Linux USB wifi 使用指南

作者：向仔州

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

```
root@ubuntu:/home/xiang/IMX6/fsl-linux/drivers/net/wireless# ls
adm8211.c  at76c50x-usb.c  atmel.h      brcm80211   iwlegacy     mac80211_hwsim.c  mwifiex  ray_cs.c     rtl818x      zd1201.c
adm8211.h  at76c50x-usb.h  atmel_pci.c  built-in.o  iwlwifi      mac80211_hwsim.h  mwl8k.c  ray_cs.h     rtlwifi      zd1201.h
airo.c     ath             b43          cw1200      Kconfig      Makefile          orinoco  rayctl.h     ti           zd1211rw
airo_cs.c  atmel.c         b43legacy    hostap      libertas     modules.builtin   p54      rndis_wlan.c wl3501_cs.c
airo.h     atmel_cs.c      bcmdhd       ipw2x00     libertas_tf  modules.order     prism54  rt2x00       wl3501.h
```

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

```
root@ubuntu:/home/xiang/IMX6/fsl-linux/drivers/net/wireless/ath# ls
ar5523  ath9k      ath.mod.o   debug.c                dfs_pattern_detec
ath10k  ath.h      ath.o       debug.o                dfs_pri_detector.
ath5k   ath.ko     built-in.o  dfs_pattern_detector.c dfs_pri_detector.
ath6kl  ath.mod.c  carl9170    dfs_pattern_detector.h dfs_pri_detector.
```

```
root@ubuntu:/home/xiang/IMX6/fsl-linux/drivers/net/wireless/ath/ath9k# ls
ahb.c              ar9002_phy.h              ar9003_paprd.o         ar9580_1
ani.c              ar9002_phy.o              ar9003_phy.c           ath9k_co
ani.h              ar9003_2p2_initvals.h     ar9003_phy.h           ath9k_co
ani.o              ar9003_buffalo_initvals.h ar9003_phy.o           ath9k_co
```

```
hif_usb.c
hif_usb.h
htc_drv_bea
```

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

```
#include <asm/unaligned.h>
#include "htc.h"

/* identify firmware images */
#define FIRMWARE_AR7010_1_1     "htc_7010.fw"
#define FIRMWARE_AR9271         "htc_9271.fw"

MODULE_FIRMWARE(FIRMWARE_AR7010_1_1);
MODULE_FIRMWARE(FIRMWARE_AR9271);

static struct usb_device_id ath9k_hif_usb_ids[] = {
        { USB_DEVICE(0x0cf3, 0x9271) }, /* Atheros */
        { USB_DEVICE(0x0cf3, 0x1006) }, /* Atheros */
        { USB_DEVICE(0x0846, 0x9030) }, /* Netgear N150 */
        { USB_DEVICE(0x07D1, 0x3A10) }, /* Dlink Wireless 150 */
        { USB_DEVICE(0x13D3, 0x3327) }, /* Azurewave */
        { USB_DEVICE(0x13D3, 0x3328) }, /* Azurewave */
        { USB_DEVICE(0x13D3, 0x3346) }, /* IMC Networks */
        { USB_DEVICE(0x13D3, 0x3348) }, /* Azurewave */
```

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

```
        Power management options  --->
    [*] Networking support  --->
        Device Drivers  --->
```

```
< >   RXRPC session sockets
[*]   Wireless  --->
< >   WiMAX Wireless Broadb
```

```
--- Wireless
<*>   cfg80211 - wireless configuration API
[ ]     nl80211 testmode command
[ ]     enable developer warnings
[ ]     cfg80211 regulatory debugging
[*]     enable powersave by default
[*]   Old wireless static regulatory definitions
[*]   Wireless extensions
[*]     Wireless extensions sysfs files
<*>   Common routines for IEEE802.11 drivers
[ ]   lib80211 debugging messages
<*>   Generic IEEE 802.11 Networking Stack (mac80211)
      Default rate control algorithm (Minstrel)  --->
[*]   Enable mac80211 mesh networking (pre-802.11s) support
[ ]   Enable LED triggers
[ ]   Select mac80211 debugging features  --->
```

配置上这项 *

配置上这项 *

配置 wifi 芯片驱动

```
    Device Drivers  --->
[*] Network device support  --->
[*]   Wireless LAN  --->
```

```
--- Wireless LAN
[ ]   Wireless LAN (pre-802.11)  --->
[*]   Wireless LAN (IEEE 802.11)  --->
```

这里面有很多 wifi 厂家的芯片驱动，看有没有你用的型号

```
-*- Wireless LAN (IEEE 802.11)
< >   Marvell 8xxx Libertas WLAN driver support
< >   Marvell 8xxx Libertas WLAN driver support with thin firmware
< >   Atmel at76c503/at76c505/at76c505a USB cards
< >   USB ZD1201 based Wireless device support
< >   Wireless RNDIS USB support
< >   Realtek 8187 and 8187B USB support
< >   Simulated radio testing tool for mac80211
< >   Softmac Prism54 support
< >   Atheros Wireless Cards  --->
< >   IEEE 802.11 for Host AP (Prism2/2.5/3 and WEP/TKIP/CCMP)
< >   Broadcom 43xx wireless support (mac80211 stack)
< >   Broadcom 43xx-legacy wireless support (mac80211 stack)
< >   ZyDAS ZD1211/ZD1211B USB-wireless support
< >   Ralink driver support  --->
< >   TI wl12xx driver support  --->
< >   Intel Wireless Multicomm 3200 WiFi driver
```

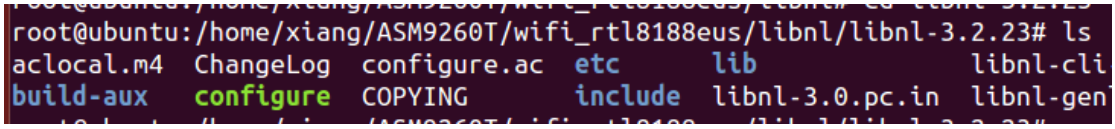
看这么多型号，如果这里没有，你就要找厂家要 wifi 芯片的驱动，然后自己编译成.ko 直接板子上 insmod，或者编译成.o 放进内核

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

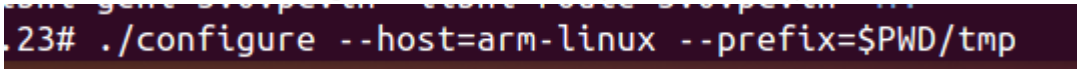
先下载 libnl 库

我用的是 libnl-3.2.23 的库，在 linux 下解压



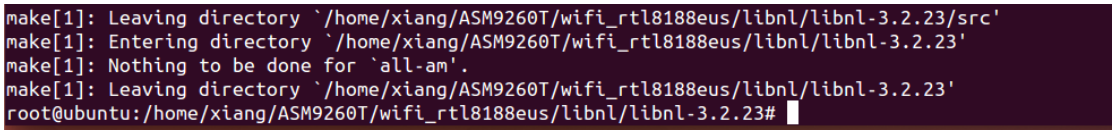
我们看到有 configure 文件

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

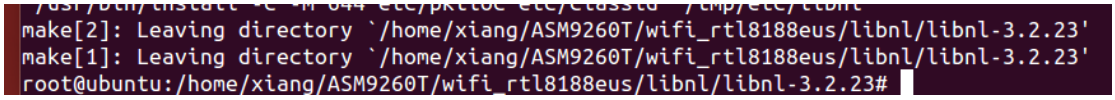


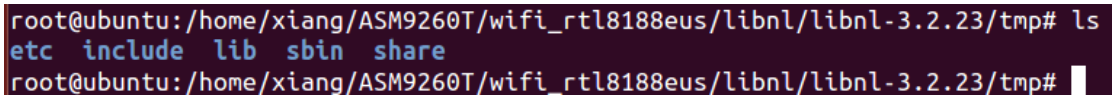
--prefix = $(PWD)/tmp 就是在 make install 的时候把编译后的文件放在当前目录 tmp 目录下

然后 make



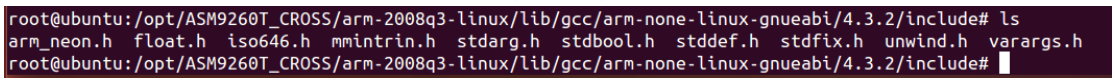
再执行 make install



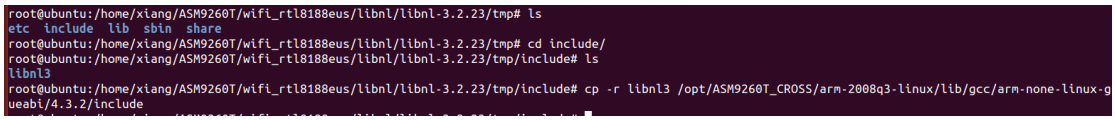


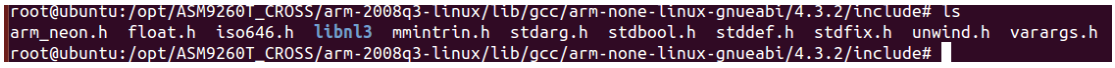
进入 tmp 目录就可以看到我们编译出来的 include，lib，sbin.....等待目录文件了

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



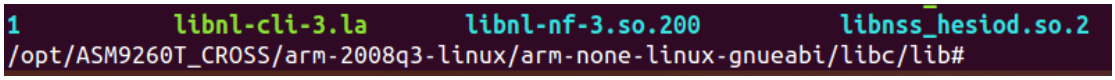
我用的是 ASM9260T ARM9 芯片，所以放在这个交叉编译器目录下



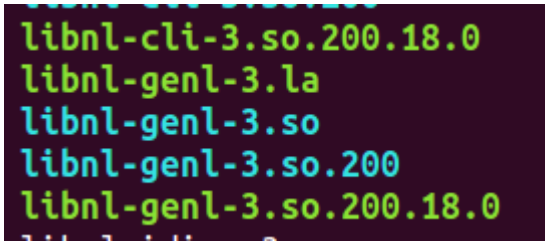


这就放进来了

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

```
root@ubuntu:/home/xiang/ASM9260T/rootfs/_rootfs/roms/lib# ls
ld-2.8.so          libc-2.8.so       libdl.so.2      libnss_compat-2.8.so  libnss_files.so.2      libnss_nisplus.so.2  libresolv.so.2
ld-linux.so.3      libcrypt-2.8.so   libgcc_s.so     libnss_compat.so.2    libnss_hesiod-2.8.so   libnss_nis.so.2      libutil-2.8.so
libasound.so       libcrypt.so.1     libgcc_s.so.1   libnss_dns-2.8.so     libnss_hesiod.so.2     libpthread-2.8.so    libutil.so.1
libasound.so.2     libc.so.6         libm-2.8.so     libnss_dns.so.2       libnss_nis-2.8.so      libpthread.so.0
libasound.so.2.0.0 libdl-2.8.so      libm.so.6       libnss_files-2.8.so   libnss_nisplus-2.8.so  libresolv-2.8.so
root@ubuntu:/home/xiang/ASM9260T/rootfs/_rootfs/roms/lib#
```

本来 ASM9260T 原生文件系统就只有这些文件

```
libnl-3.la          libnl-cli-3.so          libnl-genl-3.so.200       libnl-idiag-3.so.200.18.0  libnl-route-3.a
libnl-3.so          libnl-cli-3.so.200      libnl-genl-3.so.200.18.0  libnl-nf-3.a               libnl-route-3.la
libnl-3.so.200      libnl-cli-3.so.200.18.0 libnl-idiag-3.a           libnl-nf-3.la              libnl-route-3.so
libnl-3.so.200.18.0 libnl-genl-3.a          libnl-idiag-3.la          libnl-nf-3.so              libnl-route-3.so.200
root@ubuntu:/home/xiang/ASM9260T/wifi_rtl8188eus/libnl/libnl-3.2.23/tmp/lib# cp * -r /opt/ASM9260T_CROSS/arm-2008q3-linux/lib
root@ubuntu:/home/xiang/ASM9260T/wifi_rtl8188eus/libnl/libnl-3.2.23/tmp/lib# cp * -r /home/xiang/ASM9260T/rootfs/_rootfs/roms/lib
root@ubuntu:/home/xiang/ASM9260T/rootfs/_rootfs/roms/lib# ls
ld-2.8.so          libgcc_s.so          libnl-cli-3.la          libnl-idiag-3.so          libnl-route-3.so.200       libnss_nisplus-2.8.so
ld-linux.so.3      libgcc_s.so.1        libnl-cli-3.so          libnl-idiag-3.so.200      libnl-route-3.so.200.18.0  libnss_nisplus.so.2
libasound.so       libm-2.8.so          libnl-cli-3.so.200      libnl-idiag-3.so.200.18.0 libnss_compat-2.8.so       libnss_nis.so.2
libasound.so.2     libm.so.6            libnl-cli-3.so.200.18.0 libnl-nf-3.a              libnss_compat.so.2         libpthread-2.8.so
libasound.so.2.0.0 libnl                libnl-genl-3.a          libnl-nf-3.la             libnss_dns-2.8.so          libpthread.so.0
libc-2.8.so        libnl-3.a            libnl-genl-3.la         libnl-nf-3.so             libnss_dns.so.2            libresolv-2.8.so
libcrypt-2.8.so    libnl-3.la           libnl-genl-3.so         libnl-nf-3.so.200         libnss_files-2.8.so        libresolv.so.2
libcrypt.so.1      libnl-3.so           libnl-genl-3.so.200     libnl-nf-3.so.200.18.0    libnss_files.so.2          libutil-2.8.so
libc.so.6          libnl-3.so.200       libnl-genl-3.so.200.18.0 libnl-route-3.a          libnss_hesiod-2.8.so       libutil.so.1
libdl-2.8.so       libnl-3.so.200.18.0  libnl-idiag-3.a         libnl-route-3.la          libnss_hesiod.so.2         pkgconfig
libdl.so.2         libnl-cli-3.a        libnl-idiag-3.la        libnl-route-3.so          libnss_nis-2.8.so
```

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

```
root@ubuntu:/home/xiang/ASM9260T/wifi_rtl8188eus/iw#
iw-3.11.tar.bz2
```

解压这个文件

```
root@ubuntu:/home/xiang/ASM9260T/wifi_rtl8188eus/iw/iw-3.11# ls
Android.mk    coalesce.c  cqm.c     hwsim.c      info.c       iw.c     Makefile  nl80211.h  phy.c     reason.c  scan.c      status.c  version.sh
android-nl.c  connect.c   event.c   ibss.c       interface.c  iw.h     mesh.c    offch.c    ps.c      reg.c     sections.c  survey.c  wowlan.c
bitrate.c     COPYING     genl.c    ieee80211.h  iw.8         link.c   mpath.c   p2p.c      README    roc.c     station.c   util.c
root@ubuntu:/home/xiang/ASM9260T/wifi_rtl8188eus/iw/iw-3.11#
```

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

```
INSTALL ?= install
CC ?= "gcc"
```

我们发现 Makefile 默认的是 x86 的 gcc 交叉编译器

```
MKDIR ?= mkdir -p
INSTALL ?= install
CC = "arm-none-linux-gnueabi-gcc-4.3.2"
```

我们强制改成我们开发板的交叉编译器

```
root@ubuntu:/home/xiang/ASM9260T/wifi_rtl8188eus/iw/iw-3.11# make
  CC    iw.o
iw.c:17:31: error: netlink/genl/genl.h: No such file or directory
iw.c:18:33: error: netlink/genl/family.h: No such file or directory
iw.c:19:31: error: netlink/genl/ctrl.h: No such file or directory
iw.c:20:25: error: netlink/msg.h: No such file or directory
iw.c:21:26: error: netlink/attr.h: No such file or directory
```

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。

```
root@ubuntu:/opt/ASM9260T_CROSS/arm-2008q3-linux/lib/gcc/arm-none-linux-gnueabi/4.3.2/include# ls
arm_neon.h  float.h  iso646.h  libnl3  mmintrin.h  stdarg.h  stdbool.h  stddef.h  stdfix.h  unwind.h  varargs.h
root@ubuntu:/opt/ASM9260T_CROSS/arm-2008q3-linux/lib/gcc/arm-none-linux-gnueabi/4.3.2/include#
root@ubuntu:/opt/ASM9260T_CROSS/arm-2008q3-linux/lib/gcc/arm-none-linux-gnueabi/4.3.2/include/libnl3# ls
netlink
root@ubuntu:/opt/ASM9260T_CROSS/arm-2008q3-linux/lib/gcc/arm-none-linux-gnueabi/4.3.2/include/libnl3#
```

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

```
root@ubuntu:/opt/ASM9260T_CROSS/arm-2008q3-linux/lib/gcc/arm-none-linux-gnueabi/4.3.2/include# ls
arm_neon.h  float.h  iso646.h  libnl3  mmintrin.h  netlink  stdarg.h  stdbool.h  stddef.h  stdfix.h  unwind.h  varargs.h
root@ubuntu:/opt/ASM9260T_CROSS/arm-2008q3-linux/lib/gcc/arm-none-linux-gnueabi/4.3.2/include#
```

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。

```
root@ubuntu:/home/xiang/ASM9260T/wifi_rtl8188eus/iw/iw-3.11# make
  CC   iw.o
iw.c:28: error: conflicting types for 'nl_socket_alloc'
/opt/ASM9260T_CROSS/arm-2008q3-linux/bin/../lib/gcc/arm-none-linux-gnueabi/4.3.2/include/netlink/socket.h:22: error: previous declaration of 'nl_socket_allo
c' was here
```

发现这个错误，这是因为我们没有指定 libnl 的版本。

```
libnl-3.2.23.tar.gz
```

我们使用的 libnl 的版本是 3.2.23

```
OBJS += $(OBJS-y) $(OBJS-Y)

ALL = iw

ifeq ($(NO_PKG_CONFIG),)
NL3xFOUND := $(shell $(PKG_CONFIG) --atleast-version=3.2 libnl-3.0 && echo Y)
ifneq ($(NL3xFOUND),Y)
NL31FOUND := $(shell $(PKG_CONFIG) --exact-version=3.1 libnl-3.1 && echo Y)
ifneq ($(NL31FOUND),Y)
NL3FOUND := $(shell $(PKG_CONFIG) --atleast-version=3 libnl-3.0 && echo Y)
ifneq ($(NL3FOUND),Y)
NL2FOUND := $(shell $(PKG_CONFIG) --atleast-version=2 libnl-2.0 && echo Y)
ifneq ($(NL2FOUND),Y)
NL1FOUND := $(shell $(PKG_CONFIG) --atleast-version=1 libnl-1 && echo Y)
```

我们选择 NL3X，将这里直接写成 Y

```
ifeq ($(NO_PKG_CONFIG),)
NL3xFOUND := Y
ifneq ($(NL3xFOUND),Y)
```

然后再 make

```
  CC    event.o
Package libnl-3.0 was not found in the pkg-config search path.
Perhaps you should add the directory containing `libnl-3.0.pc'
to the PKG_CONFIG_PATH environment variable
No package 'libnl-3.0' found
  CC    info.o
info.c: In function 'print_phy_handler':
info.c:493: error: implicit declaration of function 'htole16'
make: *** [info.o] Error 1
root@ubuntu:/home/xiang/ASM9260T/wifi_rtl8188eus/iw/iw-3.11# vim Makefil
```

前面编译过了，但是出现了 htole16 错误

```
/iw-3.11# vi info.c
```

我们打开 info.c

```
if (cm->cap_info & htole16(IEEE80211_HT_CAP_MAX_AMSDU))
        printf("\t\t * maximum A-MSDU length\n");
```

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

```
#include "nl80211.h"
#include "iw.h"

#define htole16(X) (((((uint16_t)(X))<<8) | ((uint16_t)(X)>>8)) & 0xffff)

static void print_flag(const char *name, int *open)
{
```

然后再次 make

```
/opt/ASM9260T_CROSS/arm-2008q3-linux/bin/../lib/gcc/arm-none-linux-gnueabi/4.3.2/../../../../arm-none-linux-gnueabi/bin/ld: cannot find -lnl-genl-3
collect2: ld returned 1 exit status
```

出现了找不到 genl 动态库的问题

```
root@ubuntu:/opt/ASM9260T_CROSS/arm-2008q3-linux/lib# ls
gcc             libnl-3.so.200.18.0     libnl-genl-3.la          libnl-idiag-3.so          libnl-nf-3.so.200       libnl-route-3.so.200.18.
libnl           libnl-cli-3.a           libnl-genl-3.so          libnl-idiag-3.so.200      libnl-nf-3.so.200.18.0  pkgconfig
libnl-3.a       libnl-cli-3.la          libnl-genl-3.so.200      libnl-idiag-3.so.200.18.0 libnl-route-3.a
libnl-3.la      libnl-cli-3.so          libnl-genl-3.so.200.18.0 libnl-nf-3.a              libnl-route-3.la
libnl-3.so      libnl-cli-3.so.200      libnl-idiag-3.a          libnl-nf-3.la             libnl-route-3.so
```

因为动态库的 pkgconfig 是放在交叉编译目录下的

```
export PKG_CONFIG_PATH=/opt/ASM9260T_CROSS/arm-2008q3-linux/lib/pkgconfig
```

执行用 export 导出 PKG，指定库的 pagconfig 目录路径

然后再 make

```
info.o          iw.c
interface.c     iw.h
interface.o     iw.o
iw              link.c
iw.8            link.o
```

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

wpa_supplicant v2.2 版本编译使用

确保交叉编译器有 libnl 和 openssl 的.so 库文件和 include 文件

```
/wifi/wpa_supplicant-2.2/wpa_supplicant#
root@ubuntu:/home/xiang/IMX6/wifi# ls
wpa_supplicant-2.2  wpa_supplicant-2.2.tar.gz
```

解压 V2.2

```
/wifi/wpa_supplicant-2.2/wpa_supplicant#
```

进入这个目录

```
defconfig
```

这是 wpa_supplicant 目录下的 defconfig 文件

```
/wpa_supplicant# cp defconfig .config
```

将 defconfig 文件拷贝成.config 文件

```
/wpa_supplicant# vim Makefile
```

修改 Makefile

```
1 ifndef CC
2 CC=gcc
3 endif
4
```

取消 Makefile 里面的 ifndef CC 只保留 CC=gcc，等下修改

```
CC=arm-poky-linux-gnueabi-gcc  -march=armv7-a -mthumb-interwork -mfloat-abi=hard -mfpu=neon -mtune=cortex-a9 --sysroot=/opt/poky/1.8/sysroots/cortexa
7hf-vfp-neon-poky-linux-gnueabi
```

设置 Makefile 的交叉编译路径

```
                  interworking.h      README-P2P
                  main.c              README-Windo
h                 main_none.c         README-WPS
t.c               main_winmain.c      scan.c
_dummy.mak        main_winsvc.c       scan.h
/wpa_supplicant-2.2/wpa_supplicant# make
```

make

```
../src/drivers/driver_nl80211.c:19:31: fatal error: netlink/genl/genl.h: No such file or directory
 #include <netlink/genl/genl.h>
                           ^
compilation terminated.
make: *** [../src/drivers/driver_nl80211.o] Error 1
```

报错

```
wpa_supplicant-2.2/src/drivers# vim drivers.mak
```

查找 drivers.mak 文件

```
35 ifdef CONFIG_LIBNL32
36   DRV_LIBS += -lnl-3
37   DRV_LIBS += -lnl-genl-3
38   DRV_CFLAGS += -DCONFIG_LIBNL20 -I/usr/include/libnl3
39 else
40   ifdef CONFIG_LIBNL_TINY
41     DRV_LIBS += -lnl-tiny
```

要在.config 配置
CONFIG_LIBNL32 才行

```
/wpa_supplicant-2.2/wpa_supplicant# vim .config
```

回到.config 文件修改

```
10 # to override previous
11 CONFIG_LIBNL32=y
12
13 # Uncomment following t
```

在.config 文件添加该参数，保存

```
root@ubuntu:/home/xiang/IMX6/wifi/wpa_supplicant-2.2/wpa_supplicant# vim .config
root@ubuntu:/home/xiang/IMX6/wifi/wpa_supplicant-2.2/wpa_supplicant# make
  CC  config.c
```

然后 make

```
../src/drivers/driver_nl80211.c:19:31: fatal error: netlink/genl/genl.h: No such file or directory
 #include <netlink/genl/genl.h>
                           ^
compilation terminated.
make: *** [../src/drivers/driver_nl80211.o] Error 1
```

还是报错，应该是 Makefile 的 CFLAGS 没有指定交叉编译器的头文件路径

```
13 CFLAGS += -I$(abspath ../src/utils)
14 CFLAGS += -I/opt/poky/1.8/sysroots/cortexa7hf-vfp-neon-poky-linux-gnueabi/usr/include/libnl3
```

添加 CFLAGS

在 Makefile 的 CFLAGS 里添加交叉编译文件的 include/libnl3 的

```
  CC  wpa_passphrase.c
  LD  wpa_passphrase
sed -e 's|\@BINDIR\@|/usr/local/sbin/|g' systemd/wpa_supplicant.service.in >systemd/wpa_supplicant.service
sed -e 's|\@BINDIR\@|/usr/local/sbin/|g' systemd/wpa_supplicant.service.arg.in >systemd/wpa_supplicant@.serv
sed -e 's|\@BINDIR\@|/usr/local/sbin/|g' systemd/wpa_supplicant-nl80211.service.arg.in >systemd/wpa_supplica
sed -e 's|\@BINDIR\@|/usr/local/sbin/|g' systemd/wpa_supplicant-wired.service.arg.in >systemd/wpa_supplicant
sed -e 's|\@BINDIR\@|/usr/local/sbin/|g' dbus/fi.epitest.hostap.WPASupplicant.service.in >dbus/fi.epitest.ho
sed -e 's|\@BINDIR\@|/usr/local/sbin/|g' dbus/fi.w1.wpa_supplicant1.service.in >dbus/fi.w1.wpa_supplicant1.s
```

make 编译成功

```
2/wpa_supplicant# make DESTDIR=$PWD/tmp install
```

然后 DESTDIR 指定安装软件目录，执行 install

```
root@ubuntu:/home/xiang/IMX6/wifi/wpa_supplicant-2.2/wpa_supplicant/tmp/usr/local/sbin# ls
wpa_cli  wpa_passphrase  wpa_supplicant
```

把软件拷贝到开发板的 bin 目录，或者是 sbin 目录，就可以使用了

iwconfig，iwlist 工具移植

```
xiang/ASM9260T/wifi_rtl8188eus/iwlist# ls
 wireless_tools.29.tar.gz
```

下载一个 wireless_tools.29 版本的包

解压压缩包

```
root@ubuntu:/home/xiang/ASM9260T/wifi_rtl8188eus/iwlist/wireless_tools.29# ls
19-udev-ifrename.rules  ifrename.c          iwconfig.d  iwgetid.c  iwlist.c       iwspy        README
CHANGELOG.h             ifrename.d          iwconfig.o  iwgetid.d  iwlist.d       iwspy.8      README.fr
COPYING                 ifrename.o          iwevent     iwgetid.o  iwlist.o       iwspy.c      sample_enc.c
cs                      IFRENAME-VS-XXX.txt iwevent.8   iwlib.c    iwmulticall.c  iwspy.d      sample_pm.c
DISTRIBUTIONS.txt       iftab.5             iwevent.c   iwlib.d    iwpriv         iwspy.o      sample_priv_ad
fr                      INSTALL             iwevent.d   iwlib.h    iwpriv.8       libiw.so.29  udev.import_de
HOTPLUG.txt             iwconfig            iwevent.o   iwlib.so   iwpriv.c       macaddr.c    wireless.10.h
ifrename                iwconfig.8          iwgetid     iwlist     iwpriv.d       Makefile     wireless.11.h
ifrename.8              iwconfig.c          iwgetid.8   iwlist.8   iwpriv.o       PCMCIA.txt   wireless.12.h
root@ubuntu:/home/xiang/ASM9260T/wifi_rtl8188eus/iwlist/wireless_tools.29#
```

修改 Makefile

```
PREFIX = /usr/local
endif

## Compiler to use (modify this for cross compile).
CC = arm-none-linux-gnueabi-gcc-4.3.2
## Other tools you need to modify for cross compile (stat
AR = ar
RANLIB = ranlib

## Uncomment this to build tools using static version of
```

指定你自己板子配套的交叉编译器

然后 make

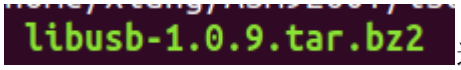
```
iftab.5
INSTALL
iwconfig

iwevent     iwgetid.o   iwlist.
iwevent.8   iwlib.c     iwmulti
iwevent.c   iwlib.d     iwpriv
iwevent.d   iwlib.h     iwpriv.
iwevent.o   iwlib.so    iwpriv.
iwgetid     iwlist      iwpriv.

iwspy       README
iwspy.8     README
iwspy.c     sample
iwspy.d     sample
iwspy.o     sample
libiw.so.29 udev.i
macaddr.c   wirele
```

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

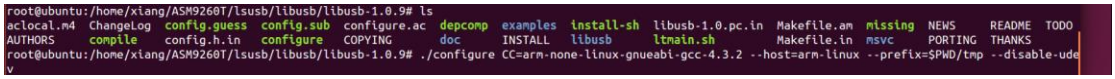
lsusb 命令移植

第 1 步：安装 libusb1.0 库

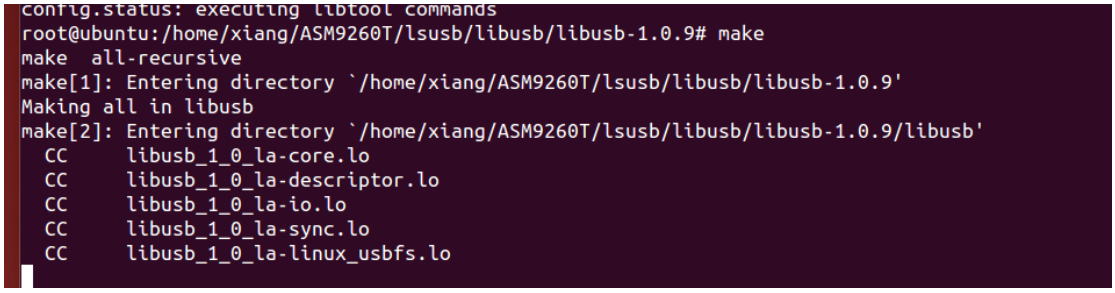


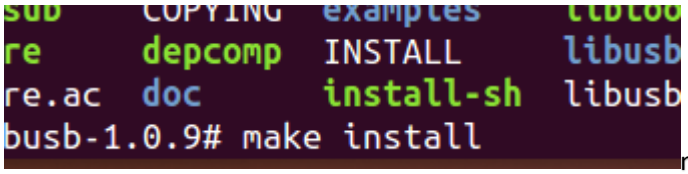
这 libusb-1.0.9 压缩包

找个目录解压

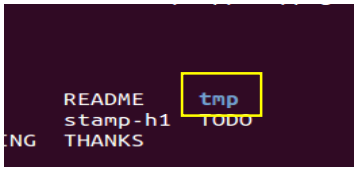


./configure CC=arm-none-linux-gnueabi-gcc-4.3.2 --host=arm-linux --prefix=$PWD/tmp --disable-udev



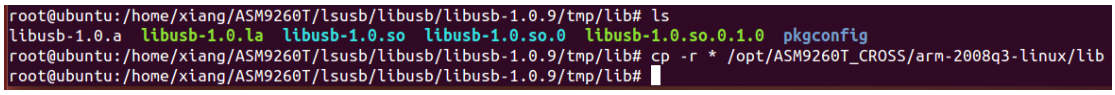


make install 安装软件到 tmp 目录

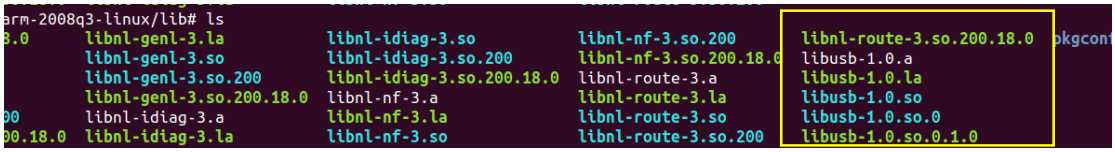


把头文件和 lib 库文件复制进交叉编译工具链

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

```
root@ubuntu:/home/xiang/ASM9260T/lsusb/libusb/libusb-compat/libusb-compat-0.1-0.1.6-rc2# ls
AUTHORS      bootstrap.sh  configure.ac  examples  libusb           libusb.pc.in  m4          NEWS
autogen.sh   ChangeLog     COPYING       INSTALL   libusb-config.in  LICENSE       Makefile.am  README
root@ubuntu:/home/xiang/ASM9260T/lsusb/libusb/libusb-compat/libusb-compat-0.1-0.1.6-rc2# ./autogen.sh
```

解压后的 libusb-compat 没有 configure 文件

```
root@ubuntu:/home/xiang/ASM9260T/lsusb/libusb/libusb-compat/libusb-compat-0.1-0.1.6-rc2# ./autogen.sh
libtoolize or glibtoolize was not found! Please install libtool.
```

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

```
root@ubuntu:/home/xiang/ASM9260T/lsusb/libusb/libusb-compat/libusb-compat-0.1-0.1.6-rc2# apt-get install libtool
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  libltdl-dev
Suggested packages:
```

一路点 Y 安装

```
root@ubuntu:/home/xiang/ASM9260T/lsusb/libusb/libusb-compat/libusb-compat-0.1-0.1.6-rc2# ./autogen.sh
libtoolize or glibtoolize was not found! Please install libtool.
root@ubuntu:/home/xiang/ASM9260T/lsusb/libusb/libusb-compat/libusb-compat-0.1-0.1.6-rc2# apt-get install libtool
Reading package lists... Done
Building dependency tree
Reading state information... Done
```

在执行 autogen.sh 就可以了

```
root@ubuntu:/home/xiang/ASM9260T/lsusb/libusb/libusb-compat/libu
aclocal.m4  autom4te.cache  compile       config.log  configure.
AUTHORS     bootstrap.sh    config.guess  config.sub  COPYING
autogen.sh  ChangeLog       config.h.in   configure   depcomp
```

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

```
root@ubuntu:/home/xiang/ASM9260T/lsusb/libusb/libusb-compat/libusb-compat-0.1-0.1.6-rc2# ./configure CC=arm-none-linux-gnueabi-gcc-4.3.2 --host=arm-none-lin
ux-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

```
make[2]: Leaving directory `/home/xiang/ASM9260T/lsusb/libusb/libusb-compat/libusb-compat-0.1-0.1.6-rc2/libusb'
make[2]: Entering directory `/home/xiang/ASM9260T/lsusb/libusb/libusb-compat/libusb-compat-0.1-0.1.6-rc2'
make[2]: Leaving directory `/home/xiang/ASM9260T/lsusb/libusb/libusb-compat/libusb-compat-0.1-0.1.6-rc2'
make[1]: Leaving directory `/home/xiang/ASM9260T/lsusb/libusb/libusb-compat/libusb-compat-0.1-0.1.6-rc2'
root@ubuntu:/home/xiang/ASM9260T/lsusb/libusb/libusb-compat/libusb-compat-0.1-0.1.6-rc2# ls
```

编译成功

```
2# make install
```

安装在 tmp 目录下

```
root@ubuntu:/home/xiang/ASM9260T/lsusb/libusb/libusb-compat/libusb-compat-0.1-0.1.6-rc2/tmp/lib# ls
libusb-0.1.so.4  libusb-0.1.so.4.4.4  libusb.a  libusb.la  libusb.so  pkgconfig
root@ubuntu:/home/xiang/ASM9260T/lsusb/libusb/libusb-compat/libusb-compat-0.1-0.1.6-rc2/tmp/lib# cp -r * /opt/ASM9260T_CROSS/arm-2008q3-linux/arm-none-linux-gn
ueabi/libc/lib
```

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

```
root@ubuntu:/opt/ASM9260T_CROSS/arm-2008q3-linux/arm-none-linux-gnueabi/libc/usr/include# ls
aio.h        bits        err.h        fts.h          ifaddrs.h   malloc.h    netdb.h     obstack.h  resolv.h    spawn.h     sysexits.h  unistd.h
aliases.h    byteswap.h  errno.h      ftw.h          inttypes.h  math.h      neteconet   paths.h    rpc         stab.h      syslog.h    ustat.h
alloca.h     complex.h   error.h      _G_config.h    langinfo.h  mcheck.h    netinet     poll.h     rpcsvc      stdint.h    tar.h       utime.h
a.out.h      cpio.h      execinfo.h   gconv.h        lastlog.h   memory.h    netipx      printf.h   sched.h     stdio_ext.h termio.h    utmp.h
```

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

```
ng/ASM9260T/lsusb/usbutils# ls
usbutils-0.80  usbutils-0.80.tar.gz
```

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

```
sbutils/usbutils-0.80# ls
COPYING     devtree.h   list.h    lsusb.c
depcomp     INSTALL     lsusb     lsusb-dev
devtree.c   install-sh  lsusb.8   lsusb-lsu
```

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。

```
list.h   lsusb.c          lsusb-names.o    Makefile.am  names.c  README    usbmisc.c
lsusb    lsusb-devtree.o  lsusb-usbmisc.o  Makefile.in  names.h  stamp-h1  usbmisc.h
lsusb.8  lsusb-lsusb.o    Makefile         missing      NEWS     usb.ids
cp usb.ids /home/xiang/ASM9260T/rootfs/_rootfs/roms/usr/local/share
```

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

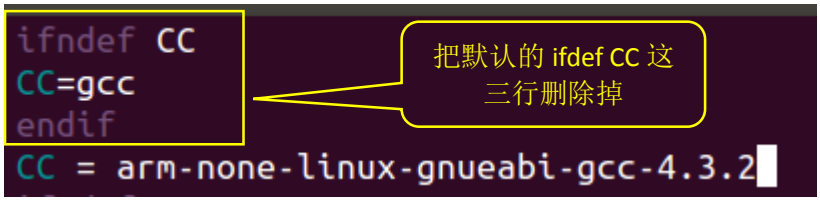
```
root@ubuntu:/home/xiang/ASM9260T/wifi_rtl8188eus/hostapd/hostapd-2.0# ls
COPYING  hostapd  patches  README  src
root@ubuntu:/home/xiang/ASM9260T/wifi_rtl8188eus/hostapd/hostapd-2.0# cd hostapd/
root@ubuntu:/home/xiang/ASM9260T/wifi_rtl8188eus/hostapd/hostapd-2.0/hostapd# ls
android.config  ctrl_iface.c  eap_register.c          hlr_auc_gw.txt  hostapd.conf            hostapd.sim_db   Makefile
Android.mk      ctrl_iface.h  eap_register.h          hostapd.8       hostapd.deny            hostapd.vlan     nt_password_hash.c
ChangeLog       defconfig     eap_testing.txt         hostapd.accept  hostapd.eap_user        hostapd.wpa_psk  README
config_file.c   dump_state.c  hlr_auc_gw.c            hostapd_cli.1   hostapd.eap_user_sqlite logwatch         README-WPS
config_file.h   dump_state.h  hlr_auc_gw.milenage_db  hostapd_cli.c   hostapd.radius_clients  main.c           wired.conf
root@ubuntu:/home/xiang/ASM9260T/wifi_rtl8188eus/hostapd/hostapd-2.0/hostapd#
```

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

```
/hostapd# cp defconfig .config
```

```
/hostapd# vim Makefile
```

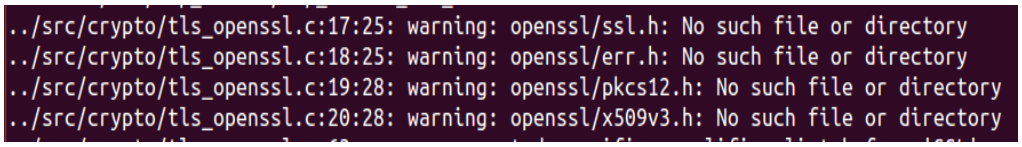
修改 makefile



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

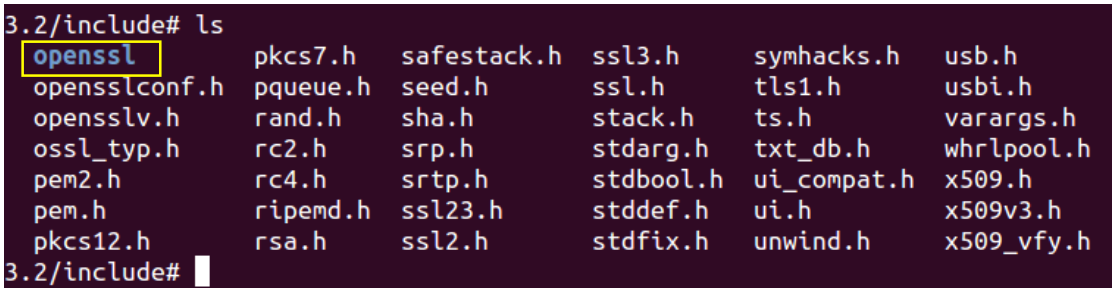


发现没有 openssl 的库，所以我们还要编译 openssl 库给 hostapd 用，编译 openssl 看下面编译 openssl 章节

再 make

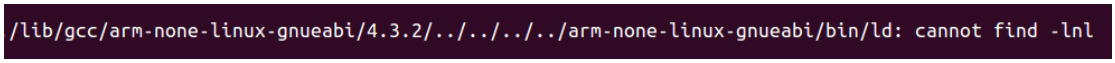
warning: openssl/ssl.h: No such file or directory　报错发现找不到 ssl.h 头文件

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？

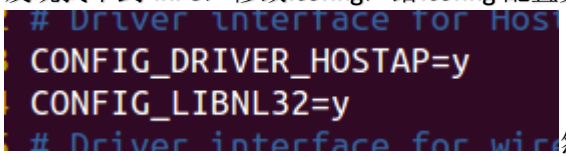


不能把 openssl 目录里面的头文件复制在交叉编译器的 include 目录下，必须把整个 openssl 目录复制过来，ssl 里面的程序是先找 openssl 这个目录，它认这个死理。所以要在 include 目录下看到 openssl 目录，所以我把整个 openssl 目录复制过来问题才得到解决。

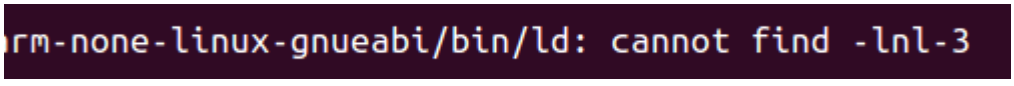
然后再 make



发现找不到-lnl-3，修改.config，给.config 配置文件里面加上 CONFIG_LIBNL32=y



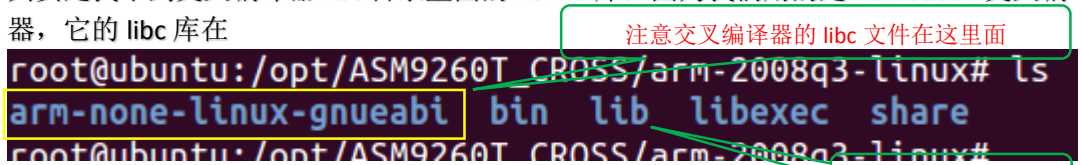
然后在 make



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



```
root@ubuntu:/home/xiang/ASM9260T/wifi_rtl8188eus/hostapd/hostapd-2.0/hostapd# make
  LD  hostapd
  CC  hostapd_cli.c
  CC  ../src/common/wpa_ctrl.c
  CC  ../src/utils/edit_simple.c
  LD  hostapd_cli
```

编译通过

```
hostapd
hostapd.8
hostapd.accept
hostapd_cli
```

你看 hostapd，hostapd_cli 软件编译出来了。

第 2 步: 配置内核

```
[*] Networking support  --->
    Device Drivers  --->
    File systems  --->
[*] Network device support  --->
[*]   Wireless LAN  --->
      *** Enable WiMAX (Networking
      USB Network Adapters  --->
--- Wireless LAN
[ ]   Wireless LAN (pre-802.11)  --->
[*]   Wireless LAN (IEEE 802.11)  --->
```

Wifi 的 AP 模式这个一定要配置上

```
< >   Softmac Prism54 support
< >   Atheros Wireless Cards  --->
<*>   IEEE 802.11 for Host AP (Prism2/2.5/3 and WEP/TKIP/CCMP)
[*]     Support downloading firmware images with Host AP driver
[*]       Support for non-volatile firmware download
```

第 3 步骤：编写没有密码的 hostapd.conf 配置文件

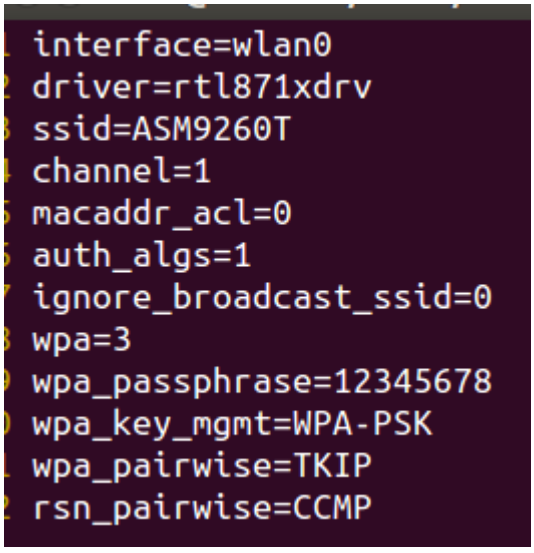
vi hostapd.conf 文件

```
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。*/
```

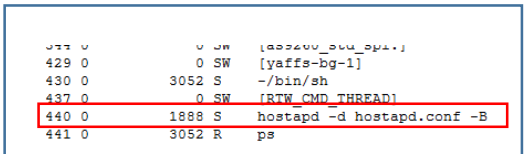
第 3 步骤，选择编写需要密码的 hostapd.conf 配置文件

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
```



配置文件写好后用 hostapd 加载



Hostapd 服务启动成功

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
# udhcpd -f dhcpd.conf
udhcpd (v1.14.4) started
udhcpd: can't open '/var/lib/misc/udhcpd.leases': No such file or directory
```

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

```
458 0          1888 S    hostapd -d hostapd.conf -B
462 0          3048 S    udhcpd -f dhcpd.conf
```

然后我们用手机连接我们的开发板

```
Sending OFFER of 192.168.2.2
Sending OFFER of 192.168.2.2
Sending ACK to 192.168.2.2
Sending OFFER of 192.168.2.3
Sending OFFER of 192.168.2.3
Sending ACK to 192.168.2.3
```

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 192.168.2.3
PING 192.168.2.3 (192.168.2.3): 56 data bytes
64 bytes from 192.168.2.3: seq=0 ttl=64 time=90.000 ms
64 bytes from 192.168.2.3: seq=1 ttl=64 time=70.000 ms
64 bytes from 192.168.2.3: seq=2 ttl=64 time=10.000 ms
64 bytes from 192.168.2.3: seq=3 ttl=64 time=30.000 ms
```

Ping 连接成功，证明手机和开发板通信没有问题了。

```
# hostapd_cli all_sta
Selected interface 'wlan0'
a4:71:74:8f:6c:7d
dot11RSNAStatsSTAAddress=a4:71:74:8f:6c:7d
dot11RSNAStatsVersion=1
dot11RSNAStatsSelectedPairwiseCipher=00-0f-ac-4
dot11RSNAStatsTKIPLocalMICFailures=0
dot11RSNAStatsTKIPRemoteMICFailures=0
hostapdWPAPTKState=11
hostapdWPAPTKGroupState=0
```

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

```
root@ubuntu:/opt/ASM9260T_CROSS/arm-2008q3-linux/lib# ls
gcc                libnl-3.so.200          libnl-genl-3.la
libcrypt-2.8.so    libnl-3.so.200.18.0     libnl-genl-3.so
libcrypt.so.1      libnl-cli-3.a           libnl-genl-3.so
```

```
CHANGES.SSLeay  demos     FAQ          INSTALL.com   INSTALL.VMS  MacOS        MakeVMS.com
config          doc       GitConfigure INSTALL.DJGPP INSTALL.W32  Makefile     ms
Configure       engines   GitMake      INSTALL.MacOS INSTALL.W64  Makefile.bak Netware
CONTRIBUTING    e_os2.h   include      INSTALL.NW    INSTALL.WCE  Makefile.org NEWS
/xiang/ASM9260T/wifi_rtl8188eus/openssl/openssl-1.0.2n# ./config shared no-asm --prefix=$PWD/tmp
```

./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

```
md2test.c => dummytest.c
rc5test.c => dummytest.c
jpaketest.c => dummytest.c
make[1]: Leaving directory `/home/xiang/ASM9260T/wifi_rtl8188eus/openssl/openssl-1.

Configured for linux-x86_64.
```

成功配置

```
  INSTALL.W32   Makefile
  INSTALL.W64   Makefile.bak
  INSTALL.WCE   Makefile.org
2n# vim Makefile
```

去 Makefile 文件里面指定交叉编译器配置

```
CC= gcc
AR= ar $(ARFLAGS) r
NM= nm
MAKEDEPPROG= gcc
```

修改这几个变量

```
CC= arm-none-linux-gnueabi-gcc-4.3.2
CFLAG= -fPIC -DOPENSSL_PIC -DOPENSSL_THREADS -D_
DEPFLAG= -DOPENSSL_NO_EC_NISTP_64_GCC_128 -DOPEN
C3779 -DOPENSSL_NO_SCTP -DOPENSSL_NO_SSL_TRACE -
PEX_LIBS=
EX_LIBS= -ldl
EXE_EXT=
ARFLAGS=
AR= arm-none-linux-gnueabi-ar $(ARFLAGS) r
RANLIB= arm-none-linux-gnueabi-ranlib
RC= windres
NM=  arm-none-linux-gnueabi-nm
PERL= /usr/bin/perl
TAR= tar
TARFLAGS= --no-recursion
MAKEDEPPROG= arm-none-linux-gnueabi-gcc-4.3.2
LIBDIR=lib
```

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"

```
CFLAG= -fPIC -DOPENSSL_PIC -DOPENSSL_THREADS -D_REENTRANT -DDSO_DLFCN -DHAVE_DLFCN_H -m64 -DL_E
DEPFLAG= -DOPENSSL_NO_EC_NISTP_64_GCC_128 -DOPENSSL_NO_GMP -DOPENSSL_NO_JPAKE -DOPENSSL_NO_LIBUN
```

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

```
make[1]: Entering directory `/home/xiang/ASM9260T/wifi_rtl8188eus/openssl/openssl-1.0.2n/tools'
make[1]: Nothing to be done for `all'.
make[1]: Leaving directory `/home/xiang/ASM9260T/wifi_rtl8188eus/openssl/openssl-1.0.2n/tools'
```

编译成功了

make install

```
libcrypto.a     Makefile          Netware
libcrypto.pc    Makefile.bak      NEWS
libssl.a        Makefile.org      openssl.dox
libssl.pc       Makefile.shared   openssl.pc
LICENSE         makevms.com       openssl.spe
MacOS           ms                os2
make INSTALL_PREFIX=$PWD/tmp install
```

```
README.ASN1     tmp
README.ENGINE   tools
rehash.time     util
shlib           VMS
```

在 tmp 目录下

```
root@ubuntu:/home/xiang/ASM9260T/wifi_rtl8188eus/openssl/openssl-1.0.2n/tmp# ls
bin  include  lib  ssl
root@ubuntu:/home/xiang/ASM9260T/wifi_rtl8188eus/openssl/openssl-1.0.2n/tmp#
```

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

```
root@ubuntu:/home/xiang/ASM9260T/wifi_rtl8188eus/openssl/openssl-1.0.2n/tmp/include/openssl# ls
aes.h       bn.h       comp.h     dh.h       ecdsa.h    hmac.h     md5.h      opensslconf.h  pkcs7.h   rsa.h        ssl23.h    tls1.h      x509.h
asn1.h      buffer.h   conf_api.h dsa.h      ec.h       idea.h     mdc2.h     opensslv.h     pqueue.h  safestack.h  ssl2.h     ts.h        x509v3.h
asn1_mac.h  camellia.h conf.h     dso.h      engine.h   krb5_asn.h modes.h    ossl_typ.h     rand.h    seed.h       ssl3.h     txt_db.h    x509_vfy.h
asn1t.h     cast.h     crypto.h   dtls1.h    e_os2.h    kssl.h     objects.h  pem2.h         rc2.h     sha.h        ssl.h      ui_compat.h
bio.h       cmac.h     des.h      ebcdic.h   err.h      lhash.h    obj_mac.h  pem.h          rc4.h     srp.h        stack.h    ui.h
blowfish.h  cms.h      des_old.h  ecdh.h     evp.h      md4.h      ocsp.h     pkcs12.h       ripemd.h  srtp.h       symhacks.h whrlpool.h
root@ubuntu:/home/xiang/ASM9260T/wifi_rtl8188eus/openssl/openssl-1.0.2n/tmp/include/openssl# cp * /opt/ASM9260T_CROSS/arm-2008q3-linux/lib/gcc/arm-none-li
ux-gnueabi/4.3.2/include
```

openssl 里面的头文件 include 复制到交叉编译工具下了

```
root@ubuntu:/home/xiang/ASM9260T/wifi_rtl8188eus/openssl/openssl-1.0.2n/tmp/lib# ls
engines  libcrypto.a  libcrypto.so  libcrypto.so.1.0.0  libssl.a  libssl.so  libssl.so.1.0.0  pkgconfig
root@ubuntu:/home/xiang/ASM9260T/wifi_rtl8188eus/openssl/openssl-1.0.2n/tmp/lib# cp -r * /opt/ASM9260T_CROSS/arm-2008q3-linux/lib
```

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

```
root@ubuntu:/home/xiang/ASM9260T/wifi_rtl8188eus/openssl/openssl-1.0.2n/tmp/lib# ls
engines  libcrypto.a  libcrypto.so  libcrypto.so.1.0.0  libssl.a  libssl.so  libssl.so.1.0.0  pkgconfig
root@ubuntu:/home/xiang/ASM9260T/wifi_rtl8188eus/openssl/openssl-1.0.2n/tmp/lib# cp -r * /home/xiang/ASM9260T/rootfs/_rootfs/roms/lib
```

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

自己写 wifi 应用程序

实现自动扫描 wifi 热点

```
/wifi# cp -r wpa_supplicant-2.2 /home/xiang/IMX6/wifi/mywpa
```

单独拷贝一份 wpa_supplicant-2.2 工程给自己，因为我们只有在 wpa_cli.c 下面修改。

所以我们要保证 wpa_supplicant-2.2 文件已经是编译过的，在开发板上跑过默认程序的

```
/wpa_supplicant-2.2/wpa_supplicant# vim wpa_cli.c
```

主要是修改这个 wpa_cli.c 文件

```
3752 int main(int argc, char *argv[])
3753 {
3754         int c;
3755         int daemonize = 0;
3756         int ret = 0;
3757         const char *global = NULL;
3758
3759         if (os_program_init())
3760                 return -1;
3761
3762         for (;;) {
3763                 c = getopt(argc, argv, "a:Bg:G:hi:p:P:v");
3764                 if (c < 0)
3765                         break;
3766                 switch (c) {
3767                 case 'a':
3768                         action_file = optarg;
3769                         break;
3770                 case 'B':
3771                         daemonize = 1;
3772                         break;
3773                 case 'g':
3774                         global = optarg;
3775                         break;
3776                 case 'G':
3777                         ping_interval = atoi(optarg);
3778                         break;
3779                 case 'h':
3780                         usage();
3781                         return 0;
3782                 case 'v':
3783                         printf("%s\n", wpa_cli_version);
3784                         return 0;
```

原版的 main 函数有很多功能，但是我们要不了这么多，所以我裁剪了很多

```
3751
3752 int main(int argc, char *argv[])
3753 {
3754         int c;
3755         int daemonize = 0;
3756         int ret = 0;
3757         const char *global = NULL;
3758
3759         if (os_program_init())
3760                 return -1;
3761
3762
3763         if (ctrl_ifname == NULL)
3764                 ctrl_ifname = wpa_cli_get_default_ifname();
3765
3766         if(wpa_cli_open_connection(ctrl_ifname, 0)<0){
3767                 fprintf(stderr, "Failed to connect to non-global "
3768                         "ctrl_ifname: %s  error: %s\n",
3769                 ctrl_ifname, strerror(errno));
3770                 return -1;
3771         }
3772         ret = wpa_request(ctrl_conn, argc - optind,&argv[optind]);
3773         while(1)
3774         {
3775                 /*scan*/
3776
3777         }
3778         os_free(ctrl_ifname);
3779         wpa_cli_cleanup();
3780
3781         return ret;
3782 }
```

这是我裁剪后的 main 程序，就这么多我们需要

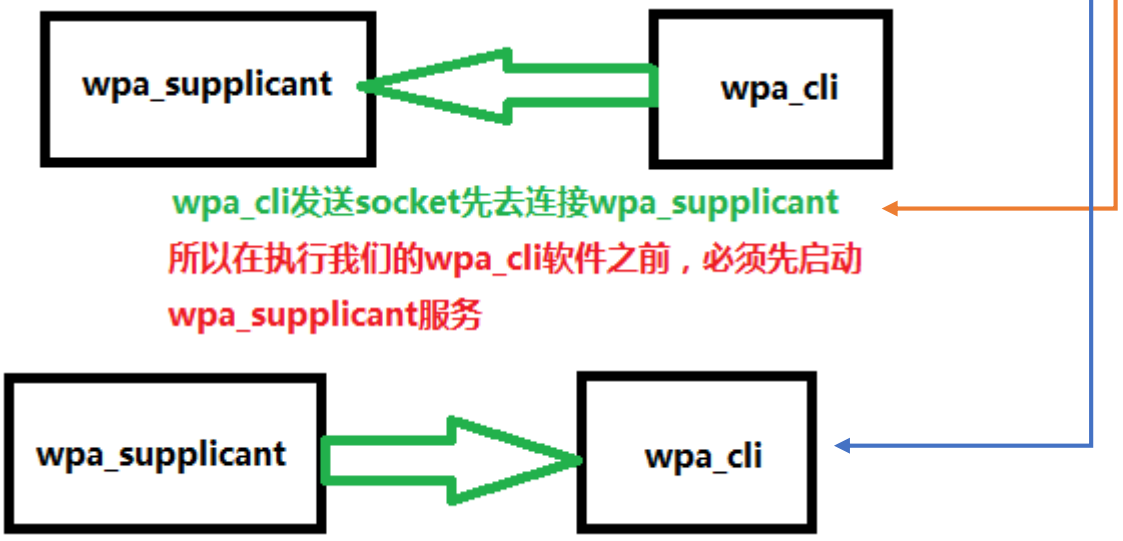
保留 wpa 初始化函数

我们没有指定哪一个网卡节点，所以它给我选择默认网卡

这一段很重要，是 wpa_cli 去连接 wpa_supplicant

Wpa_request 就是获取 wpa_supplicant 返回给 wpa 的数据

我们主要在这里写查询 wifi 热点和信号的程序



```
3753 int main(int argc, char *argv[])
3754 {
3755         int c;
3756         int daemonize = 0;
3757         int ret = 0;
3758         const char *global = NULL;
3759
3760         char* recevie_char[10];
3761         if (os_program_init())
3762                 return -1;
3763
3764
3765         if (ctrl_ifname == NULL)
3766                 ctrl_ifname = wpa_cli_get_default_ifname();
3767
3768         if(wpa_cli_open_connection(ctrl_ifname, 0)<0){
3769                 fprintf(stderr, "Failed to connect to non-global "
3770                                 "ctrl_ifname: %s  error: %s\n",
3771                 ctrl_ifname, strerror(errno));
3772                 return -1;
3773         }
3774         while(1)
3775         {
3776                 /*scan*/
3777                 recevie_char[0]="scan";
3778                 ret = wpa_request(ctrl_conn, 1,recevie_char);
3779                 if(ret)
3780                 {
3781                         continue;
3782                 }
3783                 recevie_char[0]="scan_results";
3784                 ret = wpa_request(ctrl_conn, 1,recevie_char);
3785                 printf("scan_results:\n %s \n",my_buf);
3786                 sleep(2);
3787         }
3788         os_free(ctrl_ifname);
3789         wpa_cli_cleanup();
```

定义个二维数组来发送命令给 wpa，这里的数组格式是 “XXXXXX” , ”XXXXXX” ….

我就先给数组 0 发送 scan 命令进去

这里写 1

这里不变

这里就是我们传入的 scan 字符串命令

```
3089 static int wpa_request(struct wpa_ctrl *ctrl, int argc, char *argv[])
3090 {
3091         struct wpa_cli_cmd *cmd, *match = NULL;
3092         int count;
3093         int ret = 0;
3094
3095         if (argc > 1 && os_strncasecmp(argv[0], "IFNAME=", 7) == 0) {
3096                 ifname_prefix = argv[0] + 7;
3097                 argv = &argv[1];
3098                 argc--;
3099         } else
3100                 ifname_prefix = NULL;
3101
3102         if (argc == 0)
3103                 return -1;
3104
3105         count = 0;
3106         cmd = wpa_cli_commands;
3107         while (cmd->cmd) {
3108                 if (os_strncasecmp(cmd->cmd, argv[0], os_strlen(argv[0])) == 0)
```

得到数组的字符数据去执行这个回调函数

```
2473 static struct wpa_cli_cmd wpa_cli_commands[] = {
2474         { "status", wpa_cli_cmd_status, NULL,
2475           cli_cmd_flag_none,
2476           "[verbose] = get current WPA/EAPOL/EAP status" },
2477         { "ifname", wpa_cli_cmd_ifname, NULL,
2478           cli_cmd_flag_none,
2479           "= get current interface name" },
2480         { "ping", wpa_cli_cmd_ping, NULL,
```

所以我们的 scan 字符串就是去匹配这里的数组项

这里有很多数组项目，经过我查阅是匹配的下面两个

```
2622         { "scan", wpa_cli_cmd_scan, NULL,
2623           cli_cmd_flag_none,
2624           "= request new BSS scan" },
2625         { "scan_results", wpa_cli_cmd_scan_results, NULL,
2626           cli_cmd_flag_none,
2627           "= get latest scan results" },
```

主函数的这段代码匹配的数组 scan

```
/*scan*/
recevie_char[0]="scan";
ret = wpa_request(ctrl_conn, 1,recevie_char);
if(ret)
{
        continue;
}
```

scan 传入进去

```
{ "scan", wpa_cli_cmd_scan, NULL,
  cli_cmd_flag_none,
  "= request new BSS scan" },
```

导致 wpa_cli_cmd_scan 函数被执行

```
1503 static int wpa_cli_cmd_scan(struct wpa_ctrl *ctrl, int argc, char *argv[])
1504 {
1505         return wpa_cli_cmd(ctrl, "SCAN", 0, argc, argv);
1506 }
```

这个函数就是命令 wpa 执行扫描操作的

然后 wpa 扫描的结果会存放在一个缓冲区

然后我主函数在发送获取缓冲区数据的命令

获取扫描结果缓冲区数据的命令

```
recevie_char[0]="scan_results";
ret = wpa_request(ctrl_conn, 1,recevie_char);
```

```
2625         { "scan_results", wpa_cli_cmd_scan_results, NULL,
2626           cli_cmd_flag_none,
2627           "= get latest scan results" },
1509 static int wpa_cli_cmd_scan_results(struct wpa_ctrl *ctrl, int argc,
1510                                     char *argv[])
1511 {
1512         return wpa_ctrl_command(ctrl, "SCAN_RESULTS");
1513 }
```

这个函数要修改

```
435 static int wpa_ctrl_command(struct wpa_ctrl *ctrl, char *cmd)
436 {
437         return _wpa_ctrl_command(ctrl, cmd, 1);
438 }
```

这个官方的函数，这里写 1 是要函数内部 printf 打印，但是我们不想函数内部打印，打印由我们来操作所以这里改成 0

```
436 static int wpa_ctrl_command(struct wpa_ctrl *ctrl, char *cmd)
437 {
438         return _wpa_ctrl_command(ctrl, cmd, 0);
439 }
```

```
399 static int _wpa_ctrl_command(struct wpa_ctrl *ctrl, char *cmd, int print)
400 {
401         char buf[4096];
402         size_t len;
403         int ret;
404
405         if (ctrl_conn == NULL) {
406                 printf("Not connected to wpa_supplicant - command dropped.\n");
407                 return -1;
408         }
409         if (ifname_prefix) {
410                 os_snprintf(buf, sizeof(buf), "IFNAME=%s %s",
411                             ifname_prefix, cmd);
412                 buf[sizeof(buf) - 1] = '\0';
413                 cmd = buf;
414         }
415         len = sizeof(buf) - 1;
416         ret = wpa_ctrl_request(ctrl, cmd, os_strlen(cmd), buf, &len,
417                                wpa_cli_msg_cb);
418         if (ret == -2) {
419                 printf("'%s' command timed out.\n", cmd);
420                 return -2;
421         } else if (ret < 0) {
422                 printf("'%s' command failed.\n", cmd);
423                 return -1;
424         }
425         if (print) {
426                 buf[len] = '\0';
427                 printf("%s", buf);
428                 if (interactive && len > 0 && buf[len - 1] != '\n')
429                         printf("\n");
430         }
431         return 0;
```

这是官方的_wpa_ctrl_command 函数，它是存放获取的热点数据

然后在函数内部打印出来，但是我的 print 传入的是 0，所以不会在函数内部打印

这样这个官方的_wpa_ctrl_command 要修改

```c
char my_buf[4096];
size_t my_len;
static int _wpa_ctrl_command(struct wpa_ctrl *ctrl, char *cmd, int print)
{
        int ret;

        if (ctrl_conn == NULL) {
                printf("Not connected to wpa_supplicant - command dropped.\n");
                return -1;
        }
        if (ifname_prefix) {
                os_snprintf(my_buf, sizeof(my_buf), "IFNAME=%s %s",
                            ifname_prefix, cmd);
                my_buf[sizeof(my_buf) - 1] = '\0';
                cmd = my_buf;
        }
        my_len = sizeof(my_buf) - 1;
        ret = wpa_ctrl_request(ctrl, cmd, os_strlen(cmd), my_buf, &my_len,
                               wpa_cli_msg_cb);
        if (ret == -2) {
                printf("'%s' command timed out.\n", cmd);
                return -2;
        } else if (ret < 0) {
                printf("'%s' command failed.\n", cmd);
                return -1;
        }
        my_buf[my_len] = '\0';
        if (print) {
                my_buf[my_len] = '\0';
                printf("%s", my_buf);
                if (interactive && my_len > 0 && my_buf[my_len - 1] != '\n')
                        printf("\n");
        }
        return 0;
}
```

我们将存放数据的 buf 定义成全局的，长度变量也定义成全局的，这样主函数好调用

这些 buf 还有 len 都换成全局变量的名字

```c
int main(int argc, char *argv[])
{
        int c;
        int daemonize = 0;
        int ret = 0;
        const char *global = NULL;

        char* recevie_char[10];
        if (os_program_init())
                return -1;


        if (ctrl_ifname == NULL)
                ctrl_ifname = wpa_cli_get_default_ifname();

        if(wpa_cli_open_connection(ctrl_ifname, 0)<0){
                fprintf(stderr, "Failed to connect to non-global "
                        "ctrl_ifname: %s  error: %s\n",
                ctrl_ifname, strerror(errno));
                return -1;
        }
        while(1)
        {
                /*scan*/
                recevie_char[0]="scan";
                ret = wpa_request(ctrl_conn, 1,recevie_char);
                if(ret)
                {
                        continue;
                }
                recevie_char[0]="scan_results";
                ret = wpa_request(ctrl_conn, 1,recevie_char);
                printf("scan_results:\n %s \n",my_buf);
                sleep(2);
        }
        os_free(ctrl_ifname);
        wpa_cli_cleanup();
```

再回顾一下主函数

这个 wifi 热点查询程序就编写完了，保存退出执行 make

```
root@ubuntu:/home/xiang/IMX6/wifi/mywpa/wpa_supplicant-2.2/wpa_supplicant# make
  CC  wpa_cli.c
  CC  ../src/common/wpa_ctrl.c
  CC  ../src/utils/edit_simple.c
  LD  wpa_cli
  CC  wpa_passphrase.c
  LD  wpa_passphrase
root@ubuntu:/home/xiang/IMX6/wifi/mywpa/wpa_supplicant-2.2/wpa_supplicant# ls
```

在当前目录下会生成新的 wpa_cli 软件

```
Makefile          scan.c          wnm_sta.h
my_cli            scan.h          wpa_cli
nfc_pw_token.c    scan.o          wpa_cli.c
```

我将 wpa_cli 拷贝成 my_cli，这样要区分，下载进开发板调试

因为我们要先启动 wpa_supplicant 服务，所以我编写一个不连接热点的配置文件

wpa_supplicant.conf

```
1 ctrl_interface=/var/run/wpa_supplicant
```

就写这么一句就可以了

我是 AP6261WIFI 模块所以启动 wifi 前，先写入脚本

echo –n “/etc/firmware/fw_bcmd43438a0.bin” >/sys/module/bcmdhd/parameters/firmware_path

ifconfig wlan0 up //启动 wifi

```
root@imx6qdlsolo:/mnt# wpa_supplicant -iwlan0 -c wpa_supplicant.conf -B
```

将我写的 wpa_supplicant.conf 配置文件启动到后台

```
root@imx6qdlsolo:/mnt# ls /var/run/wpa_supplicant/ -l
total 0
srwxrwx--- 1 root root 0 Mar 20 15:53 wlan0
```

查看到 wlan0 和 wpa_supplicant 链接运行起来了

```
root@imx6qdlsolo:/mnt# ./my_cli
Selected interface 'wlan0'
scan_results:
 bssid / frequency / signal level / flags / ssid

scan_results:
 bssid / frequency / signal level / flags / ssid
08:10:79:fa:fc:e2       2437    -44     [WPA-PSK-CCMP][WPA2-PSK-CCMP][ESS]      UNIACLTK
04:95:e6:1b:e5:11       2462    -62     [WPA-PSK-CCMP][WPA2-PSK-CCMP][WPS][ESS] Chinanet-2.4G-E510
84:d9:31:72:fc:30       2412    -63     [WPA2-PSK-CCMP][ESS]    TSARIMAKERSPACE
d4:ee:07:51:16:42       2412    -71     [WPA-PSK-CCMP][WPA2-PSK-CCMP][ESS]      cdatatest
50:da:00:11:0b:b0       2437    -74     [WPA2-PSK-CCMP][ESS]    TSARIMAKERSPACE
84:d9:31:63:fe:50       2462    -80     [WPA2-PSK-CCMP][ESS]    TSARIMAKERSPACE
84:d9:31:64:04:10       2437    -81     [WPA2-PSK-CCMP][ESS]    TSARIMAKERSPACE
84:d9:31:72:f2:50       2462    -81     [WPA2-PSK-CCMP][ESS]    TSARIMAKERSPACE
dc:fe:18:ab:18:92       2412    -82     [WPA-PSK-CCMP+TKIP][WPA2-PSK-CCMP+TKIP][ESS]    TP-LINK_1892
a0:8c:fd:3d:5f:64       2437    -83     [WPA2-PSK-CCMP][WPS][ESS]       DIRECT-63-HP DeskJet 4530 serie
b8:f8:83:3d:69:56       2412    -85     [WPA-PSK-CCMP][WPA2-PSK-CCMP][ESS]      TP-LINK_F661
50:da:00:11:14:10       2437    -89     [WPA2-PSK-CCMP][ESS]    TSARIMAKERSPACE
84:d9:31:72:f0:d0       2462    -91     [WPA2-PSK-CCMP][ESS]    TSARIMAKERSPACE
1c:15:1f:46:8d:50       2452    -92     [WPA-PSK-CCMP][WPA2-PSK-CCMP][WPS][ESS] WIFI-Receiver1
94:d9:b3:41:60:4b       2462    -94     [WPA-PSK-CCMP][WPA2-PSK-CCMP][ESS]      BY
```

在执行 my_cli，热点就经过 printf 打印出来了。

获取 wifi 多个热点的其中一个热点去连接路由器，这个技术也可以对多个 wifi 热点进行存储

```
        while(1)
        {
                /*scan*/
                recevie_char[0]="scan";
                ret = wpa_request(ctrl_conn, 1,recevie_char);
                if(ret)
                {
                        continue;
                }
                recevie_char[0]="scan_results";
                ret = wpa_request(ctrl_conn, 1,recevie_char);
                printf("scan_results:\n %s \n",my_buf);
                sleep(2);
        }
        os_free(ctrl_ifname);
        wpa_cli_cleanup();
```

主要是解析 my_buf 里面的数据

这个程序主要是基于前面扫描 wifi 热点程序进行修改

S	S	I	D		/r	S	I	N	G	L	/r	X	X	Z	Z	/n	S	S	I	D	A	P		/r	/n				

my_buf 的 wifi 数据格式，一个 buf 里面有多个 wifi 热点，相互之间在 buf 里面写/r/n 隔离开，所以你发现一个一维数组也能打印出几行 wifi 热点，就是这种排列方式

所以我们要写一个解析 buf 的函数

```
static int get_line_from_buf(int index,char *buf,char *line)
{
        int i=0,j=0;
        int endcnt = -1;
        char* linestart = buf;
        int len;
        while(1)
        {
                if(buf[i] == '\n' || buf[i] == '\r' || buf[i] == '\0')
                {
                        endcnt++;
                        if(index == endcnt)
                        {
                                len = &buf[i] - linestart;
                                strncpy(line,linestart,len);
                                line[len] = '\0';
                                return 0;
                        }
                        else
                        {
                                for(j=i+1;buf[j];)
                                {
                                        if(buf[j] == '\n' || buf[j] == '\r' || buf[j] == '\0')
                                        {
                                                j++;
                                        }
                                        else
                                        {
                                                break;
                                        }
                                }
                                if(!buf[j])
                                {
                                        return -1;
                                }
                                linestart = &buf[j];
                                i=j;
                        }
                }
                if(!buf[i])
                {
                        return -1;
                }
                i++;
        }
}
```

index 就是给每一行 line 加上对应的序号

将 mybuf 的数据一行一行解析出来，每一行复制给 line，主程序读取一次 line 就是读取一行数据

1.一般 buf 数组第 1 个字符都不是换行符什么的

3.循环几次发现到了 buf 的\n,\0 处。然后将该处的地址减去 buf 的首地址，得到这行字符串的长度

4.从 buf 首地址开始循环拷贝字符串给 line，一直拷贝到现在停下来的 buf 地址

5.line 得到这行数据后，最后加个\0 表示该行结束

2.所以 i++寻找 buf 后面的字符

```
3800 int main(int argc, char *argv[])
3801 {
3802         int c,i;
3803         int daemonize = 0;
3804         int ret = 0;
3805         const char *global = NULL;
3806
3807         char* recevie_char[10];
3808         char xline[1024];
```

主函数申请个 xline 来获取行数据

```
while(1)
{
        /*scan*/
        recevie_char[0]="scan";
        ret = wpa_request(ctrl_conn, 1,recevie_char);
        if(ret)
        {
                continue;
        }
        recevie_char[0]="scan_results";
        ret = wpa_request(ctrl_conn, 1,recevie_char);

        for(i = 1;!get_line_from_buf(i,my_buf,xline);i++)
        {
                printf("line %d = %s \n",i,xline);
        }
        sleep(2);
}
```

如果传入的 i 有对应的行，就继续循环

```
root@imx6qdlsolo:/mnt# ./ap_cli
Selected interface 'wlan0'
line 1 = 08:10:79:fa:fc:e2      2437    -37     [WPA-PSK-CCMP][WPA2-PSK-CCMP][ESS]      UNIACLTK
line 2 = 04:95:e6:1b:e5:11      2412    -50     [WPA-PSK-CCMP][WPA2-PSK-CCMP][WPS][ESS] Chinanet-2.4G-E510
line 3 = 84:d9:31:72:fc:30      2412    -66     [WPA2-PSK-CCMP][ESS]    TSARIMAKERSPACE
line 4 = d4:ee:07:51:16:42      2412    -71     [WPA-PSK-CCMP][WPA2-PSK-CCMP][ESS]      cdatatest
line 5 = 50:da:00:11:0b:b0      2437    -74     [WPA2-PSK-CCMP][ESS]    TSARIMAKERSPACE
line 6 = 84:d9:31:72:f2:50      2462    -78     [WPA2-PSK-CCMP][ESS]    TSARIMAKERSPACE
line 7 = 84:d9:31:63:fe:50      2462    -74     [WPA2-PSK-CCMP][ESS]    TSARIMAKERSPACE
line 8 = 50:da:00:11:14:10      2437    -85     [WPA2-PSK-CCMP][ESS]    TSARIMAKERSPACE
line 9 = dc:fe:18:ab:18:92      2412    -81     [WPA-PSK-CCMP+TKIP][WPA2-PSK-CCMP+TKIP][ESS]    TP-LINK_1892
line 10 = 84:d9:31:64:04:10     2437    -84     [WPA2-PSK-CCMP][ESS]    TSARIMAKERSPACE
line 11 = a0:8c:fd:3d:5f:64     2437    -78     [WPA2-PSK-CCMP][WPS][ESS]       DIRECT-63-HP DeskJet 4530 series
line 12 = 84:d9:31:72:f1:d0     2437    -80     [WPA2-PSK-CCMP][ESS]    TSARIMAKERSPACE
line 13 = ec:3d:fd:ed:06:c2     2467    -95     [ESS]   B-LINK-2.4G_ED06C2
line 14 = b8:f8:83:3d:69:56     2412    -81     [WPA-PSK-CCMP][WPA2-PSK-CCMP][ESS]      TP-LINK_F661
line 15 = 84:d9:31:72:f0:f0     2462    -88     [WPA2-PSK-CCMP][ESS]    TSARIMAKERSPACE
line 16 = 84:d9:31:72:f0:d0     2462    -91     [WPA2-PSK-CCMP][ESS]    TSARIMAKERSPACE
line 17 = 1c:15:1f:46:8d:50     2427    -86     [WPA-PSK-CCMP][WPA2-PSK-CCMP][WPS][ESS] WIFI-Receiver1
line 18 = f4:83:cd:38:fd:91     2412    -95     [WPA-PSK-CCMP][WPA2-PSK-CCMP][ESS]      15-6
line 1 = 08:10:79:fa:fc:e2      2437    -38     [WPA-PSK-CCMP][WPA2-PSK-CCMP][ESS]      UNIACLTK
line 2 = 04:95:e6:1b:e5:11      2412    -49     [WPA-PSK-CCMP][WPA2-PSK-CCMP][WPS][ESS] Chinanet-2.4G-E510
line 3 = 84:d9:31:72:fc:30      2412    -64     [WPA2-PSK-CCMP][ESS]    TSARIMAKERSPACE
line 4 = d4:ee:07:51:16:42      2412    -72     [WPA-PSK-CCMP][WPA2-PSK-CCMP][ESS]      cdatatest
```

这就将每行热点都对应上了序号

如果我只想获取每行热点的某几段怎么办？

```
3808        char xline[1024];
3809        char bssid[1024];
3810        char freq[1024];
3811        char signal[1024];
3812        char flags[1024];
3813        char ssid[1024];
3814        if (os_program_init())
3815                return -1;
3816
3817
3818        if (ctrl_ifname == NULL)
3819                ctrl_ifname = wpa_cli_get_default_ifname();
3820
3821        if(wpa_cli_open_connection(ctrl_ifname, 0)<0){
3822                fprintf(stderr, "Failed to connect to non-global "
3823                                "ctrl_ifname: %s  error: %s\n",
3824                ctrl_ifname, strerror(errno));
3825                return -1;
3826        }
3827        while(1)
3828        {
3829                /*scan*/
3830                recevie_char[0]="scan";
3831                ret = wpa_request(ctrl_conn, 1,recevie_char);
3832                if(ret)
3833                {
3834                        continue;
3835                }
3836                recevie_char[0]="scan_results";
3837                ret = wpa_request(ctrl_conn, 1,recevie_char);
3838
3839                for(i = 1;!get_line_from_buf(i,my_buf,xline);i++)
3840                {
3841                        sscanf(xline,"%s %s %s %s %s",bssid,freq,signal,flags,ssid);
3842                        printf("line %d =  %s  %s \n",i,ssid,signal);
3843                }
3844                sleep(2);
3845        }
```

先申请每行热点获取某几段的存放空间

将 xline 一行数据的每段取出来赋值给对应的数组存放

最后只显示我想要的段

```
t# cp wpa_cli ap_cli
```

因为修改的还是 wpa_cli，所以改一个我熟悉的名字

按照实现自动扫描 wifi 热点的方法启动 wifi，执行 wpa_supplicant 后台服务

```
root@imx6qdlsolo:/mnt# ./ap_cli
Selected interface 'wlan0'
line 1 =  UNIACLTK  -38
line 2 =  TSARIMAKERSPACE  -59
line 3 =  Chinanet-2.4G-E510  -60
line 4 =  cdatatest  -70
line 5 =  TSARIMAKERSPACE  -75
line 6 =  TSARIMAKERSPACE  -77
line 7 =  TSARIMAKERSPACE  -80
line 8 =  TSARIMAKERSPACE  -82
line 9 =  TSARIMAKERSPACE  -83
line 10 =  DIRECT-63-HP  -84
line 11 =  \xe4\xbf\x9d\xe5\x88\xa9\xe7\x89\xa9\xe4\xb8\x9a\xe4\xba\x91WIFI-1  -87
line 12 =  \xe4\xbf\x9d\xe5\x88\xa9\xe7\x89\xa9\xe4\xb8\x9a\xe4\xba\x91WIFI  -87
line 1 =  UNIACLTK  -39
line 2 =  TSARIMAKERSPACE  -60
line 3 =  Chinanet-2.4G-E510  -61
line 4 =  cdatatest  -70
line 5 =  TSARIMAKERSPACE  -76
line 6 =  TSARIMAKERSPACE  -80
line 7 =  TSARIMAKERSPACE  -78
line 8 =  TSARIMAKERSPACE  -82
line 9 =  TSARIMAKERSPACE  -81
```

这就实现了读取你指定的 SSID 对应信号强度的程序

所以获取每一行 SSID 信号 加密 mac 的代码就是解析 my_buf 里面的字符串

这样你就可以把你获取的每行字符串写入创建的文件供其它开发者读取

选中 SSID 链接路由器网络，实现 WPA/WPA2 加密连接

```
799 int main(int argc, char *argv[])
800 {
801         int c,i,addnum;
802         int daemonize = 0;
803         int ret = 0;
804         const char *global = NULL;
805
806         char* recevie_char[10];
807         char xline[1024];
808         char bssid[1024];
809         char freq[1024];
810         char signal[1024];
811         char flags[1024];
812         char ssid[1024];
813         char str[10];
814         if (os_program_init())
815                 return -1;
816
817
818         if (ctrl_ifname == NULL)
819                 ctrl_ifname = wpa_cli_get_default_ifname();
820
821         if(wpa_cli_open_connection(ctrl_ifname, 0)<0){
822                 fprintf(stderr, "Failed to connect to non-global "
823                         "ctrl_ifname: %s  error: %s\n",
824                 ctrl_ifname, strerror(errno));
825                 return -1;
826         }
3827     while(1)
3828     {
3829
3830
3831                 recevie_char[0]="add_network";
3832                 ret = wpa_request(ctrl_conn, 1,recevie_char);
3833                 if(ret)
3834                 {
3835                         continue;
3836                 }
3837                 sscanf(my_buf,"%d",&addnum);
3838                 sprintf(str,"%d",addnum);
3839                 recevie_char[0]="set_network";
3840                 recevie_char[1]=str;
3841                 recevie_char[2]="ssid";
3842                 recevie_char[3]="\"TSARIMAKERSPACE\"";
3843                 ret = wpa_request(ctrl_conn, 4,recevie_char);
3844                 if(ret)
3845                 {
3846                         continue;
3847                 }
3848                 recevie_char[0]="set_network";
3849                 recevie_char[1]=str;
3850                 recevie_char[2]="psk";
3851                 recevie_char[3]="\"20160928\"";
3852                 ret = wpa_request(ctrl_conn, 4,recevie_char);
3853                 if(ret)
3854                 {
3855                         continue;
3856                 }
3857
3858                 recevie_char[0]="enable_network";
3859                 recevie_char[1]=str;
3860                 ret = wpa_request(ctrl_conn, 2,recevie_char);
```

Str 是用来将整型转换成字符型的变量

wpa_链接默认网卡的套路不变 wlan0

我们要链接路由器，就必须给板子新增加一个网络序号

在 wpa_cli 交互模式下 list_network 就显示了加载的网络序号

```
> list_network
network id / ssid / bssid / flags
0               any     [DISABLED]
1               any     [DISABLED]
2       TSARIMAKERSPACE any     [CUR
>
```

这里写 1 因为你传入的 recevie_char[]数组只有元素 0

用 sprintf 将序号转换成字符串

这里一定是填写字符串，千万不要把格式搞错了

因为我不知道 add_network 随机申请的序号是多少，所以我马上将申请的序号获取出来，序号存在 mybuf 里面,mybuf 是什么时候申请的，请看前面自己写 wifi 扫描程序章节

set_network 就是设置 SSID，这里是 4 个数组元素传入，所以写 4

set_network 同样的方法设置密码 psk

注意 SSID 和密码的字符串引号必须是这种"\"形式

将序列号，SSID 和密码加载进网络列表

```
> list_network
network id / ssid / bssid / flags
0               any     [DISABLED]
1               any     [DISABLED]
2       TSARIMAKERSPACE any     [CURRENT]
>
```

加载进网络列表的内容就是在 list_network 的内容

```
3861            if(ret)
3862            {
3863                    continue;
3864            }
3865
3866            recevie_char[0]="select_network";
3867            recevie_char[1]=str;
3868            ret = wpa_request(ctrl_conn, 2, recevie_char);
3869            if(ret)
3870            {
3871                    continue;
3872            }
3873            return 0;
3874        }
3875        os_free(ctrl_ifname);
3876        wpa_cli_cleanup();
3877
3878        return ret;
3879 }
```

启动 wifi 链接

这启动链接哪一个 SSID 的 wifi 完全靠这个序列号来指定

然后你执行 udhcpc –i wlan0 就能自动获取 ip 地址

然后你 ping 百度就没有问题了

流程整理

```
root@imx6qdlsolo:~# ifconfig wlan0 up
```

加载 wifi 固件，启动板子 wifi 网卡

```
root@imx6qdlsolo:/mnt# wpa_supplicant -iwlan0 -c wpa_supplicant.conf -B
```

启动 wpa_supplicant 服务

以前链接 wifi 在 wpa_supplicant 里面要加入 SSID 和密码，现在不用了，因为我是代码直接写入，所以这里只需要启动 ctrl_interface 就 OK 了

wpa_supplicant.conf

```
1 ctrl_interface=/var/run/wpa_supplicant
```

执行我写的 wifi 链接程序，也就是我上面写的代码

```
root@imx6qdlsolo:/mnt# ./connect_wpa
Selected interface 'wlan0'
root@imx6qdlsolo:/mnt# Connectting with 84:d9:31:72:fc:30 channel (1) ssid "TSARIMAKERSPACE", len (15)

wl_bss_connect_done succeeded with 84:d9:31:72:fc:30
wl_bss_connect_done succeeded with 84:d9:31:72:fc:30

root@imx6qdlsolo:/mnt# udhcpc -i wlan0
udhcpc (v1.22.1) started
Sending discover...
Sending select for 172.16.1.201...
Lease of 172.16.1.201 obtained, lease time 7200
/etc/udhcpc.d/50default: Adding DNS 221.5.203.98
root@imx6qdlsolo:/mnt# ping www.baidu.com
PING www.baidu.com (61.135.169.121): 56 data bytes
```

链接成功

开发板 wifi 与路由器网络断开检测，重新连接成功检测

```
3818        if (ctrl_ifname == NULL)
3819                ctrl_ifname = wpa_cli_get_default_ifname();
3820
3821        if(wpa_cli_open_connection(ctrl_ifname, 0)<0){
3822                fprintf(stderr, "Failed to connect to non-global "
3823                        "ctrl_ifname: %s  error: %s\n",
3824                ctrl_ifname, strerror(errno));
3825                return -1;
3826        }
3827        while(1)
3828        {
3829
3830
3831                recevie_char[0]="add_network";
3832                ret = wpa_request(ctrl_conn, 1,recevie_char);
3833                if(ret)
3834                {
3835                        continue;
3836                }
3837                sscanf(my_buf,"%d",&addnum);
3838                sprintf(str,"%d",addnum);
3839                recevie_char[0]="set_network";
3840                recevie_char[1]=str;
3841                recevie_char[2]="ssid";
3842                recevie_char[3]="\"OP3\"";
3843                ret = wpa_request(ctrl_conn, 4,recevie_char);
3844                if(ret)
3845                {
3846                        continue;
3847                }
3848                recevie_char[0]="set_network";
3849                recevie_char[1]=str;
3850                recevie_char[2]="psk";
3851                recevie_char[3]="\"123456789\"";
3852                ret = wpa_request(ctrl_conn, 4,recevie_char);
3853                if(ret)
3854                {
3855                        continue;
3856                }
3857
```

链接 wifi 的程序和上一章节一样，我只是修改了下路由器 SSID 和密码，因为 OP3 路由器可以关机开机

```
3858            recevie_char[0]="enable_network";
3859            recevie_char[1]=str;
3860            ret = wpa_request(ctrl_conn, 2,recevie_char);
3861            if(ret)
3862            {
3863                    continue;
3864            }
3865
3866            recevie_char[0]="select_network";
3867            recevie_char[1]=str;
3868            ret = wpa_request(ctrl_conn, 2,recevie_char);
3869            if(ret)
3870            {
3871                    continue;
3872            }
3873            while(1)
3874            {
3875                    recevie_char[0]="status";
3876                    ret = wpa_request(ctrl_conn, 1,recevie_char);
3877                    if(ret)
3878                    {
3879                            continue;
3880                    }
3881                    if(strstr(my_buf,"COMPLETED"))
3882                    {
3883                            printf("connect wifi success ...\n");
3884                    }
3885                    else
3886                    {
3887                            printf("connect wifi failed...\n");
3888                    }
3889
3890                    sleep(5);
3891            }
3892            return 0;
3894        os_free(ctrl_ifname);
3895        wpa_cli_cleanup();
3896
3897        return ret;
3898 }
```

网络连接后主要是发送 status 命令，然后会返回一堆字符串给我定义的全局缓存区 my_buf

我们判断 my_buf 里面其中一段字符是不是"COMPLETED"，如果是证明连接 wifi 成功

如果不是证明连接 wifi 失败

测试结果

```
root@imx6qdlsolo:/mnt# ./wifistatus
Selected interface 'wlan0'
connect wifi failed...
Connectting with 94:65:2d:21:be:47 channel (1) ssid "OP3", len (3)

wl_bss_connect_done succeeded with 94:65:2d:21:be:47
wl_bss_connect_done succeeded with 94:65:2d:21:be:47

connect wifi success ...

connect wifi success ...
```

```
connect failed event=0 e->status 1 e->reason 0
CFG80211-ERROR) wl_bss_connect_done : Report connect result - connection failed
connect wifi failed...
connect wifi failed...
connect wifi failed...
connect wifi failed...
connect wifi failed...
Connectting with 94:65:2d:21:be:47 channel (11) ssid "OP3", len (3)

wl_bss_connect_done succeeded with 94:65:2d:21:be:47
wl_bss_connect_done succeeded with 94:65:2d:21:be:47
connect wifi success ...
connect wifi success ...
connect wifi success ...
connect wifi success ...
```

完全没问题，断开后还可以自动连接上

我们还可以将我们连接的路由器 SSID 名字获取出来

```
root@imx6qdlsolo:/mnt# ./wifistatus
Selected interface 'wlan0'
connect wifi failed...
Connectting with 94:65:2d:21:be:47 channel (1) ssid "OP3", len (3)

wl_bss_connect_done succeeded with 94:65:2d:21:be:47
wl_bss_connect_done succeeded with 94:65:2d:21:be:47
```

```
root@imx6qdlsolo:/mnt# wpa_cli
wpa_cli v2.2
Copyright (c) 2004-2014, Jouni Malinen <j@w1.f

This software may be distributed under the ter
See README for more details.


Selected interface 'wlan0'

Interactive mode

> status
bssid=94:65:2d:21:be:47
ssid=OP3
id=0
mode=station
pairwise_cipher=CCMP
group_cipher=CCMP
key_mgmt=WPA2-PSK
wpa_state=COMPLETED
address=cc:b8:a8:1d:60:62
uuid=83ce0958-1d36-5dad-8eb0-6591c05e2b7d
```

2.现在我们要获取链接 wifi 的哪一个路由器 SSID 值

1.我们上面的 status 命令，我只截取了 wpa_state 的 wifi 链接通断状态

```
        while(1)
        {
                recevie_char[0]="status";
                ret = wpa_request(ctrl_conn, 1,recevie_char);
                if(ret)
                {
                        continue;
                }
                if(strstr(my_buf,"COMPLETED"))
                {
                        printf("connect wifi success ...\n");
                        get_line_from_buf(1,my_buf,xline);
                        sscanf(xline,"%s",ssid);
                        printf("ssid = %s\n",ssid);
                }
                else
                {
                        printf("connect wifi failed...\n");
                }

                sleep(5);
        }
        return 0;
}
os_free(ctrl_ifname);
wpa_cli_cleanup();
```

其实就是在 status 命令执行后，成功连接上 wifi 了，获取当前 status 返回的字符串

把 ssid 那一段截取下来

```
CFG80211-ERROR) wl_bss_connect_done : Report connect
connect wifi failed...
connect wifi failed...
connect wifi failed...
connect wifi failed...
connect wifi failed...
connect wifi failed...
Connectting with 94:65:2d:21:be:47 channel (6) ssid "

wl_bss_connect_done succeeded with 94:65:2d:21:be:47
wl_bss_connect_done succeeded with 94:65:2d:21:be:47
connect wifi success ...
ssid = ssid=OP3
connect wifi success ...
ssid = ssid=OP3
connect wifi success ...
ssid = ssid=OP3
connect wifi success ...
ssid = ssid=OP3
```

链接失败打印 else

链接成功打印 ssid