000000000"

MessageContent2="5553424312345678000000000000061e000000000000000000000000000000"

MessageContent3="55534243123456702000000080000c85010101180101010101000000000000"

NeedResponse=1