wifi驱动---wpa_supplicant 移植操作指南

1.获取源码

> 创建个文件夹 下载源码并解压

```
mkdir wifi1
cd wifi1
wget http://w1.fi/releases/wpa_supplicant-2.7.tar.gz
wget https://www.openssl.org/source/old/1.1.1/openssl-1.1.1q.tar.gz
wget https://www.infradead.org/~tgr/libnl/files/libnl-3.2.23.tar.gz
tar -xvf libnl-3.2.23.tar.gz
tar -xvf openssl-1.1.1q.tar.gz
tar -xvf wpa_supplicant-2.7.tar.gz
```

2.移植libopenssl

> wpa_supplicant 依赖于 libopenssl，因此需要先移植 libopenssl
>
> prefix=/home/ljh/mydemo/wifi1/tool/libopenssl/ 这里的路径是自己定义配置输出的库文件的路径 根据自己路径创建即可

1.配置生成 Makefile

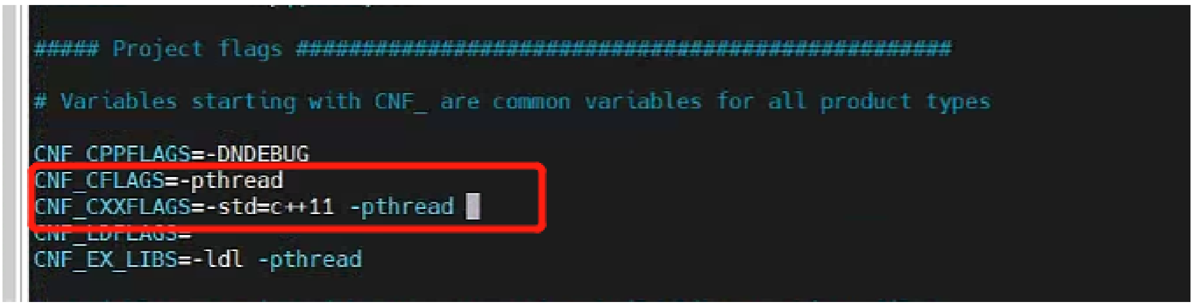
```
cd wifi1
mkdir tool
cd tool
mkdir libopenssl
cd ../openssl-1.1.1q/
./config shared no-asm --prefix=/home/ljh/mydemo/wifi1/tool/libopenssl/
```

2.声明编译器路径并修改Makefile

> /home/ljh/mydemo/sophpi-huashan/host-tools/gcc/riscv64-linux-musl-x86_64/bin根据源码路径修改

```
 export PATH=$PATH:/home/ljh/mydemo/sophpi-huashan/host-tools/gcc/riscv64-linux-musl-x86_64/bin
vi Makefile
```

找到所有包含“-m64”的内容，一共两处分别为变量 `CNF_CFLAGS 和 CNF_CXXFLAGS，将这两个变量中的“-m64”删除掉`

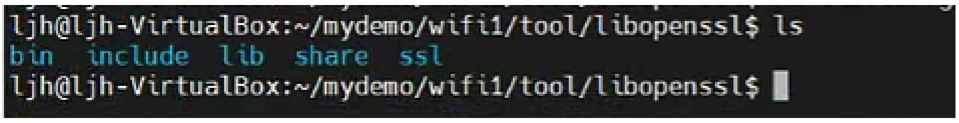


```
##### Project flags ##################################################

# Variables starting with CNF_ are common variables for all product types

CNF_CPPFLAGS=-DNDEBUG
CNF_CFLAGS=-pthread
CNF_CXXFLAGS=-std=c++11 -pthread
CNF_LDFLAGS=
CNF_EX_LIBS=-ldl -pthread
```

3.编译

```
make CROSS_COMPILE=riscv64-unknown-linux-musl- -j4
make install
```

> 编译成功如下图所示



```
ljh@ljh-VirtualBox:~/mydemo/wifi1/tool/libopenssl$ ls
bin  include  lib  share  ssl
ljh@ljh-VirtualBox:~/mydemo/wifi1/tool/libopenssl$
```

4.生成文件压缩并传输到开发板

> 注意：需要提前在开发板的mnt/data/创建wifi文件夹 并且在wifi文件下创建lib文件夹

```
cd ../tool/libopenssl/lib/

tar -czvf lib.tar.gz *
scp lib.tar.gz root@192.168.150.2:/mnt/data/wifi/lib
```

3.移植libnl库

> wpa_supplicant 也依赖于 libnl，因此还需要移植一下 libnl 库

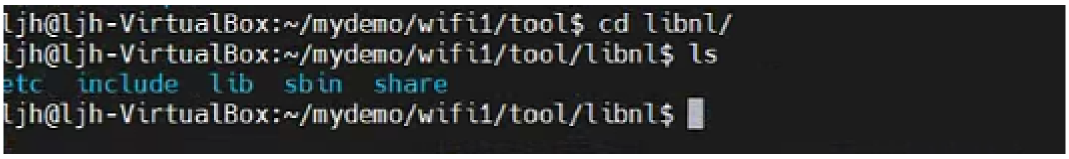
1.配置生成Makefile

```
cd wifi1/tool
mkdir libnl
cd ../libnl-3.2.23/
export PATH=$PATH:/home/ljh/mydemo/sophpi-huashan/host-tools/gcc/riscv64-linux-
musl-x86_64/bin
./configure --host=riscv64-unknown-linux-musl --
prefix=/home/ljh/mydemo/wifi1/tool/libnl/
```

2.编译

```
make -j6
make install
```

> 编译安装完成后的libnl目录下如图 所示：



```
ljh@ljh-VirtualBox:~/mydemo/wifi1/tool$ cd libnl/
ljh@ljh-VirtualBox:~/mydemo/wifi1/tool/libnl$ ls
etc  include  lib  sbin  share
ljh@ljh-VirtualBox:~/mydemo/wifi1/tool/libnl$ 
```

3.生成文件压缩并传输到开发板

`需要用到libnl文件下的lib库文件`，文件传输拷贝lib下的库文件到开发板的mnt/data/wifi/lib目录下。

> 注意：需要提前在开发板的mnt/data/创建wifi文件夹 并且在wifi文件下创建lib文件夹

```
cd ../tool/libnl/lib

tar -czvf libnl.tar.gz *
scp libnl.tar.gz root@192.168.150.2:/mnt/data/wifi/lib
```

4.移植wpa_supplicant

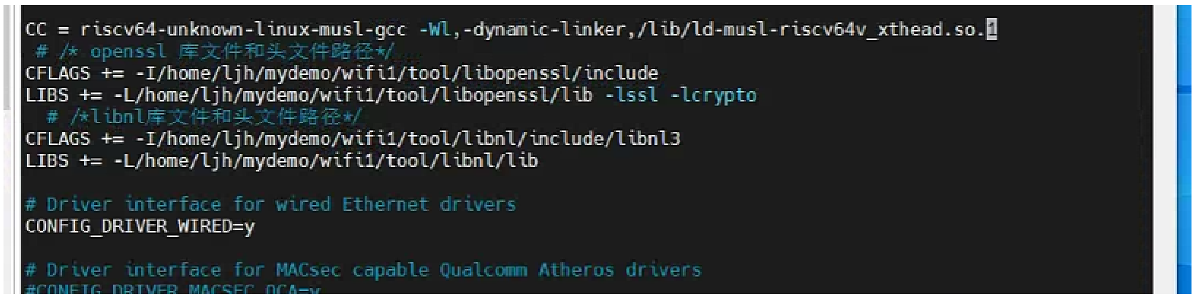
> 接下来移植wpa_supplicant

1.配置.config 指定交叉编译器

```
cd wpa_supplicant-2.7/wpa_supplicant/
cp defconfig .config
```

> 完成以后打开.config 文件，在里面指定交叉编译器、openssl、libnl 库和头文件路径，设置如下
>
> /home/ljh/mydemo/wifi1/tool 这里的路径是前面创建存放lib库文件的文件夹 根据自己设置而改

```
CC = riscv64-unknown-linux-musl-gcc -Wl,-dynamic-linker,/lib/ld-musl-
riscv64v_xthead.so.1
 # /* openssl 库文件和头文件路径*/
CFLAGS += -I/home/ljh/mydemo/wifi1/tool/libopenssl/include
LIBS += -L/home/ljh/mydemo/wifi1/tool/libopenssl/lib -lssl -lcrypto
  # /*libnl库文件和头文件路径*/
CFLAGS += -I/home/ljh/mydemo/wifi1/tool/libnl/include/libnl3
LIBS += -L/home/ljh/mydemo/wifi1/tool/libnl/lib
```



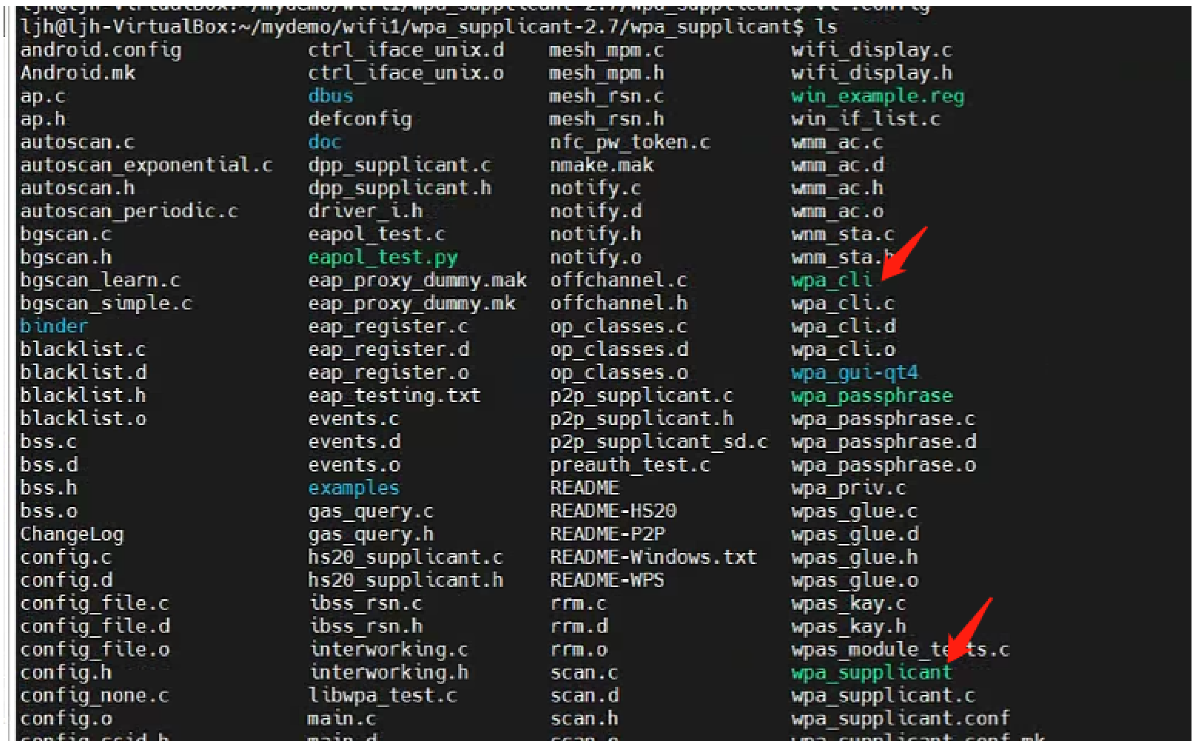
```
CC = riscv64-unknown-linux-musl-gcc -Wl,-dynamic-linker,/lib/ld-musl-riscv64v_xthead.so.1
 # /* openssl 库文件和头文件路径*/
CFLAGS += -I/home/ljh/mydemo/wifi1/tool/libopenssl/include
LIBS += -L/home/ljh/mydemo/wifi1/tool/libopenssl/lib -lssl -lcrypto
  # /*libnl库文件和头文件路径*/
CFLAGS += -I/home/ljh/mydemo/wifi1/tool/libnl/include/libnl3
LIBS += -L/home/ljh/mydemo/wifi1/tool/libnl/lib

# Driver interface for wired Ethernet drivers
CONFIG_DRIVER_WIRED=y

# Driver interface for MACsec capable Qualcomm Atheros drivers
#CONFIG_DRIVER_MACSEC_QCA=y
```

2.编译wap_supplicant

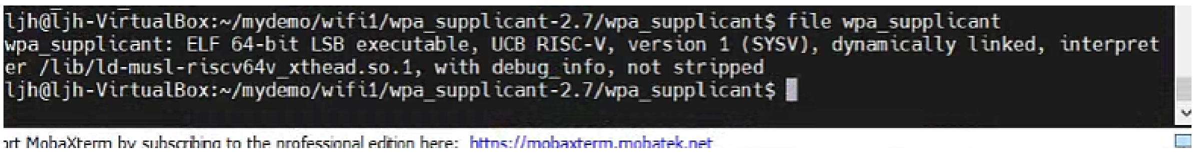
> /home/ljh/mydemo/wifi1/tool 这里的路径是前面创建存放lib库文件的文件夹 根据自己设置而改
>
> `编译 wpa_supplicant 的时候是需要指定 libnl 的 pkgconfig 路径`，否则会提示“libnl-3.0”或者“libnl-3.0.pc”找不到等错误。

```
export
PKG_CONFIG_PATH=/home/ljh/mydemo/wifi1/tool/libnl/lib/pkgconfig:$PKG_CONFIG_PATH
make
```

编译完成以后就会在本目录下生成 wpa_supplicant 和 wpa_cli



```
ljh@ljh-VirtualBox:~/mydemo/wifi1/wpa_supplicant-2.7/wpa_supplicant$ vi .config
ljh@ljh-VirtualBox:~/mydemo/wifi1/wpa_supplicant-2.7/wpa_supplicant$ ls
android.config            ctrl_iface_unix.d    mesh_mpm.c           wifi_display.c
Android.mk                ctrl_iface_unix.o    mesh_mpm.h           wifi_display.h
ap.c                      dbus                 mesh_rsn.c           win_example.reg
ap.h                      defconfig            mesh_rsn.h           win_if_list.c
autoscan.c                doc                  nfc_pw_token.c       wmm_ac.c
autoscan_exponential.c    dpp_supplicant.c     nmake.mak            wmm_ac.d
autoscan.h                dpp_supplicant.h     notify.c             wmm_ac.h
autoscan_periodic.c       driver_i.h           notify.d             wmm_ac.o
bgscan.c                  eapol_test.c         notify.h             wnm_sta.c
bgscan.h                  eapol_test.py        notify.o             wnm_sta.h
bgscan_learn.c            eap_proxy_dummy.mak  offchannel.c         wpa_cli
bgscan_simple.c           eap_proxy_dummy.mk   offchannel.h         wpa_cli.c
binder                    eap_register.c       op_classes.c         wpa_cli.d
blacklist.c               eap_register.d       op_classes.d         wpa_cli.o
blacklist.d               eap_register.o       op_classes.o         wpa_gui-qt4
blacklist.h               eap_testing.txt      p2p_supplicant.c     wpa_passphrase
blacklist.o               events.c             p2p_supplicant.h     wpa_passphrase.c
bss.c                     events.d             p2p_supplicant_sd.c  wpa_passphrase.d
bss.d                     events.o             preauth_test.c       wpa_passphrase.o
bss.h                     examples             README               wpa_priv.c
bss.o                     gas_query.c          README-HS20          wpas_glue.c
ChangeLog                 gas_query.h          README-P2P           wpas_glue.d
config.c                  hs20_supplicant.c    README-Windows.txt   wpas_glue.h
config.d                  hs20_supplicant.h    README-WPS           wpas_glue.o
config_file.c             ibss_rsn.c           rrm.c                wpas_kay.c
config_file.d             ibss_rsn.h           rrm.d                wpas_kay.h
config_file.o             interworking.c       rrm.o                wpas_module_tests.c
config.h                  interworking.h       scan.c               wpa_supplicant
config_none.c             libwpa_test.c        scan.d               wpa_supplicant.c
config.o                  main.c               scan.h               wpa_supplicant.conf
config_ssid.h             main.d               scan.o               wpa_supplicant.conf.mk
```



```
ljh@ljh-VirtualBox:~/mydemo/wifi1/wpa_supplicant-2.7/wpa_supplicant$ file wpa_supplicant
wpa_supplicant: ELF 64-bit LSB executable, UCB RISC-V, version 1 (SYSV), dynamically linked, interpret
er /lib/ld-musl-riscv64v_xthead.so.1, with debug_info, not stripped
ljh@ljh-VirtualBox:~/mydemo/wifi1/wpa_supplicant-2.7/wpa_supplicant$ 
```

rt MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net

3.生成文件压缩并传输到开发板

编译好的wpa_cli 和 wpa_supplicant 这两个文件拷贝到开发板的mnt/data/wifi

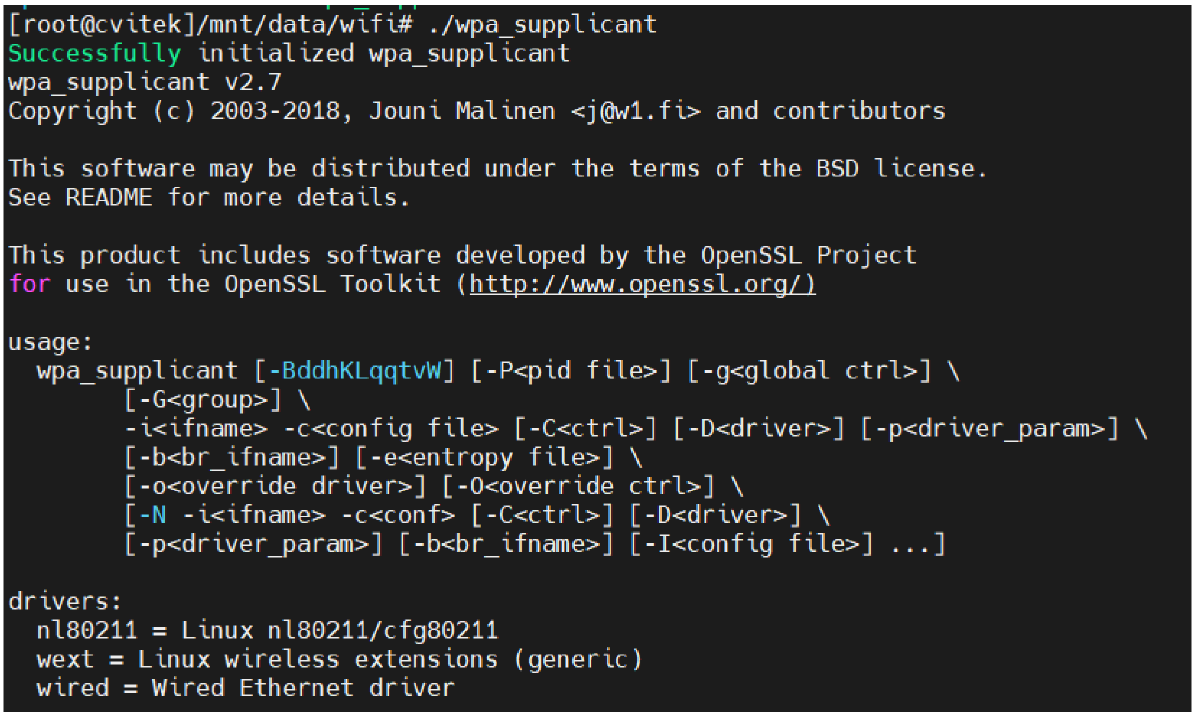
> 注意：需要提前在开发板的mnt/data/创建wifi文件夹

```
scp -r wpa_cli wpa_supplicant root@192.168.150.2:/mnt/data/wifi
```

在开发板测试使用：

```
cd /mnt/data/wifi/lib
gzip -d *.gz
tar xvf lib.tar
tar xvf libnl.tar
#声明库文件路径
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/mnt/data/wifi/lib/
```

> 正常输出一下内容表示移植成功！



```
[root@cvitek]/mnt/data/wifi# ./wpa_supplicant
Successfully initialized wpa_supplicant
wpa_supplicant v2.7
Copyright (c) 2003-2018, Jouni Malinen <j@w1.fi> and contributors

This software may be distributed under the terms of the BSD license.
See README for more details.

This product includes software developed by the OpenSSL Project
for use in the OpenSSL Toolkit (http://www.openssl.org/)

usage:
  wpa_supplicant [-BddhKLqqtvW] [-P<pid file>] [-g<global ctrl>] \
        [-G<group>] \
        -i<ifname> -c<config file> [-C<ctrl>] [-D<driver>] [-p<driver_param>] \
        [-b<br_ifname>] [-e<entropy file>] \
        [-o<override driver>] [-O<override ctrl>] \
        [-N -i<ifname> -c<conf> [-C<ctrl>] [-D<driver>] \
        [-p<driver_param>] [-b<br_ifname>] [-I<config file>] ...]

drivers:
  nl80211 = Linux nl80211/cfg80211
  wext = Linux wireless extensions (generic)
  wired = Wired Ethernet driver
```

5.WIFI联网测试

1.在开发板解压库文件压缩包并声明库文件路径

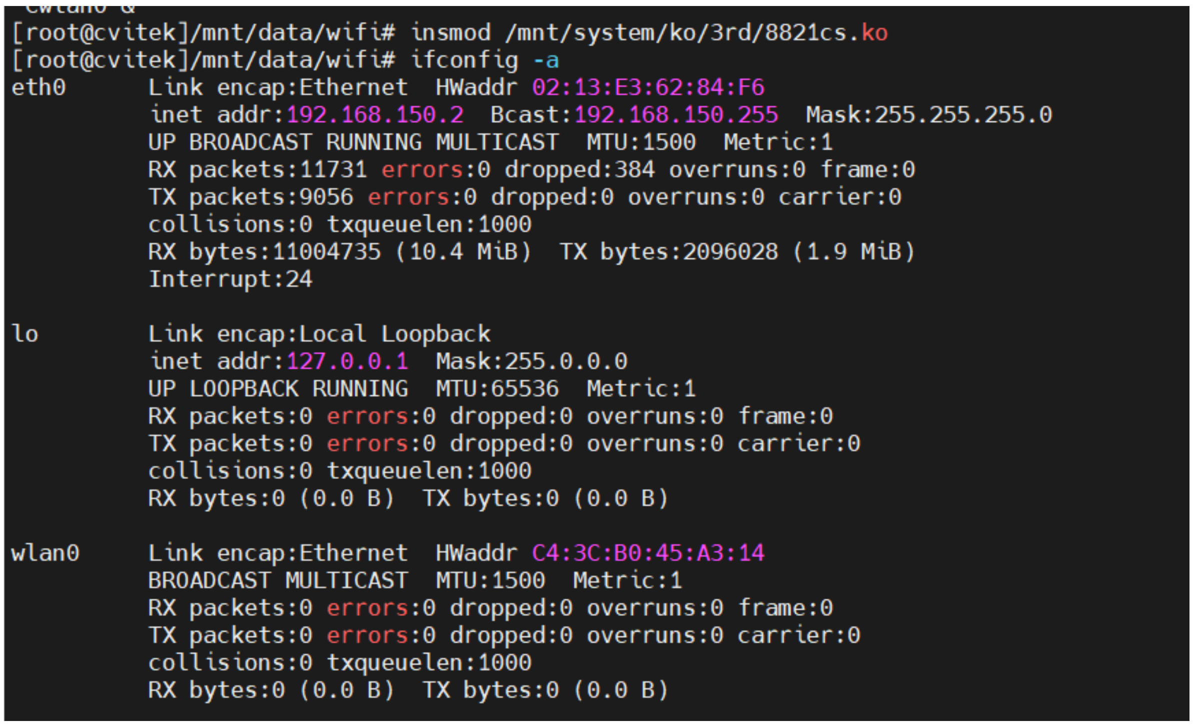
> 注意：上面操作过这步骤可忽略

```
cd /mnt/data/wifi/lib
gzip -d *.gz
tar xvf *.tar
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/mnt/data/wifi/lib/
export PATH=$PATH:/mnt/data/wifi/
```

2.联网使用

加载wifi驱动：

```
insmod /mnt/system/ko/3rd/8821cs.ko
ifconfig -a
```



```
[root@cvitek]/mnt/data/wifi# insmod /mnt/system/ko/3rd/8821cs.ko
[root@cvitek]/mnt/data/wifi# ifconfig -a
eth0      Link encap:Ethernet  HWaddr 02:13:E3:62:84:F6
          inet addr:192.168.150.2  Bcast:192.168.150.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:11731 errors:0 dropped:384 overruns:0 frame:0
          TX packets:9056 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:11004735 (10.4 MiB)  TX bytes:2096028 (1.9 MiB)
          Interrupt:24

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

wlan0     Link encap:Ethernet  HWaddr C4:3C:B0:45:A3:14
          BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

在开发板的mnt/data/wifi中

```
mkdir wpa
vi wpa_supplicant.conf
```

编写以下内容保存退出

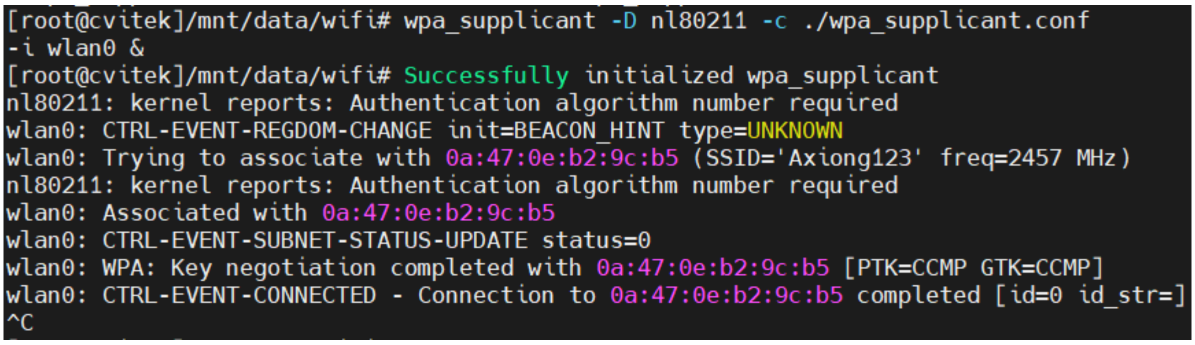
> ctrl_interface= 这里指的是上边的wpa路径
>
> ssid是wifi的名称 psk是密码 这里我使用是手机热点

```
ctrl_interface=./wpa
ap_scan=1
network={
 ssid="Axiong123"
 psk="11111111"
}
```

输入命令使用：

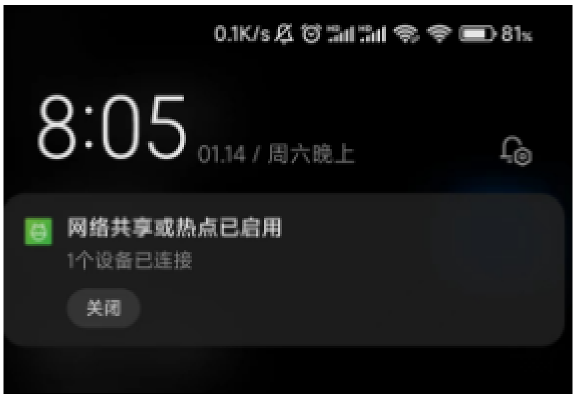
```
wpa_supplicant -D nl80211 -c ./wpa_supplicant.conf -i wlan0 &
```

成功输出：



```
[root@cvitek]/mnt/data/wifi# wpa_supplicant -D nl80211 -c ./wpa_supplicant.conf
-i wlan0 &
[root@cvitek]/mnt/data/wifi# Successfully initialized wpa_supplicant
nl80211: kernel reports: Authentication algorithm number required
wlan0: CTRL-EVENT-REGDOM-CHANGE init=BEACON_HINT type=UNKNOWN
wlan0: Trying to associate with 0a:47:0e:b2:9c:b5 (SSID='Axiong123' freq=2457 MHz)
nl80211: kernel reports: Authentication algorithm number required
wlan0: Associated with 0a:47:0e:b2:9c:b5
wlan0: CTRL-EVENT-SUBNET-STATUS-UPDATE status=0
wlan0: WPA: Key negotiation completed with 0a:47:0e:b2:9c:b5 [PTK=CCMP GTK=CCMP]
wlan0: CTRL-EVENT-CONNECTED - Connection to 0a:47:0e:b2:9c:b5 completed [id=0 id_str=]
^C
```

观察手机：



0.1K/s 81%

8:05 01.14 / 周六晚上

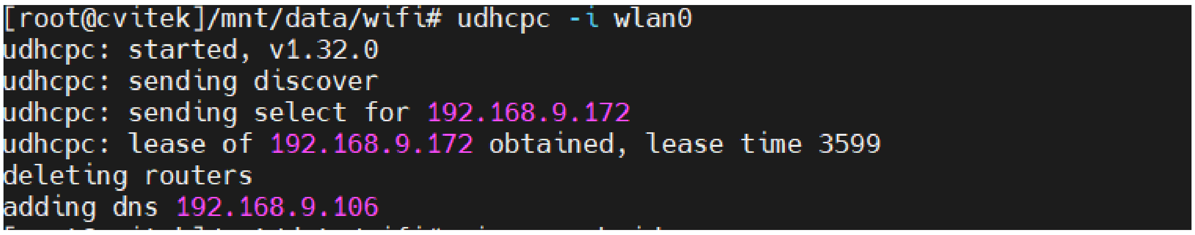
网络共享或热点已启用

1个设备已连接

关闭

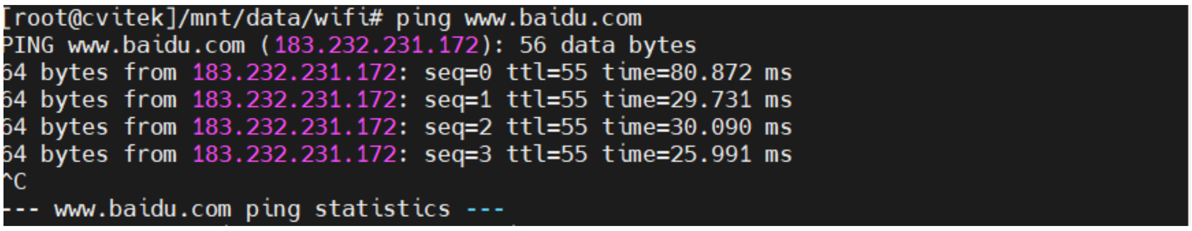
获取ip地址：

```
udhcpc -i wlan0
```



```
[root@cvitek]/mnt/data/wifi# udhcpc -i wlan0
udhcpc: started, v1.32.0
udhcpc: sending discover
udhcpc: sending select for 192.168.9.172
udhcpc: lease of 192.168.9.172 obtained, lease time 3599
deleting routers
adding dns 192.168.9.106
```

可以ping 通百度 联网成功！



```
[root@cvitek]/mnt/data/wifi# ping www.baidu.com
PING www.baidu.com (183.232.231.172): 56 data bytes
64 bytes from 183.232.231.172: seq=0 ttl=55 time=80.872 ms
64 bytes from 183.232.231.172: seq=1 ttl=55 time=29.731 ms
64 bytes from 183.232.231.172: seq=2 ttl=55 time=30.090 ms
64 bytes from 183.232.231.172: seq=3 ttl=55 time=25.991 ms
^C
--- www.baidu.com ping statistics ---
```