Linux USB wifi使用指南

**作者：向仔州**

**内核配置指定型号的USB网卡………………………………………………………………2**

**iw ，wpa\_supplicant，dhcp，这三个命令工具的编译………………………..4**

**iwconfig，iwlist工具移植……………………………………………………………………….8**

**lsusb命令移植…………………………………………………………………………………………9**

**移植libusb-compat-0.1-0.1.6…………………………………………………………………..10**

**usbutils-0.80移植…………………………………………………………………………………….11**

**USB WIFI AP模式……………………………………………………………………………………..11**

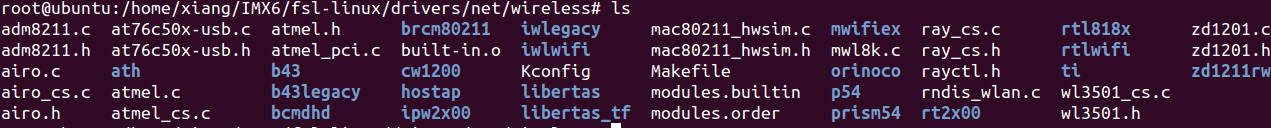
**编译openssl库………………………………………………………………………………………..16**

**内核配置指定型号的USB网卡**

根据购买的USB wifi 网卡的VID/PID号来确定你去内核里面找哪一个代码来编译USB网卡驱动

比如我们使用的是Atheros公司的AR9271 USB wifi模块。

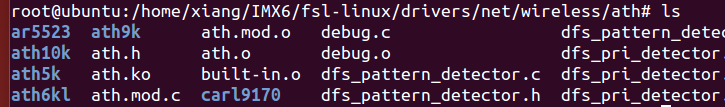
那么我们查找该USB接口wifi模块型号的VID=0x13d3 PID=0x3372

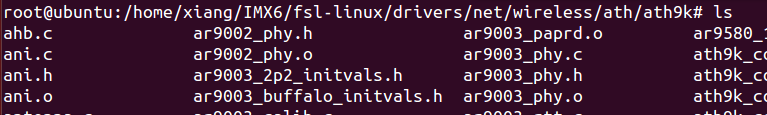


一般各个厂家的wifi驱动代码都是在内核的/drivers/net/wireless目录下

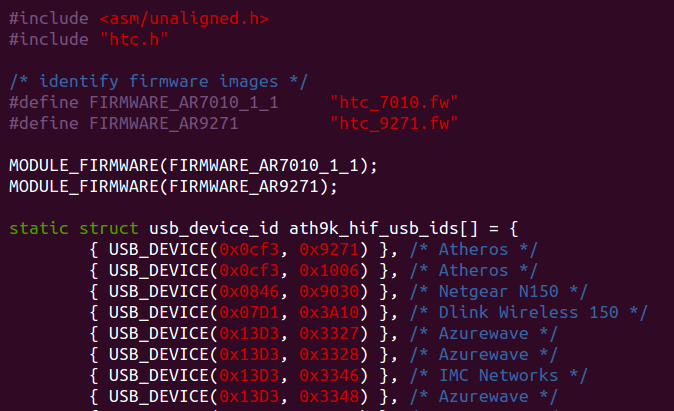
Atheros公司的wifi核心都是以ath目录开头。所以我们进入ath目录

我们AR9271 USB wifi内核为ath9K系列



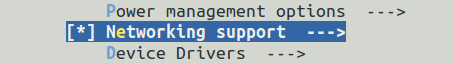


我们查找hif\_usb.c目录里面有没有支持AR9271 USB wifi的VID/PID

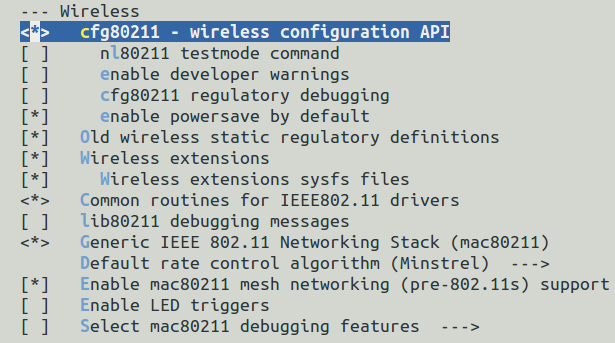


发现内核里面支持AR9271 USBwifi网卡的VID/PID，那么我们就不需要去官网找源代码编译和insmod了，直接配置make menuconfig 内核，然后make就可以了

配置网络协议







配置上这项 \*

配置上这项 \*

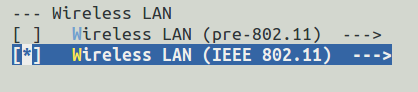
配置wifi芯片驱动







这里面有很多wifi厂家的芯片驱动，看有没有你用的型号





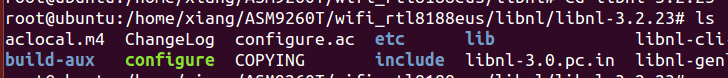
看这么多型号，如果这里没有，你就要找厂家要wifi芯片的驱动，然后自己编译成.ko直接板子上insmod，或者编译成.o放进内核

还有就是要找厂家拿固件，一般厂家都会给你内核编译wifi的文档

**iw ，wpa\_supplicant，dhcp，这三个命令工具的编译**

先下载libnl库

我用的是libnl-3.2.23的库，在linux下解压



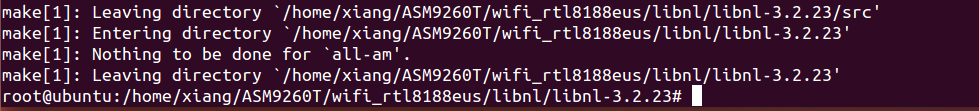
我们看到有configure文件

这种情况万能方法是./configure --host=arm-linux --prefix=$PWD/tmp

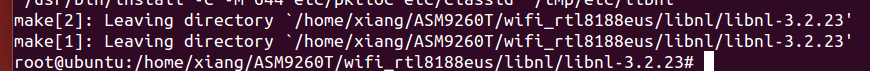


--prefix = $(PWD)/tmp 就是在make install的时候把编译后的文件放在当前目录tmp目录下

然后make



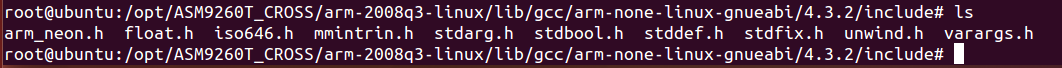
再执行make install



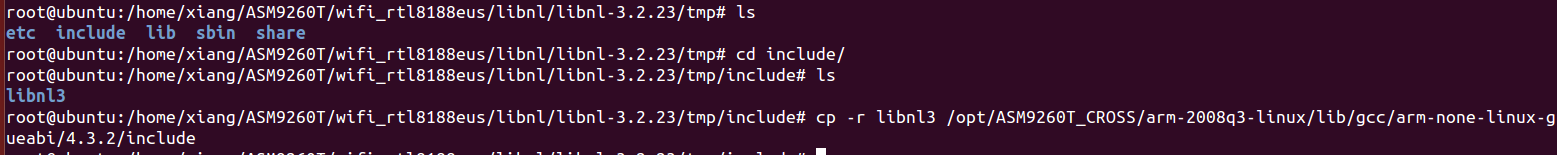


进入tmp目录就可以看到我们编译出来的include，lib，sbin…..等待目录文件了

第1步：把编译出来的头文件放入你开发板平台使用的交叉编译器的include目录



我用的是ASM9260T ARM9芯片，所以放在这个交叉编译器目录下



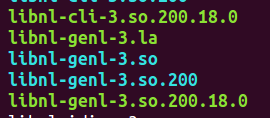
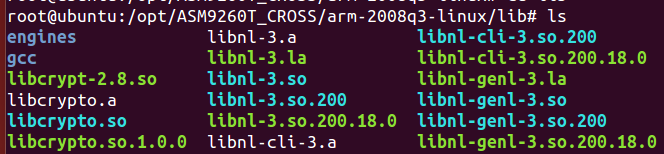


这就放进来了

第2步：把编译出来的lib库文件放入你开发板平台使用的交叉编译器的libc/lib目录



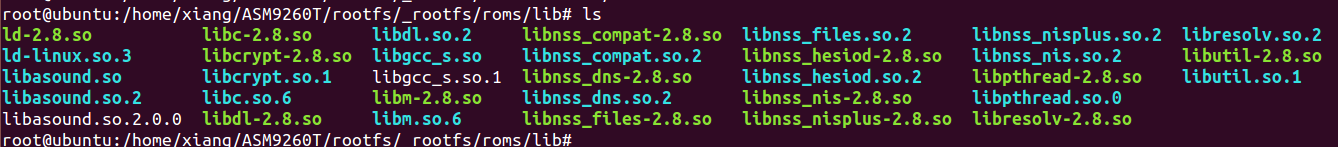
本来交叉编译器的arm-2008q3-linux/arm-none-linux-gnueabi/libc/lib里面是没有的

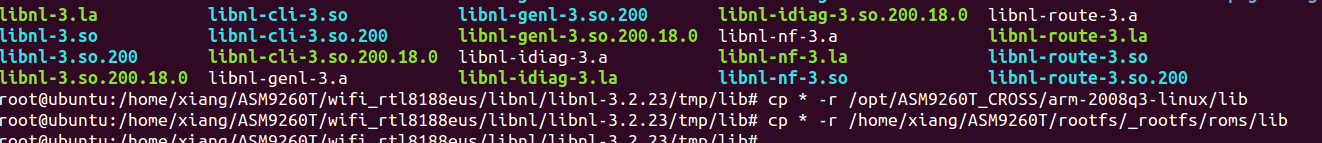
现在交叉编译器的lib里面有了 顺便在拷贝一份libnl到arm-2008q3-linux目录下的顶层lib目录里面

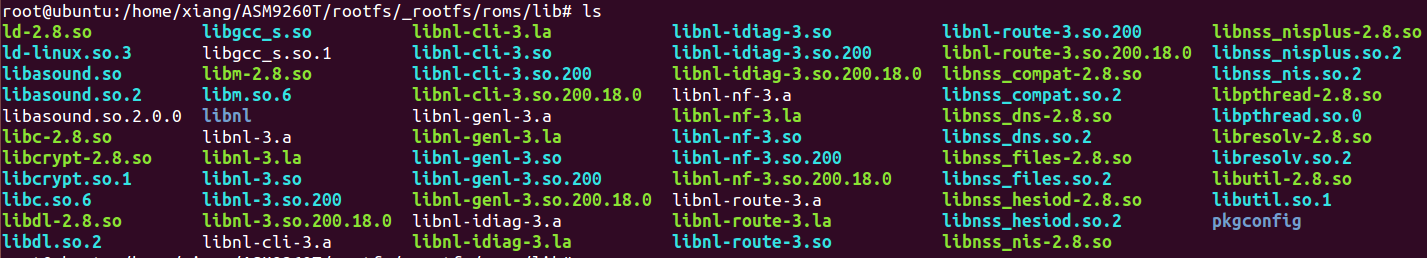
第3步:还要将上面编译器出来的libnl的lib库文件放在开发板文件系统上

重复第2步，只是拷贝的路径变成了开发板的文件系统路径



本来ASM9260T原生文件系统就只有这些文件



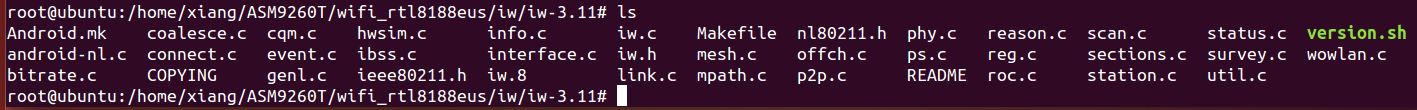


现在ASM9260T原生文件系统里面的lib库就多了libnl的库了。

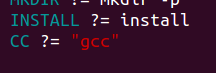
现在我们来编译iw，wpa\_supplicant，dhcp工具

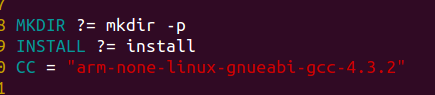


解压这个文件

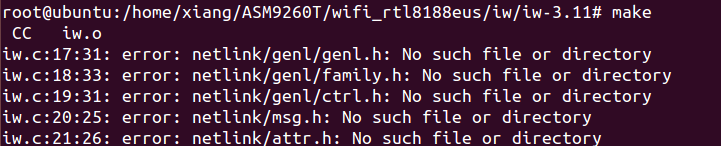
****

发现这个文件里面没有configuer文件，那我们就是要修改Makefile了

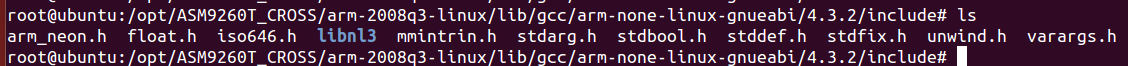
我们发现Makefile默认的是x86的gcc交叉编译器

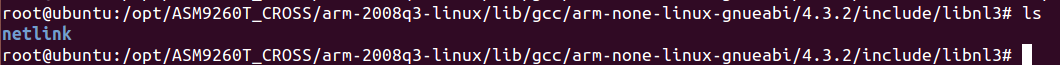


我们强制改成我们开发板的交叉编译器



然后make发现报错，这个意思是它找不到libnl库netlink目录下的头文件，但是我们前面编译了libnl库了啊，头文件也放在交叉编译器下面了。

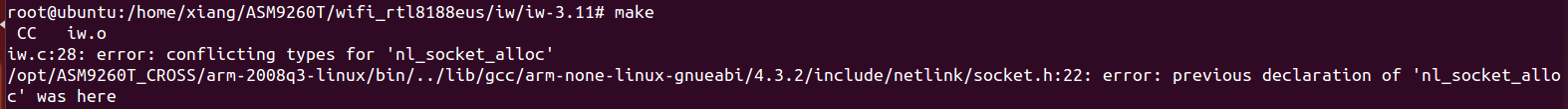




我们发现netlink不是完全暴露在include目录下的，前面多了一个无用的libnl3目录

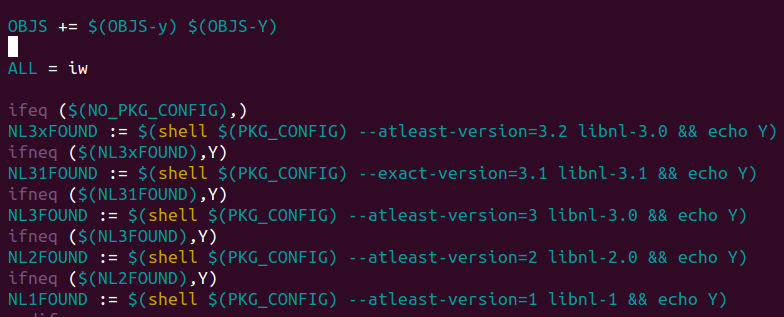


我们把netlink目录从libnl3目录中拿出来放在交叉编译器include下就是了。

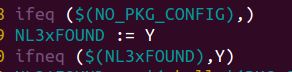


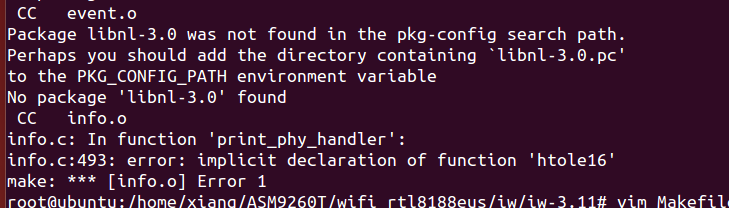
发现这个错误，这是因为我们没有指定libnl的版本。

我们使用的libnl的版本是3.2.23



我们选择NL3X，将这里直接写成Y

然后再make

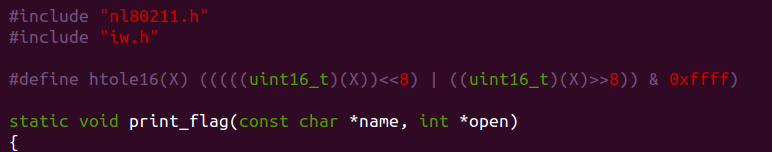


前面编译过了，但是出现了htole16错误

我们打开info.c



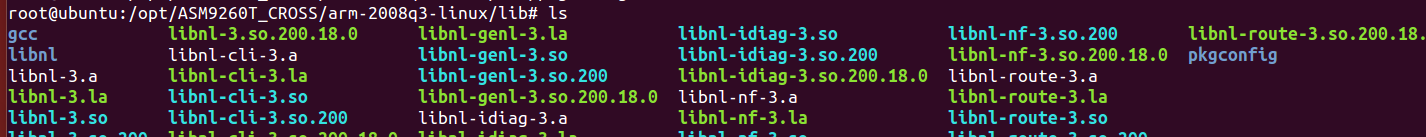
发现使用了hotle16，但是hotle16函数未定义。按照网上的文档我们在info.c前面加个宏



然后再次make



出现了找不到genl动态库的问题

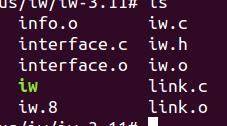


因为动态库的pkgconfig是放在交叉编译目录下的



执行用export导出PKG，指定库的pagconfig目录路径

然后再make



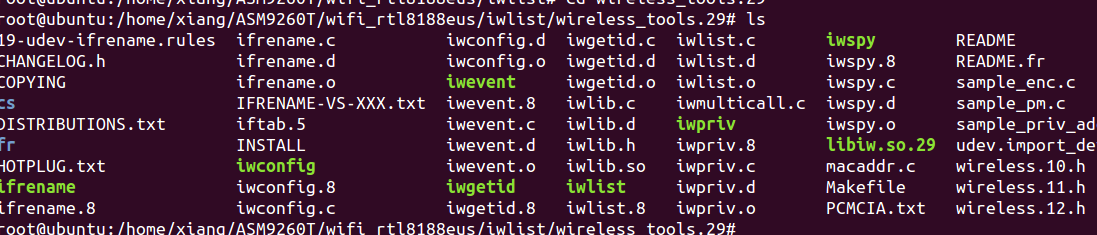
成功编译出iw应用程序，现在只需要将iw复制到开发板文件系统就可以运行

**iwconfig，iwlist工具移植**

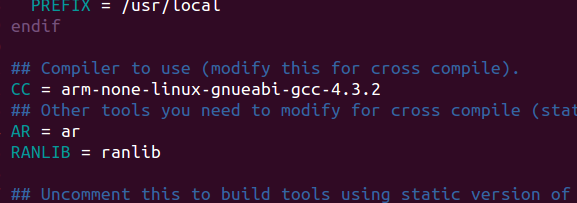


下载一个wireless\_tools.29版本的包

解压压缩包

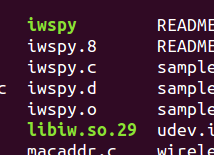


修改Makefile



指定你自己板子配套的交叉编译器

然后make

将iwconfig，和iwlist软件拷贝到开发板bin目录，以方便好使用。

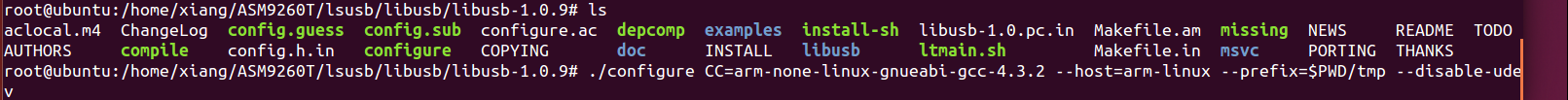
将libiw.so.29库拷贝到开发板文件系统lib目录下

**lsusb命令移植**

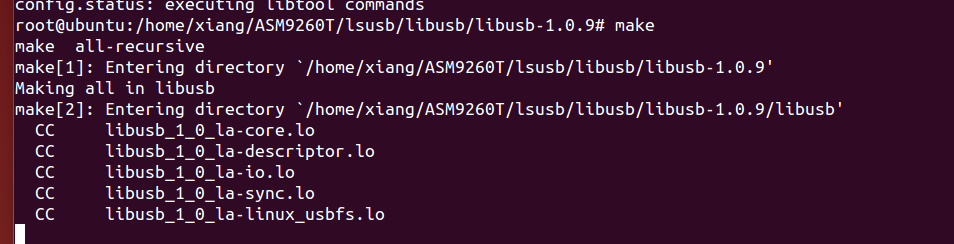
第1步：安装libusb1.0库

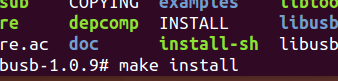
这libusb-1.0.9压缩包

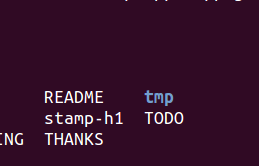
找个目录解压



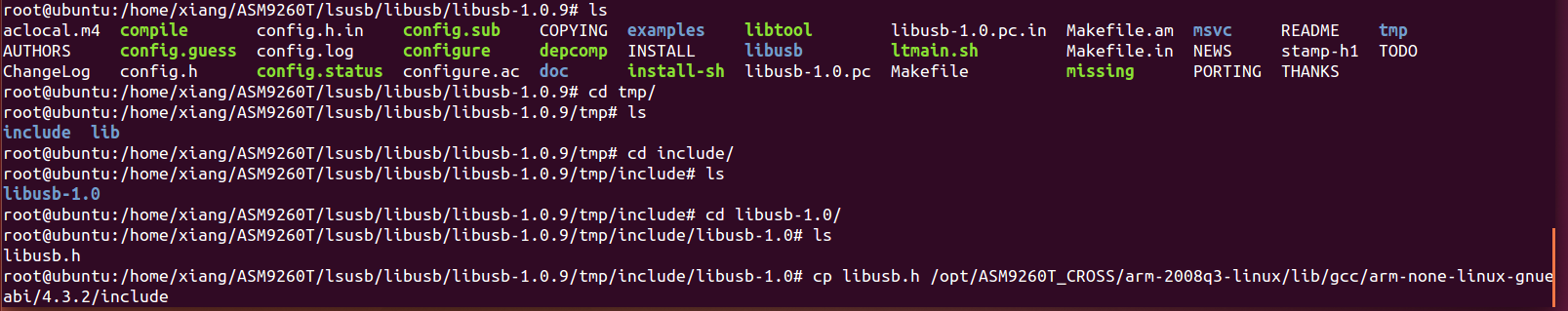
./configure CC=arm-none-linux-gnueabi-gcc-4.3.2 --host=arm-linux --prefix=$PWD/tmp --disable-udev



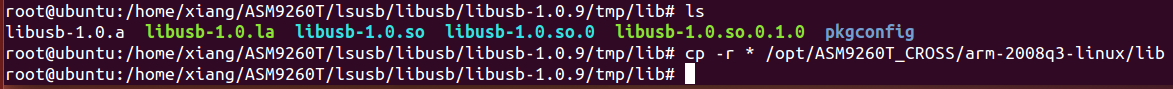
make install安装软件到tmp目录



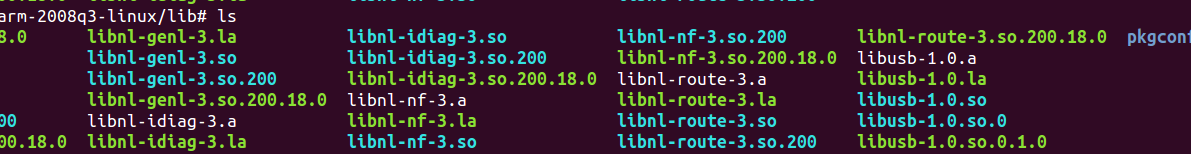
把头文件和lib库文件复制进交叉编译工具链



我要把libusb.h复制到交叉工具链include目录里



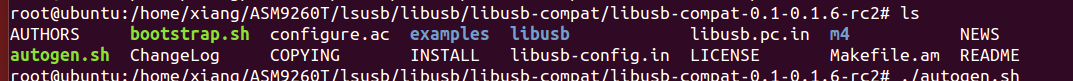
我还要把lib库复制进交叉工具链里面



你看交叉编译器有libusb-1.0的库了

然后把上面libusb-1.0的头文件和库文件在复制一次到开发板文件系统

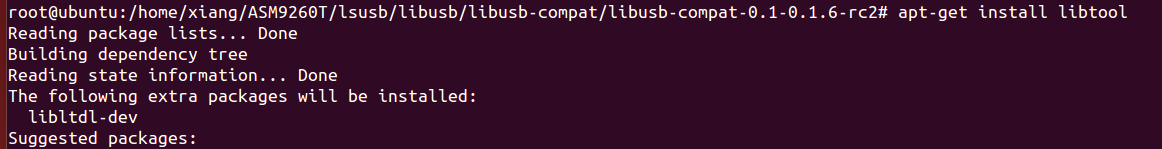
**移植libusb-compat-0.1-0.1.6**



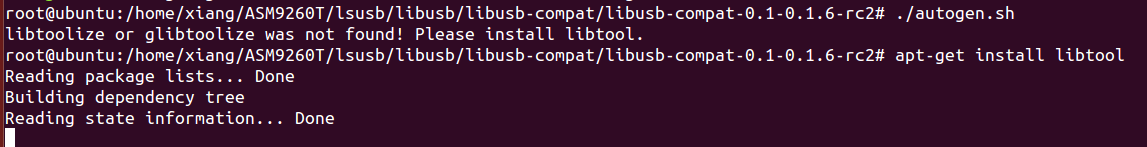
解压后的libusb-compat没有configure文件



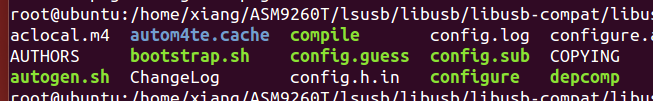
我们执行autogen.sh获取configure失败。看来要先安装libtool工具



一路点Y安装



在执行autogen.sh就可以了



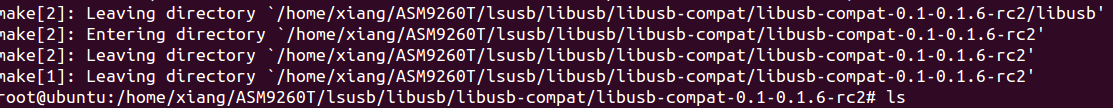
Configure文件生成了。

./configure CC=arm-none-linux-gnueabi-gcc-4.3.2 --host=arm-none-linux-gnueabi --prefix=$PWD/tmp PKG\_CONFIG\_PATH=/opt/ASM9260T\_CROSS/arm-2008q3-linux/lib/pkgconfig



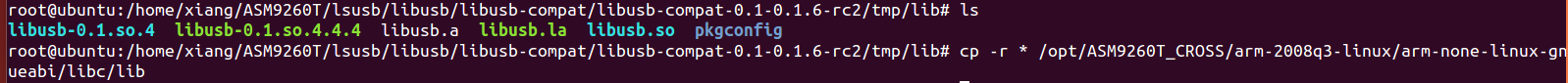
一定要用PKG\_CONFIG\_PATH指定你前面编译的libusb pkgconfig目录

然后make

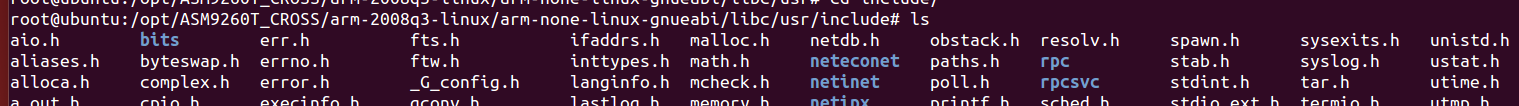


编译成功

安装在tmp目录下



将lib库复制到交叉编译器的libc目录下的lib目录，一定是libc，千万不要复制错了



将include文件复制到交叉编译器libc/usr/include目录下，记住一定是libc下的usr/include

也可以复制到交叉编译器本目录下的include，每个平台交叉编译器include位置都不同。所以建议复制到libc下的usr/include

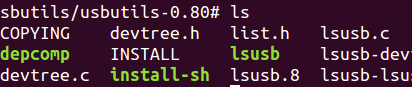
**usbutils-0.80**

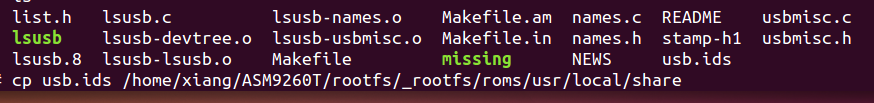
usbutils就是编译lsusb的文件，但是你一定要编译好libusb库，然后用libusb-compat文件把libusb-0.1.so.4.4.4库和头文件编译出来。usbutils要依靠libusb库和libusb-0.1.so.4.4.4库

进入usbutils目录

./configure CC=arm-none-linux-gnueabi-gcc-4.3.2 --host=arm-none-linux-gnueabi LIBUSB\_CFLAGS="/opt/ASM9260T\_CROSS/arm-2008q3-linux/lib/gcc/arm-none-linux-gnueabi/4.3.2/include" LIBUSB\_LIBS="/opt/ASM9260T\_CROSS/arm-2008q3-linux/arm-none-linux-gnueabi/libc/lib/libusb.so" CPPFLAGS=-I"/opt/ASM9260T\_CROSS/arm-2008q3-linux/lib/gcc/arm-none-linux-gnueabi/4.3.2/include" CFLAGS="-O2"

然后make

然后lsusb软件就编译出来了，把软件复制到开发板文件系统下就行。



然后把usb.ids复制到文件系统下就行。

**USB WIFI AP模式**

### Wifi芯片：RTL8188EUS ETV

第1步：编译hostapd软件

编译hostapd软件需要libnl库，在我们前面编译iw ，wpa\_supplicant，dhcp，这三个命令工具的时候已经编译了libnl库

现在我们就直接用libnl库来依赖就行了。

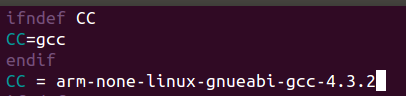
需要下载hostapd-2.0



这是解压后的目录，进入hostapd目录把defconfig复制成.config

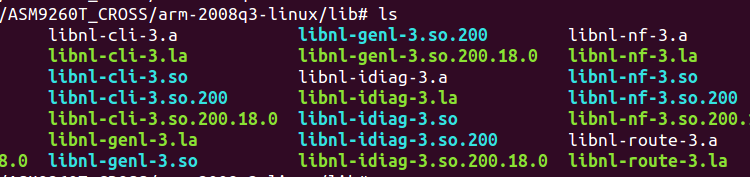


修改makefile



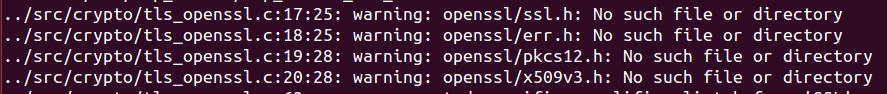
把默认的ifdef CC这三行删除掉

自己用CC指定你开发板使用的交叉编译器



在编译之前我们确定交叉编译器里面有libnl的库

然后我们make

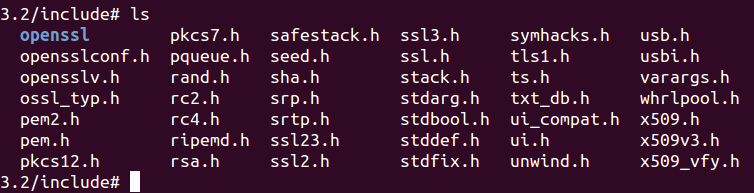


发现没有openssl的库，所以我们还要编译openssl库给hostapd用，编译openssl看下面编译openssl章节

再make

warning: openssl/ssl.h: No such file or directory 报错发现找不到ssl.h头文件

其实我前面已经将头文件复制给交叉编译器的include了，为什么还出这个问题呢？

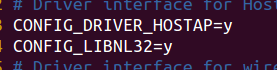


不能把openssl目录里面的头文件复制在交叉编译器的include目录下，必须把整个openssl目录复制过来，ssl里面的程序是先找openssl这个目录，它认这个死理。所以要在include目录下看到openssl目录，所以我把整个openssl目录复制过来问题才得到解决。

然后再make



发现找不到-lnl-3，修改.config，给.config配置文件里面加上CONFIG\_LIBNL32=y

然后在make



这个问题是因为交叉编译libc目录里面没有libnl库

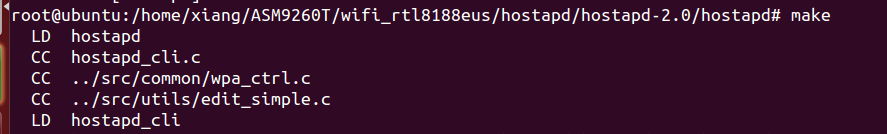
怎么又出现错误了，我以前在IMX6的poky交叉编译器上就没出现这个问题。

其实是找不到交叉编译器libc目录里面的libnl-3库，因为我们用的是ASM9260T交叉编译器，它的libc库在

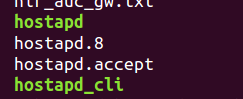
注意交叉编译器的libc文件在这里面



不是这个lib目录



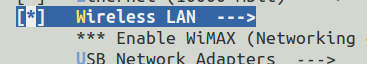
编译通过

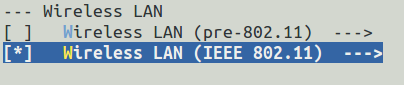
你看hostapd，hostapd\_cli软件编译出来了。

第2步: 配置内核

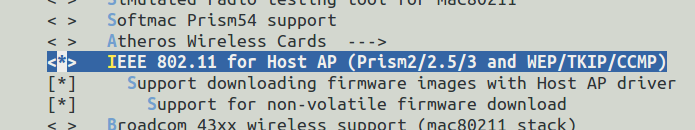








Wifi的AP模式这个一定要配置上



第3步骤：编写没有密码的hostapd.conf配置文件

vi hostapd.conf文件

interface=wlan0 //指定网卡节点

driver=nl80211 /\*这是支持的协议，但是rtl818 rtl871Xdrv，所以driver变量值不是

固定的nl80211，根据wifi网卡情况

而定\*/

ssid=ASM9260T //你wifi ap模式的热点名字

channel=1 //这是指定通道(信道)，具体问题看下面加密conf了解

hw\_mode=g /\*wifi分为b/g/n三个模式，这三个模式

代表数据传输速度

b = 11M速度

g = 54M速度

n = 150M或者300M

选择b模，g模，n模，根据你使用的路

由器来决定，必须和路由器兼容，普通

家庭都是选择g模。\*/

第3步骤，选择编写需要密码的hostapd.conf配置文件

vi hostapd.conf文件

interface = wlan0

driver=nl80211

ssid=ASM9260T

channel=1 /\*这是指定(通道)信道，为了 信道间不相互干扰，我们一般选择

1，6，11\*/

macaddr\_acl=0 /\*指定mac地址的过滤规则

0表示禁止列表的mac地址不同意

1表示同意列表的mac地址同意

这个功能就是给指定mac地址的手

机上网，没指定的不能上网，减少

路由器负担\*/

auth\_algs=1 /\*这个变量指定是用什么模式

0表示OPEN模式，就是没有密码

1表示WEP模式，需要密码

对于WPA/WPA2这里也必须写1\*/

ignore\_broadcast\_ssid=0 //这个是广播ssid什么的

wpa=3 /\*支持哪种WPA,

0表示支持WPA

1表示支持WPA2

3表示既支持WPA，又支持WPA2\*/

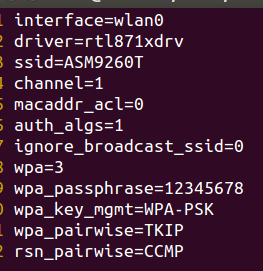
wpa\_passphrase=12345678 //这是你的AP密码1234567，密码一定要8位以上

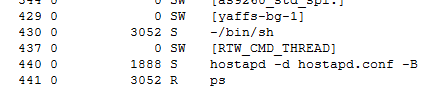
wpa\_key\_mgmt=WPA-PSK /\*WPA-PSK就是手机和开发板直接

认证密码，不经过服务器认证。\*/

wpa\_pairwise=TKIP //TKIP为wpa的加密方式

rsn\_ pairwise=CCMP //CCMP为wpa2的加密方式





Hostapd服务启动成功



配置文件写好后用hostapd加载

这样hostapd就是成功使用了，手机上面也能看到ASM9260T的wifi热点

启动dhcpd服务(有些arm linux平台是udhcpd)

在文件系统/etc目录下创建udhcpd.conf

start 192.168.2.2 //动态给手机分配IP地址的起始地址

end 192.168.2.254 //这是结束地址，最多能连接252个手机

interface wlan0 //网卡节点

opt dns 192.168.2.1 //分配dns

option subnet 255.255.255.0 //子网掩码

opt router 192.168.2.1 //路由

option domain local

option lease 864000

编写完成后最好将udhcpd.conf放在文件系统的/etc目录下

然后启动dhcpd服务

在S3C2440的文件系统里面是用dhcpd软件启动

dhcpd –cf /etc/uhcpd.conf wlan0

但是在IMX6的文件系统，或者ASM9260的Cramfs文件系统下是用udhcpd启动

udhcpd -f /etc/udhcpd.conf

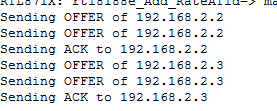


在启动udhcpd的过程中发现找不到relaease文件，所以你要根据错误路径，去该路径下建立leases文件，/var/lib/misc/udhcpd.leases



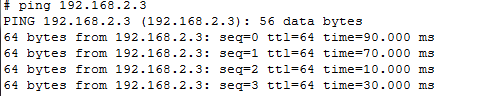
然后我们用手机连接我们的开发板

我们在启动hostapd之前一定要先安照udhcpd.conf分配的ip地址范围去设置ifconfig wlan0 IP地址



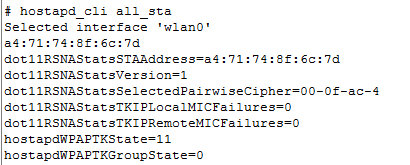
这是分配给手机的IP地址

然后我们用ping链接手机看看是否成功



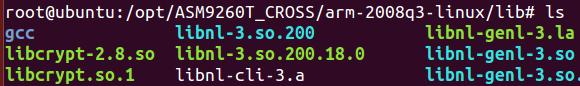
Ping连接成功，证明手机和开发板通信没有问题了。

WIF网卡AP模式就相当于路由器，所以我们可以用hostapd\_cli all\_sta来查看有多少网卡连接上了我开发板的wifi网卡，(记住使用hostapd\_cli之前，一定要在hostapd.conf文件里面加ctrl\_interface接口路径，我上面没有加，所以你一定记得加)

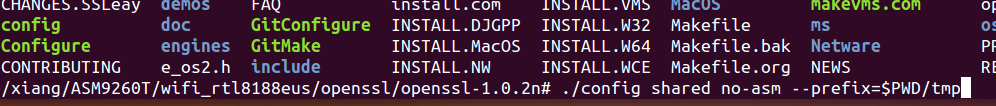


**编译openssl库**

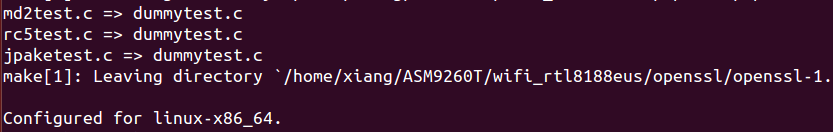
下载解压openssl文件，openssl依赖libcrypt库，然后我们用的AS9260T开发板自带libcrypt库，但是交叉编译器不带，所以把开发板的libcrypt库拷贝给编译器



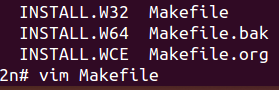




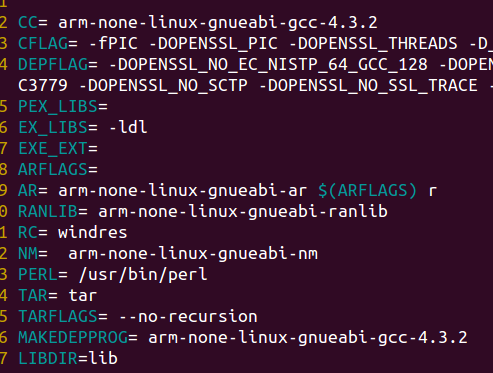
./config shared no-asm --prefix=$PWD/tmp 先配置config文件得到Makefile。Shared是要求编译的时候要输出动态库，no-asm是不优化，我们这里是给arm平台用，所以不需要优化



成功配置

去Makefile文件里面指定交叉编译器配置

   修改这几个变量



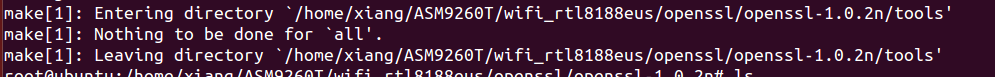
都修改成了我的交叉编译器

然后make 编译的时候报错 cc1: error: unrecognized command line option "-m64"



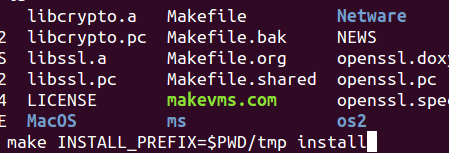
我们CPU是32位系统，交叉编译器也是32位的，所以把-m64去掉

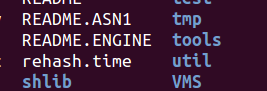
等待5分钟左右



编译成功了

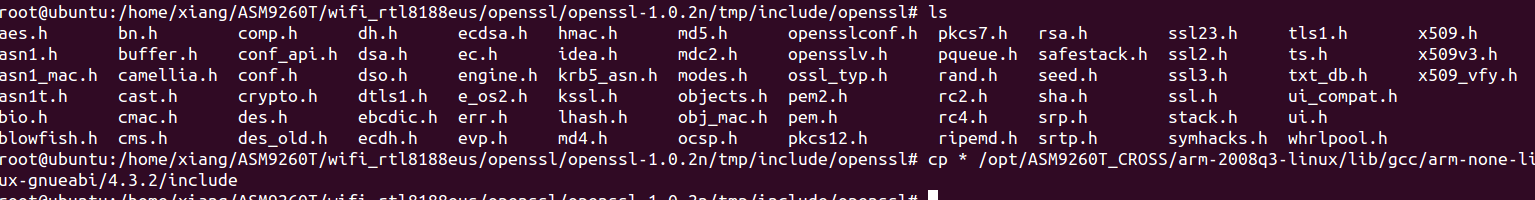
make install



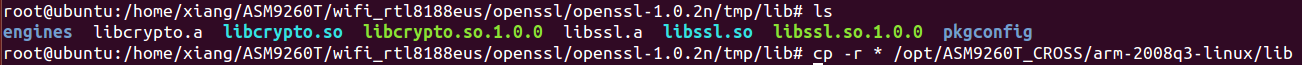
在tmp目录下



把include放入交叉编译器的include，把lib库放入交叉编译器的lib库



openssl里面的头文件include复制到交叉编译工具下了



Openssl里面的库文件复制到交叉编译工具下了

再复制一份openssl库文件到arm-2008q3-linux/arm-none-linux-gnueabi/libc/lib 目录下

记住还有把库文件复制到开发板的文件系统lib目录下



我们的开发板是ASM9260T，所以我把库复制到了ASM9260T文件系统下的lib目录里面