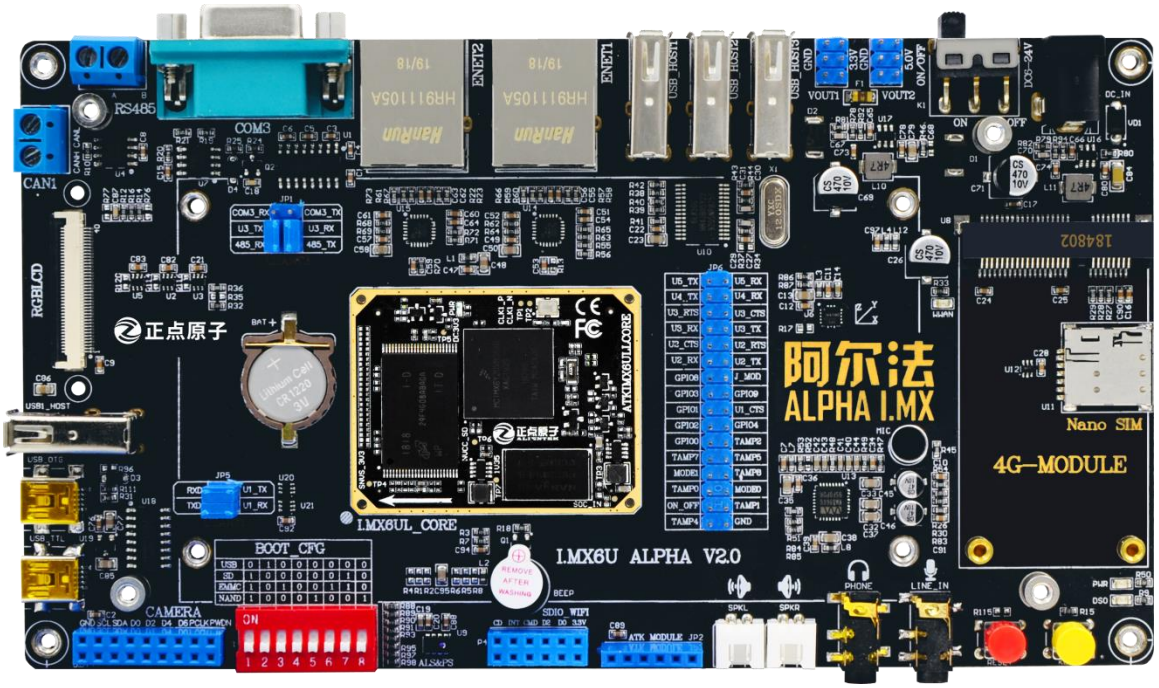
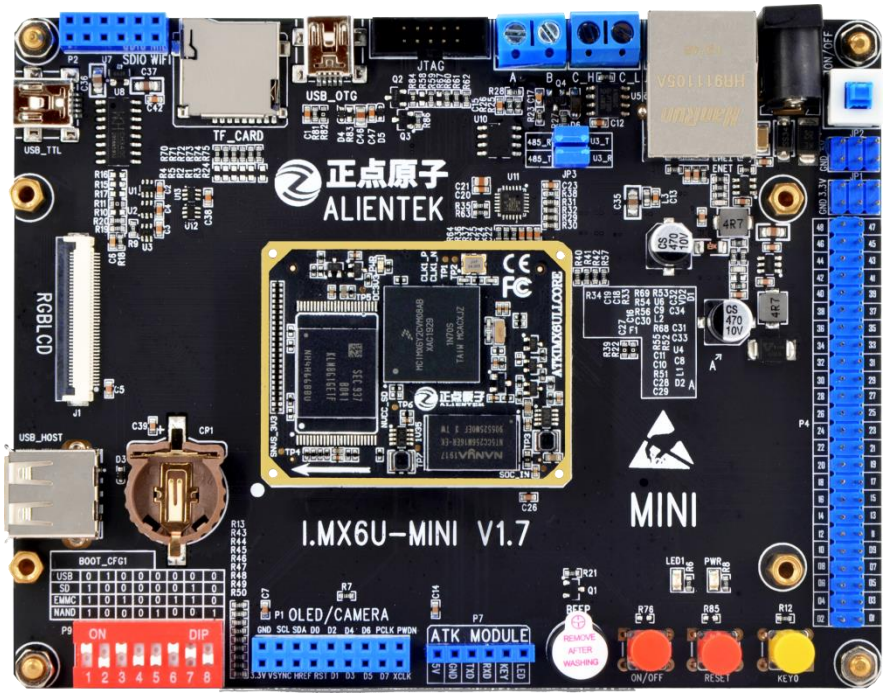
I.MX6U 常见问题汇总 V1.3





正点原子

正点原子公司名称 ：广州市星翼电子科技有限公司

原子哥在线教学平台：www.yuanzige.com

开源电子网 / 论坛 ：http://www.openedv.com/forum.php

正点原子淘宝店铺 ：https://openedv.taobao.com

正点原子官方网站 ：www.alientek.com

正点原子 B 站视频 ：https://space.bilibili.com/394620890

电话：020-38271790 传真：020-36773971

请关注正点原子公众号，资料发布更新我们会通知。

请下载原子哥 APP，数千讲视频免费学习，更快更流畅。

扫码关注正点原子公众号

扫码下载“原子哥”APP

文档更新说明

版本	版本更新说明	负责人	校审	发布日期
V1.0	初稿：	正点原子 linux 团队	正点原子 linux 团队	2020.4.30
V1.1	修改： 修改文档中部分错误 重新排版 添加一部分常见问题解决办法	正点原子 linux 团队	正点原子 linux 团队	2020.7.28
V1.2	修改： 修改文档中部分错误 重新排版 添加一部分常见问题解决办法	正点原子 linux 团队	正点原子 linux 团队	2021.3.29
V1.3	添加一些论坛上的分享链接和解决问题的链接	正点原子 linux 团队	正点原子 linux 团队	2021.6.21

目录

前言

这份文档总结了一些常见的问题分享给大家，特别是新人，在遇到一些问题的时候，可以看看这份文档，可能对您有一点点帮助。因为涉及的问题多，不好分类，还不知道怎么排版才能方便大家查找，目前暂时这样排版，如果您有什么好的建议也可以反馈给我们。在查阅这份文档的时候，您可以根据报错信息，**尝试使用关键字进行搜索找到对应的解决办法**。如果大家在操作过程中还会遇到什么常见的问题，也可以反馈给我们，可以在论坛发帖和大家分享您遇到问题或者解决办法，这份文档将会不断更新，感谢你们的支持！

第一章 资料获取和硬件检查

1.1 资料

1.1.1 资料获取

资料可以在论坛里下载：http://www.openedv.com/thread-300792-1-1.html
A 盘是基础资料，其它盘是视频资料，视频资料可以不用下载，视频可以在线观看：
B 站哔哩哔哩学习视频：
https://search.bilibili.com/all?keyword=%E6%AD%A3%E7%82%B9%E5%8E%9F%E5%AD%90
原子哥在线教学网站：https://www.yuanzige.com/
也可以手机商店 APP 下载和安装原子哥 APP，在 APP 上查看教学视频。

图 1.1.1-1 手机商店搜索原子哥

1.1.2 资料介绍

A 盘资料总览

图 1.1.2-1 A 盘资料

上图是笔者编辑这份文档的时候 A 盘的目录结构，后期这个目录结构还会变化，我们还在不断添加例程。 1、例程源码 里是《【正点原子】I.MX6U 嵌入式 Linux 驱动开发指南》教程

里用到的例程源码以及出厂系统的源码，跟着教程来学习的时候，建议教程里指定用哪个源码来开学习就用哪个，希望大家尽量跟着教程步骤来学习，特别是 3、正点原子Uboot和Linux出厂源码 和 4、NXP官方原版Uboot和Linux 以及 10、开发板教程对应的uboot和linux源码这三个源码不要混淆使用。

图 1.1.2-2 例程源码文件夹

以下是对几个常用的源码说明。

3、正点原子Uboot和Linux出厂源码

开发板出厂预装的系统中，uboot、内核和设备树是用这个源码编译出来的，使用这个源码并按照《【正点原子】I.MX6U 用户快速体验》这个文档编译出来的镜像可以直接用于测试开发板的外设。注意：此源码不是教程源码，在驱动开发指南中仅做体验编译使用，与教程无特别大的关系。

4、NXP官方原版Uboot和Linux

这个是 NXP 官方的 uboot 和内核源码，教程是基于这个源码来进行移植的。

10、开发板教程对应的uboot和linux源码

这个源码是按照《【正点原子】I.MX6U 嵌入式 Linux 驱动开发指南》移植好的教程 uboot 和内核源码。教程中的驱动实验都是基于此内核源码，而不是出厂内核源码。其实还是建议大家跟着教程顺序来学习，不建议跳过教程前面的章节内容去直接操作后面驱动开发的内容，因为会忽略很多细节。

A 盘里有个《资料更新记录.txt》，我们每次更新资料，更新的时间以及更新了些什么内容都记录在了这个文本里。

关于 A 盘里资料的目录详细结构，大家可以看看《资料目录结构（2 级结构图说明）.png》。

1.1.3 技术咨询

（1）可以加入技术交流群和大家讨论

（2）可以论坛发帖子（鼓励大家在论坛里发帖子，问题分享）

（3）可以在淘宝上联系客服和技术支持

（4）电话咨询 02038271790

1.2 开箱检查

1.2.1 开箱视频

大家拿到开发板应先进行检查，先检查外观再检查功能。

B 站开箱检查的视频：https://www.bilibili.com/video/av74037066

注意：这个开箱检查视频是在早期录制的，当时使用的是第一版的 Qt 界面。截止此文档更新时，使用的是第三版的 Qt 界面。所以用户拿到手上的开发板的系统界面可能不太一样。

图 1.2.1-1 历代出厂 Qt 版本界面

开箱需要检查的内容：

（1）检查开发板上的元器件有没有存在脱落或者破损的情况；

（2）检查核心板的类型

1.2.2 核心板类型

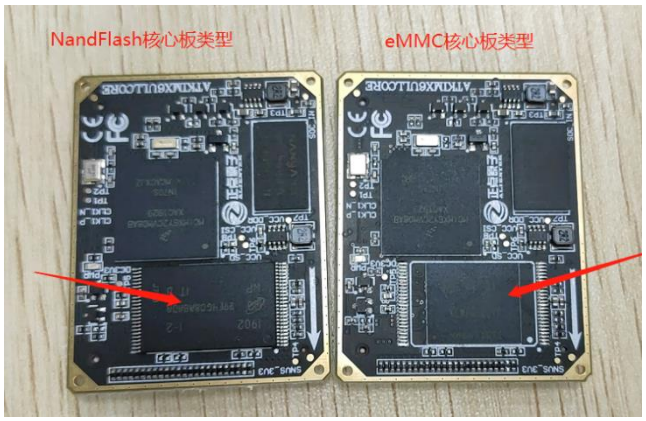


图 1.2.2-1 NandFlash 核心板和 EMMC 核心板

① 外观区别

查看存储芯片，如上图，外观上 EMMC 芯片长度比 NAND FLASH 芯片的长度要短、小巧。

② 查看芯片上的丝印

存储芯片上的丝印：

丝印 MT29F2G08ABAEAWP-IT 或 MT29F4G08ABADAWP-IT 型号的是 NAND FLASH 版本的核心板，对应的存储容量分别是 256MB 和 512MB；

丝印是 KLM8G1GET 的表示 EMMC 的核心板，对应的 EMMC 存储容量是 8GB。

外扩 DDR3L 上的丝印：

型号 NT5CC**256**M16EP-EK 的 DDR3L 大小为 **512**MB，为 EMMC 版本。

型号 NT5CC**128**M16JR-EK 的 DDR3L 大小为 **256**MB，为 NAND 版本。

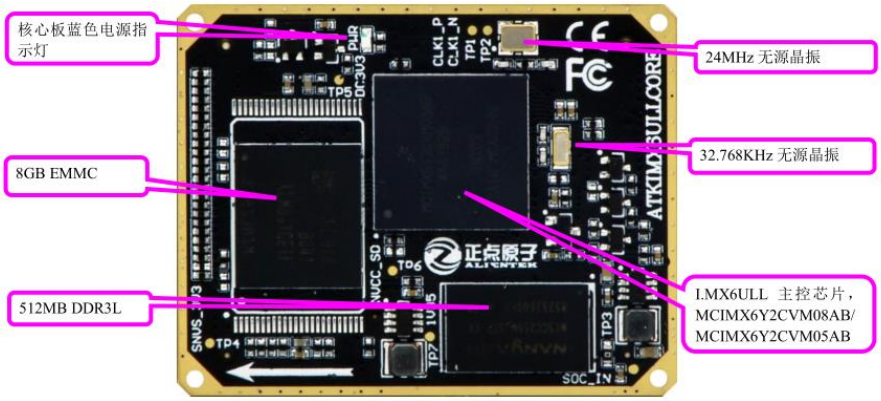


图 1.2.2-2 EMMC 核心板

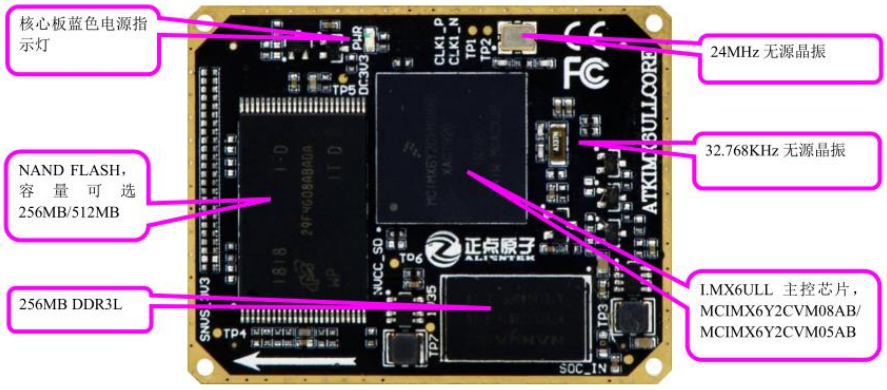


图 1.2.2-3 Nand Flash 核心板

③已经安装了 SecureCRT 或 MobaXterm 等串口终端软件以及 CH340 串口驱动的话，用出厂的系统启动开发板，查看 uboot 启动过程中的打印信息，出货的时候，我们已经将系统烧录到 eMMC 或者 NAND FLASH 了，根据打印的 DRAM 的数值判断是 eMMC 的还是 NAND FLASH 的核心板。（安装软件和连接等操作具体可以参考《【正点原子】I.MX6U 嵌入式 Linux 驱动开发指南》教程或者视频。）

```
U-Boot 2016.03-gd9420c3 (Nov 01 2019 - 12:03:59 +0800)

CPU:   Freescale i.MX6ULL rev1.1 69 MHz (running at 396 MHz)
CPU:   Industrial temperature grade (-40C to 105C) at 29C
Reset cause: POR
Board: MX6ULL 14x14 EVK
I2C:   ready
DRAM:  512 MiB
MMC:   FSL_SDHC: 0, FSL_SDHC: 1
Display: ATK-LCD-7-800x480 (800x480)
Video: 800x480x24
reading alientek.bmp
** Unable to read file alientek.bmp **
Error: no valid bmp image at 88000000
In:    serial
Out:   serial
Err:   serial
switch to partitions #0, OK
mmc1(part 0) is current device
Net:   FEC1
Error: FEC1 address not set.
```

图 1.2.2-4 EMMC 版本启动 uboot 信息

```
U-Boot 2016.03-gd9420c3 (Nov 01 2019 - 12:04:07 +0800)

CPU:   Freescale i.MX6ULL rev1.1 69 MHz (running at 396 MHz)
CPU:   Industrial temperature grade (-40C to 105C) at 34C
Reset cause: POR
Board: MX6ULL 14x14 EVK
I2C:   ready
DRAM:  256 MiB
NAND:  512 MiB
MMC:   FSL_SDHC: 0
*** Warning - bad CRC, using default environment

Display: ATK-LCD-7-800x480 (800x480)
Video: 800x480x24

NAND read: device 0 offset 0x420000, size 0x100000
 1048576 bytes read: OK
Error: no valid bmp image at 88000000
In:    serial
Out:   serial
Err:   serial
Net:   FEC1
Error: FEC1 address not set.
```

图 1.2.2-5 Nand Flash 版本启动 uboor 信息

上面两个图分别为 EMMC 版本和 Nand Flash 版本的核心板的 uboot 信息。

第一个图打印 mmc1(part 0) is current device，即设备是 mmc1，说明是 EMMC 启动方式，DRAM 大小是 512MB，即 EMMC 的核心板，DDR3 是 512MB。

第二个图打印设备是 NAND read，DRAM 是 256MB，说明是 NAND FLASH 的核心板，DDR3 是 256MB 的。

1.3 启动和关闭开发板

1.3.1 电源适配器

开发板使用直流电源 DC6V~16V，ALPHA 开发板配是 12V 1A 的电源适配器，这里要注意一下，推荐使用我们配套的电源适配器，如果是用自己的电源适配器，请严格按照规格要求来，超过电压最大范围会烧坏开发板。检查开发板开关是否是 OFF 状态，如果不是就将开关拨到 OFF 状态，然后插上电源适配器，如下图位置。（有些用户是把电源适配器接到排插上，在使用时请注意排插接口是否松动、排插开关是否打开）

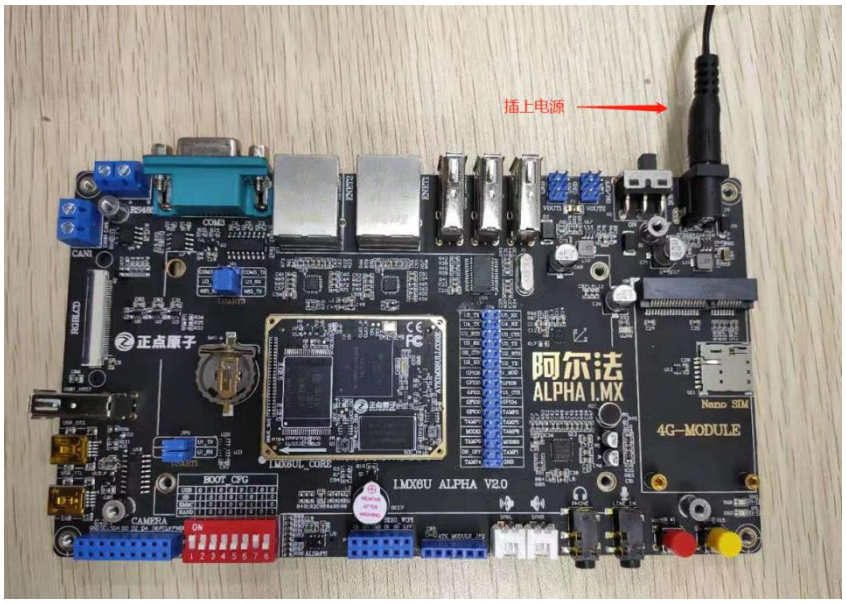


图 1.3.1-1 接入电源适配器

1.3.2 检查拨码开关位置

在上电开机前，请检查开发板的拨码开关位置。因为拨码开关的位置就决定了开发板系统启动的方式。简单来说就是系统要从哪种设备启动，拨错了系统可能无法启动，这里大家需要注意一下。如果是 EMMC 的核心板，可以选择从 SD 卡启动或者 EMMC 启动，如果是 NAND FLASH 的核心板，可以选择从 SD 卡启动或者 NAND FLASH 启动，如果使用 mfgtool 上位机固化系统，则选择从 USB 启动。

查看看拨码开关附近的丝印，“1”代表的是拨码开关往 ON 方向拨，“0”代表拨码开关往下 ON 的反方向拨。开发板分为 SD 卡启动、EMMC 启动和 NAND FLASH 启动方式，启动方式要根据核心板的型号来。

EMMC 启动的拨码方式 10100110，如下图：

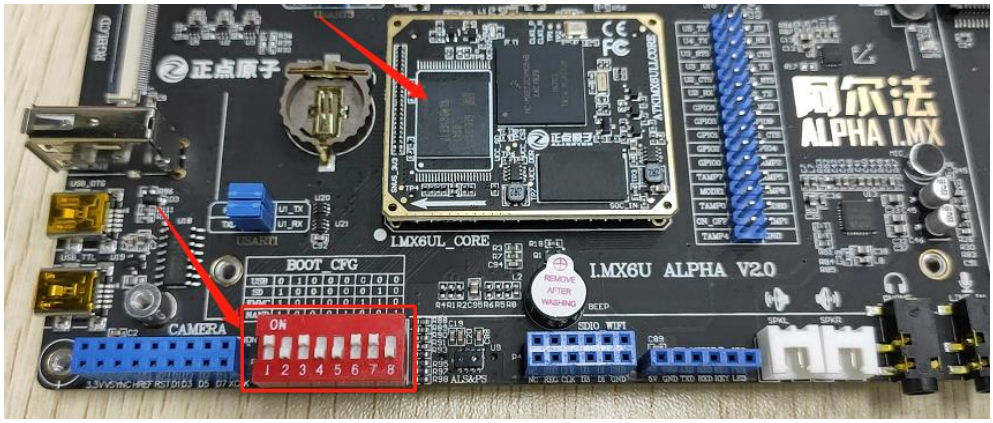


图 1.3.2-1 拨码开关 EMMC 启动

NAND FLASH 启动的拨码方式 10001001，如下图：

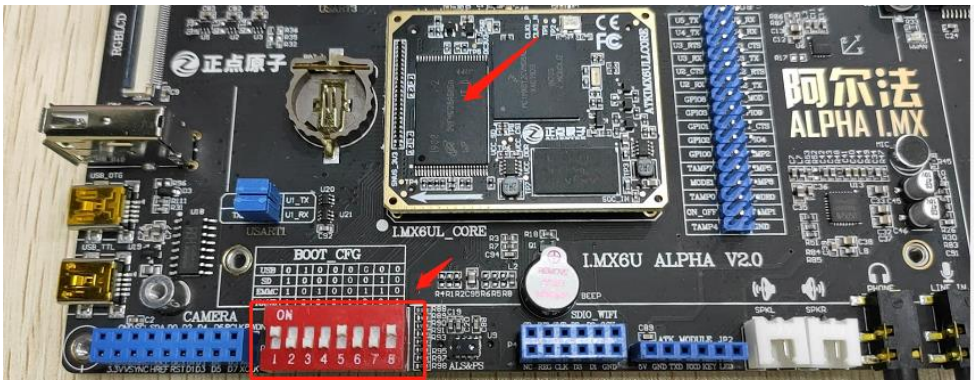


图 1.3.2-2 拨码开关 Nand Flash 启动

SD 卡启动方式是 10000010，如下图：

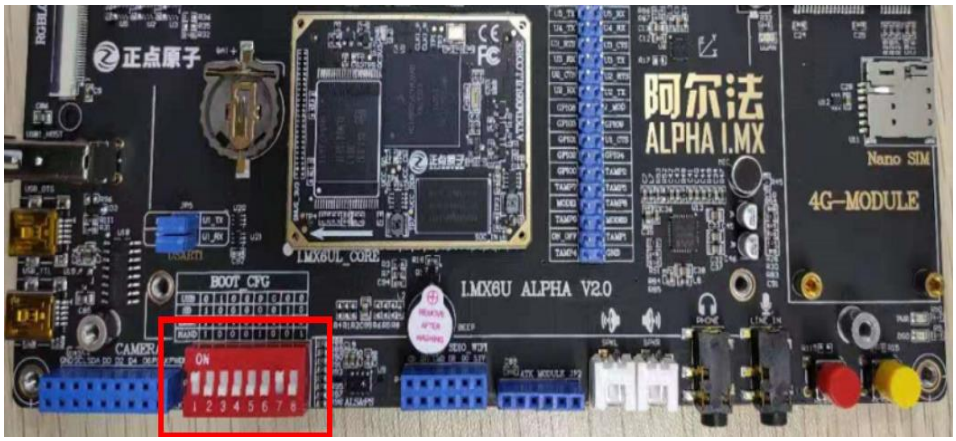


图 1.3.2-3 拨码开关 SD 卡启动

1.3.3 上电启动

接上电源适配器以后，打开开关，开发板上电启动。上电进入系统需要 18s 左右的时间。

选购屏幕的用户在系统启动以后可以看到 QT 界面，可以操作一下界面，点击播放音乐查看是否有声音，可以检查喇叭是否正常，插入耳机可以检查耳机接口是否正常等等。

图 1.3.3-1 启动后屏幕会出现界面（此图为第二版 Qt）

如果用户安装了串口调试软件（比如 SecureCRT、XShell、MobaXterm 或者 Putty 等）和 CH340 驱动，可以在串口终端查看启动的打印信息。

```
done.
Starting Xserver
Starting system message bus: dbus.
Starting Connection Manager
Starting Dropbear SSH server: dropbear.
Starting rpcbind daemon...done.
starting statd: [   10.811860] fec 2188000.ethernet eth1: Freescale FEC PHY driver [SMSC LAN8710/LAN8720] (mii_bus:phy_addr=20
[   10.871284] IPv6: ADDRCONF(NETDEV_UP): eth1: link is not ready
done
Starting advanced power management daemon: No APM support in kernel
(failed.)
Starting atd: OK
exportfs: can't open /etc/exports for reading
NFS daemon support not enabled in kernel
Starting system log daemon...0
Starting kernel log daemon...0
[ ok ]rting Avahi mDNS/DNS-SD Daemon: avahi-daemon
Starting Telephony daemon
Starting Linux NFC daemon
Starting crond: OK
Running local boot scripts (/etc/rc.local).

root@ALIENTEK-IMX6U:~# [   14.971541] fec 2188000.ethernet eth1: Link is Up - 100Mbps/Full - flow control rx/tx
[   14.979431] IPv6: ADDRCONF(NETDEV_CHANGE): eth1: link becomes ready
[   55.411097] random: nonblocking pool is initialized

root@ALIENTEK-IMX6U:~#
root@ALIENTEK-IMX6U:~#
root@ALIENTEK-IMX6U:~#
```

图 1.3.3-2 启动后最终进入文件系统

1.3.4 开机无法启动，核心板电源灯亮

启动板子后看核心板电源灯有没有亮。

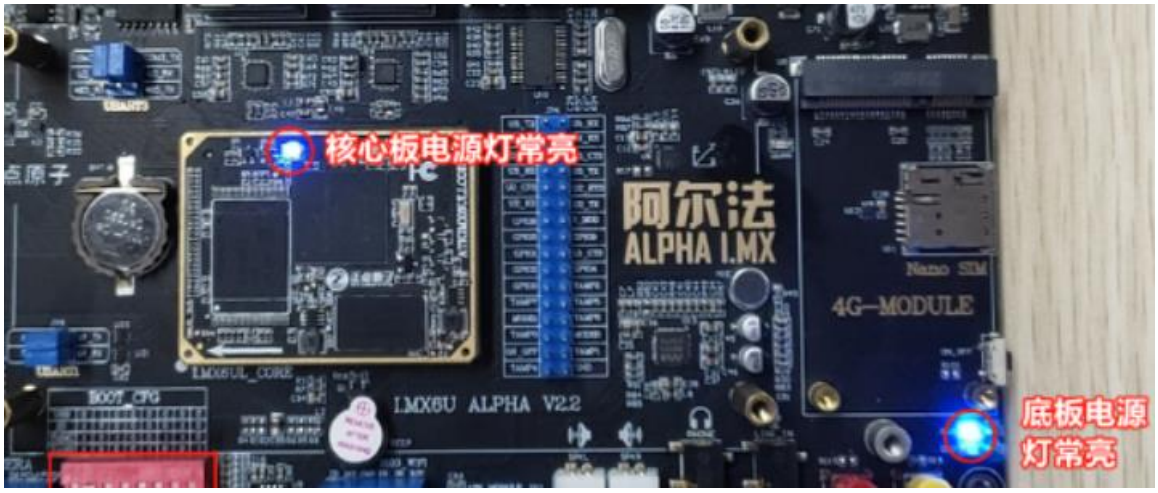


图 1.3.4-1 核心板电源灯

（1）先检查电源适配器是否插好，usb 线接在 USB_TTL 接口只能部分供电，在接屏幕的时候要接电源适配器来供电，否则供电不足系统无法启动。

（2）检查拨码开关的位置是否正确，开关是否已经打开。

（3）请检查是否有误触到其他金属物体，造成短路或者检查核心板是否没有插好等情况。

（4）还有一种情况的就是拨码开关损坏，可以按照下图的方法检查下拨码开关。

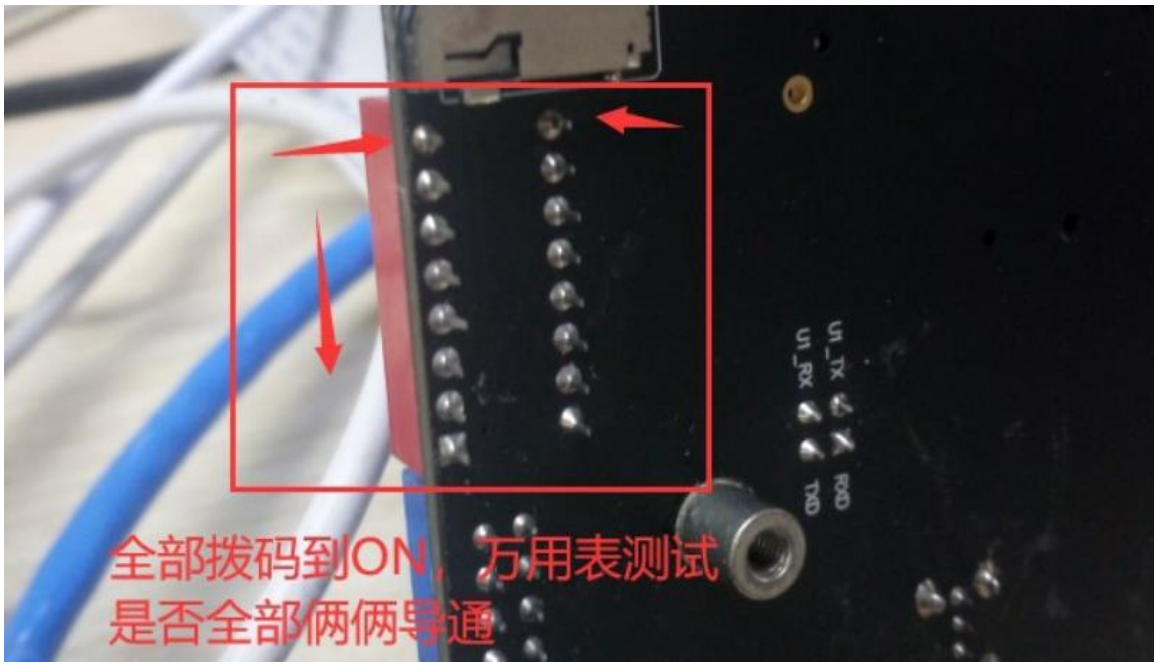


图 1.3.4-2 测量拨码开关导通性

1.3.5 点了 Qt 界面的关机后，开发板无法再次启动

可以检查下核心板上蓝色的电源灯有没有亮，可能核心板没在工作，上电后长按 3 秒左右 4G 模块丝印那附近的白色小按键就行（因为用户可能使用了 Qt 界面的关机或者使用指令关机，执行的是软关机，重新上电需要按下图 ON/OFF 键重启。或者取下 RTC 电池或重新插拔核心板也能启动。MINI 开发板也是同理，上电按 ON/OFF 按键）

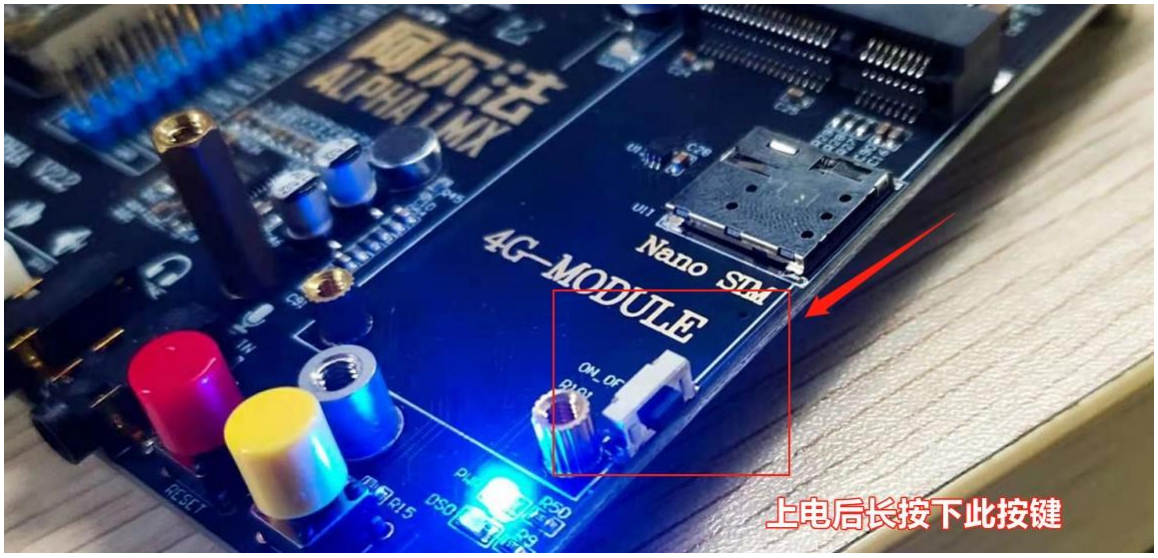


图 1.3.5-1 阿尔法板重启示意图

图 1.3.5-2 mini 板重启示意图

1.3.6 出厂系统屏幕无法触摸

因为之前老版本的系统没有兼容好部分屏幕的触摸驱动，所以可能会存在此类问题。解决方法就是更新下最新的出厂系统烧录工具，重新烧录系统。同时也要检查屏幕排线、电源适配器有没有接好。

1.3.7 串口无法输出

出厂系统启动信息默认从 USB_TTL 这个口打印的（RS232 和 RS485 这两个串口不是启动信息调试口，如果要用这两个口输出系统启动信息，需要修改 uboot），如果 USB_TTL 这个口无法输出启动信息，请注意检查圈圈地方的跳线帽是否方向接错了，跳线帽是否坏了以及 usb 线是否接好。重新插拔下 USB_TTL 接口的 USB 线和核心板，接触不良也可能会导致这个问题的产生。注意下串口的波特率为 115200，8N1。

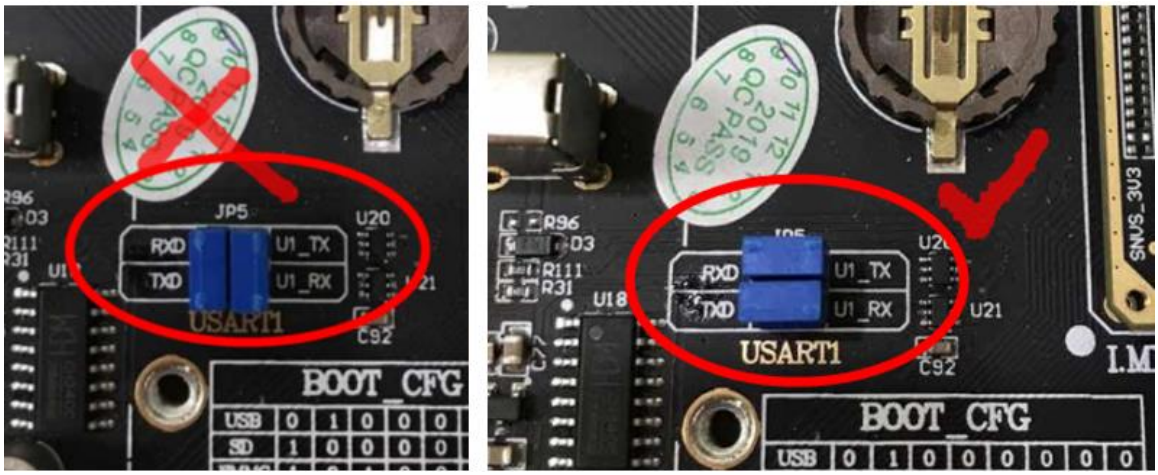


图 1.3.7-1 左图为错误示范，右图为正确示范

1.3.8 串口无法输入

重新插拔下 USB_TTL 接口的 USB 线和核心板，接触不良也可能会导致这个问题的产生。注意下串口的波特率为 115200，8N1，可以参考上面的。如果是使用 SecureCRT 串口软件的，默认是有串口的流控设置，需要自己去掉流控设置，重新连接下串口终端。

图 1.3.8-1 不勾选 RTS/CTS 流控

1.3.9 屏幕花屏

开发板启动，进入系统如果有出现屏幕花屏的情况，请关闭开关先断电，请先断电！

检查屏幕的 FPC 软排线是否与开发板接好，如下图，屏的一端是否与开发板连接整齐，没有歪，FPC 排线是否有接反等等。用指甲将压紧 FPC 排线的黑色压条向上翻起就可以取出 FPC 软排线。（不建议用户经常卸下屏幕，因为 FPC 软排线有使用寿命，易老化脱落）

图 1.3.9-1 接入屏幕

取下排线请检查是否有“起毛”或者“脱落”的情况。如下图这种情况就是 FPC 排线脱落的情况。如果脱落了有关触摸的几根线，会造成触摸不正常。脱落了与显示相关的几根线，那么将会造成显示不正常，出现颜色不对的情况。下图为 FPC 软排线脱落的的例子。

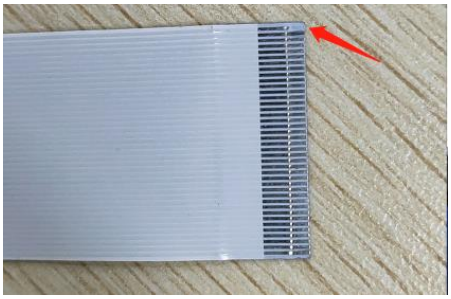


图 1.3.9-2 软排线金属片脱落

1.3.10 关机

关掉电源开关即可关机，如果有对系统的文件进行修改或者有替换系统的文件，关机前，先在串口终端输入指令“sync”同步一下缓存数据以防止数据丢失，执行完指令后再直接拨动电源开关关闭电源就可以了。（每次关机前，如果有对系统的文件进行修改，都建议执行一下“sync”指令同步一下缓存了再关机。）

```
Starting Linux NFC daemon
Starting crond: OK
Running local boot scripts (/etc/rc.local).

root@ATK-IMX6U:~# [   46.603334] random: nonblocking pool is initialized

root@ATK-IMX6U:~#
root@ATK-IMX6U:~#
root@ATK-IMX6U:~# vi /etc/rc.local
root@ATK-IMX6U:~# sync
root@ATK-IMX6U:~#
```

图 1.3.10-1 同步

如果是使用了第三版 Qt 的关机控件（后面出厂系统把这个功能取消了），如下图所示。

图 1.3.10-2 第三版 Qt 界面设置

使用此控件关机后，可能会导致开发板无法直接启动。可以检查下核心板上蓝色的电源灯有没有亮，可能核心板没在工作，上电后长按下 4G 模块丝印那附近的白色小按键就行（因为用户可能使用了 Qt 界面的关机或者使用指令关机，执行的是软关机，重新上电需要按下图 ON/OFF 键重启。或者取下 RTC 电池或重新插拔核心板也能启动。MINI 开发板也是同理，上电按 ON/OFF 按键，MINI 板有单独做这个按键）

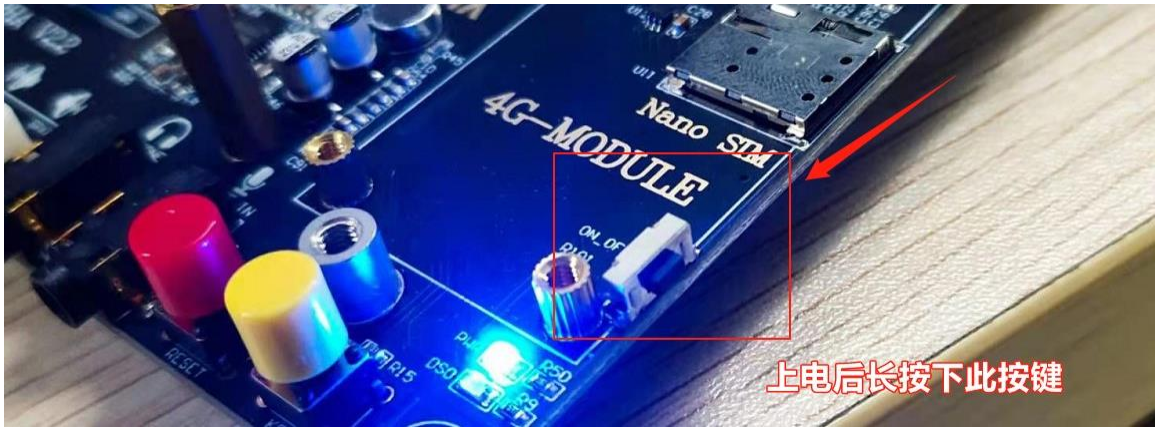


图 1.3.10-3 开发板软关机后上电

1.3.11 如何取下核心板

如果需要取下核心板或者更换核心板，如下图所示，大拇指按住核心板的两端，再取下核心板。如果有装了亚克力板的开发板，可以在亚克力板与核心板的空隙处取下核心板。

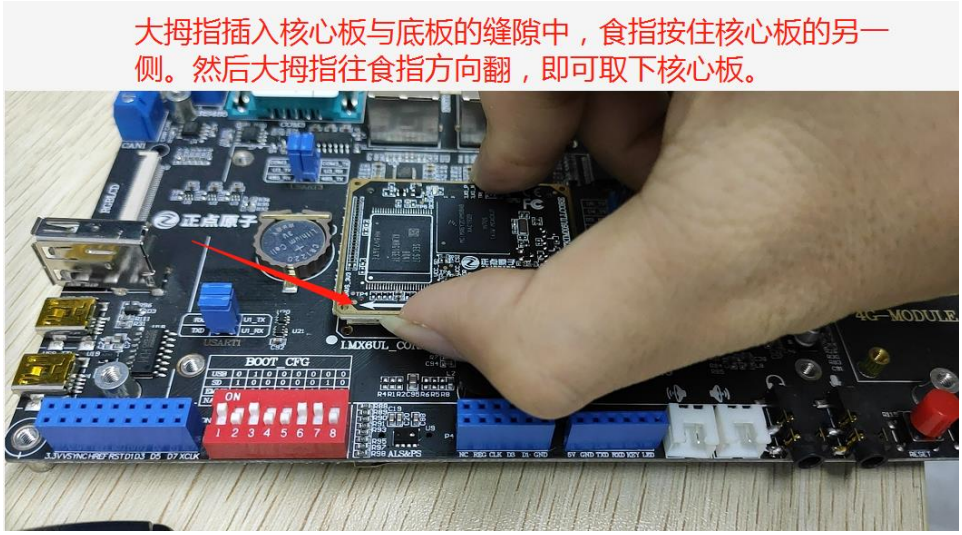


图 1.3.11-1 取下核心板

1.3.12 如何安装核心板

核心板上的丝印箭头方向要与底板的丝印箭头方向一致，将核心板插到 BTB 连接器上。

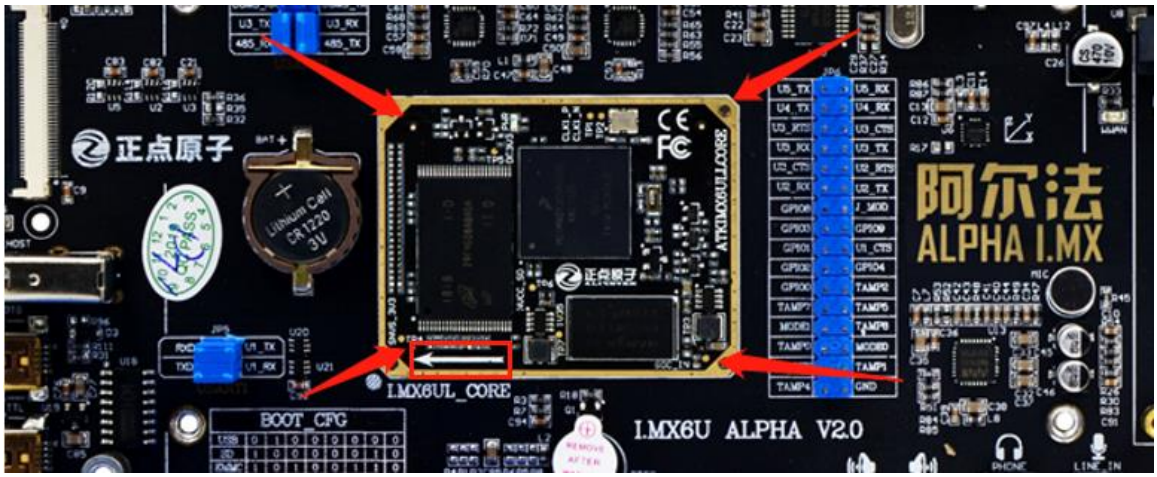


图 1.3.12-1 安装核心板

检查核心板 BTB 连接器是否与底板连接紧密！这个很重要！要不然可能出现花屏或者触摸不灵敏的情况！下图是错误的示例，可以看到核心板连接明显翘起，这样有部分座子接触不良！

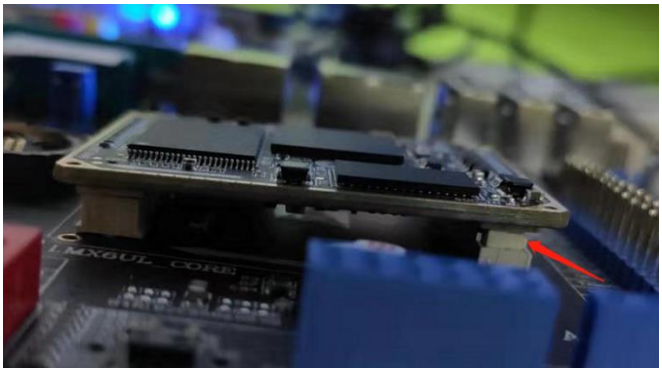


图 1.3.12-2 安装错误示例

1.3.13 其他外设检查

出货的时候正点原子已经将系统烧录到 EMMC 或者 NAND FLASH 了，那么可以使用出厂的系统进行简单的外设测试，可以按照文档《【正点原子】I.MX6U 用户快速体验》进行测试。

1.4 其他硬件问题

1.4.1 出厂系统 DDR 剩余容量问题

正点原子 IMX6U 核心板有 EMMC 和 NAND 版本，其中 EMMC 版本的 DDR 是 512M，NAND 版本的 DDR 是 256M。

启动系统后，在串口终端执行以下指令可以查看到剩余 DDR 容量：

```
cat /proc/meminfo
```

例如我这里使用的是 Nand Flash 版本的核心板，可以看到总 DDR 为 246968 kB，剩余 103324 kB。因为出厂系统版本不同，所以实际剩余 DDR 容量大小可能不同，以实际测试为准。

```
root@ATK-IMX6U:~#
root@ATK-IMX6U:~# cat /proc/meminfo
MemTotal:         246968 kB
MemFree:          103324 kB
MemAvailable:      89512 kB
Buffers:               0 kB
```

图 1.4.1-1 查看 DDR 容量

1.4.2 出厂系统 Flash 剩余容量问题

正点原子 IMX6U 核心板有 EMMC 和 NAND 版本，其中 EMMC 版本的 Flash 是 8G，NAND 版本的 Flash 是 512M。

启动系统后，在串口终端执行以下指令可以查看到剩余 Flash 容量：

```
df
```

例如我这里使用的是 Nand Flash 版本的核心板，可以看到剩余 Flash 为 146928kb。

```
root@ATK-IMX6U:~# df
Filesystem     1K-blocks    Used Available Use% Mounted on
ubi0:rootfs       444636  292872    146928  67% /
devtmpfs           57684       4     57680   1% /dev
tmpfs                 40       0        40   0% /mnt/.psplash
tmpfs             123484     132    123352   1% /run
tmpfs             123484     132    123352   1% /var/volatile
```

图 1.4.2-1 查看剩余 Flash

第二章 烧录系统问题

2.1 烧录出厂系统

参考用户快速体验或博客 https://blog.csdn.net/weixin_42004926/article/details/114229368

2.2 烧录自制系统

2.2.1 No Device Connected

操作：

用原子的烧录工具烧录自己的 uboot、设备树、内核、文件系统，卡在 No Device Connected。

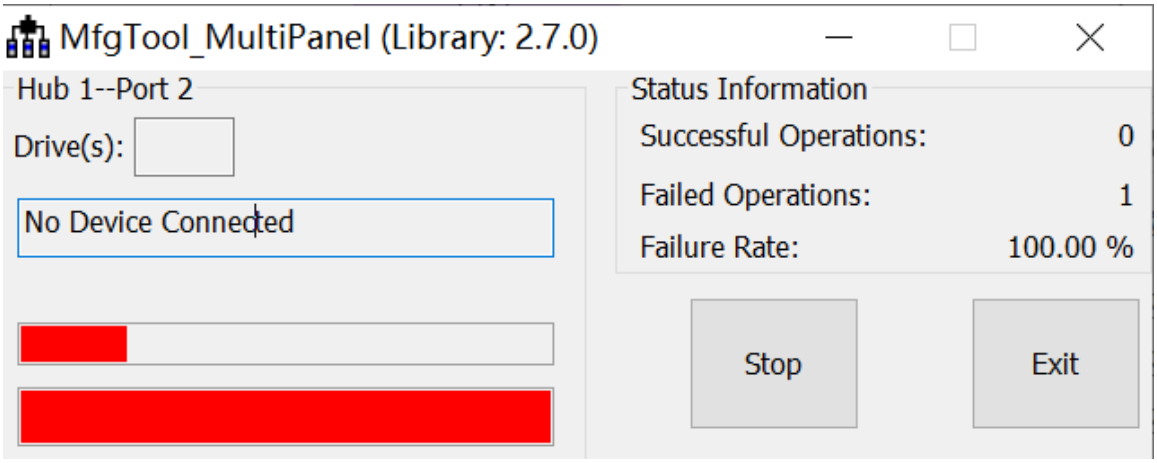


图 2.2.1-1 报错信息

解决：自己编译的 uboot、设备树、内核、文件系统，只能放在 mfgtools-with-rootfs/mfgtools/Profiles/Linux/OS Firmware/files 下对应的文件夹中，而且要重命名成和原烧录工具中文件名一样。

OS Firmware/firmware 不能动，要用官方的。

拓展：

如果想修改 firmware 烧录，需要修改内核配置文件选项。

```
make distclean
make imx_v7_mfg_defconfig
make zImage -j16
make dtbs
```

然后就可以替换 firmware 了，不过不推荐大家这样做，没啥必要。

2.3 其他问题

2.3.1 mfgtool2-yocto-mx-evk-emmc.vbs 无法打开

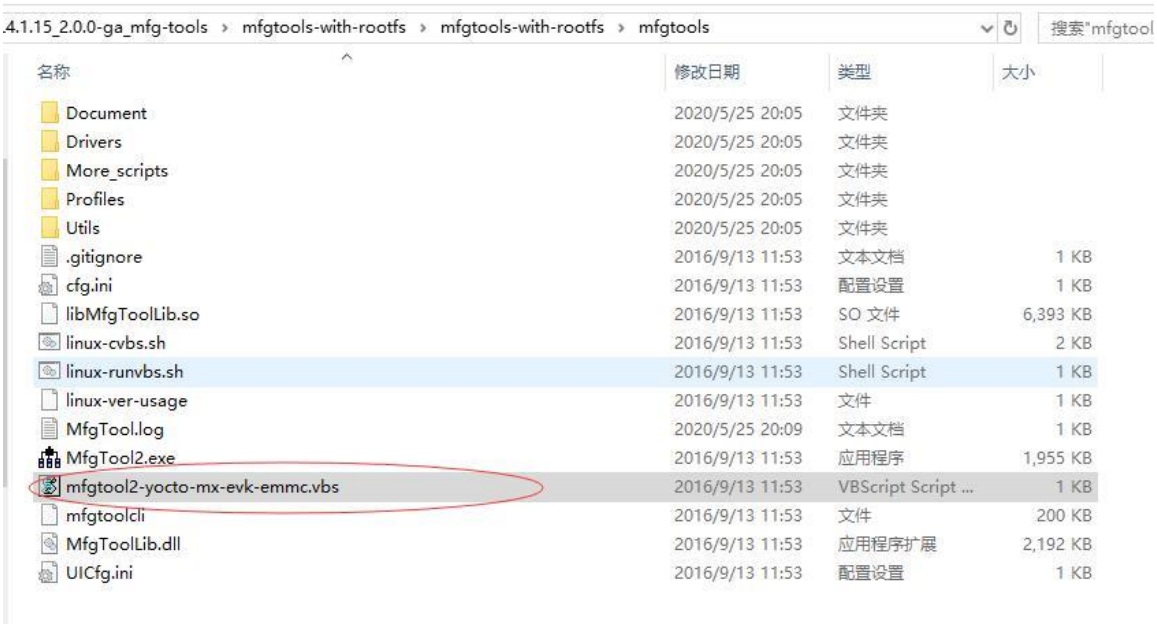


图 2.3.1-1 2.3.1 mfgtool2-yocto-mx-evk-emmc.vbs

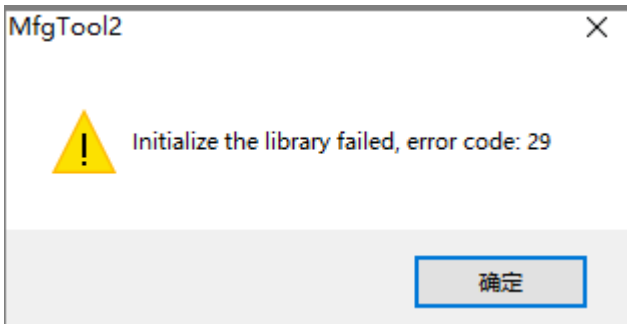


图 2.3.1-2 无法打开

解决方法：将 mfgtools-with-rootfs.tar.gz 这个压缩包文件传到 Ubuntu 中，在 Ubuntu 下解压后再拷贝到 windows 下就好了，就可以直接双击运行了。

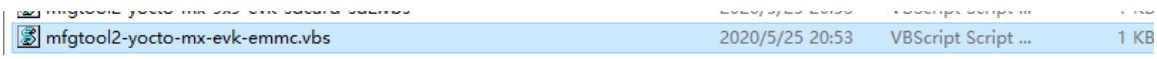


图 2.3.1-3 双击运行

第三章 环境搭建问题

3.1 安装 ubuntu 需要设置 BIOS

3.1.1 此主机支持 Intel VT-x,但 Intel VT-x 处于禁用状态

安装 Ubuntu 时报错：此主机支持 Intel VT-x,但 Intel VT-x 处于禁用状态。

图 3.1.1-1 安装时报错信息

PS：报错需要设置 BIOS 的情况会有很多，可以按照报错提示来在网上搜索如何设置 BIOS，这里只是列举其中一种例子。

解决办法：

重启系统，进入 BIOS（系统启动的过程中不停按住 F2 按键或者按住 Delete 按键进入 BIOS），在 CPU 设置那里找到 Intel (VMX) Virtualization Technology，将其开启，保存退出，再重启系统，打开虚拟机以后 Ubuntu 就可以继续成功安装了。（进入 BIOS 设置页面，在这里鼠标是不起作用的，通过左右方向键选中“Configuration”，再通过上下方向键选中“Intel Virtual Technology”，按“Enter”键，通过上下方向键选中“Enabled”，再按“Enter”键确认，最后按“F10”键保存并退出 BIOS 设置页面。然后重启计算机接着按照安装教程操作。）

有的电脑上 BIOS 界面这样：

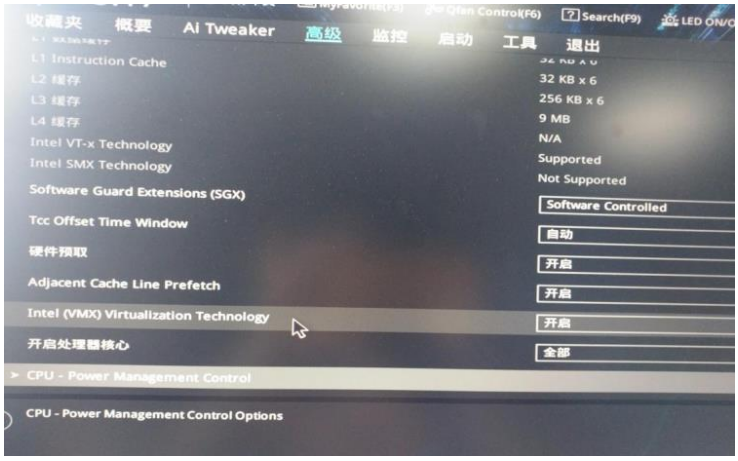


图 3.1.1-2 示例一

有的是这样的：

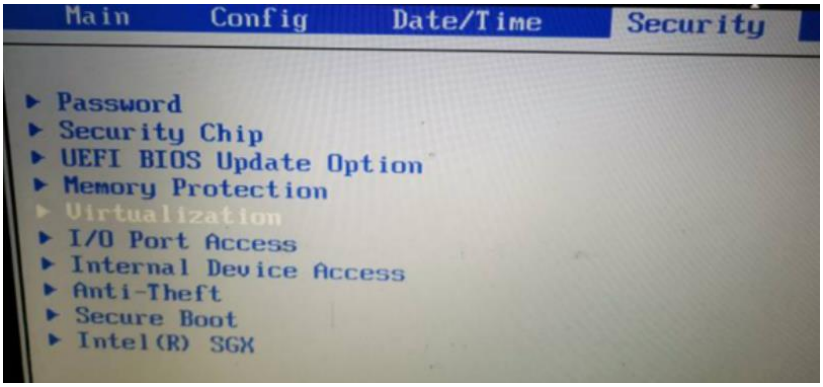


图 3.1.1-3 示例二

或者这样的：

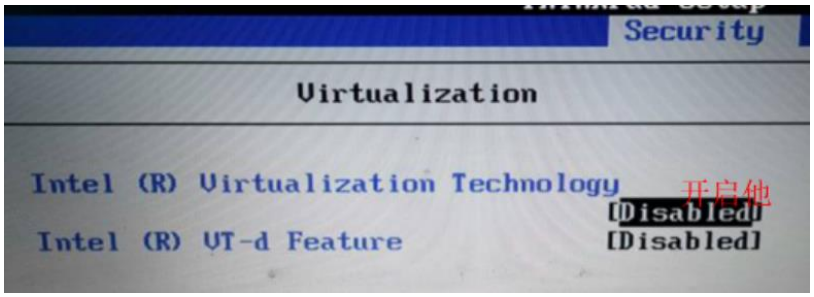


图 3.1.1-4 示例三

每台电脑的处理器不一样，BIOS 界面可能不一样，大家可以在百度上搜索下对应的解决方法。

3.2 VMware 虚拟机提示找不到 vmnetbridge.dll 文件

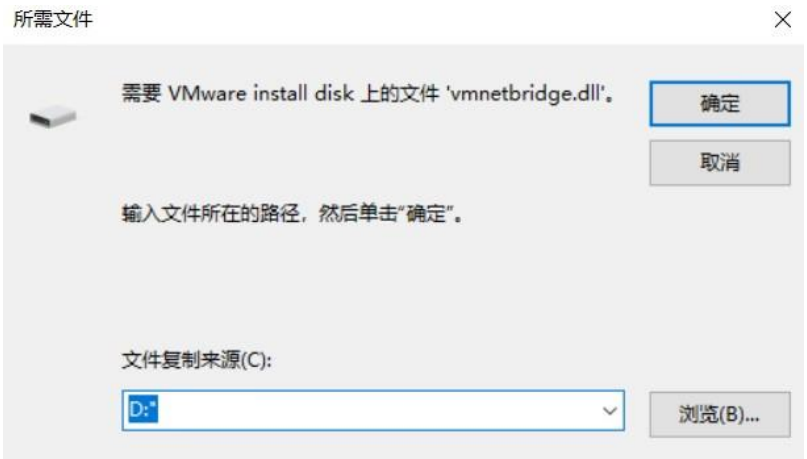


图 3.1.1-1 报错信息

解决：重装 VMware

3.3 Ubuntu 无法识别 USB 设置

参考如下方法：

3.3.1 开启 VMware USB Arbitration Service 服务

按下键盘的 win+r 键，输入 services.msc，点击确定，服务列表找到 VMware USB Arbitration Service 并双击，修改启动类型为自动，点击应用，然后点击启动，等待启动完成后，点击确定即可，如果设置完此项以后还是不行，可以试试重启电脑。

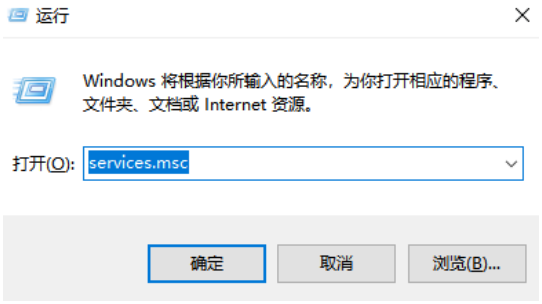


图 3.3.1-1 运行栏输入 services.msc

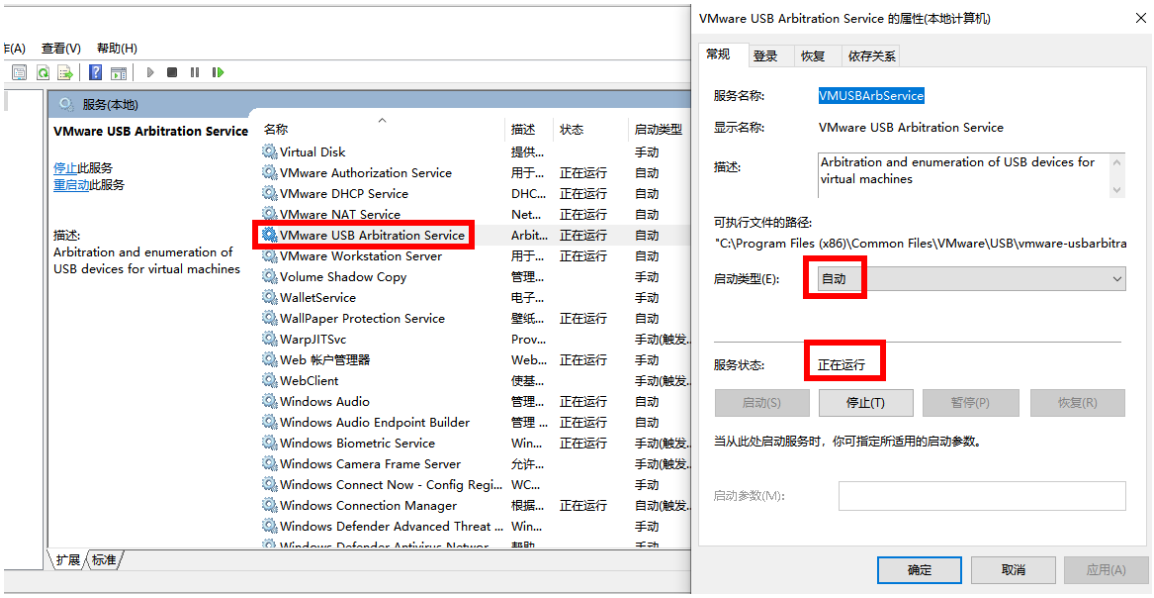


图 3.3.1-2 设置 VMware USB Arbitration Service 为自动

3.3.2 虚拟机设置 USB 兼容性 3.0

打开虚拟机设置那里，兼容性可以选 2.0 或者 3.0 试试。

图 3.3.2-1 设置 USB 兼容性

3.3.3 重装 VMware

以上方法都不行的话，可以考虑重新安装 VMWare（ubuntu 不用重新安装，只需要安装虚拟机就可以），有的电脑，可能需要安装其他版本的虚拟机才可以，例如 VMWare15.1.0。

3.4 SD 卡挂入 Ubuntu 报错不能挂载 16GB 卷

报错信息：

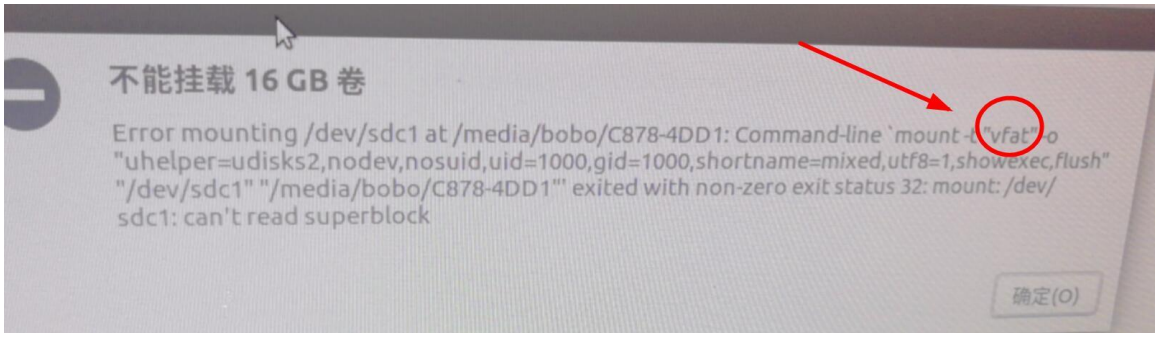


图 3.3.3-1 报错信息

解决：在 ubuntu 下格式化一下 SD 卡，执行以下指令。（这里报错/dev/sdc1，这此为例）

```
sudo mkfs.vfat -F 32    -n "boot" /dev/sdc1
```

3.5 Ubuntu 如何扩容

可以参考这个笔记的方法来扩容：https://bd37e1e0.wiz06.com/wapp/pages/view/share/s/2Zd-7w3jBN7G22Uj5K2eelKh0kSCzb3wDA_r2ALVYM38T1PP

3.6 修改完/etc/profile 后无法登陆 Ubuntu

3.6.1 修改完/etc/profile 后无法登陆 Ubuntu

问题：

按照教程按照交叉编译器，配置/etc/profile 时写错，重启 Ubuntu 后导致无法登陆 Ubuntu，一直停留在输入密码界面。

解决：

在 Ubuntu 登陆界面，按下 ctrl+alt+F1（有的需要同时按下 ctrl+alt+f1+fn），进入 tty1 非图形化界面。

输入你的账户名回车。注意：这里是帐户名，而不是密码。

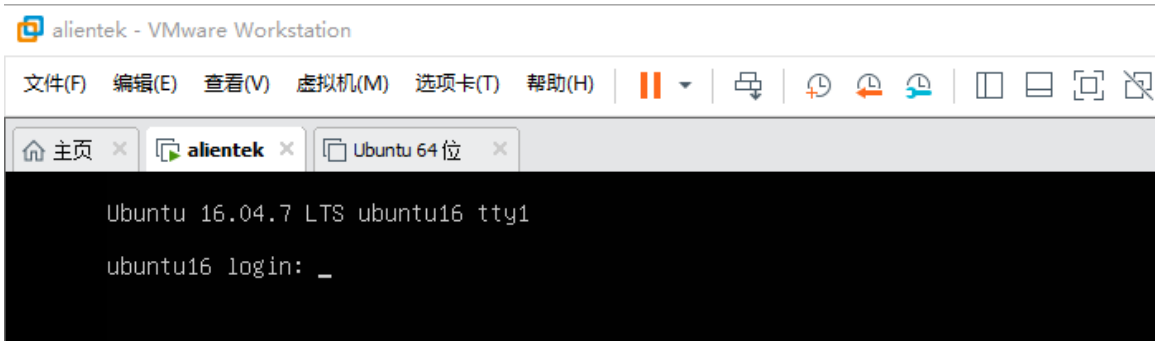


图 3.6.1-1 输入用户名

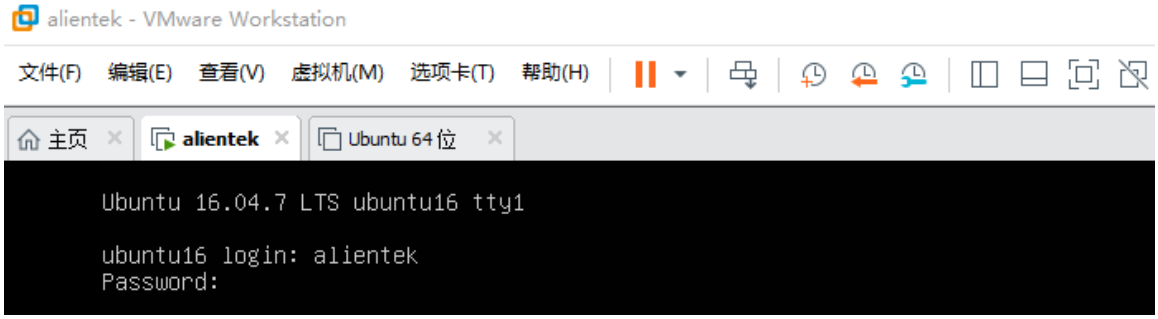


图 3.6.1-2 输入密码

输入你的密码回车 ok，注意：数字键盘默认是关闭的。此时就已经进入系统了。

执行命令 /usr/bin/sudo vi /etc/profile，打开 profile 修改成正确的内容。注意：该系统下无法直接使用 sudo，需要/usr/bin/sudo 指令。

图 3.6.1-3 打开文件

输入密码后打开文件，检查下添加的环境变量是否有问题，比如我这里把$打成 S 了，改过来就可以了。

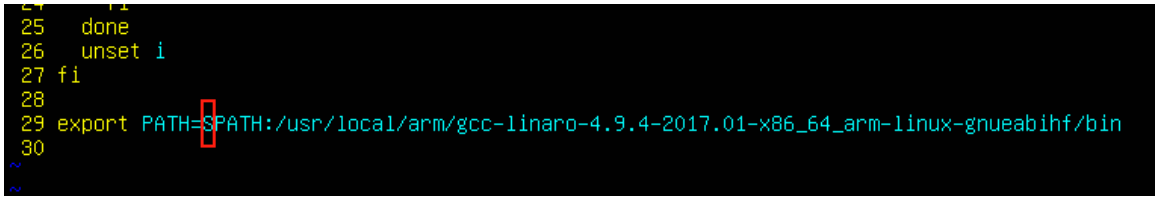


图 3.6.1-4 检查错误

修改好错误后，保存文件。重启 Ubuntu 即可正常开机。

3.7 关闭 Ubuntu 自动升级提示

正点原子 IMX6U 教程推荐使用的是 Ubuntu16 版本的，但是系统会老是会提示升级到 Ubuntu18 版本，这会导致后面出现很多问题，因此建议把系统升级提示关掉，防止不小心点到升级系统。

图 3.6.1-1 点击不升级系统

按照如下步骤关闭自动升级提示：

图 3.6.1-2 关闭自动升级提示

3.8 Ubuntu 无法上网

建议先按照《【正点原子】I.MX6U 网络环境 TFTP&NFS 搭建手册》的方法搭建网络环境。还是不行再参考以下方法。

3.8.1 Ubuntu 无法获取 IP 地址

如果是新安装好的 ubuntu，可能还无法上网，无法获取 IP 地址，需要开启网络服务。按下键盘的 win+r 键，输入 services.msc，点击确定，在服务列表找到 VMware NAT Service 并双击，修改启动类型为自动或手动，点击应用，然后点击启动，等待启动完成后，点击确定即可。

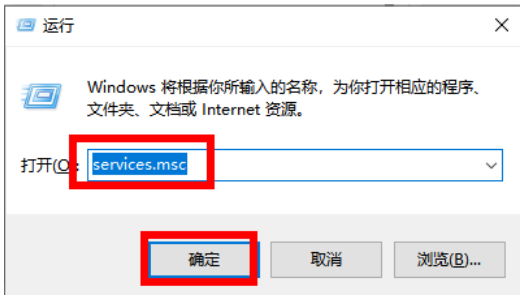


图 3.8.1-1 运行栏输入 services.msc

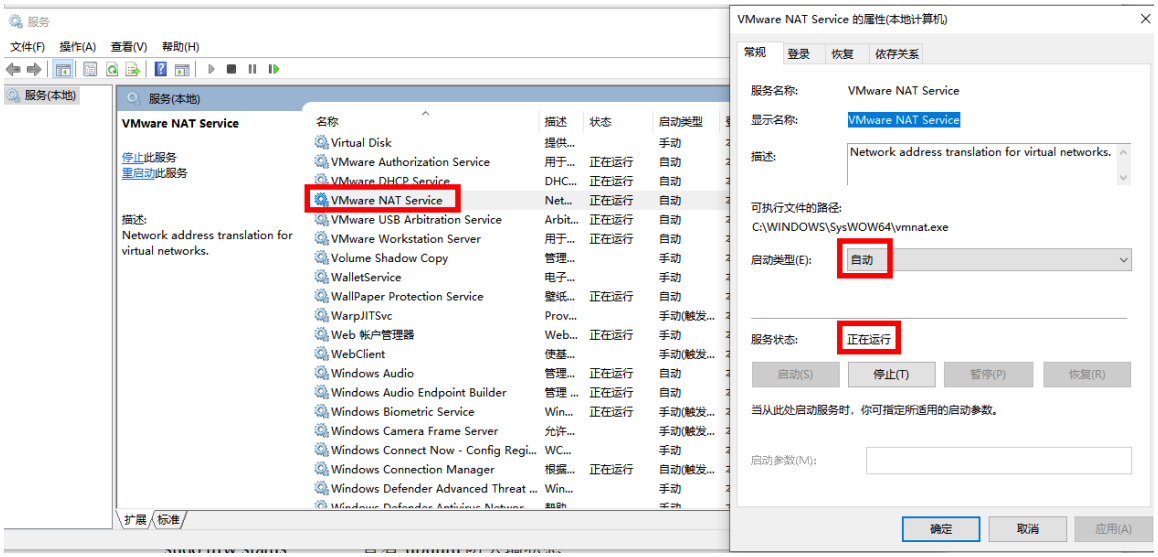


图 3.8.1-2 设置 VMware NAT Service 自动

检查看看 控制面板\所有控制面板项\网络连接下是否有网络适配器，例如这里有 VMnet1 和 VMnet8，如果没有，可能是虚拟机网卡没安装成功。

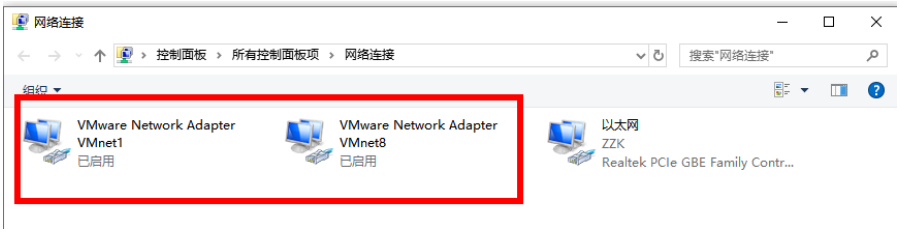


图 3.8.1-3 查看网络适配器

尝试点开虚拟机右上角的网络图标，手动点开网络连接。

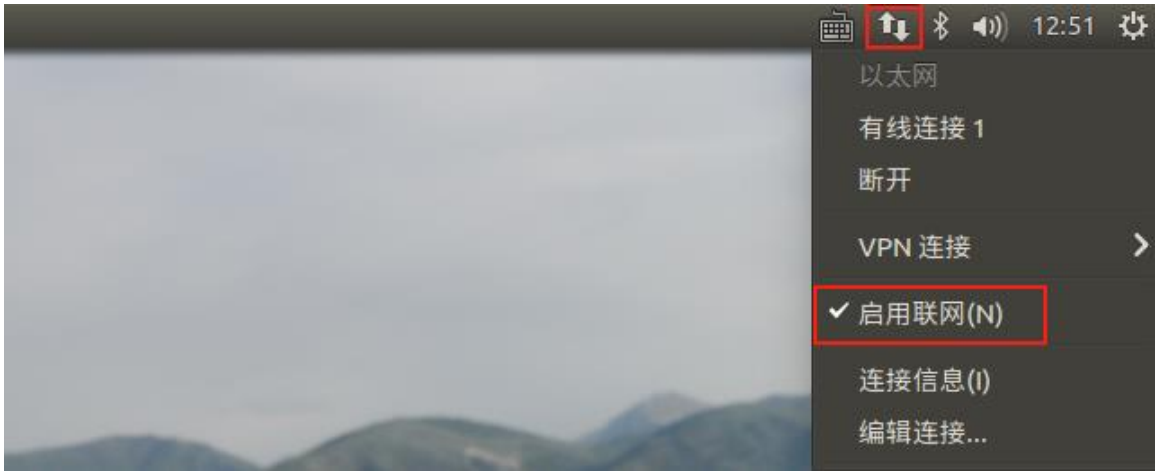


图 3.8.1-4 启动联网

如果还是不能联网，就使用虚拟网络编辑器的还原配置的功能，以下摘自网络搭建文档。打开菜单栏的 编辑 -> 虚拟网络编辑器。

图 3.8.1-5 打开虚拟网络编辑器

点击虚拟网络编辑器的 更改设置 选项。

图 3.8.1-6 更改设置

这个需要电脑管理员权限，如果有提示用户账户控制的提示框，点击 是 就可以了。

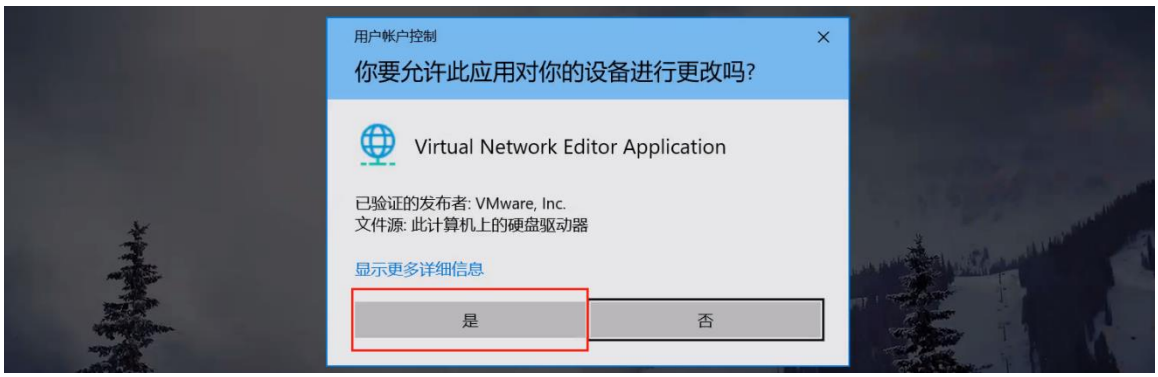


图 3.8.1-7 赋予修改权限

重新打开虚拟网络编辑器，可以看到如下界面。点击还原默认设置。

图 3.8.1-8 还原默认设置

这里会提示我们是否恢复到默认网络设置，点击 是 即可。

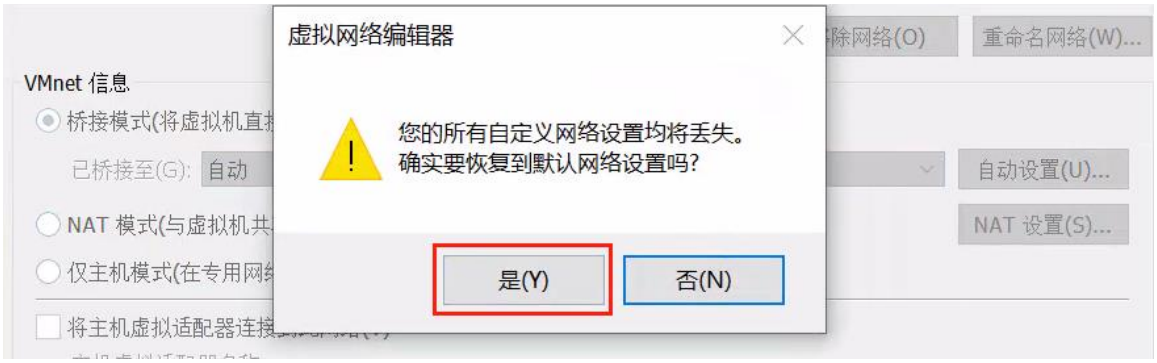


图 3.8.1-9 确认恢复到默认网络设置

恢复到默认网络设置后如下图所示，全部的 VMnet 子网地址会重新随机分配。这时候重启下虚拟机看看能不能上网。

3.8.2 没有未桥接的主机网络适配器

关闭虚拟机，点击 编辑---虚拟网络编辑器，查看网络适配器中没 VMnet0，且其它网络适配器设置为桥接的话会提示 **VM 无法将网络更改为桥接状态：没有未桥接的主机网络适配器。**

图 3.8.2-1 虚拟网络编辑器

可以参考链接尝试解决：

https://blog.csdn.net/czg13548930186/article/details/77099074

3.9 如何卸载 Ubuntu 上安装的 Qt creator

①进入 QT 安装的目录，如果安装的 QT 版本是 Qt5.3.1：例如我的目录：cd /home/warsllon/SoftWare/Qt5.3.1；

②运行命令：./MaintenanceTool；

③进入图形化界面，按照操作卸载即可。

或者在 qt create 的安装目录下，我的是/opt/qtqtcreator-1.3.0/bin/ 找到 unistall 文件 执行 sudo ./unistall

3.10 如何卸载 VSCode

```
sudo dpkg -r code
sudo dpkg --purge code
sudo dpkg --remove code
```

3.11 开发板和 Ubuntu 无法 ping 通

3.11.1 开发板 uboot 和 Ubuntu 无法 ping 通

建议先按照《【正点原子】I.MX6U 网络环境 TFTP&NFS 搭建手册》的方法搭建网络环境。还是不行再参考以下方法。注意：这里只介绍开发板 uboot 去 ping 虚拟机 Ubuntu 的 IP 地址，虚拟机是不需要去 ping 开发板 uboot 的 IP 地址的，因为 uboot 是没做反馈处理的。

①检查网线插在哪个网口，正点原子教程中 uboot 默认使用的是底板的 ETH2 网口，所以网线要插在底板的 ETH2 网口。

②检查 uboot 下的环境变量配置是否正确，主要是检查 ipaddr 和 ethaddr 这两个环境变量设置的是否正确，例如 ubuntu 的 IP 地址是 192.168.1.25，可以设置开发板的 IP 地址与 ubuntu 的 IP 地址在同一个网段（注意设置的 IP 地址要与当前网络其他设备的 IP 地址不要有冲突），例如设置开发板的 IP 地址是 192.168.1.100，开发板的 MAC 地址是 00:04:9f:04:d2:35，指令如下，指令尽量手打，避免因复制粘贴出现格式错误：

```
setenv ipaddr 192.168.1.100
setenv ethaddr 00:04:9f:04:d2:35
saveenv
```

③尝试关闭电脑和 ubuntu 的防火墙，一般 ubuntu 的防火墙是默认关着的，也有可能防火墙开启了：

```
sudo ufw status          //查看 ubuntu 防火墙状态
sudo ufw disable         //关闭 ubuntu 防火墙
```

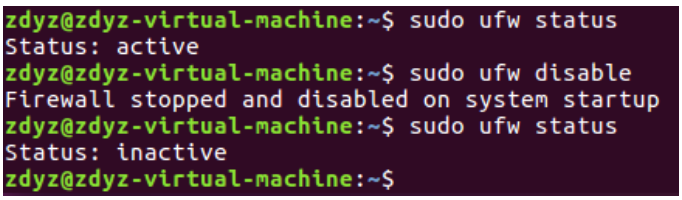


图 3.11.1-1 关闭防火墙

④如果是自己移植的 uboot，不排除移植的 uboot 哟普问题。可以用原子移植好的 uboot 试试，如果原子的 uboot 可以，自己移植的不行，在 ping 操作无差别的情况下，可能自己移植的 uboot 有问题。

⑤可能是硬件问题，可以用原子出厂的系统来 ping 测试，如果出厂的系统都无法 ping 通，检查硬件连接是否正常，检查网口的灯闪烁情况。底板网口，在没有插网线的情况下，只有橘色的灯在亮，插上网线以后，网口橘色的灯在闪烁而黄绿色的灯常亮。如果网口灯不亮，可能 RJ45 接口有问题或者其他，可以淘宝联系我们售后客服。注意：uboot 阶段是只有一个网口灯（ENET2）的，出厂系统启动后才有两个网口灯。

图 3.11.1-2 启动出厂系统后的网口灯

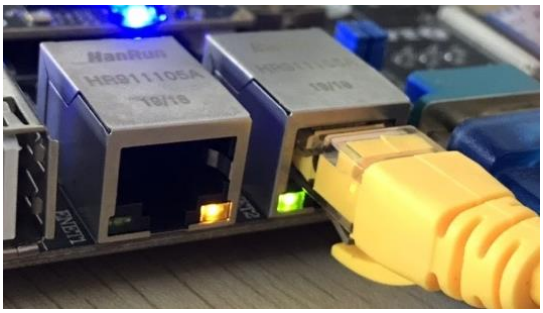


图 3.11.1-3 网口接入网线

3.11.2 开发板文件系统下无法 ping 通 Ubuntu

这部分比较复杂，建议直接参考《【正点原子】I.MX6U 网络环境 TFTP&NFS 搭建手册》的方法搭建网络环境。

3.12 开发板能 ping 通虚拟机，但是 TFTP\NFS 挂载失败

解决思路：

1、检查下 TFTP 或者 NFS 的配置、文件夹权限问题，bootcmd 和 bootargs 的格式、路径、权限。

2、防火墙问题，关闭电脑主机和虚拟机的防火墙，虚拟机执行 sudo ufw disable 关闭防火墙。

3、关闭防火墙也没用，可能防火墙有出站和入站规则，可能是这个规则使得无法传输文件，可以按照以下步骤。

尝试执行这个指令

```
sudo iptables -I INPUT -p udp --dport    69 -j ACCEPT
```

上述指令是开启对应的 69 端口，主板和主机的端口不一样可能会导致无法传输。

4、如果还不行，就卸载 Ubuntu 防火墙

```
sudo apt-get remove iptables
```

3.13 交叉编译器问题（arm-linux-gnueabihf-gcc）

3.13.1 交叉编译器没安装---'arm-linux-gnueabihf-gcc' is currently not installed

按照驱动开发指南教程 **4.3 Ubuntu 交叉编译工具链安装** 配置好/etc/profile 文件以后，如果找不到交叉编译器，可以先执行指令 source /etc/profile 来使能环境变量，再用指令 arm-linux-gnueabihf-gcc -v 查看是否已经有交叉编译器了，如果没有，检查一下/etc/profile 配置文件路径是否有些错的，配置指令尽量手打，避免复制粘贴出错（初学者容易把配置指令里的$打成 S，这样会导致重启 Ubuntu 时蓝屏），如果还是不行可以尝试重启 ubuntu。

如果提示没有安装交叉编译器，不建议用指令 sudo apt-get install gcc-arm-linux-gnueabihf 来安装别的版本的交叉编译器，因为用指令来下载的交叉编译器版本和教程的交叉编译器版本不一致，会导致后面更多的问题。

以下为常见报错：

The program 'arm-linux-gnueabihf-gcc' is currently not installed. To run 'arm-linux-gnueabihf-gcc' please ask your administrator to install the package 'gcc-arm-linux-gnueabihf'

```
MY@IMX6ULL:~$ arm-linux-gnueabihf-gcc -v
The program 'arm-linux-gnueabihf-gcc' is currently not installed. To run 'arm-linux-gnueabihf-gcc' please ask your administ
rator to install the package 'gcc-arm-linux-gnueabihf'
MY@IMX6ULL:~$ source /etc/profile
MY@IMX6ULL:~$ arm-linux-gnueabihf-gcc -v
The program 'arm-linux-gnueabihf-gcc' is currently not installed. To run 'arm-linux-gnueabihf-gcc' please ask your administ
rator to install the package 'gcc-arm-linux-gnueabihf'
MY@IMX6ULL:~$
```

图 3.13.1-1 报错没有安装交叉编译器

3.13.2 交叉编译器版本不对---提示 Resetting CPU

开发板启动提示 Resetting CPU …的问题

```
=> nfs 80800000 192.168.0.68:/home/yzy/Code/IMX6ULL/nfs/zImage
FEC1 Waiting for PHY auto negotiation to complete.... done
Using FEC1 device
File transfer via NFS from server 192.168.0.68; our IP address is 192.168.0.129
Filename '/home/yzy/Code/IMX6ULL/nfs/zImage'.
Load address: 0x80800000
Loading: data abort
pc : [<9ff83d70>]          lr : [<9ff83e18>]
reloc pc : [<8783bd70>]    lr : [<8783be18>]
sp : 9ef45450  ip : 00000040     fp : 0000006f
r10: 000003e8  r9 : 9ef45eb8     r8 : 9ffef570
r7 : 0000e803  r6 : 00006f00     r5 : 00000038  r4 : 9ffed70e
r3 : 14000045  r2 : 8100a8c0     r1 : 9ef45458  r0 : 9ffed70e
Flags: nZCv  IRQs off  FIQs off  Mode SVC_32
Resetting CPU ...

resetting ...


U-Boot 2016.03 (Mar 19 2020 - 11:26:05 +0800)

CPU:   Freescale i.MX6ULL rev1.1 69 MHz (running at 396 MHz)
CPU:   Industrial temperature grade (-40C to 105C) at 51C
Reset cause: WDOG
```

图 3.13.2-1 报错信息

这种是编译器版本问题导致的，驱动教程里用的编译器版本是 4.9.4 版本的，如果使用其它版本的交叉编译器，例如是用 7 版本的交叉编译器来编译 uboot 或者内核，就会报上面的错。输入 arm-linux-gnueabihf-gcc -v 查看交叉编译器版本，如果版本不是教程 4.9.4 的版本，是其它高版本的编译器，可以将高版本的交叉编译命令删除，再安装驱动教程里版本的编译器。要删

除原来高版本的交叉编译命令，先找到可执行文件。例如 7 版本的交叉编译命令，这个编译器名字是 arm-linux-gnueabihf-gcc-7，则执行指令查询它在哪个路径下：

```
which arm-linux-gnueabihf-gcc-7
```

执行指令后根据出来的路径，进入那个目录下，执行以下指令将可执行文件删除：

```
rm -rf arm-linux-gnueabihf-gcc*
```

最后再按照教程步骤检查，安装教程指定版本的交叉编译器。

```
MY@IMX6ULL:~$ arm-linux-gnueabihf-gcc -v
Using built-in specs.
COLLECT_GCC=arm-linux-gnueabihf-gcc
COLLECT_LTO_WRAPPER=/usr/local/arm/gcc-linaro-4.9.4-2017.01-x86_64_arm-linux-gnueabihf/bin/../libexec/gcc/arm-linux-gnueabihf/4.9.4/lto-wrapper
Target: arm-linux-gnueabihf
Configured with: /home/tcwg-buildslave/workspace/tcwg-make-release/label/docker-trusty-amd64-tcwg-build/target/arm-linux-gnueabihf/snapshots/gcc-linaro-
4.9-2017.01/configure SHELL=/bin/bash --with-mpc=/home/tcwg-buildslave/workspace/tcwg-make-release/label/docker-trusty-amd64-tcwg-build/target/arm-linux
-gnueabihf/_build/builds/destdir/x86_64-unknown-linux-gnu --with-mpfr=/home/tcwg-buildslave/workspace/tcwg-make-release/label/docker-trusty-amd64-tcwg-b
uild/target/arm-linux-gnueabihf/_build/builds/destdir/x86_64-unknown-linux-gnu --with-gmp=/home/tcwg-buildslave/workspace/tcwg-make-release/label/docker
-trusty-amd64-tcwg-build/target/arm-linux-gnueabihf/_build/builds/destdir/x86_64-unknown-linux-gnu --with-gnu-as --with-gnu-ld --disable-libmudflap --en
able-lto --enable-objc-gc --enable-shared --without-included-gettext --enable-nls --disable-sjlj-exceptions --enable-gnu-unique-object --enable-linker-b
uild-id --disable-libstdcxx-pch --enable-c99 --enable-clocale=gnu --enable-libstdcxx-debug --enable-long-long --with-cloog=no --with-ppl=no --with-isl=n
o --disable-multilib --with-float=hard --with-mode=thumb --with-tune=cortex-a9 --with-arch=armv7-a --with-fpu=vfpv3-d16 --enable-threads=posix --enable-
multiarch --enable-libstdcxx-time=yes --with-build-sysroot=/home/tcwg-buildslave/workspace/tcwg-make-release/label/docker-trusty-amd64-tcwg-build/target
/arm-linux-gnueabihf/_build/sysroots/arm-linux-gnueabihf --with-sysroot=/home/tcwg-buildslave/workspace/tcwg-make-release/label/docker-trusty-amd64-tcwg
-build/target/arm-linux-gnueabihf/_build/builds/destdir/x86_64-unknown-linux-gnu/arm-linux-gnueabihf/libc --enable-checking=release --disable-bootstrap
--enable-languages=c,c++,fortran,lto --build=x86_64-unknown-linux-gnu --host=x86_64-unknown-linux-gnu --target=arm-linux-gnueabihf --prefix=/home/tcwg-b
uildslave/workspace/tcwg-make-release/label/docker-trusty-amd64-tcwg-build/target/arm-linux-gnueabihf/_build/builds/destdir/x86_64-unknown-linux-gnu
Thread model: posix
gcc version 4.9.4 (Linaro GCC 4.9-2017.01)
MY@IMX6ULL:~$
```

图 3.13.2-2 教程的交叉编译器版本为 4.9.4

3.14 Filezilla 与虚拟机实现文件互传

3.14.1 Filezilla 与虚拟机连接不上的问题

Filezilla 与虚拟机连接不上一般排查步骤：

①查看站点管理的配置是否正确，比如主机（ubuntu）的 IP 地址和密码是否正确，端口可以试试 21 或者 22（默认端口是 21,），协议可以试试 SSH 或者 FTP。

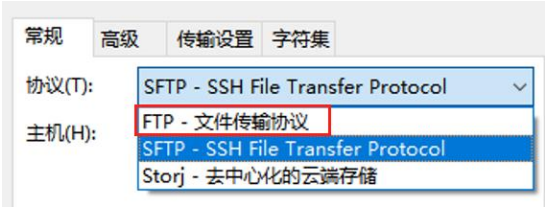


图 3.14.1-1 选择 FTP 或者 SFTP 协议

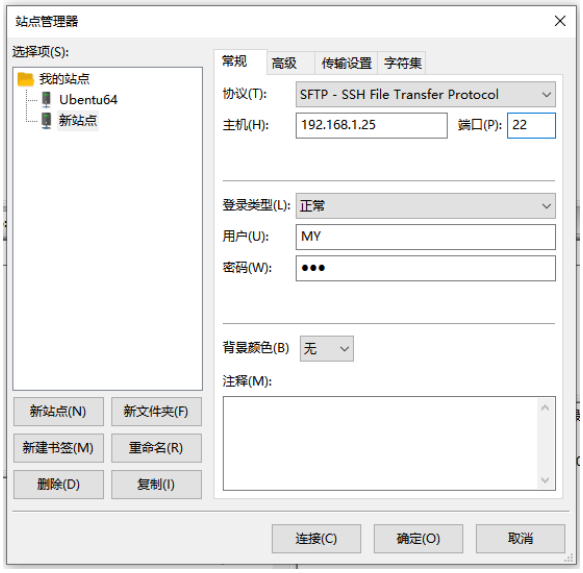


图 3.14.1-2 虚拟机主机信息

②查看 windows 和 ubuntu 之间是否可以 ping 通，如果不能 ping 通，解决 ping 通的问题。查看 ubuntu 的 IP 地址是不是会变化，VMWare 可以试试换桥接模式以及 NAT 模式，电脑可以上网的话，可以设置 ubuntu 为自动获取 IP，尝试关闭 windows 下的防火墙，同时也要检查 ubuntu 的防火墙。这里只是觉得关闭防火墙直接一些，也可以不关闭防火墙，直接设置防火墙入站规则（设置方法可以网上查询）。对于 ubuntu，一般 ubuntu 的防火墙是默认关着的，可能防火墙开启了。

```
sudo ufw status                 //查看防火墙状态
sudo ufw disable                //关闭防火墙
```

```
zdyz@zdyz-virtual-machine:~$ sudo ufw status
Status: active
zdyz@zdyz-virtual-machine:~$ sudo ufw disable
Firewall stopped and disabled on system startup
zdyz@zdyz-virtual-machine:~$ sudo ufw status
Status: inactive
zdyz@zdyz-virtual-machine:~$
```

图 3.14.1-3 关闭防火墙

③试试重启 FTP 与 SSH 服务器（Filezilla 下可以用这两个服务中的一个和 windows 进行通信，要用这两个服务的话，先按照教程搭建好 FTP 与 SSH 服务）

```
sudo /etc/init.d/vsftpd restart          //重启 FTP 服务
```

如下图，显示 OK，表示 FTP 服务开启了，然后 Filezilla 的协议可以选中 FTP-文件数传输协议 进行连接。

```
zdyz@zdyz-virtual-machine:~$ sudo /etc/init.d/vsftpd restart
[sudo] password for zdyz:
[ ok ] Restarting vsftpd (via systemctl): vsftpd.service.
zdyz@zdyz-virtual-machine:~$
```

图 3.14.1-4 开启 FTP 服务

```
/etc/init.d/ssh restart          //重启 SSH 服务
ps -e |grep ssh
```

如下图，显示 ssh，表示 SSH 服务开启了，如果仅有 agent，表示服务没开启。可以试试执行/etc/init.d/ssh restart 开启服务。开启 SSH 服务以后，Filezilla 的协议可以选中 SFTP-SSH File Transfer Protocol 进行连接。

```
zdyz@zdyz-virtual-machine:/mnt/hgfs/567$ /etc/init.d/ssh restart
[ ok ] Restarting ssh (via systemctl): ssh.service.
zdyz@zdyz-virtual-machine:/mnt/hgfs/567$ ps -e |grep ssh
 27205 ?        00:00:00 sshd
zdyz@zdyz-virtual-machine:/mnt/hgfs/567$
```

图 3.14.1-5 开启 SSH 服务

③如果 FTP 或者 SSH 服务开启失败，检查文件的配置，看看教程中需要修改的部分是否已经按照要求进行修改。

3.14.2 Filezilla 无法传输文件到 ubuntu

如果传输文件过程中提示：permission denied

如下，我想将文件传输至右边的 P1 目录，传输失败并提示

命令:　put "F:\mnt\1\2\record.sh" "record.sh"

错误: /home/MY/led/p1/record.sh: open for write: permission denied
错误: 文件传输失败
状态: 已从服务器断开

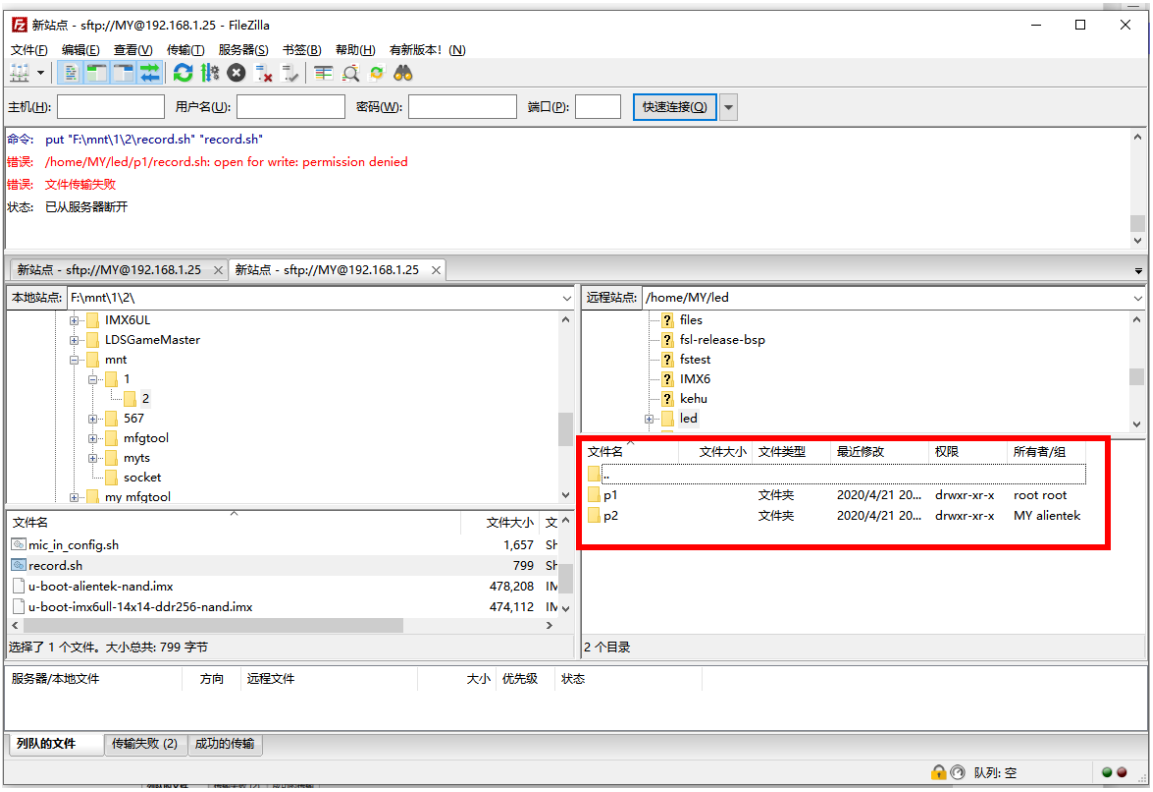


图 3.14.2-1 连接断开

从以上的提示中可以看出是权限问题，因为我的 ubuntu 的 所有者/组 是 root 用户，然而我目前 ubuntu 登录的是普通用户 MY，解决办法可以是：

① ubuntu 切换到 root 用户，再进行拷贝：

```
MY@IMX6ULL:~/led$ ls
p1  p2
MY@IMX6ULL:~/led$ ll
total 16
drwxr-xr-x  4 MY   alientek 4096  4月 21 20:29 ./
drwxr-xr-x 59 MY   alientek 4096  4月 21 21:06 ../
drwxr-xr-x  2 root root     4096  4月 21 20:29 p1/
drwxr-xr-x  2 MY   alientek 4096  4月 21 20:29 p2/
MY@IMX6ULL:~/led$ su root
Password:
root@IMX6ULL:/home/MY/led#
```

图 3.14.2-2 文件为 root 权限，需要切换到 root 用户

切换到 root 用户以后可以传输成功，如下，提示 状态:列出“/home/MY/led”的目录成功。

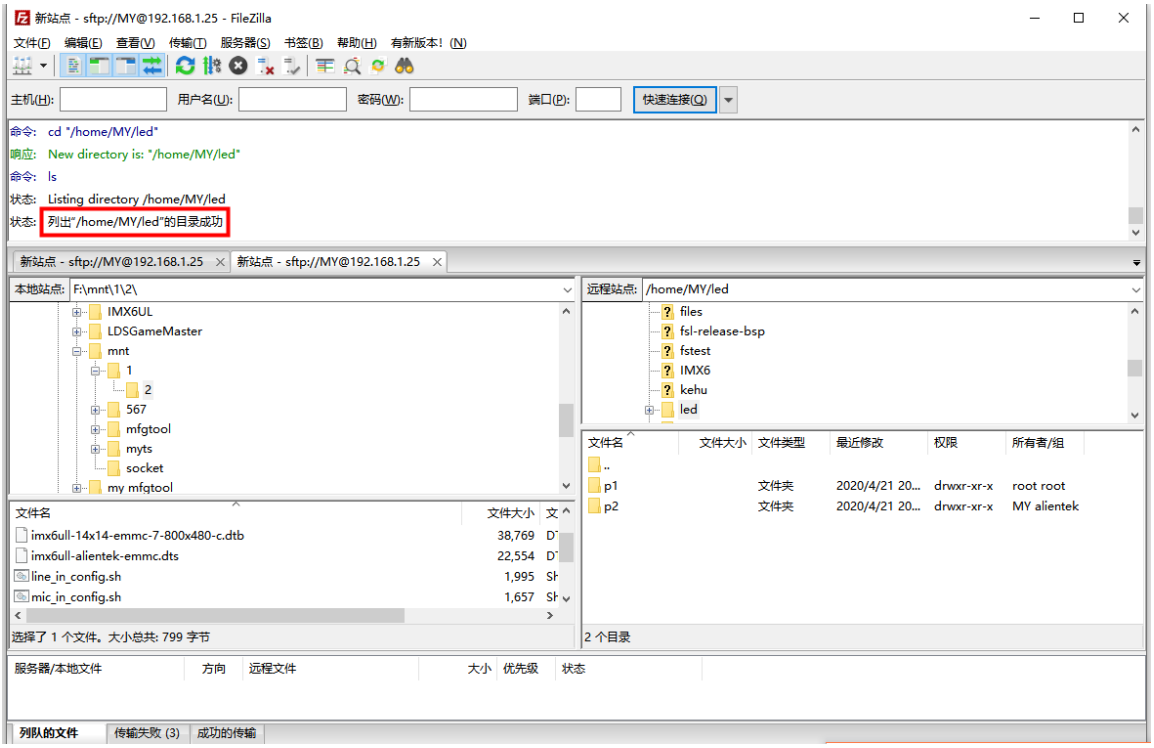


图 3.14.2-3 传输成功

②我们除了切换用户，还可以直接修改文件权限。上面的 P1 原来权限等级为 drwxr-xr-x，表示该目录所有组和其他组能对该目录进行读操作和删除等操作，但是不能进行写操作，所以导致我们无法拷贝文件。可以将其权限改为 777，即权限等级为 drwxrwxrwx，表示该目录所有者、所有组、其他组的成员都可以有权限对该目录进行访问和读写以及删除等操作。

```
MY@IMX6ULL:~/led$ ll
total 16
drwxr-xr-x  4 MY   alientek 4096  4月 21 20:29 ./
drwxr-xr-x 59 MY   alientek 4096  4月 21 21:06 ../
drwxr-xr-x  2 root root     4096  4月 21 20:29 p1/
drwxr-xr-x  2 MY   alientek 4096  4月 21 22:05 p2/
MY@IMX6ULL:~/led$ sudo chmod 777 /home/MY/led/p
p1/ p2/
MY@IMX6ULL:~/led$ sudo chmod 777 /home/MY/led/p1/
MY@IMX6ULL:~/led$ ll
total 16
drwxr-xr-x  4 MY   alientek 4096  4月 21 20:29 ./
drwxr-xr-x 59 MY   alientek 4096  4月 21 21:06 ../
drwxrwxrwx  2 root root     4096  4月 21 20:29 p1/
drwxr-xr-x  2 MY   alientek 4096  4月 21 22:05 p2/
MY@IMX6ULL:~/led$
```

图 3.14.2-4 修改文件权限

修改 P1 的权限等级为 drwxrwxrwx 后，传输成功：

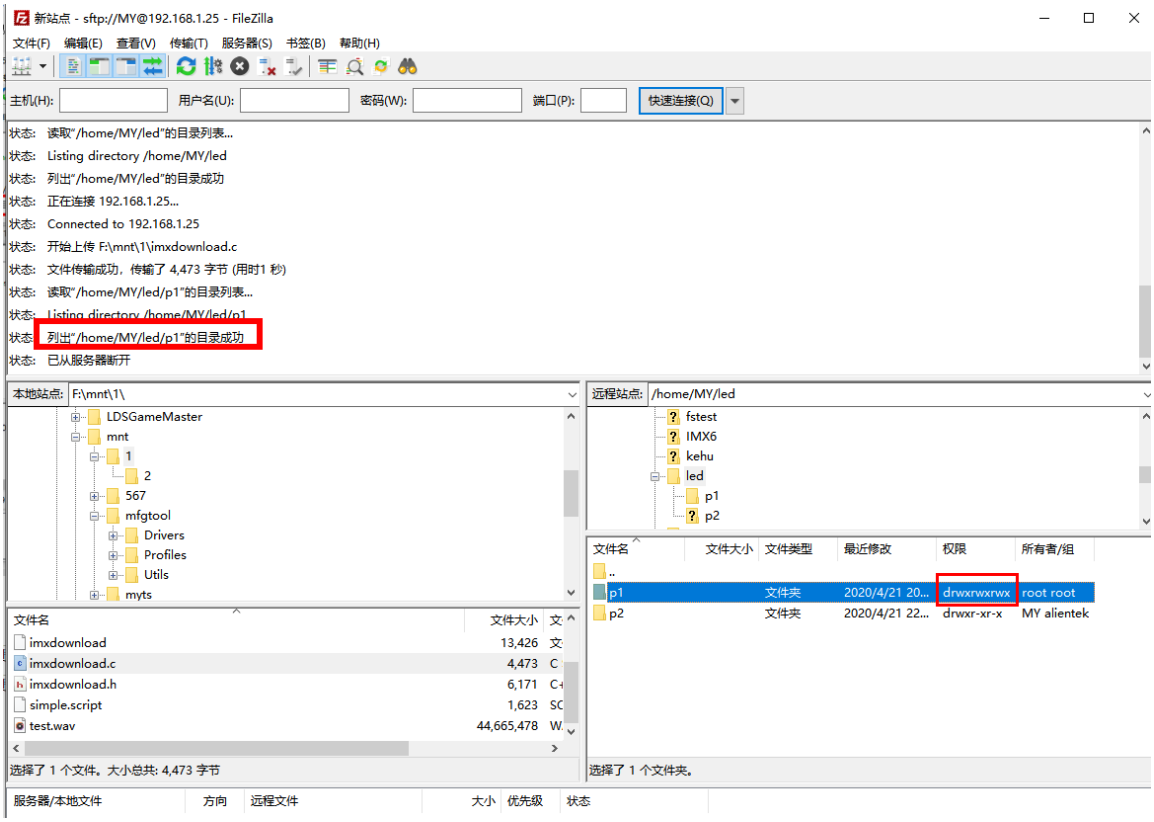


图 3.14.2-5 传输成功

如果是普通用户的目录，比如我上面的 P2，这个显示的所有者/组 分别是 MY 和 alientek，这两个都是普通用户，无须考虑权限问题，可以直接传输。

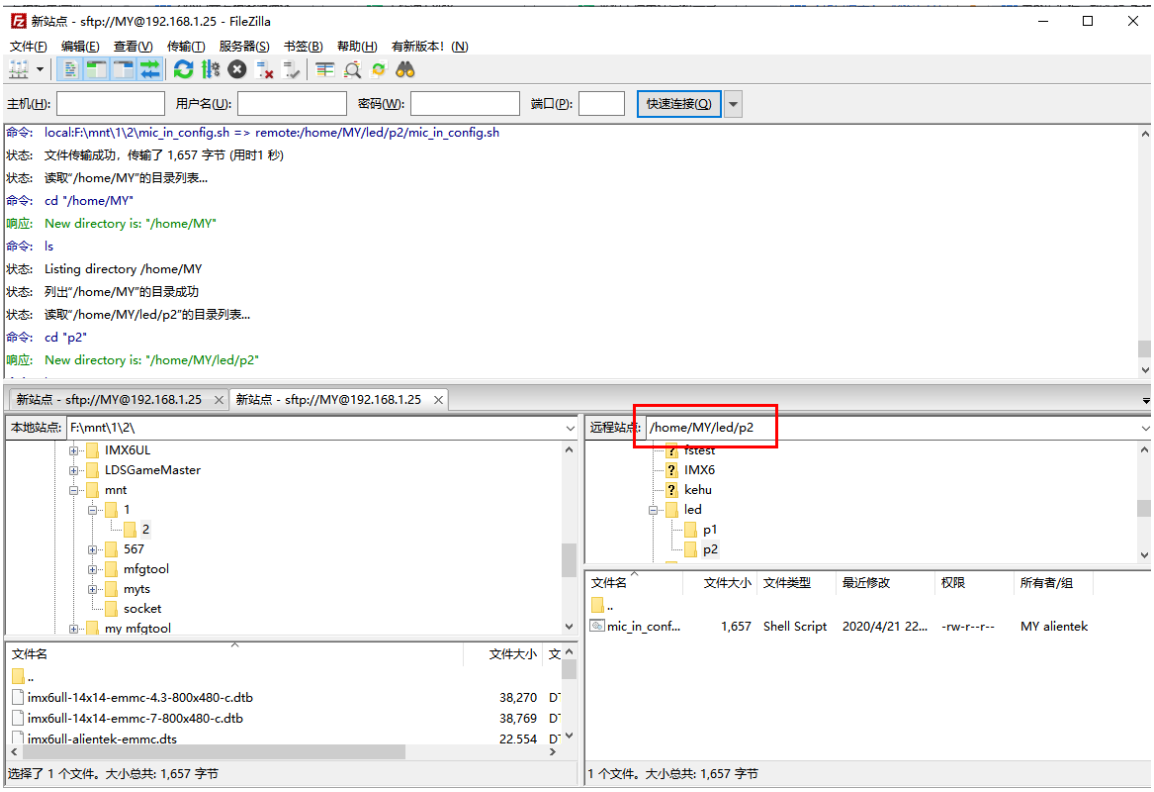


图 3.14.2-6 传输成功

3.14.3 响应：553 Could not create file 错误：严重文件传输错误

报错信息：

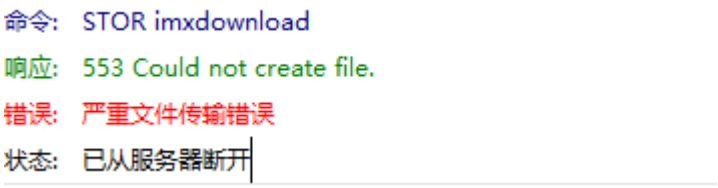


图 3.14.3-1 报错信息

原因：在 Ubuntu 中创建命令时，使用 sudo mkdir ，切换到了 root 用户，与 FTP 连接到用户不同，无法传输文件。

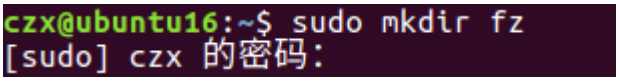


图 3.14.3-2 不小心使用 sudo 权限创建目录

解决：创建目录时，使用 mkdir 命令即可。或者直接在 filezilla 软件中创建目录。

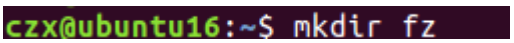


图 3.14.3-3 不使用 sudo 权限创建目录

或者直接在 Filezilla 软件上创建目录。

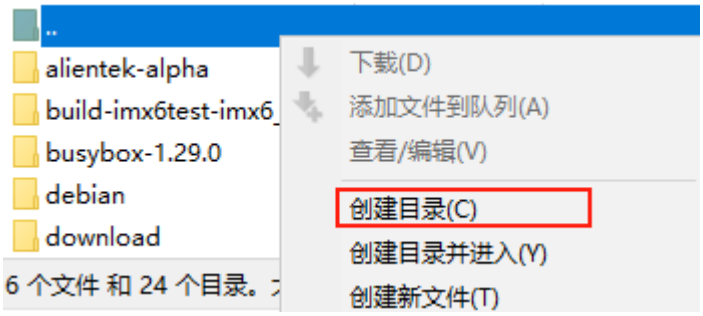


图 3.14.3-4 使用 Filezilla 创建目录

如果不是创建目录的问题，检查下传输目标路径是否在用户空间目录（/home/用户名）下，如果与 filezilla 登录的用户不同，则会报这个错。比如 filezilla 登录的是个人用户，但传输文件去到了 Ubuntu 的根目录，就会报错。

图 3.14.3-5 示例

如图，filezilla 登录的是 czx 用户，文件传到 Ubuntu 根目录下则出错。解决办法就是在用户目录下新建目录，把文件发到这个新建目录下。

3.15 段错误 （核心已转储）

3.15.1 运行/opt/Qt5.5.1/Tools/QtCreator/bin/qtcreator.sh &报错段错误

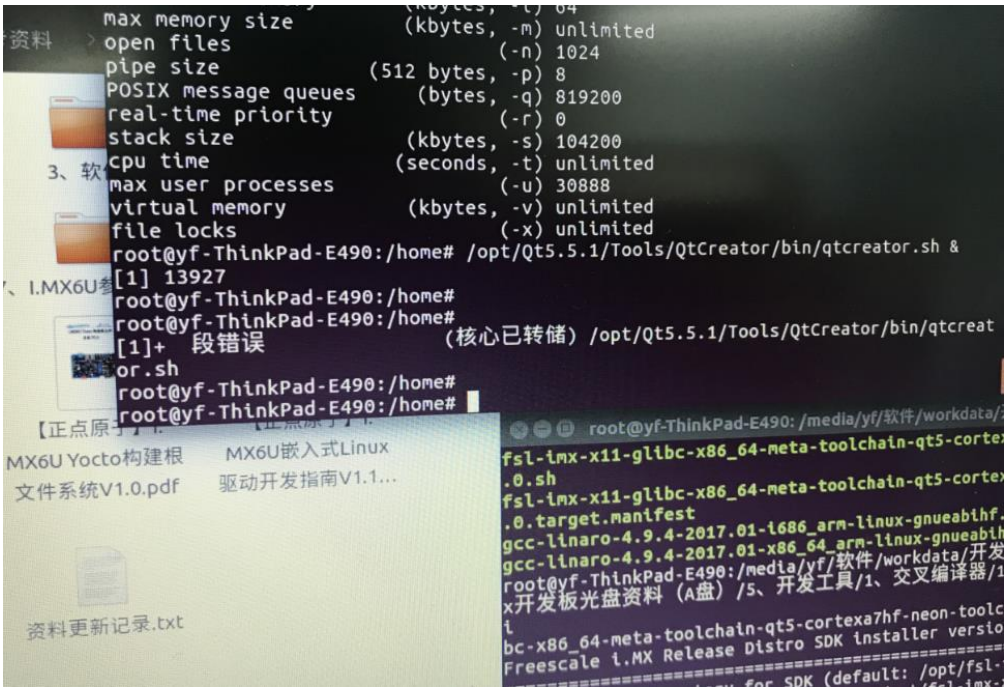


图 3.15.1-1 报错段错误

报核心已转储，段错误的，可以按照以下检查：

图 3.15.1-2 在脚本前添加使能环境

3.16 sudo apt 安装软件问题

3.16.1 安装软件时，出现死锁问题 Could not get lock

用 sudo apt-get install 安装软件时，会出现一下的情况

```
E: Could not get lock /var/lib/dpkg/lock - open (11: Resource temporarily unavailable)
E: Unable to lock the administration directory (/var/lib/dpkg/), is another process using it?
```

```
miklewu@Miklewu-virtual-machine:~$ sudo apt-get install vsftpd
E: 无法获得锁 /var/lib/dpkg/lock-frontend - open (11: 资源暂时不可用)
E: Unable to acquire the dpkg frontend lock (/var/lib/dpkg/lock-frontend), is an
other process using it?
miklewu@Miklewu-virtual-machine:~$
```

图 3.16.1-1 报错死锁

用 ps -a 查看一下是否像提示那样真的有进程在占用，检查是否在运行 apt,apt-get 相关的进程，如果查出来占用就 kill 掉进程，比如查询到进程 pid 号是 123，那么久执行指令杀掉占用的进程

```
sudo kill 123
```

杀掉进程的指令：

```
ps -A | grep apt
sudo kill        //线程 id 号
```

或者最简单的解决办法就是重启大法：重启 ubuntu 就好。

3.16.2 Unable to locate package hexdump

```
MY@IMX6ULL:~$ sudo apt-get install hexdump
[sudo] password for MY:
Reading package lists... Done
Building dependency tree
Reading state information... Done
E: Unable to locate package hexdump
MY@IMX6ULL:~$
```

图 3.16.2-1 hexdump 警告

出现无法定位软件包 hexdump 的警告。一般出现这种报错的，可以考虑一下，是不是要安装的这个东西名字写错了还是之前更新了源以后，没有执行 sudo apt-get update，所以导致无法定位？

解决办法：

（1）检查 ubuntu 是否可以上网，是否是因为无法上网引起

（2）执行 sudo apt-get update 来使之前更新的源在本地更新一下

（3）重装系统后，或者刚装好系统，软件源还来不及更新，系统没有最新的软件包列表，所以需要执行 sudo apt-get update 才行

（4）执行上面的更新指令以后仍不成功的话，那就很有可能是安装包名字写错了

经过检查发现，原来是安装包的名字写错了，真正的安装包名字是：libdata-hexdumper-perl，所以执行指令 sudo apt-get install libdata-hexdumper-perl 来安装。

（5）最后尝试换源

3.16.3 文件名列表文件缺少最后结尾的换行符

操作：安装软件包时报错 文件名列表文件缺少最后结尾的换行符

报错信息：

```
下列【新】软件包将被安装：
  tftp-hpa tftpd-hpa
升级了 0 个软件包，新安装了 2 个软件包，要卸载 0 个软件包，
需要下载 0 B/57.5 kB 的归档。
解压缩后会消耗 172 kB 的额外空间。
正在预设定软件包 ...
正在选中未选择的软件包 tftp-hpa。
dpkg: 无法恢复的致命错误，中止：
 软件包 usbutils 的文件名列表文件缺少最后结尾的换行符
E: Sub-process /usr/bin/dpkg returned an error code (2)
```

图 3.16.3-1 报错信息

解决：

```
sudo su
cd /var/lib/dpkg/info
ls
```

//我这是因为 gtk 文件名列表文件出问题了，我查看相关文件有四个：gtkterm.listgtkterm.md5sums，gtkterm.postinst，gtkterm.postrm

```
rm gtkterm.* //将相关文件全删除
sudo apt upgrade //成功
```

参考 https://blog.csdn.net/dengshuai_super/article/details/52034555

3.16.4 E: 无法修正错误，因为您要求某些软件包保持现状，就是它们破坏了软件包间的依赖关系

```
czx@ubuntu16:~/test/QWebBrowser/build-QWebBrowser-imx6-Debug$ sudo apt install l
ibgstreamer0.10-dev
[sudo] czx 的密码：
正在读取软件包列表... 完成
正在分析软件包的依赖关系树
正在读取状态信息... 完成
有一些软件包无法被安装。如果您用的是 unstable 发行版，这也许是
因为系统无法达到您要求的状态造成的。该版本中可能会有一些您需要的软件
包尚未被创建或是它们已被从新到(Incoming)目录移出。
下列信息可能会对解决问题有所帮助：

下列软件包有未满足的依赖关系：
 libgstreamer0.10-dev : 依赖: libxml2-dev 但是它将不会被安装
E: 无法修正错误，因为您要求某些软件包保持现状，就是它们破坏了软件包间的依赖关系
。
```

图 3.16.4-1 报错信息

解决：

打开 Ubuntu 的“软件和更新”选项卡，选择“更新”

勾选“重要安全更新”和“推荐更新”

图 3.16.4-2 选择更新

在“其他软件”选项卡中，去掉有问题等源，没有可以不做这步。

保存，关闭。

在命令行输入命令 sudo apt-get update

再重新下载需要的安装包即可成功。

```
czx@ubuntu16:~/test/QWebBrowser/build-QWebBrowser-imx6-Debug$ sudo apt install l
ibgstreamer0.10-dev
正在读取软件包列表... 完成
正在分析软件包的依赖关系树
正在读取状态信息... 完成
将会同时安装下列软件：
  gir1.2-gstreamer-0.10 icu-devtools libgstreamer0.10-0 libicu-dev libxml2-dev
建议安装：
  gstreamer0.10-tools gstreamer0.10-plugins-base gstreamer0.10-doc icu-doc
下列【新】软件包将被安装：
  gir1.2-gstreamer-0.10 icu-devtools libgstreamer0.10-0 libgstreamer0.10-dev
  libicu-dev libxml2-dev
升级了 0 个软件包，新安装了 6 个软件包，要卸载 0 个软件包，有 125 个软件包未被升
级。
需要下载 11.0 MB 的归档。
解压缩后会消耗 55.4 MB 的额外空间。
您希望继续执行吗？ [Y/n]
获取:1 http://security.ubuntu.com/ubuntu xenial-security/main amd64 icu-devtools
 amd64 55.1-7ubuntu0.5 [166 kB]
```

图 3.16.4-3 安装成功

或者按照报错信息执行以下执行：

sudo apt-get -f install

sudo apt-get -f install 软件包名称

3.17 VIM 问题

3.17.1 Found a swap file by the name "/etc/.vsftpd.conf.swp"

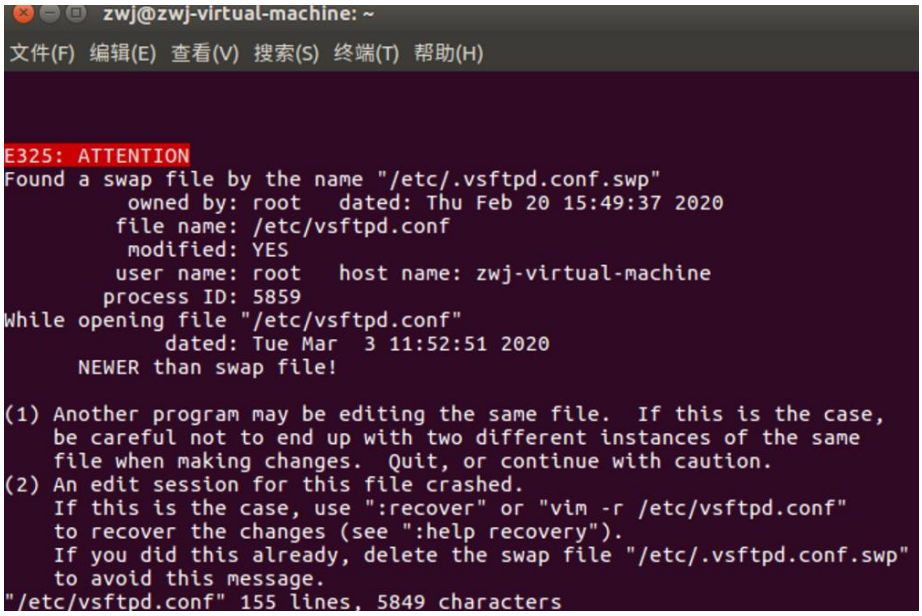


图 3.17.1-1 提示信息

原因：可能是由于上一次在编辑 vsftpd.conf 时，文件没有退出就直接（强行）重启或关机造成的，产生.vsftpd.conf.swp 文件。

解决方法：在 vsftpd.conf 文件所在的目录，执行 ls -a，会发现该文件，删除该文件—

```
rm /etc/vsftpd/.vsftpd.conf.swp/
```

第四章 裸机相关问题

4.1 裸机程序编译报错

4.1.1 Makefile:4: *** missing separator. Stop. 或者提示 Makefile:4:***空变量名。停止。

图 4.1.1-1 Makefile:4:***空变量名

解答：

Makefile 的命令行里该按下 TAB 的地方写成了空格了或者顶格写了，一定要 TAB 键开头：

```
objs := start.o main.o

ledc.bin:$(objs)
arm-linux-gnueabihf-ld -Timx6ul.lds -o ledc.elf $^
	arm-linux-gnueabihf-objcopy -O binary -S ledc.elf $@
	arm-linux-gnueabihf-objdump -D -m arm ledc.elf > ledc.dis

%.o:%.s
	arm-linux-gnueabihf-gcc -Wall -nostdlib -c -O2 -o $@ $<

%.o:%.S
	arm-linux-gnueabihf-gcc -Wall -nostdlib -c -O2 -o $@ $<

%.o:%.c
	arm-linux-gnueabihf-gcc -Wall -nostdlib -c -O2 -o $@ $<

clean:
	rm -rf *.o ledc.bin ledc.elf ledc.dis
```

图 4.1.1-2 命令前没有使用 Tab 键

按照 Makefile 的格式要求来编写，将命令行出进行 TAB 键开头，修改如下，修改好后编译通过：

```
objs := start.o main.o

ledc.bin:$(objs)
	arm-linux-gnueabihf-ld -Timx6ul.lds -o ledc.elf $^
	arm-linux-gnueabihf-objcopy -O binary -S ledc.elf $@
	arm-linux-gnueabihf-objdump -D -m arm ledc.elf > ledc.dis

%.o:%.s
	arm-linux-gnueabihf-gcc -Wall -nostdlib -c -O2 -o $@ $<

%.o:%.S
	arm-linux-gnueabihf-gcc -Wall -nostdlib -c -O2 -o $@ $<

%.o:%.c
	arm-linux-gnueabihf-gcc -Wall -nostdlib -c -O2 -o $@ $<

clean:
	rm -rf *.o ledc.bin ledc.elf ledc.dis
```

图 4.1.1-3 在命令前加入 Tab 键

如果 Tab 键显示的格数过长或者还是提示一样的报错，请设置 VIM 的 Tab 键为 4 格。

4.1.2 error:no such instruction 错误

报错信息如下：

```
as      -o start.o start.s
start.s: Assembler messages:
start.s:21: Error: no such instruction: `mrs r0,cpsr'
start.s:22: Error: no such instruction: `bic r0,r0,'
start.s:23: Error: no such instruction: `orr r0,r0,'
start.s:24: Error: no such instruction: `msr cpsr,r0'
start.s:26: Error: no such instruction: `ldr sp,=0X80200000'
start.s:27: Error: no such instruction: `b main'
make: *** [start.o] Error 1
```

```
MY@IMX6ULL:~/program/2_ledc$ ls
imx6ul.lds  imxdownload  ledc.code-workspace  load.imx  main.c  main.h  Makefile  s  start.s
MY@IMX6ULL:~/program/2_ledc$ make
as   -o start.o start.s
start.s: Assembler messages:
start.s:21: Error: no such instruction: `mrs r0,cpsr'
start.s:22: Error: no such instruction: `bic r0,r0,'
start.s:23: Error: no such instruction: `orr r0,r0,'
start.s:24: Error: no such instruction: `msr cpsr,r0'
start.s:26: Error: no such instruction: `ldr sp,=0X80200000'
start.s:27: Error: no such instruction: `b main'
make: *** [start.o] Error 1
MY@IMX6ULL:~/program/2_ledc$
```

图 4.1.2-1 no such instruction

解答：

在 Makefile 中 start.s 的这个 s 大小写写错了，查看文件名是 start.s，但是 Makefile 下却写得是大写的，所以编译不通过。注意的是，在 Linux 下和 Windows 下是不同的，Linux 下严格区分大小写。如下，将箭头的大写的 S 改成小写的 s 就可以解决问题：

```
objs := start.o main.o

ledc.bin:$(objs)
        arm-linux-gnueabihf-ld -Timx6ul.lds -o ledc.elf $^
        arm-linux-gnueabihf-objcopy -O binary -S ledc.elf $@
        arm-linux-gnueabihf-objdump -D -m arm ledc.elf > ledc.dis


%.o:%.S
        arm-linux-gnueabihf-gcc -Wall -nostdlib -c -O2 -o $@ $<

%.o:%.c
        arm-linux-gnueabihf-gcc -Wall -nostdlib -c -O2 -o $@ $<

clean:
        rm -rf *.o ledc.bin ledc.elf ledc.dis
```

图 4.1.2-2 修改 Makefile

4.1.3 make: arm-linux-gnueabihf-gcc: Command not found

```
MY@IMX6ULL:~/program/2_ledc$ ls
imx6ul.lds  imxdownload  ledc.code-workspace  load.imx  main.c  main.h  Makefile  s  start.s
MY@IMX6ULL:~/program/2_ledc$ make
arm-linux-gnueabihf-gcc -Wall -nostdlib -c -O2 -o start.o start.s
make: arm-linux-gnueabihf-gcc: Command not found
make: *** [start.o] Error 127
MY@IMX6ULL:~/program/2_ledc$
```

图 4.1.3-1 找不到交叉编译器

提示 arm-linux-gnueabihf-gcc 交叉编译器找不到，要么就是没有搭建好交叉编译器环境，要么就是搭建好了环境，但没使能（source /etc/profile）环境，所以环境就不生效，系统编译的时候就找不到编译器了就报错。

解答：在 ubuntu 上输入 arm-linux-gnueabihf-gcc -v 可以查看交叉编译器版本：

```
MY@IMX6ULL:~/program/2_ledc$ arm-linux-gnueabihf-gcc -v
The program 'arm-linux-gnueabihf-gcc' is currently not installed. To run 'arm-linux-gnueabihf-gcc' please ask your administrator to ins
tall the package 'gcc-arm-linux-gnueabihf'
MY@IMX6ULL:~/program/2_ledc$
```

图 4.1.3-2 查看能否找到交叉编译器

提示 The program 'arm-linux-gnueabihf-gcc' is currently not installed.的，那就是没有安装交叉编译器，按照《【正点原子】I.MX6U 嵌入式 Linux 驱动开发指南 V1.5》的第四章 4.3 Ubuntu 交叉编译工具链安装的章节来操作，安装一下交叉编译器。

如果已经按照教程安装了交叉编译器，用 sudo vi /etc/profile 指令查看文件配置是否正确，特别是中英文字符以及大小写这些容易写错：

```
export PATH=$PATH:/usr/local/arm/gcc-linaro-4.9.4-2017.01-x86_64_arm-linux-gnueabihf/bin
```

```
23 if [ -d /etc/profile.d ]; then
24   for i in /etc/profile.d/*.sh; do
25     if [ -r $i ]; then
26       . $i
27     fi
28   done
29   unset i
30 fi
31
32 export PATH=$PATH:/usr/local/arm/gcc-linaro-4.9.4-2017.01-x86_64_arm-linux-gnueabihf/bin
```

图 4.1.3-3 /etc/profile

如果已经按照教程安装交叉编译器，配置也正确执行 source /etc/profile 指令使能一下环境变量：

```
MY@IMX6ULL:~/program/2_ledc$ arm-linux-gnueabihf-gcc -v
Using built-in specs.
COLLECT_GCC=arm-linux-gnueabihf-gcc
COLLECT_LTO_WRAPPER=/usr/local/arm/gcc-linaro-4.9.4-2017.01-x86_64_arm-linux-gnueabihf/bin/../libexec/gcc/arm-linux-gnueabihf/4.9.4/lto
-wrapper
Target: arm-linux-gnueabihf
Configured with: /home/tcwg-buildslave/workspace/tcwg-make-release/label/docker-trusty-amd64-tcwg-build/target/arm-linux-gnueabihf/snap
shots/gcc-linaro-4.9-2017.01/configure SHELL=/bin/bash --with-mpc=/home/tcwg-buildslave/workspace/tcwg-make-release/label/docker-trusty
-amd64-tcwg-build/target/arm-linux-gnueabihf/_build/builds/destdir/x86_64-unknown-linux-gnu --with-mpfr=/home/tcwg-buildslave/workspace
/tcwg-make-release/label/docker-trusty-amd64-tcwg-build/target/arm-linux-gnueabihf/_build/builds/destdir/x86_64-unknown-linux-gnu --wit
h-gmp=/home/tcwg-buildslave/workspace/tcwg-make-release/label/docker-trusty-amd64-tcwg-build/target/arm-linux-gnueabihf/_build/builds/d
estdir/x86_64-unknown-linux-gnu --with-gnu-as --with-gnu-ld --disable-libmudflap --enable-lto --enable-objc-gc --enable-shared --withou
t-included-gettext --enable-nls --disable-sjlj-exceptions --enable-gnu-unique-object --enable-linker-build-id --disable-libstdcxx-pch -
-enable-c99 --enable-clocale=gnu --enable-libstdcxx-debug --enable-long-long --with-cloog=no --with-ppl=no --with-isl=no --disable-mult
ilib --with-float=hard --with-mode=thumb --with-tune=cortex-a9 --with-arch=armv7-a --with-fpu=vfpv3-d16 --enable-threads=posix --enable
-multiarch --enable-libstdcxx-time=yes --with-build-sysroot=/home/tcwg-buildslave/workspace/tcwg-make-release/label/docker-trusty-amd64
-tcwg-build/target/arm-linux-gnueabihf/_build/sysroots/arm-linux-gnueabihf --with-sysroot=/home/tcwg-buildslave/workspace/tcwg-make-rel
ease/label/docker-trusty-amd64-tcwg-build/target/arm-linux-gnueabihf/_build/builds/destdir/x86_64-unknown-linux-gnu/arm-linux-gnueabihf
/libc --enable-checking=release --disable-bootstrap --enable-languages=c,c++,fortran,lto --build=x86_64-unknown-linux-gnu --host=x86_64
-unknown-linux-gnu --target=arm-linux-gnueabihf --prefix=/home/tcwg-buildslave/workspace/tcwg-make-release/label/docker-trusty-amd64-tc
wg-build/target/arm-linux-gnueabihf/_build/builds/destdir/x86_64-unknown-linux-gnu
Thread model: posix
gcc version 4.9.4 (Linaro GCC 4.9-2017.01)
MY@IMX6ULL:~/program/2_ledc$
```

图 4.1.3-4 使能环境

如上图，可以看到，在执行 source /etc/profile 指令以后使能了环境变量了，再执行 arm-linux-gnueabihf-gcc -v 指令以后可以看到交叉编译器了，说明搭建成功。执行

```
make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf-
```

或者 make 可以直接编译成功。

在上图中，交叉编译器 Target: arm-linux-gnueabihf，具体版本是 Linaro GCC 4.9-2017.01，这个编译器就是我们教程中用于编译运行在开发板上的程序的编译器，属于 ARM 架构的。在 ubuntu 上执行指令 gcc -v 指令查看的是 ubuntu 自带的编译器，这个编译器是 x86 架构的，如果用这个编译器编译的程序放在阿尔法板子上是不能运行的，因为他是 x86 架构的，而我们的开发板是 ARM 架构的。

4.1.4 main.o: file not recognized: File format not recognized

如下图，编译报错

```
main.o: file not recognized: File format not recognized
```

提示的是 main.o 文件的格式不是正确的格式，那么可能是之前用了 ubuntu 自带的编译器来编译的，那个编译器是 x86 架构的，所以继续用 arm-linux-gnueabihf-gcc 来编译的话，会提示格式错误。用 file main.o 指令来查看其格式，提示 main.o: ELF 64-bit LSB executable, x86-64, 果不其然，这个是 x86 架构的，所以用 arm-linux-gnueabihf-gcc 来编译就报错了。

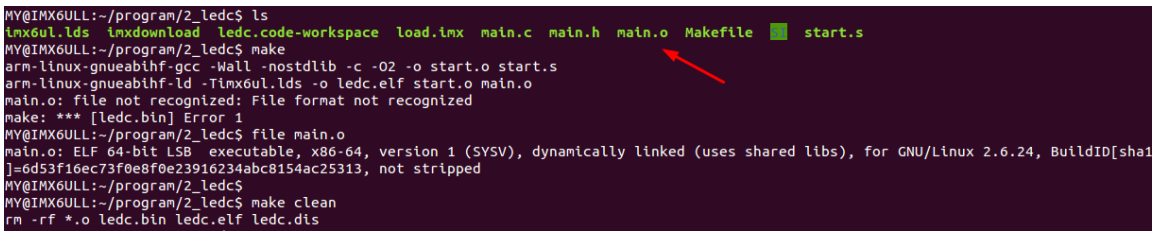


图 4.1.4-1 提示文件格式错误

解答：

先执行 make clean 指令将之前编译生成的中间产物 main.o 清除掉，在执行 make 指令成功编译，如下：

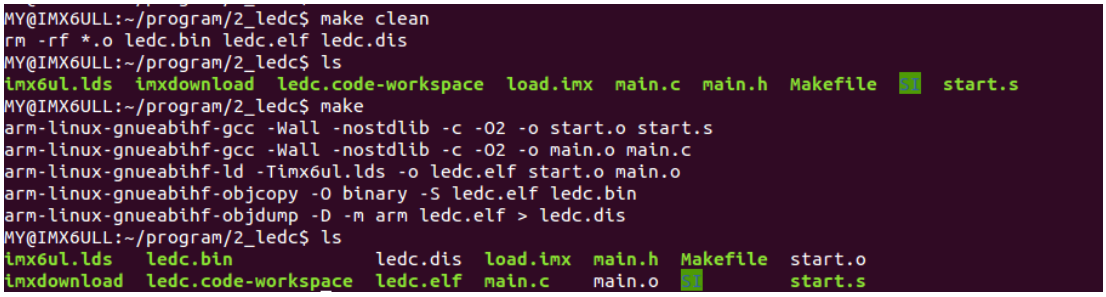


图 4.1.4-2 清除工程后用交叉编译器重新编译

4.1.5 arm-linux-gnueabihf-ld:警告：无法找到项目符号_start;缺省为 0000000087800000

或者报错

arm-linux-gnueabihf-ld: warning: cannot find entry symbol _start; defaulting to 0000000087800000

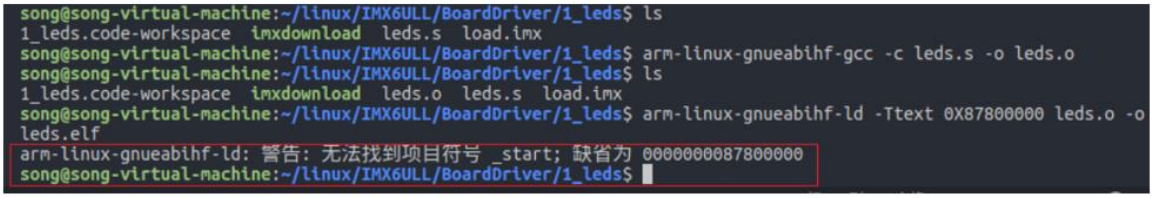


图 4.1.5-1 报错无法找到项目符号_start

解答：

提示找不到_start，应该是 led.s 里边的_start 写错了，检查看看，如下图，原来是将_start 写成了_START 了，这个词只要写错，编译就会报错：

```
.global _start  /* 全局标号 */

/*
 * 描述：        _start函数，程序从此函数开始执行此函数完成时钟使能、
 *                 GPIO初始化、最终控制GPIO输出低电平来点亮LED灯。
 */
_START:
        /* 例程代码 */
        /* 1、使能所有时钟 */
        LDR R0, =0X020C4068     /* CCGR0 */
        LDR R1, =0XFFFFFFFF
        STR R1, [R0]

        LDR R0, =0X020C406C     /* CCGR1 */
        STR R1, [R0]
```

图 4.1.5-2 led.s 里边的_start

4.1.6 报错未定义的引用

```
make[1]: Leaving directory '/home/zys/linux/application/gateway'
gcc -o gateway built-in.o -lpthread
built-in.o：在函数‘test’中：
/home/zys/linux/application/gateway/handware/up_smart_config.c:24：对‘response_fun_read_in_out’未定义的引用
collect2: error: ld returned 1 exit status
Makefile:47: recipe for target 'gateway' failed
make: *** [gateway] Error 1
```

图 4.1.6-1 报错信息

可能原因：程序或者 Makefile 里写错了，以此报错为例，解决方法如下。

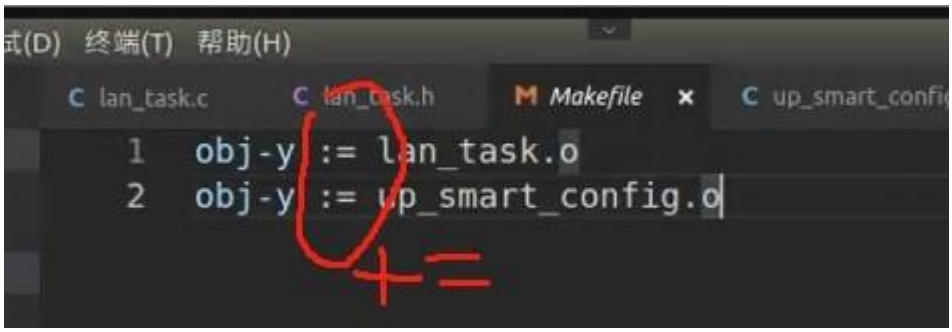


图 4.1.6-2 修改 Makefile

参考链接：

http://blog.sina.com.cn/s/blog_bcaa53900102vd0f.html

https://www.cnblogs.com/qiumingcheng/p/8366159.html

4.1.7 无法编译裸机程序（-O2）

问题：无法编译裸机程序，涉及到-O2 无法编译，如下图报错。

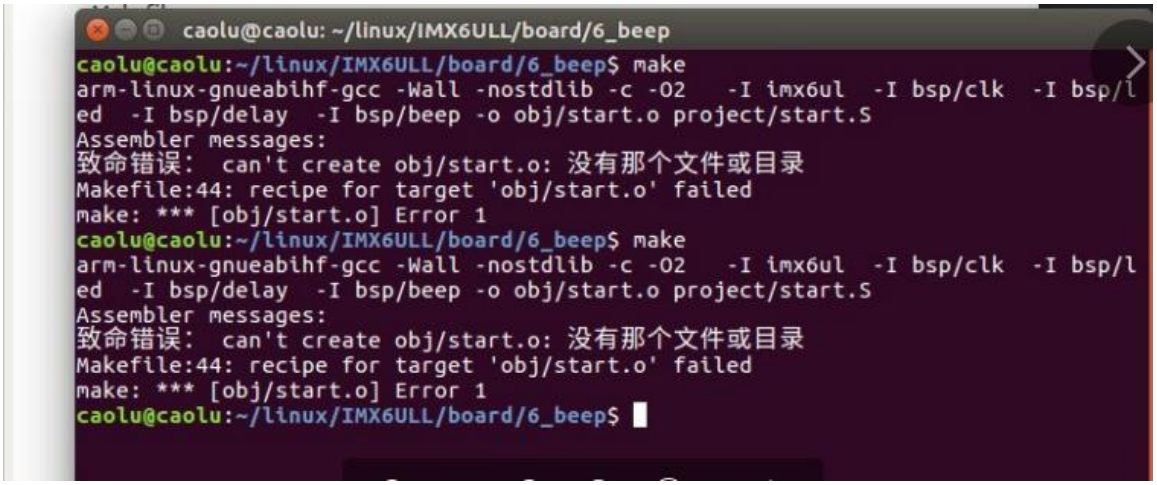


图 4.1.7-1 报错信息

解决思路：

1、 检查下 gcc 版本是否为 5.4，交叉编译器版本是否为 4.9.4。

2、 如果是自己写的程序，对比下自己源码和例程源码，以及 Makefile。

3、 重新解压一份例程源码，编译例程源码看能否运行。

4、 如果重新解压也不行，可能是解压软件的问题。有个案例，有个客户在 windows 下使用 BreeZip 解压软件解压例程源码后，拷贝到 Ubuntu 无法编译（上图报错信息）。解决方法是重新下载另一个解压软件，如 7Z。或者直接拷贝源码包，在 Ubuntu 下解压。

4.2 裸机程序运行不成功

裸机程序（**uboot 也相当于一个裸机程序**）烧写到 SD 卡后，测试 SD 卡启动无现象，可能的原因：

①编写的程序有问题，达不到实验现象；
②程序烧录到 SD 卡里没成功；
③SD 卡坏了；
④开发板底板的 SD 卡的卡槽坏了；
⑤开发板上查了 SDIO WIFI 模块，开发板无法从 SD 卡启动。
如果是后面说的四种情况，可以按照如下方法排查：

4.2.1 程序烧录到 SD 卡里没成功

（1）SD 卡借助 USB 读卡器插在电脑的 USB 口

检查烧写的指令是否正确了，如果 SD 卡用 USB 读卡器插到 ubuntu 上，在 ubuntu 上执行指令 `sudo fdisk -l` 可以查看磁盘的信息，可以查看 ubuntu 挂载了哪些设备。如下图，查看有一个设备/dev/sdc 大小是 15.9GB，我插的 SD 卡刚好是 16GB 的，15.9GB 接近 16GB，所以可以判定/dev/sdc 是我的 SD 卡设备节点。是/dev/sdb 还是/dev/sdc 或者其他，就根据个人的实际情况来，如果有其他设备大小相似，不好判断，那么可以把 SD 卡拔出来再执行 `sudo fdisk -l` 查看，缺少名称的那个就是 SD 卡对应的设备了。

```
MY@IMX6ULL:~$ sudo fdisk -l
[sudo] password for MY:

Disk /dev/sda: 536.9 GB, 536870912000 bytes
255 heads, 63 sectors/track, 65270 cylinders, total 1048576000 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x000a513d

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *        2048  1044383743   522190848   83  Linux
/dev/sda2      1044385790  1048573951     2094081    5  Extended
/dev/sda5      1044385792  1048573951     2094080   82  Linux swap / Solaris

Disk /dev/sdb: 21.5 GB, 21474836480 bytes
255 heads, 63 sectors/track, 2610 cylinders, total 41943040 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000

Disk /dev/sdb doesn't contain a valid partition table

Disk /dev/sdc: 15.9 GB, 15931539456 bytes
64 heads, 32 sectors/track, 15193 cylinders, total 31116288 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x500a0dff

This doesn't look like a partition table
Probably you selected the wrong device.

   Device Boot      Start         End      Blocks   Id  System
/dev/sdc1   ?  1948285285  3650263507   850989111+  6e  Unknown
/dev/sdc2   ?           0           0           0   74  Unknown
/dev/sdc4        28049408    28049848         220+   0  Empty

Partition table entries are not in disk order
MY@IMX6ULL:~$
```

图 4.2.1-1 查看卡设备

如上图，有/dev/sdc 和/dev/sdc1 和/dev/sdc2 以及/dev/sdc4，在这里要注意，/dev/sdc 是设备名，/dev/sdc1 和/dev/sdc2 以及/dev/sdc4 是分区名，在裸机实验中我们要将程序烧录到设备里，不是烧录到分区里，所以执行如下指令进行烧录：

```
./imxdownload ledc.bin /dev/sdc
```

这里要注意了，有的人指令写成了./imxdownload ledc.bin /dev/sdc1，这个是错的，所以烧录完成放到开发板上运行没有成功。

```
MY@IMX6ULL:~/program/2_ledc$ ls
imx6ul.lds   ledc.bin            ledc.dis  load.imx  main.h  Makefile  start.o
imxdownload  ledc.code-workspace  ledc.elf  main.c    main.o  obj       start.s
MY@IMX6ULL:~/program/2_ledc$ ./imxdownload ledc.bin /dev/sdc
I.MX6UL bin download software
Edit by:zuozhongkai
Date:2018/8/9
Version:V1.0
file ledc.bin size = 601Bytes
Delete Old load.imx
Create New load.imx
Download load.imx to /dev/sdc  ......
[sudo] password for MY:
7+1 records in
7+1 records out
3673 bytes (3.7 kB) copied, 0.0106068 s, 346 kB/s
```

图 4.2.1-2 烧写到卡设备成功

或者在插入 SD 卡前和插入 SD 卡后，执行 ls /dev/sd*指令查看对应的设备节点，多出来的那个就是 SD 卡对应的设备节点了。

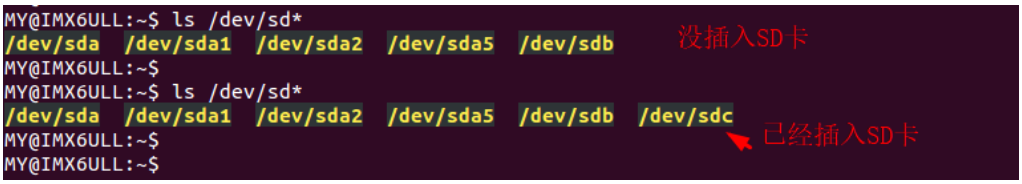


图 4.2.1-3 对比插入卡前后的设备盘符

这里要注意的是，有的人执行指令 ls /dev/sd*查看的结果如下图，可以发现，/dev/sdb 显示灰色，这个说明 SD 卡没挂载成功，所以后期执行指令./imxdownload ledc.bin /dev/sdb 来烧录程序是不成功的。产生这个的原因是之前 SD 卡没有接入 Ubuntu，就执行烧录指令导致生成的假的设备盘符。

解决的办法就是用 sudo rm -rf /dev/sdb 来删除这个假的设备节点，然后 ubuntu 再重新插好卡，然后再查看/dev/sdb 颜色变成黄色了，说明 SD 卡挂载成功。

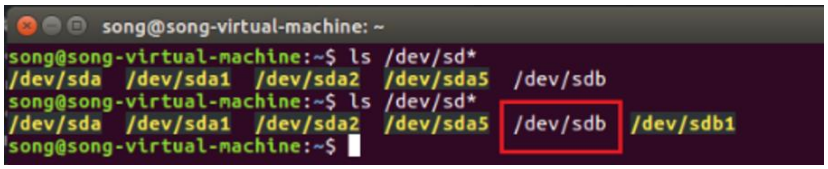


图 4.2.1-4 空白设备盘符

有的如果烧录还是不成功，可以试试格式化 SD 卡了再烧录，在正点原子开发板光盘 A-基础资料\3、软件 资料里有提供一个格式化 SD 卡的工具：

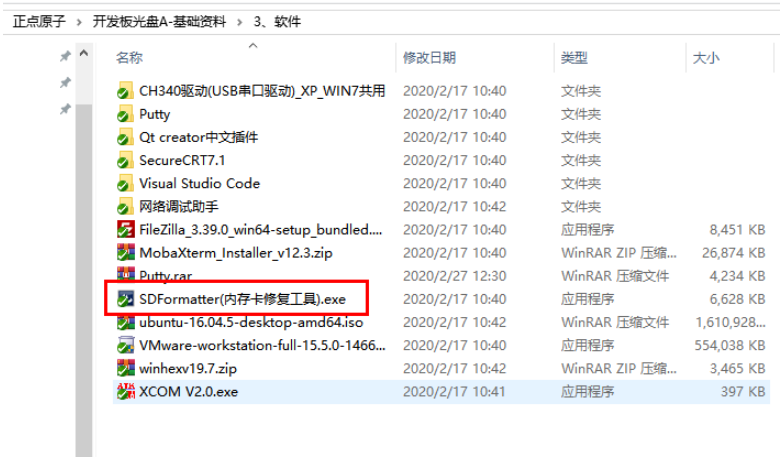


图 4.2.1-5 SD 卡格式化工具

（2）SD 卡插在卡套上，卡套插在笔记本 SD 卡接口上

有的笔记本上有 SD 卡接口，可以直接将 SD 卡插在笔记本的 SD 卡接口上，执行 sudo fdisk -l 查看：

```
Disk /dev/mmcblk0: 14.6 GiB, 15640559616 bytes, 30547968 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x00000000

设备          启动 Start    末尾    扇区  Size Id 类型
/dev/mmcblk0p1       8192 30547967 30539776 14.6G  c W95 FAT32 (LBA)
cx@cx:~/imx6ull_project/led_s$ ./imxdownload led.bin /dev/mmcblk0p1
I.MX6UL bin download software
Edit by:zuozhongkai
Date:2018/8/9
Version:V1.0
file led.bin size = 513Bytes
Delete Old load.imx
Create New load.imx
Download load.imx to /dev/mmcblk0p1  ......
记录了7+1 的读入
记录了7+1 的写出
3585 bytes (3.6 kB, 3.5 KiB) copied, 0.0406686 s, 88.2 kB/s
cx@cx:~/imx6ull_project/led_s$
```

图 4.2.1-6 接 SD 卡套的 SD 卡设备盘符

如上图，/dev/mmcblk0 就是设备名字，/dev/mmcblk0p1 是分区名字，所以烧录的指令是：

```
./imxdownload leds.bin /dev/mmcblk0
```

4.2.2 SD 卡坏了或者是低速卡

如果是 SD 卡坏了，在不确定 SD 卡是否坏的时候，可以试试 SD 卡在 windows 下是否可以识别，以及 ubuntu 下是否可以识别，如果不能识别，这个 SD 卡应该是坏了，要是可以识别，可能 SD 卡存在坏的扇区，可以试试用一些 SD 卡修复工具进行修复。手上有其他的 SD 卡也可以拿来试试。

4.2.3 开发板底板的卡槽坏了

出货的时候正点原子已经将系统固化到了核心板的 EMMC 或者 NAND FLASH 了，开发板从 EMMC 或者 NAND FLASH 启动，进入系统。比如，我的是 EMMC 版本的核心板，开发板从 EMMC 启动，进入系统输入指令 df 查看开发板磁盘挂载情况，然后再插入一张 SD 卡，查看磁盘挂载情况：

```
root@ATK-IMX6U:~#
root@ATK-IMX6U:~#
root@ATK-IMX6U:~# df                                          插入SD卡前
[   30.491938] random: nonblocking pool is initialized
Filesystem     1K-blocks    Used Available Use% Mounted on
/dev/root        7244864  510560   6359624   8% /
devtmpfs          187640       4    187636   1% /dev
tmpfs                 40       0        40   0% /mnt/.psplash
tmpfs             253436     152    253284   1% /run
tmpfs             253436     136    253300   1% /var/volatile
/dev/mmcblk1p1    129039    6897    122143   6% /run/media/mmcblk1p1
root@ATK-IMX6U:~# [   76.980835] mmc0: error -123 whilst initialising SD card
[   77.092629] mmc0: host does not support reading read-only switch, assuming write-enable
[   77.107585] mmc0: new high speed SDHC card at address 0001
[   77.115327] mmcblk0: mmc0:0001 00000 7.44 GiB
[   77.130897]  mmcblk0: p1 p2                                 插入SD卡过程中
[   77.857353] kjournald starting.  Commit interval 5 seconds
[   77.875520] EXT3-fs (mmcblk0p2): using internal journal
[   77.880786] EXT3-fs (mmcblk0p2): recovery complete
[   77.886519] EXT3-fs (mmcblk0p2): mounted filesystem with ordered data mode
[   77.902556] FAT-fs (mmcblk0p1): Volume was not properly unmounted. Some data may be corrupt. Please run fsck.

root@ATK-IMX6U:~# df
Filesystem     1K-blocks    Used Available Use% Mounted on
/dev/root        7244864  510560   6359624   8% /
devtmpfs          187640       4    187636   1% /dev
tmpfs                 40       0        40   0% /mnt/.psplash
tmpfs             253436     164    253272   1% /run
tmpfs             253436     140    253296   1% /var/volatile
/dev/mmcblk1p1    129039    6897    122143   6% /run/media/mmcblk1p1
/dev/mmcblk0p2   7414160  415468   6615412   6% /run/media/mmcblk0p2
/dev/mmcblk0p1    129039    6818    122221   6% /run/media/mmcblk0p1
                                                              插入SD卡后
```

图 4.2.3-1 出厂系统测试 SD 卡读写

如上图，我插入一张已经做好分区的 SD 卡，插入过程中打印 mmc0 或者 mmcblk0 的字眼，说明 SD 卡已经被识别到了。执行指令 df 可以查看有/dev/mmcblk0p1 和/dev/mmcblk0p2，其中/dev/mmcblk0p1 是 SD 卡的第一个分区，这个分区经常是用于存放内核以及设备树镜像的，另

外一个是/dev/mmcblk0p2，这个叫做 rootfs 分区，即文件系统分区，这个分区经常用于存放文件系统的。如果插入的是一张空的 FAT 格式的 SD 卡，会显示/dev/sda。这说明开发板的 SD 卡卡槽识别到 SD 卡了。如果 SD 卡未能识别，可能是 SD 卡卡槽的问题。

（1）开发板上已经插了 SDIO WIFI 模块

SDIO WIFI 接口原理图：

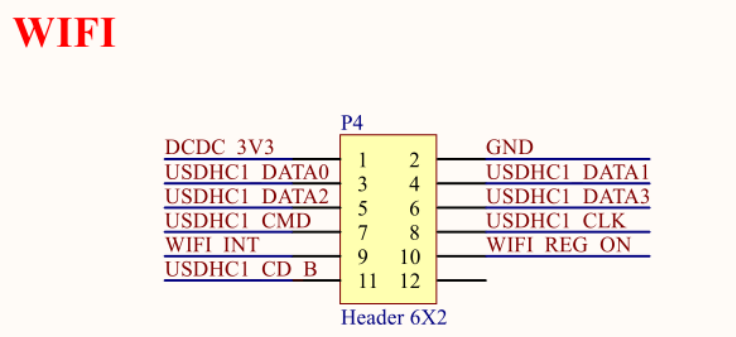


图 4.2.3-2 SDIO WIFI 接口原理图

SD 卡原理图接口：

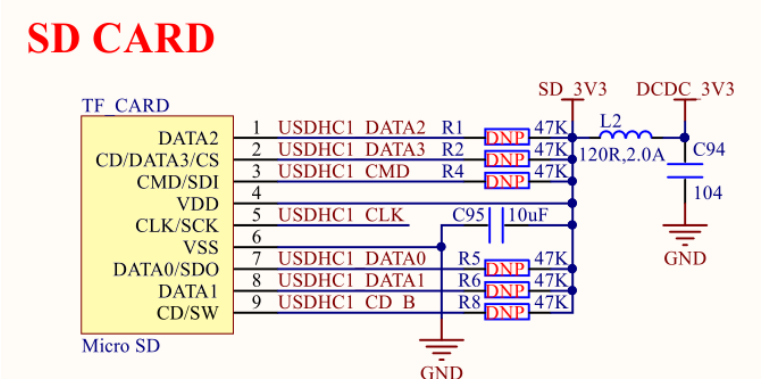


图 4.2.3-3 SD 卡接口原理图

由阿尔法开发板的底板原理图，可以看出 TF 卡接口和 SDIO WIFI 接口共用一个 SDIO，因此 TF 卡和 SDIO 不能同时使用，二者选其一。

4.3 imxdownload 相关问题

4.3.1 imxdownload 无法烧写 Nand Flash 的裸机程序、uboot

```
(base) zhang@ubuntu:~/linux/uboot$ sudo ./imxdownload u-boot.bin /dev/sdb -256m
I.MX6UL bin download software
Edit by:zuozhongkai
Date:2018/8/9
Version:V1.0
Error Usage! Reference Below:
sudo ./imxdownload <source_bin> <sd_device>
(base) zhang@ubuntu:~/linux/uboot$
```

图 4.3.1-1 报错信息

原因：imxdownload 可能使用的是老版本的，需要重新编译一个 imxdownload。例如上图报错信息中就提示 Version 版本是 V1.0 版本，这个版本是不支持 Nand Flash 的。

解决：把网盘 Ubuntu 下裸机烧写软件下这两个文件拷贝到 Ubuntu。

执行 gcc imxdownload.c -o imxdownload，用新生成的 imxdownload 来下载。

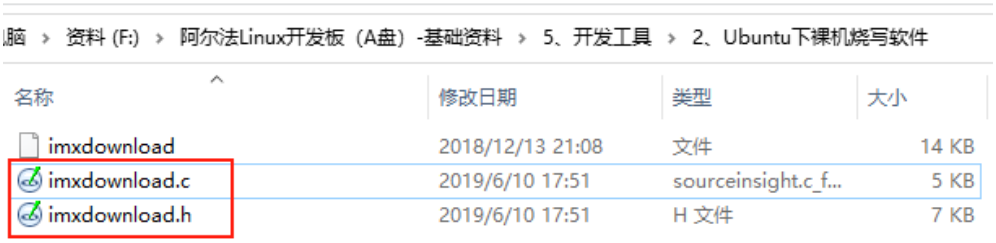


图 4.3.1-2 imxdownload 源文件

或者更新下网盘资料，网盘最新资料里的 imxdownload 已经是 V1.1 版本，支持 Nand Flash。

执行烧写 Nand Flash 版本的烧录指令，以烧写 u-boot 为例：

```
sudo ./imxdownload u-boot.bin /dev/sdb -256m
```

最终烧写成功，如下图所示可以看到当前 imxdownload 版本为 V1.1。

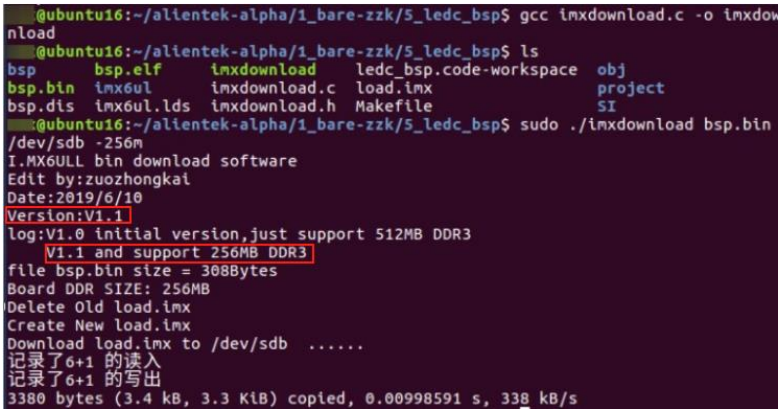


图 4.3.1-3 烧录

4.4 清 bss 段问题

4.4.1 第 9 个实验中断，汇编文件里面加入 bss 和清 bss 后程序无法运行

参考链接：http://www.openedv.com/forum.php?mod=viewthread&tid=309678&highlight=bss

A 盘的例程源码和视频盘的例程源码不太一样，A 盘的是最终的例程源码（最终的），视频盘的是左工录制视频敲的。在视频盘第 9 个实验中需要改一下 bss 段位置如图。

```
isb
MCR p15,0,r0,c12,c0,0  /* 设置VBAR寄存器=0X87800000 */
dsb
isb
#endif

.global _bss_start
_bss_start:
    .word __bss_start

.global _bss_end
_bss_end:
    .word __bss_end

    /*清除BSS段*/
    ldr r0, _bss_start
    ldr r1, _bss_end
    mov r2, #0
bss_loop:
    stmia r0!, {r2}
    cmp r0, r1      /* 比较R0和R1里面的值 */
    ble bss_loop    /*如果r0地址小于等于r1 继续清除bss段*/
```

图 4.4.1-1 修改

第五章 uboot 移植相关问题

5.1 menuconfig 报错

5.1.1 执行 menuconfig 报错，出不来菜单界面

报错信息：

```
scripts/kconfig/mconf  Kconfig
Your display is too small to run Menuconfig!
It must be at least 19 lines by 80 columns.
make[1]: *** [menuconfig] Error 1
make: *** [menuconfig] Error 2
```

```
MY@IMX6ULL:~/test/uboot-imx-rel_imx_4.1.15_2.1.0_ga$ make ARCH
=arm CROSS_COMPILE=arm-linux-gnueabihf- menuconfig
scripts/kconfig/mconf  Kconfig
Your display is too small to run Menuconfig!
It must be at least 19 lines by 80 columns.
make[1]: *** [menuconfig] Error 1
make: *** [menuconfig] Error 2
MY@IMX6ULL:~/test/uboot-imx-rel_imx_4.1.15_2.1.0_ga$
```

图 5.1.1-1 报错信息

报错原因是命令行界面太小了，错误提示里：您的显示太小，无法运行 Menuconfig！它必须至少为 19 行乘 80 列。所以把终端设置大一些、拉大一些就不会报错了。

5.2 uboot 编译报错

5.2.1 初学者按照文档初次编译 uboot 报错

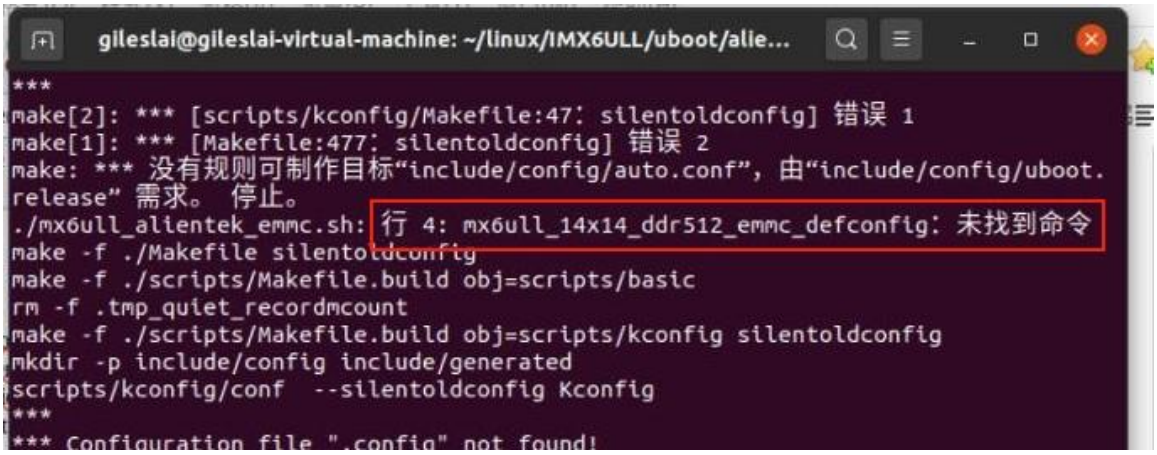


图 5.2.1-1 报错信息

行 4：mx6u11_14×14_dd512_emmc_defconfig：未找到命令

解决思路：可能是初学者按照文档学习，写错脚本，例如这里用户是这样的：

图 5.2.1-2 用户写的脚本

发现问题，用户编译脚本写错，第 3、4 行应该为完整的一条命令：

```
make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- mx6ull_14x14_ddr512_emmc_defconfig
```

不能换行（要换行也得加\回车，才能换行），文档是因为排版要求才换行的，初学者会以为这个是要分两行写指令，其实不是的。

最终编译成功。

5.2.2 uboot 编译报错 lib/asm-offsets.c:1:0: error: bad value (armv5) for -march= switch

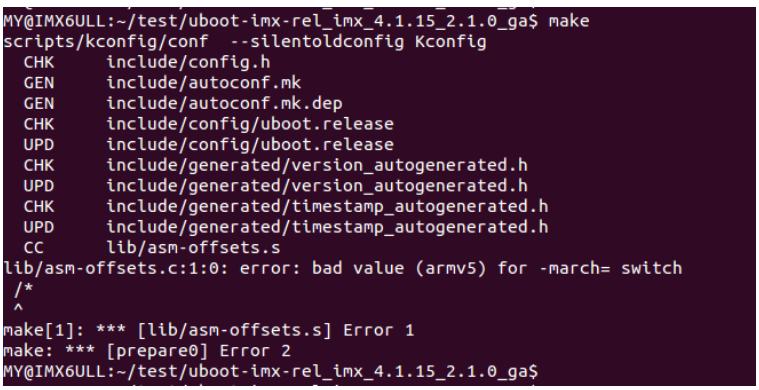


图 5.2.2-1 报错信息

解决办法：

①可以在编译指令中指定 架构 和 交叉编译器

```
make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf-
```

```
MY@IMX6ULL:~/test/uboot-imx-rel_imx_4.1.15_2.1.0_ga$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf-
  CHK     include/config/uboot.release
  CHK     include/generated/version_autogenerated.h
  UPD     include/generated/version_autogenerated.h
  CHK     include/generated/timestamp_autogenerated.h
  UPD     include/generated/timestamp_autogenerated.h
  CC      lib/asm-offsets.s
  CHK     include/generated/generic-asm-offsets.h
  UPD     include/generated/generic-asm-offsets.h
  CC      arch/arm/lib/asm-offsets.s
  CHK     include/generated/asm-offsets.h
  UPD     include/generated/asm-offsets.h
  HOSTCC  tools/bmp_logo
  HOSTCC  tools/gen_eth_addr
  HOSTCC  tools/img2srec
  HOSTCC  tools/mkenvimage.o
  HOSTCC  tools/os_support.o
```

图 5.2.2-2 指定交叉编译器

②在 uboot 源码根目录的 Makefile 文件中指定 架构 和 交叉编译器 ，指定了编译器以后可以直接执行 make 进行编译。

```
ARCH=arm
CROSS_COMPILE=arm-linux-gnueabihf-
```

```
240 export  HOSTARCH HOSTOS
241
242 ######################################################################
243
244 # set default to nothing for native builds
245 ifeq ($(HOSTARCH),$(ARCH))
246 CROSS_COMPILE ?=
247 endif
248
249 ARCH=arm
250 CROSS_COMPILE=arm-linux-gnueabihf-
251
252
253 KCONFIG_CONFIG  ?= .config
254 export KCONFIG_CONFIG
255
256 # SHELL used by kbuild
257 CONFIG_SHELL := $(shell if [ -x "$$BASH" ]; then echo $$BASH; \
```

图 5.2.2-3 修改 Makefile

5.2.3 编译报错 Configuration file ".config" not found!

报错信息：

```
Configuration file ".config" not found!
scripts/kconfig/conf    --silentoldconfig Kconfig
***
*** Configuration file ".config" not found!
***
*** Please run some configurator (e.g. "make oldconfig" or
*** "make menuconfig" or "make xconfig").
***
make[2]: *** [silentoldconfig] Error 1
make[1]: *** [silentoldconfig] Error 2
make: *** No rule to make target `include/config/auto.conf', needed by `include/config/uboot.release'.  Stop.
```

这种是因为 uboot 没配置成功就直接编译，所以会报错 Configuration file ".config" not found! 重新配置 uboot 再编译就 OK。

```
MY@IMX6ULL:~/test/uboot-imx-rel_imx_4.1.15_2.1.0_ga$ make
scripts/kconfig/conf  --silentoldconfig Kconfig
***
*** Configuration file ".config" not found!
***
*** Please run some configurator (e.g. "make oldconfig" or
*** "make menuconfig" or "make xconfig").
***
make[2]: *** [silentoldconfig] Error 1
make[1]: *** [silentoldconfig] Error 2
make: *** No rule to make target `include/config/auto.conf', needed by `include/config/uboot.release'.  Stop.
MY@IMX6ULL:~/test/uboot-imx-rel_imx_4.1.15_2.1.0_ga$
```

图 5.2.3-1 报错信息

5.2.4 Can't find default configuration "arch/../configs/mx6ull_alientek__emmc_defconfig"!

找不到配置文件

```
Can't find default configuration "arch/../configs/mx6ull_alientek__emmc_defconfig"!
```

这种一般是配置文件名字写错的，或者格式有误，尽量手敲指令。如下图，错将将 mx6ull_alientek_emmc_defconfig 写成了 mx6ull_alientek__emmc_defconfig 就报错了，提示找不到 mx6ull_alientek__emmc_defconfig。

```
MY@IMX6ULL:~/test/uboot-imx-rel_imx_4.1.15_2.1.0_ga$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- mx6ull_alientek__emmc_defconfig
  HOSTCC  scripts/basic/fixdep
  HOSTCC  scripts/kconfig/conf.o
  SHIPPED scripts/kconfig/zconf.tab.c
  SHIPPED scripts/kconfig/zconf.lex.c
  SHIPPED scripts/kconfig/zconf.hash.c
  HOSTCC  scripts/kconfig/zconf.tab.o
  HOSTLD  scripts/kconfig/conf
***
*** Can't find default configuration "arch/../configs/mx6ull_alientek__emmc_defconfig"!
***
make[1]: *** [mx6ull_alientek__emmc_defconfig] Error 1
make: *** [mx6ull_alientek__emmc_defconfig] Error 2
MY@IMX6ULL:~/test/uboot-imx-rel_imx_4.1.15_2.1.0_ga$
```

图 5.2.4-1 找不到配置文件

5.2.5 recipe for target 'scripts_basic' failed

报错信息：

```
HOSTCC scripts/basic/fixdep
/bin/sh:1: cc: not found
scripts/Makefile. host:94: recipe for target 'scripts/basic/fixdep' failed
make[1]:***[scripts/basic/fixdep] Error 127
recipe for target 'scripts_basic' failed
```

报错 recipe for target 'scripts_basic' failed 的，是 ubuntu 少了 32 位的库，执行如下指令进行安装：

```
sudo apt-get install lsb-core lib32stdc++6
```

5.2.6 scripts/basic/fixdep: Permission denied

编译 uboot 源码报错：

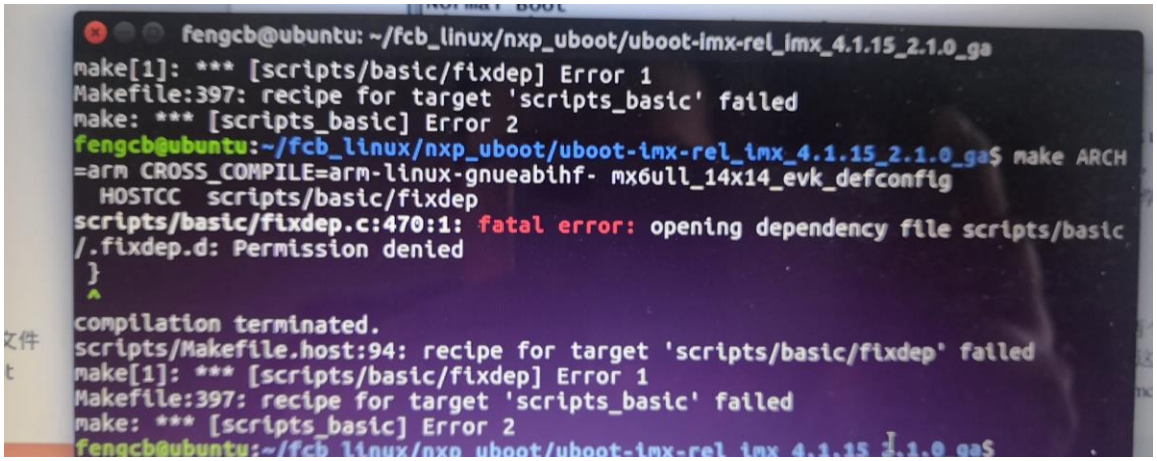


图 5.2.6-1 报错信息

解决：权限问题，重新解压一份源码，注意不要使用 sudo 权限

参考：https://blog.csdn.net/xlinxss/article/details/84700728

5.3 uboot 启动问题

5.3.1 uboot 启动出现 *** Warning - bad CRC, using default environment

提示用的默认环境变量，清除一下环境变量就好

```
env default -a
saveenv
```

5.3.2 MMC: no card present

报错 MMC: no card present 的，一般是在移植 uboot 的时候配置有误，排查的时候，按照教程的操作步骤来进行检查，例如在 uboot 源码的./configs/mx6ull_alientek_emmc_defconfig 下，将 CONFIG_SYS_EXTRA_OPTIONS="IMX_CONFIG=board/freescale/mx6ull_alientek_emmc/imximage.cfg,MX6ULL_EVK_EMMC_REWORK"这个配置中的 MX6ULL_EVK_EMMC_REWORK 删除掉，运行编译好的 uboot 以后就会报错 MMC: no card present。

图 5.3.2-1 报错信息

5.3.3 移植的 uboot 无法 ping 通 ubuntu（已经排除环境搭建问题）

```
=>
=> setenv serverip 192.168.1.219
=> setenv ipaddr 192.168.1.100
=> setenv ethaddr 00:04:9f:04:d2:35
=> setenv gatewayip 192.168.1.1
=> saveenv
Saving Environment to MMC...
Writing to MMC(1)... done
=> ping 192.168.1.32
Using FEC1 device

ARP Retry count exceeded; starting again
ping failed; host 192.168.1.32 is not alive
=>
```

图 5.3.3-1 uboot 无法 ping 虚拟机

检查看看是否有将#define CONFIG_PHY_MICREL 改成了#define CONFIG_PHY_SMSC，如果没有改的话，uboot 是不通的：

第 345 行定了一个宏 CONFIG_PHY_MICREL，此宏用于使能 uboot 中 Micrel 公司的 PHY 驱动，KSZ8081 这颗 PHY 芯片就是 Micrel 公司生产的，不过 Micrel 已经被 Microchip 收购了。如果要使用 LAN8720A，那么就得将 CONFIG_PHY_MICREL 改为 CONFIG_PHY_SMSC，也就是使能 uboot 中的 SMSC 公司中的 PHY 驱动，因为 LAN8720A 就是 SMSC 公司生产的。所以示例代码 33.2.7.1 有三处要修改：

①、修改 ENET1 网络 PHY 的地址。

②、修改 ENET2 网络 PHY 的地址。

③、使能 SMSC 公司的 PHY 驱动。

修改后的网络 PHY 地址参数如下所示：

示例代码 33.2.7.2 网络 PHY 地址配置参数

```
325 #ifdef CONFIG_CMD_NET
326 #define CONFIG_CMD_PING
327 #define CONFIG_CMD_DHCP
328 #define CONFIG_CMD_MII
329 #define CONFIG_FEC_MXC
330 #define CONFIG_MII
331 #define CONFIG_FEC_ENET_DEV        1
332
333 #if (CONFIG_FEC_ENET_DEV == 0)
334 #define IMX_FEC_BASE               ENET_BASE_ADDR
335 #define CONFIG_FEC_MXC_PHYADDR     0x0
336 #define CONFIG_FEC_XCV_TYPE        RMII
337 #elif (CONFIG_FEC_ENET_DEV == 1)
338 #define IMX_FEC_BASE               ENET2_BASE_ADDR
339 #define CONFIG_FEC_MXC_PHYADDR     0x1
340 #define CONFIG_FEC_XCV_TYPE        RMII
341 #endif
342 #define CONFIG_ETHPRIME            "FEC"
343
344 #define CONFIG_PHYLIB
345 #define CONFIG_PHY_SMSC
```

867

图 5.3.3-2 检查修改

5.3.4 u-boot 启动的时候会提示 Error:FEC1 adress not set

u-boot 启动的时候会提示 Error:FEC1 adress not set，其实这个并不是报错，可以忽略。 这个是没设置 ethaddr 地址才会这样的，可以自己在 uboot 下设置这个变量：

```
setenv ethaddr 00:04:9f:04:d2:35;saveenv
```

执行以上指令再重启系统，会发现没那个了提示了。

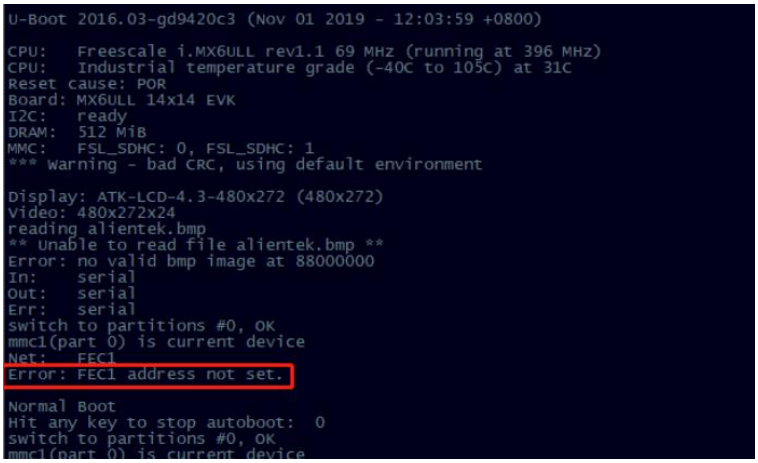


图 5.3.4-1 报错信息

5.4 DNS 相关问题

5.4.1 出厂系统 uboot 不能直接用 DNS

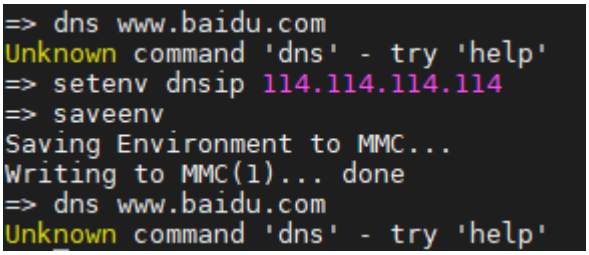


图 5.4.1-1 报错信息

出现这种情况时，其实是出厂的 uboot 配置没勾选 dns。自己在编译出厂内核的时候可以勾选 dns。执行 make menuconfig 来配置。

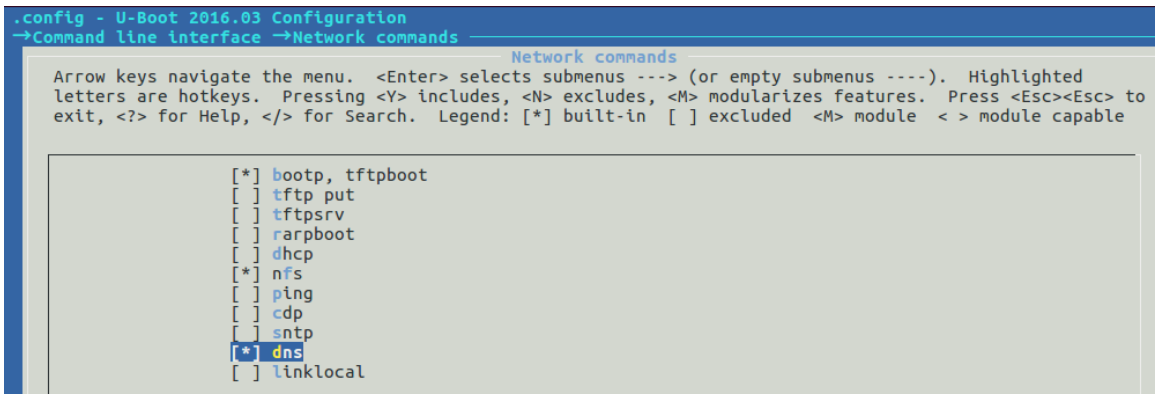


图 5.4.1-2 勾选 dns

保存好配置后，不急着编译，先把 build.sh 脚本里的清除配置屏蔽，避免我们的配置不生效。

```
vi build.sh
```

```
10 #使用Yocto SDK里的GCC 5.3.0交叉编译器编译出厂Linux源码,可不用指定ARCH等，直接执行Make
11 source /opt/fsl-imx-x11/4.1.15-2.1.0/environment-setup-cortexa7hf-neon-poky-linux-gnueabi
12 #!/bin/bash
13 #编译前先清除
14 #make distclean
15 make mx6ull_14x14_ddr256_nand_sd_defconfig
16 make all -j16
17 mv u-boot.imx u-boot-imx6ull-14x14-ddr256-nand-sd.imx
```

图 5.4.1-3 屏蔽清除指令

使能环境变量后，执行脚本编译。

```
source /opt/fsl-imx-x11/4.1.15-2.1.0/environment-setup-cortexa7hf-neon-poky-linux-gnueabi
./build.sh
```

5.4.2 移植 uboot 后，dns 失败

```
=> setenv dnsip 114.114.114.114
=> saveenv
Saving Environment to MMC...
Writing to MMC(0)... done
=>
=>
=> dns www.baidu.com
Unknown command 'dns' - try 'help'
```

图 5.4.2-1 报错信息

可能原因：

在按教程 34.1 图形配置好 uboot 后，执行了 mx6ull_alientek_emmc.sh 的脚本。该脚本第一条指令就是清除配置，导致图形配置全部无效。

解决办法：

重新按照教程 34.1 配置，最后直接使用以下命令编译 uboot。

```
make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- -j16
```

教程中也有相关提示：

使用如下命令编译 uboot：

```
make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- -j16
```

千万不能使用如下命令：

```
./mx6ull_alientek_emmc.sh
```

因为 mx6ull_alientek_emmc.sh 在编译之前会清理工程，会删除掉.config 文件！通过图形化界面配置所有配置项都会被删除，结果就是竹篮打水一场空。

图 5.4.2-2 教程提示

第六章 内核移植相关问题

6.1 编译出厂内核报错

6.1.1 64-bit kernel (64BIT) [Y/n/?] (NEW)

```
alientek@ubuntu16:~/alpha/alientek-alpha/kernel-alientek$ ls
arch            firmware  lib                                          samples
block           fs        linux-imx-4.1.15-2.1.0-gb78e551-v1.4.tar.xz  scripts
build.sh        include   MAINTAINERS                                  security
COPYING         init      Makefile                                     sound
CREDITS         ipc       mm                                           tmp
crypto          Kbuild    net                                          tools
Documentation   Kconfig   README                                       usr
drivers         kernel    REPORTING-BUGS                               virt
alientek@ubuntu16:~/alpha/alientek-alpha/kernel-alientek$ make
  HOSTCC  scripts/kconfig/conf.o
  HOSTCC  scripts/kconfig/zconf.tab.o
  HOSTLD  scripts/kconfig/conf
scripts/kconfig/conf  --silentoldconfig Kconfig
*
* Restart config...
*
*
* Linux/x86 4.1.15 Kernel Configuration
*
64-bit kernel (64BIT) [Y/n/?] (NEW)
```

图 6.1.1-1 报错信息

原因：没有指定编译的配置文件。这个不能直接通过 make 编译，可以看下编译脚本 build.sh。

6.1.2 编译报一大堆错

```
***
*** Can't find default configuration "arch/x86/configs/imx_v7_defconfig"!
***
scripts/kconfig/Makefile:105: recipe for target 'imx_v7_defconfig' failed
make[1]: *** [imx_v7_defconfig] Error 1
Makefile:541: recipe for target 'imx_v7_defconfig' failed
make: *** [imx_v7_defconfig] Error 2
scripts/kconfig/conf    --silentoldconfig Kconfig
***
*** Configuration file ".config" not found!
***
*** Please run some configurator (e.g. "make oldconfig" or
*** "make menuconfig" or "make xconfig").
***
scripts/kconfig/Makefile:37: recipe for target 'silentoldconfig' failed
make[2]: *** [silentoldconfig] Error 1
Makefile:541: recipe for target 'silentoldconfig' failed
make[1]: *** [silentoldconfig] Error 2
make: *** No rule to make target 'zImage'。 停止。
```

原因：没有使能出厂交叉编译器进行编译。

```
source /opt/fsl-imx-x11/4.1.15-2.1.0/environment-setup-cortexa7hf-neon-poky-linux-gnueabi
```

具体编译步骤看用户快速体验第五章。

6.1.3 提示没有找到 environment-setup-cortexa7hf-neon-poky-linux-gnueabi

按照《【正点原子】I.MX6U 用户快速体验》这个文档的第五章检查下是否安装了出厂系统交叉编译器，安装的路径是否正确，是否使能环境了再编译。

6.2 编译教程内核报错

6.2.1 Linux/x86 4.1.15 Kernel Configuration

或者报错提示：

```
arch/i486-openwrt-linux/Makefile
```

如下图，在编译内核或者 uboot 或者驱动模块的时候报错，根据提示，这里显示即将编译的是 x86 架构的或者其它架构的，这个是因为没有指定架构或者编译器版本导致的。

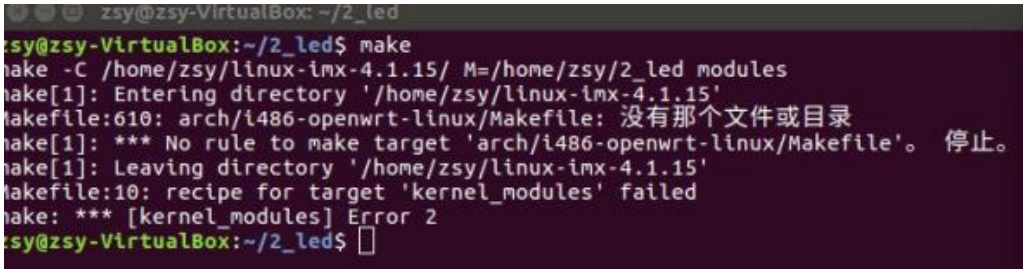


图 6.2.1-1 报错信息

图 6.2.1-2 报错信息

解决办法：

执行 source /etc/profile 使能环境变量。

如果编译的是内核或者驱动模块，则在内核源码根目录的 Makefile 文件下修改，添加如下两句：

37.2.1 修改顶层 Makefile

修改顶层 Makefile，直接在顶层 Makefile 文件里面定义 ARCH 和 CROSS_COMPILE 这两个的变量值为 arm 和 arm-linux-gnueabihf-，结果如图 37.2.1 所示：

```
242  # CROSS_COMPILE specify the prefix used for all executables used
243  # during compilation. Only gcc and related bin-utils executables
244  # are prefixed with $(CROSS_COMPILE).
245  # CROSS_COMPILE can be set on the command line
246  # make CROSS_COMPILE=ia64-linux-
247  # Alternatively CROSS_COMPILE can be set in the environment.
248  # A third alternative is to store a setting in .config so that plain
249  # "make" in the configured kernel build directory always uses that.
250  # Default value for CROSS_COMPILE is not to prefix executables
251  # Note: Some architectures assign CROSS_COMPILE in their arch/*/Makefile
252  ARCH        ?= arm
253  CROSS_COMPILE   ?= arm-linux-gnueabihf-
```

图 37.2.1 修改顶层 Makefile

图 37.2.1 中第 252 和 253 行分别设置了 ARCH 和 CROSS_COMPILE 这两个变量的值，这样在编译的时候就不用输入很长的命令了。

图 6.2.1-3 修改 Makefile

如果是编译 uboot，那么在 uboot 源码的根目录下进行修改：

图 33.1.2.1 编译结果

从图 33.1.2.1 可以看出，编译成功。我们在编译的时候需要输入 ARCH 和 CORSS_COMPILE 这两个变量的值，这样太麻烦了。我们可以直接在顶层 Makefile 中直接给 ARCH 和 CORSS_COMPILE 赋值，修改如图 33.1.2.2 所示：

```
245  # set default to nothing for native builds
246  ifeq ($(HOSTARCH),$(ARCH))
247  CROSS_COMPILE ?=
248  endif
249
250  ARCH = arm
251  CROSS_COMPILE = arm-linux-gnueabihf-
```

图 33.1.2.2 添加 ARCH 和 CROSS_COMPILE 值

图 33.1.2.2 中的 250、251 行就是直接给 ARCH 和 CROSS_COMPILE 赋值，这样我们就可以使用如下简短的命令来编译 uboot 了：

```
make mx6ull_14x14_evk_emmc_defconfig
make V=1 -j16
```

如果既不想修改 uboot 的顶层 Makefile，又想编译的时候不用输入那么多，那么就直接创

图 6.2.1-4 修改 uboot

编译指令里指定用什么编译器什么架构：

```
make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf-
```

6.2.2 编译内核报错 fatal error:curses.h:没有那个文件或目录

报错信息：fatal error: curses.h: No such file or directory

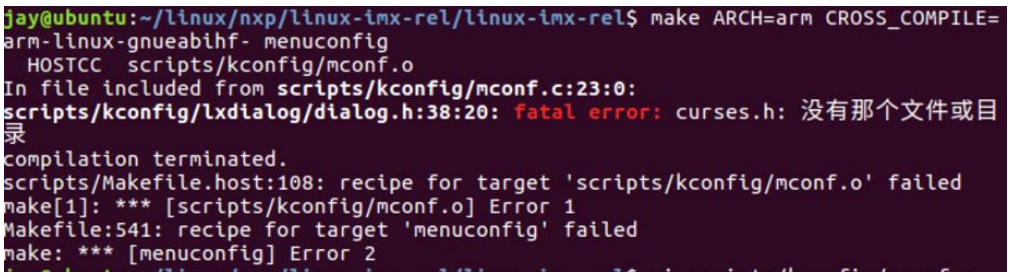


图 6.2.2-1 报错信息

执行 make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- menuconfig 后报错 fatal error:curses.h:没有那个文件或目录，这个是少了某个库，执行如下指令安装库：

```
sudo apt-get install libncurses*
```

6.2.3 编译内核提示：Linux/x86 4.1.15 Kernel Configuration

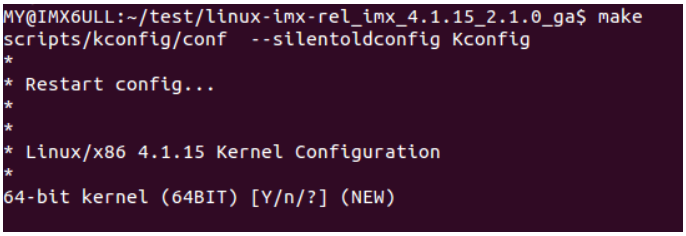


图 6.2.3-1 提示信息

这种是编译的时候用错了 ubuntu 自带的 x86 架构的编译器，不是 ARM 架构（阿尔法开发板上是 ARM 架构）上用的交叉编译器（arm-linux-gnueabihf-gcc）

解决办法：检查交叉编译器是否已经安装，并使能交叉编译器。

6.2.4 fatal error: dt-bindings/clock/imx5-clock.h: 没有那个文件或目录

编译原子的教程源码设备树报错：

```
arch/arm/boot/dts/imx50.dtsi:16:42: fatal error: dt-bindings/clock/imx5-clock.h: 没有那个文件或目录
compilation terminated.
```

```
zzk@zzk-virtual-machine: ~/linux/kernal1/linux-imx-rel_imx_4.1.15_2.1.0_ga_alientek
zzk@zzk-virtual-machine:~/linux/kernal1/linux-imx-rel_imx_4.1.15_2.1.0_ga_alient
ek$ make dtbs
  CHK     include/config/kernel.release
  CHK     include/generated/uapi/linux/version.h
  CHK     include/generated/utsrelease.h
make[1]: 'include/generated/mach-types.h' is up to date.
  CHK     include/generated/bounds.h
  CHK     include/generated/asm-offsets.h
  CALL    scripts/checksyscalls.sh
  DTC     arch/arm/boot/dts/imx50-evk.dtb
In file included from arch/arm/boot/dts/imx50-evk.dts:15:0:
arch/arm/boot/dts/imx50.dtsi:16:42: fatal error: dt-bindings/clock/imx5-clock.h:
 没有那个文件或目录
compilation terminated.
scripts/Makefile.lib:293: recipe for target 'arch/arm/boot/dts/imx50-evk.dtb' fa
iled
make[1]: *** [arch/arm/boot/dts/imx50-evk.dtb] Error 1
arch/arm/Makefile:327: recipe for target 'dtbs' failed
make: *** [dtbs] Error 2
zzk@zzk-virtual-machine:~/linux/kernal1/linux-imx-rel_imx_4.1.15_2.1.0_ga_alient
ek$
```

图 6.2.4-1 报错信息

解决步骤：参考 https://blog.csdn.net/polaris_zgx/article/details/103803234

（1）执行命令

```
cd arch/arm/boot/dts/include                    //进入该目录。
```

（2）执行

```
ln -s ../../../../../include/dt-bindings/ ./                    //软连接文件。
```

6.2.5 执行脚本没有生成 zImage，只有 Image

解决：缺少库

```
sudo apt-get install lzop
```

编译 Linux 内核的时候可能会提示“recipe for target ‘arch/arm/boot/compressed/piggy.lzo’ failed”，如图 35.2.5 所示：

```
/bin/sh: 1: lzop: not found
arch/arm/boot/compressed/Makefile:180: recipe for target 'arch/arm/boot/compressed/piggy.lzo' failed
make[2]: *** [arch/arm/boot/compressed/piggy.lzo] Error 1
make[2]: *** 正在等待未完成的任务....
  CC      arch/arm/boot/compressed/misc.o
arch/arm/boot/Makefile:52: recipe for target 'arch/arm/boot/compressed/vmlinux' failed
make[1]: *** [arch/arm/boot/compressed/vmlinux] Error 2
arch/arm/Makefile:316: recipe for target 'zImage' failed
```

图 35.2.5 lzop 未找到

图 35.2.5 中的错误提示 lzop 未找到，原因是没有安装 lzop 库，输入如下命令安装 lzop 库即可解决：

```
sudo apt-get install lzop
```

lzop 库安装完成以后在重新编译一下 Linux 内核即可。

图 6.2.5-1 教程内容

6.3 自己移植的内核问题

6.3.1 开发板 LAN8720A 网口网线直连频繁 up 和 down 切换

自己移植的内核，开发板 LAN8720A 网口网线直连频繁 up 和 down 切换

```
fec 20b4000.ethernet eth0:Link is Down
fec 20b4000.ethernet eth0:Link is Up -100Mbps/Ful1- flow control rx/tx
```

```
13.740087] fec 20b4000.ethernet eth0: Link is Down
14.820073] fec 20b4000.ethernet eth0: Link is Up - 100Mbps/Full - flow control rx/tx
15.889835] fec 20b4000.ethernet eth0: Link is Down
17.939969] fec 20b4000.ethernet eth0: Link is Up - 100Mbps/Full - flow control rx/tx
19.009828] fec 20b4000.ethernet eth0: Link is Down
20.020095] fec 20b4000.ethernet eth0: Link is Up - 100Mbps/Full - flow control rx/tx
21.089498] fec 20b4000.ethernet eth0: Link is Down
22.100204] fec 20b4000.ethernet eth0: Link is Up - 100Mbps/Full - flow control rx/tx
23.169825] fec 20b4000.ethernet eth0: Link is Down
24.181746] fec 20b4000.ethernet eth0: Link is Up - 100Mbps/Full - flow control rx/tx
25.249823] fec 20b4000.ethernet eth0: Link is Down
27.300108] fec 20b4000.ethernet eth0: Link is Up - 100Mbps/Full - flow control rx/tx
28.369832] fec 20b4000.ethernet eth0: Link is Down
29.380100] fec 20b4000.ethernet eth0: Link is Up - 100Mbps/Full - flow control rx/tx
30.449851] fec 20b4000.ethernet eth0: Link is Down
31.460090] fec 20b4000.ethernet eth0: Link is Up - 100Mbps/Full - flow control rx/tx
32.529811] fec 20b4000.ethernet eth0: Link is Down
36.660104] fec 20b4000.ethernet eth0: Link is Up - 100Mbps/Full - flow control rx/tx
37.729831] fec 20b4000.ethernet eth0: Link is Down
38.740055] fec 20b4000.ethernet eth0: Link is Up - 100Mbps/Full - flow control rx/tx
38.901608] can-3v3: disabling
```

图 6.3.1-1 报错信息

这种情况，可能是硬件问题也可能是软件问题，可以直接用原子的出厂的整套系统（原子的 uboot 和内核以及设备树）来测试看看是否还有这个现象，如果用原子的整套系统来测试没这个现象，说明是自己移植的内核和设备树的问题，请按照教程来检查内核移植网口部分，看看哪里没做好。这里主要修改设备树的 pinctrl_lcdif_dat 和 pinctrl_lcdif_ctrl 节点，可以参考下面的数值。改完后，重新编译设备树，并替换设备树，注意，设备树要替换成功。

```
pinctrl_lcdif_dat: lcdifdatgrp {
        fsl,pins = <
                MX6UL_PAD_LCD_DATA00__LCDIF_DATA00  0x49
                MX6UL_PAD_LCD_DATA01__LCDIF_DATA01  0x49
                MX6UL_PAD_LCD_DATA02__LCDIF_DATA02  0x49
                MX6UL_PAD_LCD_DATA03__LCDIF_DATA03  0x49
                MX6UL_PAD_LCD_DATA04__LCDIF_DATA04  0x49
                MX6UL_PAD_LCD_DATA05__LCDIF_DATA05  0x49
                MX6UL_PAD_LCD_DATA06__LCDIF_DATA06  0x49
                MX6UL_PAD_LCD_DATA07__LCDIF_DATA07  0x51
                MX6UL_PAD_LCD_DATA08__LCDIF_DATA08  0x49
                MX6UL_PAD_LCD_DATA09__LCDIF_DATA09  0x49
                MX6UL_PAD_LCD_DATA10__LCDIF_DATA10  0x49
                MX6UL_PAD_LCD_DATA11__LCDIF_DATA11  0x49
                MX6UL_PAD_LCD_DATA12__LCDIF_DATA12  0x49
                MX6UL_PAD_LCD_DATA13__LCDIF_DATA13  0x49
                MX6UL_PAD_LCD_DATA14__LCDIF_DATA14  0x49
                MX6UL_PAD_LCD_DATA15__LCDIF_DATA15  0x51
                MX6UL_PAD_LCD_DATA16__LCDIF_DATA16  0x49
                MX6UL_PAD_LCD_DATA17__LCDIF_DATA17  0x49
                MX6UL_PAD_LCD_DATA18__LCDIF_DATA18  0x49
                MX6UL_PAD_LCD_DATA19__LCDIF_DATA19  0x49
                MX6UL_PAD_LCD_DATA20__LCDIF_DATA20  0x49
                MX6UL_PAD_LCD_DATA21__LCDIF_DATA21  0x49
                MX6UL_PAD_LCD_DATA22__LCDIF_DATA22  0x49
                MX6UL_PAD_LCD_DATA23__LCDIF_DATA23  0x51

        >;
};
pinctrl_lcdif_ctrl: lcdifctrlgrp {
        fsl,pins = <
                MX6UL_PAD_LCD_CLK__LCDIF_CLK        0x49
                MX6UL_PAD_LCD_ENABLE__LCDIF_ENABLE  0x49
                MX6UL_PAD_LCD_HSYNC__LCDIF_HSYNC    0x49
                MX6UL_PAD_LCD_VSYNC__LCDIF_VSYNC    0x49

        >;
};
```

图 6.3.1-2 修改设备树节点信息

或者直接使用原子的出厂系统，如果用原子出厂的整套系统来做还有这个问题，可以检查看看是不是网线没接好或者网线有问题，路由器也可能有问题，曾经遇见有的用户，换了一个路由器就好了，还可以观察底板的 ETH2 网口的灯亮灭的情况，底板网口，在没有插网线的情况下，只有橘色的灯在亮，插上网线以后，网口橘色的灯在闪烁而黄绿色的灯常亮。如果网口灯不亮，可能 RJ45 接口有问题或者其他，可以淘宝联系我们售后客服。

图 6.3.1-3 进入出厂系统后的网口灯

第七章 文件系统移植相关问题

7.1 开发板启动过程中报错

7.1.1 can't run '/etc/init.d/rcS': Permission denied

解决办法

```
chmod +x /etc/init.d/rcS
```

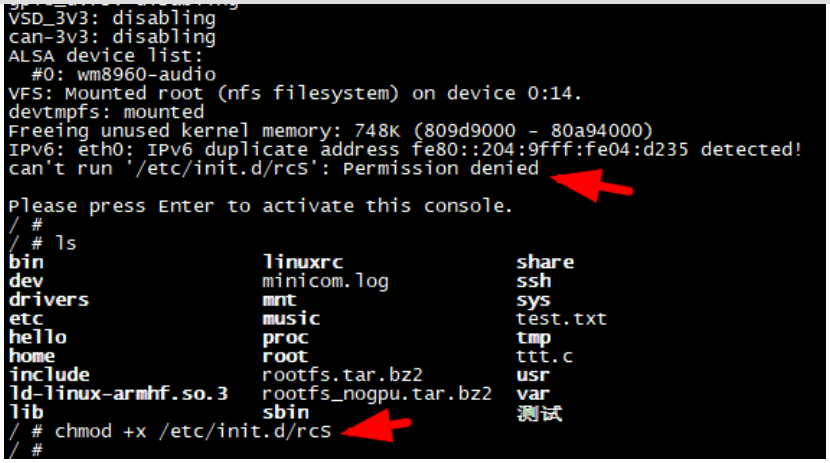


图 7.1.1-1 报错信息

7.1.2 Read-only file system

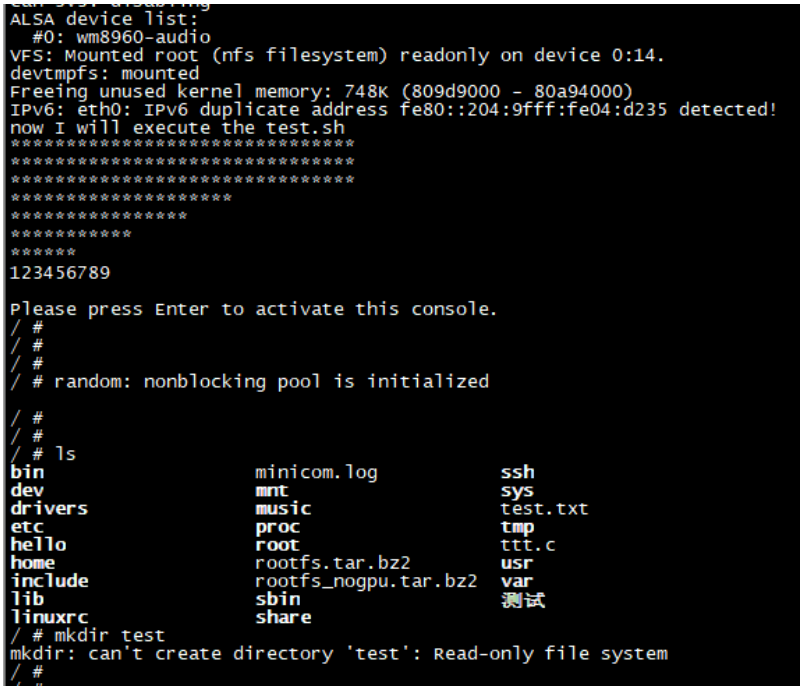


图 7.1.2-1 报错信息

解决办法:

方法①：文件系统只读的，是因为 uboot 下环境变量没设置好，在 bootargs 下加上 rw 字符，文件系统即可进行读写。

```
setenv bootargs 'console=ttymxc0,115200 root=/dev/nfs rw nfsroot=192.168.1.25:/home/MY/NFS/myrootfs
ip=192.168.1.110:192.168.1.25:192.168.1.1:255.255.255.0::eth0:off'
```

修改好 uboot 的环境变量后，重启开发板，进入文件系统后文件系统里的文件可以进行读写操作。

```
=>
=> printenv bootargs
bootargs=console=ttymxc0,115200 root=/dev/nfs nfsroot=192.168.1.25:/home/MY/NFS/myrootfs ip=192.168.1.1
10:192.168.1.25:192.168.1.1:255.255.255.0::eth0:off
=> <INTERRUPT>
=>
=>
=> setenv bootargs 'console=ttymxc0,115200 root=/dev/nfs rw nfsroot=192.168.1.25:/home/MY/NFS/myrootfs
ip=192.168.1.110:192.168.1.25:192.168.1.1:255.255.255.0::eth0:off'
=> saveenv
Saving Environment to MMC...
Writing to MMC(1)... done
=>
```

图 7.1.2-2 添加 rw 权限

方法②或者在根文件系统下执行 mount rw -o remount / 指令重新挂载文件系统，这种方式的话仅对此次开机有效，关机再开机以后就无效了。

```
/ #
/ # mount rw -o remount /
```

图 7.1.2-3 挂载文件系统

7.1.3 文件系统或者内核加载显示乱码

解决方法：

1、如果是网络挂载的话，可能是 uboot 的环境变量 bootargs 里 console 的值错误了，注意空格问题，正确的 console 是没有空格的。例如正确的：

```
setenv bootargs 'console=ttymxc0,115200 root=/dev/nfs nfsroot=以下省略
```

常见的错误就是 console 值格式不对，有空格问题，如错误的红色示范：

```
setenv bootargs 'console= ttymxc0,115200 root=/dev/nfs nfsroot=以下省略
```

或者

```
setenv bootargs 'console=ttymxc0, 115200 root=/dev/nfs nfsroot=以下省略
```

2、可能是串口终端软件没有设置正确，需要设置下串口波特率 115200，8N1。

3、可能是串口连接不对，或者核心板接触不良。检查下串口跳线帽是否正确，重新插拔下串口和核心板。

4、还有一种可能就是本身的内核、文件系统问题，用网盘提供的教程源码验证下。

7.2 加载文件系统问题

7.2.1 Kernel panic - not syncing: VFS: Unable to mount root fs on unknown-block

开发板启动后，串口打印 Kernel panic - not syncing: VFS: Unable to mount root fs on unknown-block 以后就停止了，这个是开发板找不到文件系统报内核恐慌 Kernel panic 提示。这里要注意的是，如果只是测试内核，例如只是把 uboot 和内核以及设备树烧录到 SD 卡里启动，或者只是 TFTP 或者 NFS 加载内核和设备树然后 bootz 启动，那么启动后提示这个是正常的，毕竟 SD 卡里没有文件系统，或者说没有设置 bootargs 加载对应的文件系统，所以系统只能运行到内核阶段，当内核找不到文件系统的时候就报 Kernel panic 了。对此，我们具体情况具体分析。

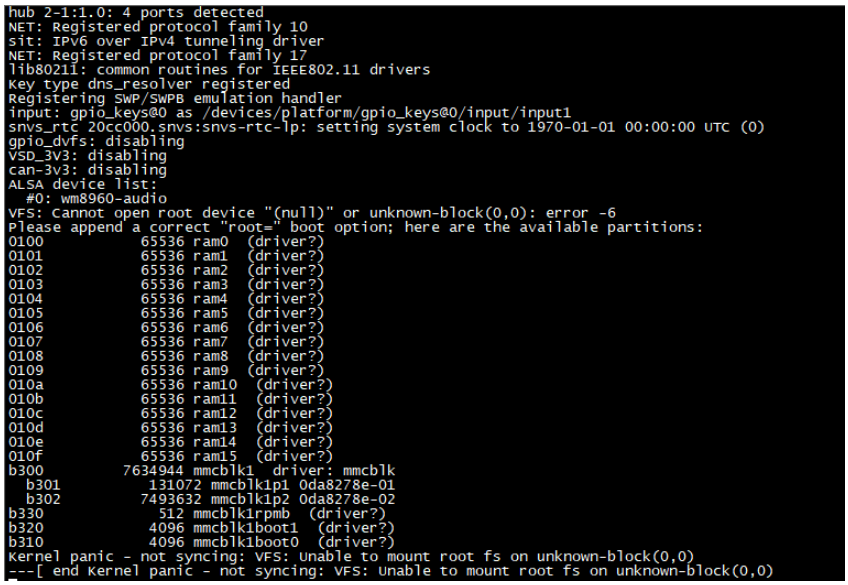


图 7.2.1-1 报错信息

如果需要挂载文件系统，以下进行分情况讲解：

①从 SD 卡或者 EMMC 启动 uboot，需要挂载 EMMC 里的文件系统：

如果挂载的是 EMMC 的文件系统，那么在 uboot 下执行以下指令，注意执行以下指令的前提是 EMMC 里已经烧录了文件系统了（原子出货的时候已经将 Linux 系统烧录到 EMMC 了）：

```
setenv bootargs 'console=ttymxc0,115200 root=/dev/mmcblk1p2 rootwait rw'
saveenv
```

②从 SD 卡或者 EMMC 启动 uboot，需要挂载 SD 卡里的文件系统（前提是 SD 卡里已经有文件系统）

```
setenv bootargs 'console=ttymxc0,115200 root=/dev/mmcblk0p2 rootwait rw'
saveenv
```

③从 SD 卡或者 EMMC 启动 uboot，需要挂载 NFS 里的文件系统（前提是 NFS 开发环境已经搭建好，文件系统已经在 ubuntu 的 NFS 目录里解压好）

```
setenv bootargs 'console=ttymxc0,115200 \
                 root=/dev/nfs rw \
                 nfsroot=192.168.1.25:/home/MY/NFS/rootfs \
                 ip=192.168.1.110:192.168.1.25:192.168.1.1:255.255.255.0::eth0:off'
saveenv
```

以上指令中，ubuntu 的 IP 地址是 192.168.1.25，开发板的 IP 地址是 192.168.1.110，网关是 192.168.1.1，子网掩码是 255.255.255.0，ubuntu 上 NFS 文件系统的目录是/home/MY/NFS/rootfs

在 NFS 挂载文件系统操作上，要注意以上的 IP 地址不能写错，指令格式也不能写错，NFS 文件系统目录要正确，如果不确定自己的 NFS 目录是什么，可以在 ubuntu 上执行指令 showmount -e 查看。执行指令 showmount -e 查看 NFS 共享目录是/home/MY/NFS，在这个目录下的 myrootfs 就存放我的文件系统，所以我的文件系统目录是/home/MY/NFS/myrootfs。

```
MY@IMX6ULL:~$ showmount -e
Export list for IMX6ULL:
/home/MY/NFS *
MY@IMX6ULL:~$ cd /home/MY/NFS
MY@IMX6ULL:~/NFS$ ls
666     hello                              my.c       rootfs.tar      ubuntu_rootfs-alientek-emmc          zuo
buildr  imx6ull-14x14-emmc-4.3-800x480-c.dtb  myrootfs  rootfs.tar.bz2  ubuntu_rootfs-alientek-emmc.tar.bz2
deng    mfgtool2                           rootfs     te              zImage
MY@IMX6ULL:~/NFS$ cd myrootfs/
MY@IMX6ULL:~/NFS/myrootfs$ ls
bin  drivers  home  linuxrc      mnt    proc  rootfs_nogpu.tar.bz2  sbin  sys       tmp    usr  测试
dev  etc      lib   minicom.log  music  root  rootfs.tar.bz2        ssh   test.txt  ttt.c  var
MY@IMX6ULL:~/NFS/myrootfs$ pwd
/home/MY/NFS/myrootfs
MY@IMX6ULL:~/NFS/myrootfs$
```

图 7.2.1-2 查看 NFS 目录

如果是 ubuntu18 版本，或者串口打印信息提示 VFS: Unable to mount root fs via NFS, trying floppy.或者或者报错 Loading:*ww ERROR:File lookup fail 的，或者 mount.nfs: mount system call failed

可能就是 uboot 支持的协议与 NFS 支持的协议不同导致报错，可以进行如下修改：

在 ubuntu 的/etc/default/nfs-kernel-server 文件中，按照如下图箭头所指的部分进行修改，改完后保存退出，再执行指令 sudo service nfs-kernel-server restart 重启 NFS 服务。

```
# Number of servers to start up
#RPCNFSDCOUNT=8
RPCNFSDCOUNT="-V 2 8"

# Runtime priority of server (see nice(1))
RPCNFSDPRIORITY=0

# Options for rpc.mountd.
# If you have a port-based firewall, you might want to set up
# a fixed port here using the --port option. For more information,
# see rpc.mountd(8) or http://wiki.debian.org/SecuringNFS
# To disable NFSv4 on the server, specify '--no-nfs-version 4' here
#RPCMOUNTDOPTS="--manage-gids"
RPCMOUNTDOPTS="-V 2 --manage-gids"

# Do you want to start the svcgssd daemon? It is only required for Kerberos
# exports. Valid alternatives are "yes" and "no"; the default is "no".
NEED_SVCGSSD=""

# Options for rpc.svcgssd.
RPCSVCGSSDOPTS="--nfs-version 2,3,4 --debug --syslog"
```

图 7.2.1-3 修改文件

7.2.2 Kernel panic - not syncing: No working init found.

系统报错信息：

Kernel panic - not syncing: No working init found. Try passing init= option to kernel. See Linux Documentation/init.txt for guidance.

---[end Kernel panic - not syncing: No working init found. Try passing init= option to kernel. See Linux Documentation/init.txt for guidance.

或者

Kernel panic - not syncing: Attempted to kill init! exitcode=0x0000000b

---[end Kernel panic - not syncing: Attempted to kill init! exitcode=0x0000000b

```
sit: IPv6 over IPv4 tunneling driver
NET: Registered protocol family 17
lib80211: common routines for IEEE802.11 drivers
Key type dns_resolver registered
Registering SWP/SWPB emulation handler
input: gpio-keys as /devices/platform/gpio-keys/input/input4
snvs_rtc 20cc000.snvs:snvs-rtc-lp: setting system clock to 1970-01-01 02:54:52 UTC (10492)
fec 20b4000.ethernet eth0: Freescale FEC PHY driver [SMSC LAN8710/LAN8720] (mii_bus:phy_addr=20b4000.et
hernet:01, irq=-1)
IPv6: ADDRCONF(NETDEV_UP): eth0: link is not ready
fec 20b4000.ethernet eth0: Link is Up - 100Mbps/Full - flow control rx/tx
IPv6: ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready
IP-Config: Complete:
     device=eth0, hwaddr=00:04:9f:04:d2:35, ipaddr=192.168.1.110, mask=255.255.255.0, gw=192.168.1.1
     host=192.168.1.110, domain=, nis-domain=(none)
     bootserver=192.168.1.25, rootserver=192.168.1.25, rootpath=
gpio_dvfs: disabling
VSD_3V3: disabling
can-3v3: disabling
ALSA device list:
  #0: wm8960-audio
VFS: Mounted root (nfs filesystem) on device 0:14.
devtmpfs: mounted
Freeing unused kernel memory: 748K (809d9000 - 80a94000)
Kernel panic - not syncing: No working init found.  Try passing init= option to kernel. See Linux Docum
entation/init.txt for guidance.
---[ end Kernel panic - not syncing: No working init found.  Try passing init= option to kernel. See Li
nux Documentation/init.txt for guidance.
```

图 7.2.2-1 报错信息

或者

```
fec 20b4000.ethernet eth0: Freescale FEC PHY driver [SMSC LAN8710/LAN8720] (mii_bus:phy_addr=20b4000.et
hernet:01, irq=-1)
IPv6: ADDRCONF(NETDEV_UP): eth0: link is not ready
fec 20b4000.ethernet eth0: Link is Up - 100Mbps/Full - flow control rx/tx
IPv6: ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready
IP-Config: Complete:
     device=eth0, hwaddr=00:04:9f:04:d2:35, ipaddr=192.168.1.110, mask=255.255.255.0, gw=192.168.1.1
     host=192.168.1.110, domain=, nis-domain=(none)
     bootserver=192.168.1.25, rootserver=192.168.1.25, rootpath=
gpio_dvfs: disabling
VSD_3V3: disabling
can-3v3: disabling
ALSA device list:
  #0: wm8960-audio
VFS: Mounted root (nfs filesystem) on device 0:14.
devtmpfs: mounted
Freeing unused kernel memory: 748K (809d9000 - 80a94000)
Kernel panic - not syncing: Attempted to kill init! exitcode=0x0000000b

---[ end Kernel panic - not syncing: Attempted to kill init! exitcode=0x0000000b
```

图 7.2.2-2 报错信息

文件系统启动报错 not syncing: No working init found. Try passing init= option to kernel.初始化失败的，会有两种情况：

①一般是少了什么库文件，例如 ld-linux-armhf.so.3 这个文件，这个是个符号链接文件，可以按照教程《【正点原子】I.MX6U 嵌入式 Linux 驱动开发指南》里 38.2.3 向根文件系统添加 lib 这章节检查一遍。

件也是个符号链接，相当于 Windows 下的快捷方式。会链接到库 ld-2.19-2014.08-1-git.so 上，输入命令“ls ld-linux-armhf.so.3 -l”查看此文件详细信息，如图 38.2.3.1 所示：

```
zuozhongkai@ubuntu:~/linux/nfs/rootfs$ cd lib/
zuozhongkai@ubuntu:~/linux/nfs/rootfs/lib$ ls ld-linux-armhf.so.3 -l
lrwxrwxrwx 1 zuozhongkai zuozhongkai 24 Jun 13 12:36 ld-linux-armhf.so.3 -> ld-2.19-2014.08-1-git.so
```

图 38.2.3.1 文件 ld-linux-armhf.so.3

从图 38.2.3.1 可以看出，ld-linux-armhf.so.3 后面有个“->”，表示其是个软连接文件，链接到文件 ld-2.19-2014.08-1-git.so，因为其是一个“快捷方式”，因此大小只有 24B。但是，ld-linux-armhf.so.3 不能作为符号链接，否则的话在根文件系统中执行程序无法执行！所以我们需要 ld-linux-armhf.so.3 完成逆袭，由“快捷方式”变为“本尊”，方法很简单，那就是重新复制 ld-linux-armhf.so.3，只是不复制软链接即可，先将 rootfs/lib 中的 ld-linux-armhf.so.3 文件删除掉，命令如下：

```
rm ld-linux-armhf.so.3
```

然后重新进入到/usr/local/arm/gcc-linaro-4.9.4-2017.01-x86_64_arm-linux-gnueabihf/arm-linux-gnueabihf/libc/lib 目录中，重新拷贝 ld-linux-armhf.so.3，命令如下：

```
cp ld-linux-armhf.so.3 /home/zuozhongkai/linux/nfs/rootfs/lib/
```

拷贝完成以后再到 rootfs/lib 目录下查看 ld-linux-armhf.so.3 文件详细信息，如图 38.2.3.2 所示：

```
zuozhongkai@ubuntu:~/linux/nfs/rootfs/lib$ rm  ld-linux-armhf.so.3
zuozhongkai@ubuntu:~/linux/nfs/rootfs/lib$ ls ld-linux-armhf.so.3 -l
-rwxr-xr-x 1 zuozhongkai zuozhongkai 724392 Jun 13 12:59 ld-linux-armhf.so.3
zuozhongkai@ubuntu:~/linux/nfs/rootfs/lib$
```

图 38.2.3.2 文件 ld-linux-armhf.so.3

从图 38.2.3.2 可以看出，此时 ld-linux-armhf.so.3 已经不是软连接了，而是实实在在的一个库文件，而且文件大小为 724392B。

图 7.2.2-3 向根文件系统添加 lib

②可能打包文件系统的时候方式错了，如下图，在 myrootfs 里是我的文件系统，如果要打包文件系统，下图的打包方法是在 myrootfs 目录外边进行操作，这个是错的：

```
MY@IMX6ULL:~/NFS$ ls
666     deng   imx6ull-14x14-emmc-4.3-800x480-c.dtb  my.c      rootfs          te    ubuntu_rootfs-alientek-emmc          zImage
buildr  hello  myrootfs                              myrootfs  rootfs.tar.bz2  test  ubuntu_rootfs-alientek-emmc.tar.bz2  zuo
MY@IMX6ULL:~/NFS$ sudo tar -vcjf rootfs.tar.bz2 myrootfs/*
```

图 7.2.2-4 错误打包：在文件系统目录外打包

正确的方法是进入 myrootfs 里边进行打包：

```
cd 文件系统
tar -vcjf rootfs.tar.bz2 *
```

```
MY@IMX6ULL:~/NFS$ ls
666     deng   imx6ull-14x14-emmc-4.3-800x480-c.dtb  my.c      rootfs          te    ubuntu_rootfs-alientek-emmc          zImage
buildr  hello  myrootfs                              myrootfs  rootfs.tar.bz2  test  ubuntu_rootfs-alientek-emmc.tar.bz2  zuo
MY@IMX6ULL:~/NFS$ cd myrootfs/
MY@IMX6ULL:~/NFS/myrootfs$ ls
bin  drivers  hello  lib64  linuxrc      mnt    proc  rootfs_nogpu.tar.bz2  sbin  ssh  test      tmp  usr  测试
dev  etc      home   lib    minicom.log  music  root  rootfs.tar.bz2        shell  sys  test.txt  ttt.c  var
MY@IMX6ULL:~/NFS/myrootfs$ tar -vcjf rootfs.tar.bz2 *
```

图 7.2.2-5 进入到文件系统后打包

写到开发板中，首先是准备好要烧写的原材料：

①、自己移植编译出来的 uboot 可执行文件：u-boot.imx。

②、自己移植编译出来的 zImage 镜像文件和开发板对应的.dtb(设备树)，对于 I.MX6U-ALPHA 开发板来说就是 imx6ull-alientek-emmc.dtb。

③、自己构建的根文件系统 rootfs，这里我们需要对 rootfs 进行打包，进入到 Ubuntu 中的 rootfs 目录中，然后使用 tar 命令对其进行打包，命令如下：

```
cd rootfs/
tar -vcjf rootfs.tar.bz2 *
```

完成以后会在 rootfs 目录下生成一个名为 rootfs.tar.bz2 的压缩包，将 rootfs.tar.bz2 发送到 windows 系统中。

图 7.2.2-6 教程说明

7.2.3 Kernel panic - not syncing: VFS: Unable to mount root fs on unknown-block(179,2)

如果说已经将 SD 卡做成了一张系统启动卡，比如说已经按照《【正点原子】I.MX6U 用户快速体验》的 2.2.1.1 固化系统到 SD 的章节内容操作，做好了一张系统启动卡，启动后报错：

```
No filesystem could mount root, tried: ext3 ext2 ext4 vfat fuseblk
Kernel panic - not syncing: VFS: Unable to mount root fs on unknown-block(179,2)
---[ end Kernel panic - not syncing: VFS: Unable to mount root fs on unknown-block(179,2)
```

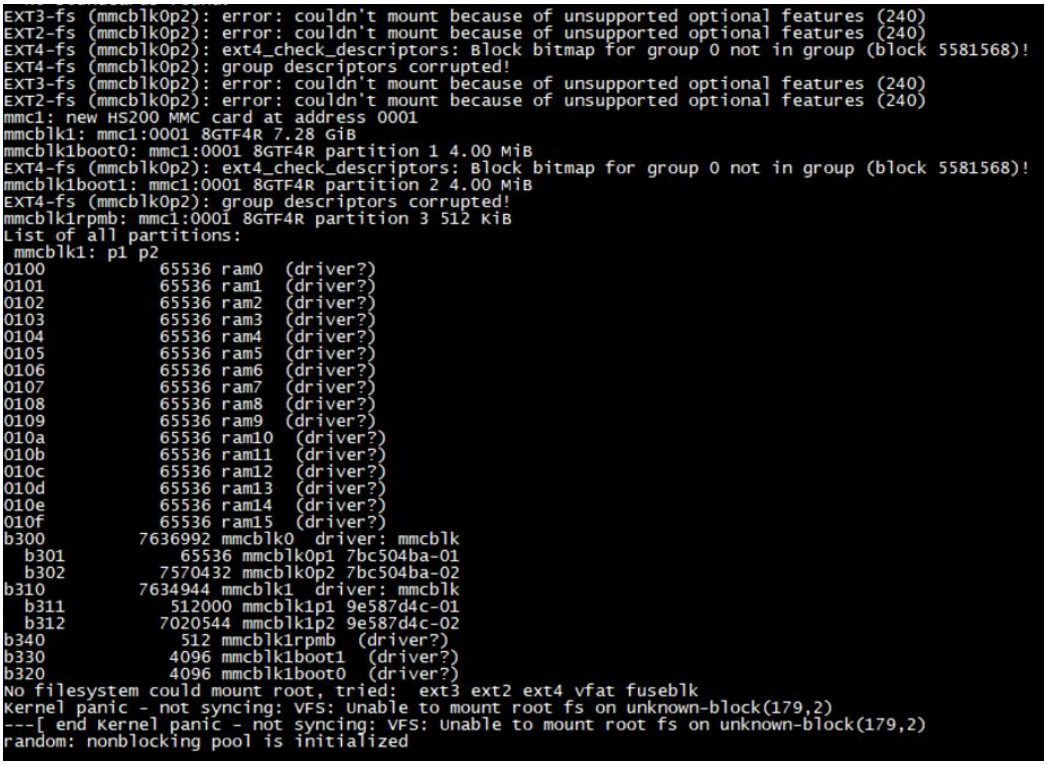


图 7.2.3-1 报错信息

这种情况也是 SD 卡里的文件系统无法加载，可以在 uboot 下用指令 printenv 查看是不是之前修改了环境变量导致的，可以在 uboot 下执行指令来清除掉之前设置的所有环境变量，恢复为 uboot 本身默认的环境变量：

```
setenv default -a
saveenv
```

在恢复环境变量以后，还是无法启动，可以试试将 TF 卡拔出插在 ubuntu 上查看，如下图，可能在 ubuntu 上显示：

```
Unable to access "rootfs"
Error mounting /dev/sdb2 at/media/alientek/rootfs:Command-line 'mount-t
"ext4" -o "uhelper=udisks2,node,nosuid" "/dev/sdb2" "/media/alientek/rootfs"
exited with non-zero exit status 32:mount:Structure needs cleaning
```

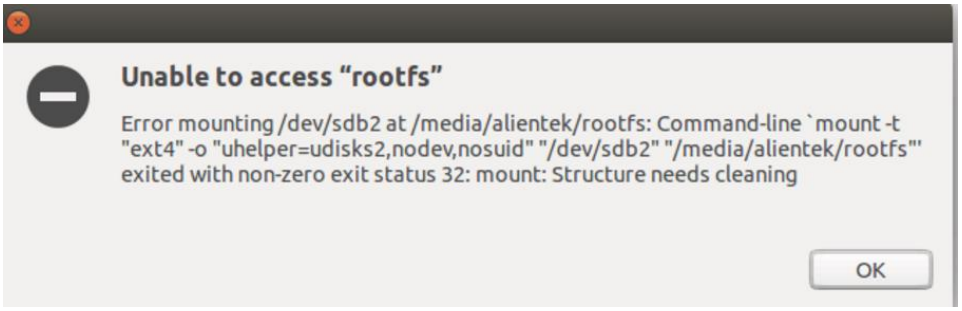


图 7.2.3-2 显示 rootfs 分区无法挂载

如上，在 ubuntu 上查看，rootfs 分区无法挂载，boot 分区可以，应该是 rootfs 分区坏了，可以重新做一张系统启动卡，或者，如果不想重新做系统启动卡的话，可以直接将 rootfs 分区格式化，再将之前制作的文件系统解压到 SD 卡的 rootfs 分区，如果采用后者：

```
sudo fdisk -l                          //查看分区，rootfs 分区对应的设备是/dev/sdb2
mkfs.ext4 -L "rootfs" /dev/sdb2        //格式化/dev/sdb2
```

就可以重新得到一个空的 rootfs 分区了，将之前制作好的文件系统解压到 SD 卡的 rootfs 分区，大功告成。这里扩展一下，格式化 boot 分区的指令是：

```
sudo mkfs.vfat -F 32 -n "boot" /dev/sdb1
```

7.2.4 mount.nfs: an incorrect mount option was specified

如果是想开发板启动进入系统以后再挂载 NFS 文件系统，开发板启动进入文件系统，开发板和 ubuntu 能互相 ping 通，在开发板文件系统下新建一个目录 you，然后执行如下指令进行挂载：

```
mkdir you
mount -t nfs -o nolock,192.168.1.25:/home/MY/NFS/myrootfs you/
```

会报错：

```
mount: can't find you/ in /etc/fstab
```

或者有的会报错：

```
mount.nfs: an incorrect mount option was specified
```

```
/ # mount -t nfs -o nolock,192.168.1.25:/home/MY/NFS/myrootfs you/
mount: can't find you/ in /etc/fstab
/ #
```

图 7.2.4-1 报错信息

解决办法就是上面的**指令里加 nfsvers=3**，这里指定 nfs 版本为 3，不加这个的话，就会提示各种问题。

```
mount -t nfs -o nolock,nfsvers=3 192.168.1.25:/home/MY/NFS/myrootfs you/
```

```
/ # ifconfig
eth0      Link encap:Ethernet  HWaddr DA:A2:B2:06:92:8A
          inet addr:192.168.1.110  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::d8a2:b2ff:fe06:928a/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:338 errors:0 dropped:0 overruns:0 frame:0
          TX packets:7 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:21964 (21.4 KiB)  TX bytes:578 (578.0 B)

/ # ping 192.168.1.25
PING 192.168.1.25 (192.168.1.25): 56 data bytes
64 bytes from 192.168.1.25: seq=0 ttl=64 time=4.242 ms
64 bytes from 192.168.1.25: seq=1 ttl=64 time=0.444 ms
64 bytes from 192.168.1.25: seq=2 ttl=64 time=0.433 ms
^C
--- 192.168.1.25 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 0.433/1.706/4.242 ms
/ #
/ #
/ # ls you/
/ # mount -t nfs -o nolock,nfsvers=3 192.168.1.25:/home/MY/NFS/myrootfs you/
/ # ls you/
bin                 mnt                 sys
dev                 music               test.txt
drivers             proc                tmp
etc                 root                ttt.c
home                rootfs.tar.bz2      usr
lib                 rootfs_nogpu.tar.bz2 var
linuxrc             sbin                测试
minicom.log         ssh
/ #
```

图 7.2.4-2 挂载

7.2.5 nfs: server 192.168.1.25 not responding, still trying

或者

Loading:*ww ERROR:File lookup fail

```
IPv6: ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready
IP-Config: Complete:
     device=eth0, hwaddr=00:04:9f:04:d2:35, ipaddr=192.168.1.110, mask=255.255.255.0, gw=192.168.1.1
     host=192.168.1.110, domain=, nis-domain=(none)
     bootserver=192.168.1.25, rootserver=192.168.1.25, rootpath=
gpio_dvfs: disabling
VSD_3V3: disabling
can-3v3: disabling
ALSA device list:
  #0: wm8960-audio
VFS: Mounted root (nfs filesystem) on device 0:14.
devtmpfs: mounted
Freeing unused kernel memory: 748K (809d9000 - 80a94000)
IPv6: eth0: IPv6 duplicate address fe80::204:9fff:fe04:d235 detected!
nfs: server 192.168.1.25 not responding, still trying
nfs: server 192.168.1.25 OK
now I will execute the test.sh
********************************
********************************
********************************
********************
****************
**********
******
123456789

nfs: server 192.168.1.25 not responding, still trying
random: nonblocking pool is initialized
```

图 7.2.5-1 报错信息

出现这种问题的，可能的情况是：

（1）IP 地址设置的问题

如图可以看出，NFS 服务器的 IP 地址为 192.168.1.25，ipaddr 是 192.168.1.110，属于 192.168.1.X 这个网段，可以检查开发板文件系统的/etc/init.d/rcS

```
#!/bin/sh
PATH=/sbin:/bin:/usr/sbin:/usr/bin
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/lib:/usr/lib
export PATH LD_LIBRARY_PATH runlevel
mount -a
mkdir /dev/pts
mount -t devpts devpts /dev/pts
echo /sbin/mdev > /proc/sys/kernel/hotplug
mdev -s
#test automatically execute
#cd /drivers
echo "now I will execute the test.sh"
echo "*******************************"
echo "*******************************"
echo "*******************************"
./drivers/test.sh
cd /

ifconfig eth0 192.168.0.88 netmask 255.255.255.0

~
~
```

图 7.2.5-2 /etc/init.d/rcS 文件

发现如上图设置的 eth0 的 IP 地址是 192.168.0.88，不在 192.168.1.X 的网段，所以修改将其修改为 192.168.1.88，重启开发板问题解决。

```
echo "*******************************"
echo "*******************************"
./drivers/test.sh
cd /

ifconfig eth0 192.168.1.88 netmask 255.255.255.0
```

图 7.2.5-3 修改正确的网段

（2）NFS 版本的问题，uboot 用的版本 2，在 ubuntu16.04 上是不用设置的，到了 18.4 就要设置用的 3 和 4，所以如果使用的是 ubuntu18.04 或者 ubuntu20.04 的话，需要在/etc/default/nfs-kernel-server 文件进行如下修改，修改完以后保存退出，再执行指令：

`sudo service nfs-kernel-server restart` 重启 NFS 服务。

```
# Number of servers to start up
#RPCNFSDCOUNT=8
RPCNFSDCOUNT="-V 2 8"

# Runtime priority of server (see nice(1))
RPCNFSDPRIORITY=0

# Options for rpc.mountd.
# If you have a port-based firewall, you might want to set up
# a fixed port here using the --port option. For more information,
# see rpc.mountd(8) or http://wiki.debian.org/SecuringNFS
# To disable NFSv4 on the server, specify '--no-nfs-version 4' here
#RPCMOUNTDOPTS="--manage-gids"
RPCMOUNTDOPTS="-V 2 --manage-gids"

# Do you want to start the svcgssd daemon? It is only required for Kerberos
# exports. Valid alternatives are "yes" and "no"; the default is "no".
NEED_SVCGSSD=""

# Options for rpc.svcgssd.
RPCSVCGSSDOPTS="--nfs-version 2,3,4 --debug --syslog"
```

图 7.2.5-4 修改 NFS 配置文件

7.2.6 nfs 挂载时出现##################TTTTTT 问题

问题：之前用 NFS 网络挂载文件系统没问题，但有时候会遇到 ###T 的情况。

```
Hit any key to stop autoboot:  0
FEC1 Waiting for PHY auto negotiation to complete... done
Using FEC1 device
TFTP from server 192.168.1.100; our IP address is 192.168.1.50
Filename 'zImage'.
Load address: 0x80800000
Loading: #T T T T T T T T T #
Retry count exceeded; starting again
Using FEC1 device
TFTP from server 192.168.1.100; our IP address is 192.168.1.50
Filename 'imx6ull-alientek-emmc.dtb'.
Load address: 0x83000000
Loading: *
```

图 7.2.6-1 现象

解决思路：之前 NFS 能使用说明 NFS 其实功能正常（在没有修改的情况下）。考虑是否是网络网速问题。其实只要是最后提示类似 Bytes transferred =5901744（5a0dbo hex）就可以了，可以试下 boot 启动。

参考解决：发现百度网盘在下载，占用网络资源。

```
FEC1 Waiting for PHY auto negotiation to complete... done
Using FEC1 device
TFTP from server 192.168.1.100; our IP address is 192.168.1.50
Filename 'zImage'.
Load address: 0x80800000
Loading: ######################T ############################T ##############
#
         ##################################################################
         ########################T ###################################T ####
#
         #######T #T #T ##########################################################T
######
         ##T ########T ####################
Retry count exceeded; starting again
Using FEC1 device
TFTP from server 192.168.1.100; our IP address is 192.168.1.50
Filename 'imx6ull-alientek-emmc.dtb'.
Load address: 0x83000000
Loading: ###
         859.4 KiB/s
done
Bytes transferred = 36971 (906b hex)
Kernel image @ 0x80800000 [ 0x000000 - 0x5a0db0 ]
## Flattened Device Tree blob at 83000000
   Booting using the fdt blob at 0x83000000
   Using Device Tree in place at 83000000, end 8300c06a

Starting kernel ...
```


图 7.2.6-2 网速受限

关闭百度网盘后，NFS 网络正常。

```
Normal Boot
Hit any key to stop autoboot:  0
FEC1 Waiting for PHY auto negotiation to complete... done
Using FEC1 device
TFTP from server 192.168.1.100; our IP address is 192.168.1.50
Filename 'zImage'.
Load address: 0x80800000
Loading: ##################################################################
         ##################################################################
         ##################################################################
         ##################################################################
         ##################################################################
         ##################################################################
         #############
         896.5 KiB/s
done
Bytes transferred = 5901744 (5a0db0 hex)
Using FEC1 device
TFTP from server 192.168.1.100; our IP address is 192.168.1.50
Filename 'imx6ull-alientek-emmc.dtb'
Load address: 0x83000000
Loading: ###
         859.4 KiB/s
done
Bytes transferred = 36971 (906b hex)
```

图 7.2.6-3 正常加载

7.2.7 mount: can't find you/ in /etc/fstab

在文件系统上挂载 NFS 目录的时候提示

```
mount: can't find you/ in /etc/fstab
```

其中/you/是文件系统根目录下的一个目录。

```
Freeing unused kernel memory: 752K (80984000 - 80a40000)
now I will execute the test.sh
**********
**********
**********
**********
**********
**********
**********
******
123456789

Please press Enter to activate this console. random: nonblocking pool is initialized
/ #
/ # mount -t nfs -o nolock,192.168.1.25:/home/MY/NFS/myrootfs you/
mount: can't find you/ in /etc/fstab
/ # ifconfig
/ #
```

图 7.2.7-1 报错信息

解决方法：指令得改一下，下面指令里加了个 nfsvers=3

```
mount -t nfs -o nolock,nfsvers=3 192.168.1.25:/home/MY/NFS/myrootfs you/
```

7.2.8 mount: wrong fs type, bad option, bad superblock on /dev/sdb

原因：挂载时未格式化，使用的文件系统格式不对

解决方案：格式化

```
sudo mkfs -t ext4 /dev/sdb
```

再挂载（/xxx/是要挂载的目录）

```
sudo mount /dev/sdb /xxx/
```

7.2.9 nfs: server 192.168.1.215 not responding, still trying

现象：网络都没问题，但 nfs 反应慢，经常掉线。

```
nfs: server 192.168.1.215 not responding, still trying
nfs: server 192.168.1.215 not responding, still trying
nfs: server 192.168.1.215 OK
nfs: server 192.168.1.215 OK
nfs: server 192.168.1.215 OK
nfs: server 192.168.1.215 OK
nfs: server 192.168.1.215 OK
```

图 7.2.9-1 现象

解决：

使用路由器或者交换机直接板子和电脑，或者板子换个 IP，重新按帖子的 nfs 命令挂载。或者重新解压文件系统，可能文件系统错误。

7.2.10 /sbin/init exists but couldn't execute it

无法挂载启动做好的内核和系统，报错系统如下：

```
VFS: Mounted root (nfs filesystem) on device 0:14.
devtmpfs: mounted
Freeing unused kernel memory: 436K (80b47000 - 80bb4000)
request_module: runaway loop modprobe binfmt-464c
Starting init: /sbin/init exists but couldn't execute it (error -8)
request_module: runaway loop modprobe binfmt-464c
Starting init: /bin/sh exists but couldn't execute it (error -8)
Kernel panic - not syncing: No working init found.  Try passing init= option to kernel. See Linux D
---[ end Kernel panic - not syncing: No working init found.  Try passing init= option to kernel. Se
```

图 7.2.10-1 报错信息

解决：

没有更改 makefile 的编译器选项，默认 x86 的编译器编译了这个文件，所以 init 阶段就跑不起来。参考：https://www.freesion.com/article/44611148946/

同 Uboot 和 Linux 移植一样，打开 busybox 的顶层 Makefile，添加 ARCH 和 CROSS_COMPILE 的值，如下所示：

示例代码 38.2.2.1 Makefile 代码段

```
164 CROSS_COMPILE ?= /usr/local/arm/gcc-linaro-4.9.4-2017.01-
x86_64_arm-linux-gnueabihf/bin/arm-linux-gnueabihf-
......
190 ARCH ?= arm
```

图 7.2.10-2 修改 Makefile 编译器选项

7.3 编译 busybox 报错

7.3.1 busybox 图形化编译出错

报错信息：

```
czx@ubuntu16:~/busybox-1.29.0$ make menuconfig
  HOSTCC  scripts/kconfig/lxdialog/checklist.o
In file included from scripts/kconfig/lxdialog/checklist.c:24:0:
scripts/kconfig/lxdialog/dialog.h:31:20: fatal error: curses.h: 没有那个文件或目
录
compilation terminated.
scripts/Makefile.host:120: recipe for target 'scripts/kconfig/lxdialog/checklist
.o' failed
make[2]: *** [scripts/kconfig/lxdialog/checklist.o] Error 1
/home/czx/busybox-1.29.0/scripts/kconfig/Makefile:14: recipe for target 'menucon
fig' failed
make[1]: *** [menuconfig] Error 2
Makefile:443: recipe for target 'menuconfig' failed
make: *** [menuconfig] Error 2
```

图 7.3.1-1 报错信息

解决：没有安装图形化界面的库

```
sudo apt-get install build-essential
sudo apt-get install libncurses5-dev
```

图 7.3.1-2 安装库

7.4 编译 buildroot 报错

7.4.1 编译器路径选错

报错信息：

package/pkg-generic.mk:254: recipe for target "/home/rog/Desktop/butldroot/buildroot-2020.02.2/output/build/toolchain-external-custom/.stamp_configured' failed

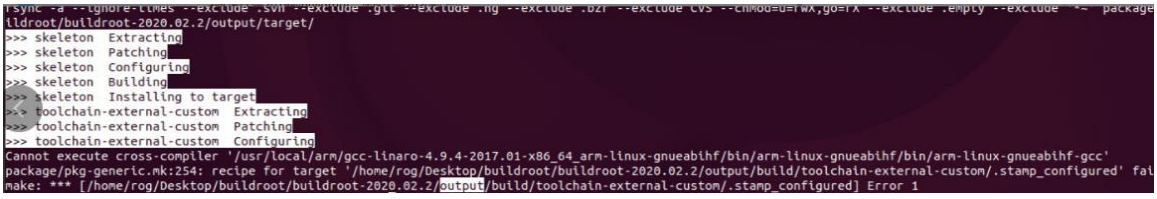


图 7.4.1-1 报错信息

编译器路径选错了，在配置的时候要选对编译器路径：

图 7.4.1-2 配置编译器路径

第八章 驱动实验相关问题

8.1 模块加载问题

8.1.1 Unknown symbol _GLOBAL_OFFSET_TABLE_ (err 0)

```
modprobe: can't open 'modules.dep': No such file or directory
/lib/modules/4.1.15 # depmod
/lib/modules/4.1.15 # modprobe 6_gpioled.ko
6_gpioled: Unknown symbol _GLOBAL_OFFSET_TABLE_ (err 0)
6_gpioled: Unknown symbol _GLOBAL_OFFSET_TABLE_ (err 0)
modprobe: can't load module 6_gpioled.ko (6_gpioled.ko): unknown symbol
in module, or unknown parameter
/lib/modules/4.1.15 #
```

图 8.1.1-1 报错信息

解决办法：在模块源码的 Makefile 文件下加上 EXTRA_CFLAGS=-fno-pic 参数，然后再重新编译模块，如下：

```
1  KERNELDIR := /home/x/workspace/01_linux_kernel/test/imx_4.1.15_2.0.0_ga_rc3/
2  CURRENT_PATH := $(shell pwd)
3  obj-m := gpioled.o
4
5  build: kernel_modules
6
7  kernel_modules:
8      $(MAKE) ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- EXTRA_CFLAGS=-fno-pic -C $(KERNELDIR) M=`pwd` modul
9
```

图 8.1.1-2 修改 Makefile

8.1.2 version magic '4.1.15 SMP preempt mod_unload modversions ARMv6 p2v8 ' should be '4.1.15-g2689732-dirty SMP preempt mod_unload modversions ARMv7 p2v8 '

```
Please press Enter to activate this console.
/ # cd lib/modules/4.1.15/
/lib/modules/4.1.15 # ls
[   47.235165] random: nonblocking pool is initialized
8188eu.ko     dtsled.ko     icm20608App   leddriver.ko
8189fs.ko     icm20608.ko   ledApp
/lib/modules/4.1.15 # insmod 8188eu.ko
[   54.217419] 8188eu: version magic '4.1.15 SMP preempt mod_unload modversions ARMv6 p2v8 ' should be
'4.1.15-g2689732-dirty SMP preempt mod_unload modversions ARMv7 p2v8 '
[   54.242926] 8188eu: version magic '4.1.15 SMP preempt mod_unload modversions ARMv6 p2v8 ' should be
'4.1.15-g2689732-dirty SMP preempt mod_unload modversions ARMv7 p2v8 '
insmod: can't insert '8188eu.ko': invalid module format
/lib/modules/4.1.15 # uname -r
4.1.15-g2689732-dirty
```

图 8.1.2-1 报错信息

这种报错是内核版本和模块版本不一致，所以导致加载不成功，这里，内核版本是 4.1.15-g2689732，而模块版本是 4.1.15，也就是内核是用 A 版本内核源码编译出来的，模块是用 B 版本的内核源码编译出来的，两个不同版本的内核源码，例如，使用的 zImage 是 开发板光盘 A-基础资料\1、例程源码\3、正点原子修改后的 Uboot 和 Linux（出厂源码）这个编译的，模块.ko 文件是用自己移植的内核源码编译出来的，所以导致加载不成功。解决办法就是内核和驱动模块都用同一个版本的内核源码编译，先编译内核再编译模块，然后将编译好的内核和模块一起放到板子上运行。

8.1.3 Unknown symbol device_create (err -22)

disagrees about version of symbol device_create
Unknown symbol device_create (err -22)
modprobe: can't load module newchrled.ko (newchrled.ko): Invalid argument

```
/ # depmod
/ # cd lib/modules/4.1.15/
/lib/modules/4.1.15 # ls
chrdevbase.ko     ledApp            modules.symbols
chrdevbaseApp     modules.alias     newchrled.ko
led.ko            modules.dep       newledApp
/lib/modules/4.1.15 # depmod
/lib/modules/4.1.15 # modprobe newchrled.ko
newchrled: disagrees about version of symbol device_create
newchrled: Unknown symbol device_create (err -22)
newchrled: disagrees about version of symbol device_destroy
newchrled: Unknown symbol device_destroy (err -22)
newchrled: disagrees about version of symbol device_create
newchrled: Unknown symbol device_create (err -22)
newchrled: disagrees about version of symbol device_destroy
newchrled: Unknown symbol device_destroy (err -22)
modprobe: can't load module newchrled.ko (newchrled.ko): Invalid argument
/lib/modules/4.1.15 # random: nonblocking pool is initialized
```

图 8.1.3-1 报错信息 -22

这种报错是内核版本和模块版本不一致导致的，重新编译内核，用编译出来的内核去编译驱动。重新加载内核和驱动运行。

8.1.4 alphaled node nost find!

报错信息：

alphaled node nost find!
insmod: can't insert 'dtsled.ko': Invalid argument

```
/lib/modules/4.1.15 # ls
8188eu.ko         icm20608.ko       leddriver.ko      modules.symbols
8189fs.ko         icm20608App       modules.alias
dtsled.ko         ledApp            modules.dep
/lib/modules/4.1.15 # insmod dtsled.ko
alphaled node nost find!
alphaled node nost find!
insmod: can't insert 'dtsled.ko': Invalid argument
/lib/modules/4.1.15 # uname -r
4.1.15
/lib/modules/4.1.15 #
```

图 8.1.4-1 报错信息

提示 xxx node nost find!的这种，很多时候是设备树没配置好，找不到节点，所以报错。解决办就是核实一下设备树是否替换成功，设备树是否修改正确，重新编译设备树再替换。

```
make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- dtbs
```

8.1.5 modprobe: can't open 'modules.dep': No such file or directory

```
/lib/modules/4.1.15 # ls
8188eu.ko      dtsled.ko      icm20608App    leddriver.ko
8189fs.ko      icm20608.ko    ledApp
/lib/modules/4.1.15 # modprobe leddriver.ko
modprobe: can't open 'modules.dep': No such file or directory
/lib/modules/4.1.15 # ls
8188eu.ko      dtsled.ko      icm20608App    leddriver.ko
8189fs.ko      icm20608.ko    ledApp
/lib/modules/4.1.15 # depmod
/lib/modules/4.1.15 # ls
8188eu.ko         icm20608.ko       leddriver.ko      modules.symbols
8189fs.ko         icm20608App       modules.alias
dtsled.ko         ledApp            modules.dep
/lib/modules/4.1.15 # modprobe leddriver.ko
led driver and device was matched!
genirq: Flags mismatch irq 50. 00000002 (tempKEY0) vs. 00000083 (GPIO Key Enter)
/lib/modules/4.1.15 # lsmod
Module                  Size  Used by    Tainted: G
leddriver               3304  0
/lib/modules/4.1.15 #
```

图 8.1.5-1 报错信息

解决办法：先执行 depmod 再执行 modprobe leddriver.ko

使用 modprobe 来加载模块的提示 can't open 'modules.dep': No such file or directory 的是因为没有执行 depmod，如果执行 depmod 的话，depmod 可以分析可加载模块的依赖性，然后生成 modules.dep 文件和映射文件，和 modules.alias、modules.symbols 文件，再执行 modprobe 就可以加载成功模块驱动了。

modprobe 需要一个最新的 modules.dep 文件，这个文件是执行 depmod 后生成的，modprobe 会分析模块的依赖关系，它会查看模块目录/lib/modules/$(uname -r)里面的所有模块和文件，然后会将所有的依赖模块都加载到内核中。

8.1.6 depmod: ERROR: could not open directory /lib/modules/4.1.15: No such file or directory

或者

```
depmod: FATAL: could not search modules: No such file or directory
```

```
root@ATK-IMX6U:/lib/modules/4.1.15-g52f6b26# ls
8188eu.ko  ft5x06.ko    leddriver.ko       modules.builtin.bin  modules.order        source
8189fs.ko  icm20608.ko  modules.alias      modules.dep          modules.softdep
build      kernel       modules.alias.bin  modules.dep.bin      modules.symbols
dtsled.ko  key.ko       modules.builtin    modules.devname      modules.symbols.bin
root@ATK-IMX6U:/lib/modules/4.1.15-g52f6b26# depmod
depmod: ERROR: could not open directory /lib/modules/4.1.15: No such file or directory
depmod: FATAL: could not search modules: No such file or directory
root@ATK-IMX6U:/lib/modules/4.1.15-g52f6b26# uname -r
4.1.15
root@ATK-IMX6U:/lib/modules/4.1.15-g52f6b26#
```

图 8.1.6-1 报错信息

这种错误和上面（1）是一个情况，就是内核版本和模块版本不一致导致的，如上图，内核版本是 4.1.15，模块版本是 4.1.15-g52f6b26。那么我们先把模块版本和内核版本设置为一致。执行以下指令，将 4.1.15-g52f6b26 命名为 4.1.15,使用 depmod 则不再报错

```
cd ..
mv 4.1.15-g52f6b26/ $(uname -r)
```

```
root@ATK-IMX6U:/lib/modules/4.1.15-g52f6b26# uname -r
4.1.15
root@ATK-IMX6U:/lib/modules/4.1.15-g52f6b26#
root@ATK-IMX6U:/lib/modules/4.1.15-g52f6b26# cd ..
root@ATK-IMX6U:/lib/modules# mv 4.1.15-g52f6b26/ $(uname -r)
root@ATK-IMX6U:/lib/modules# ls
4.1.15
root@ATK-IMX6U:/lib/modules# cd 4.1.15/
root@ATK-IMX6U:/lib/modules/4.1.15# depmod
```

图 8.1.6-2 重命名

8.1.7 depmod:can't open 'modules.dep': Read-only file system

问题：nfs 挂载文件系统，执行 depmod 时出现错误。

```
depmod:can't open 'modules.dep': Read-only file system
```

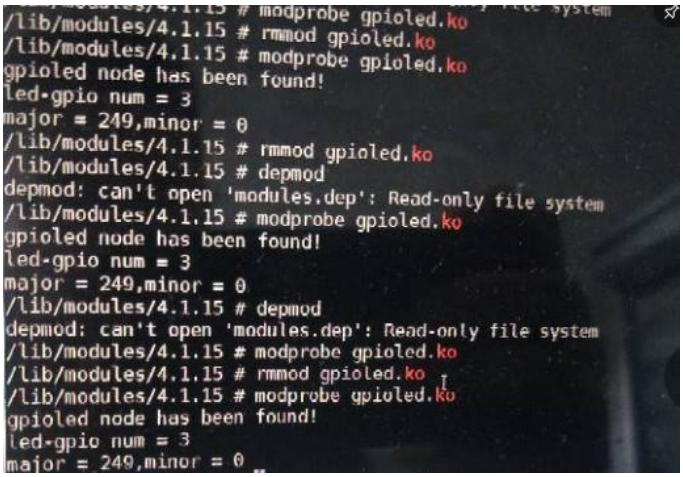


图 8.1.7-1 报错信息

解决：nfs 挂载时的 bootargs 没有添加`rw`可读写权限，要添加这个权限。或者 Ubuntu 下的 nfs 目录没有设置权限。

示例：

```
setenv bootargs 'console=(省略) nfsroot=192.168.1.165:/home/rootfs,proto=tcp rw ip=（省略）'
saveenv
```

8.1.8 -sh: depmod: not found

```
[root@alpha_imx6ull]:/lib/modules/4.1.15$:depmod
-sh: depmod: not found
```

图 8.1.8-1 报错信息

解决：重新按照教程配置 busybox。如果是 buildroot，则在 buildroot 中图形配置 busybox，如下图所示。

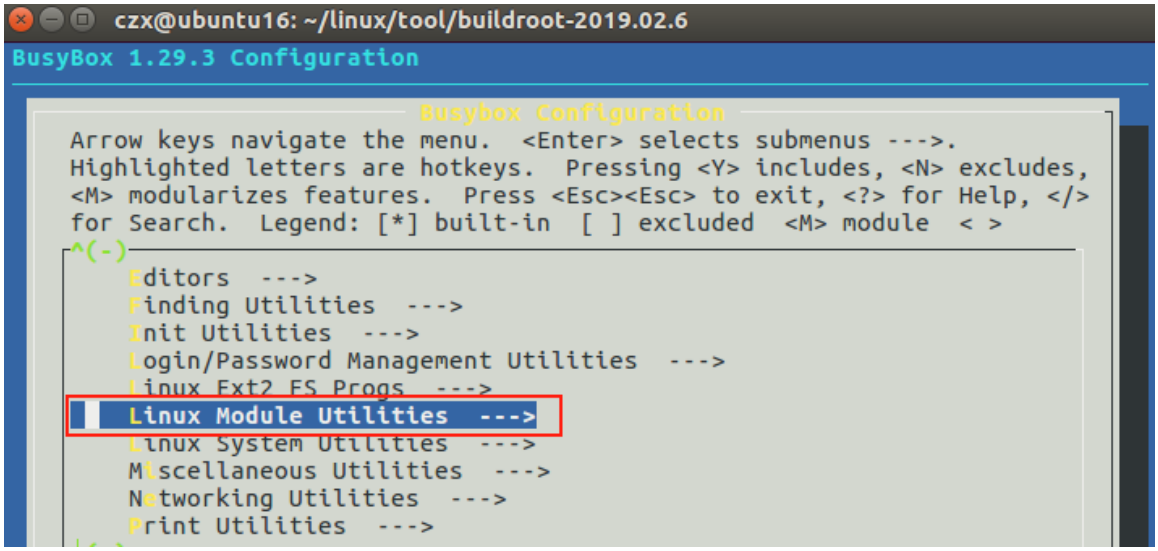


图 8.1.8-2 配置

选到 depmod。

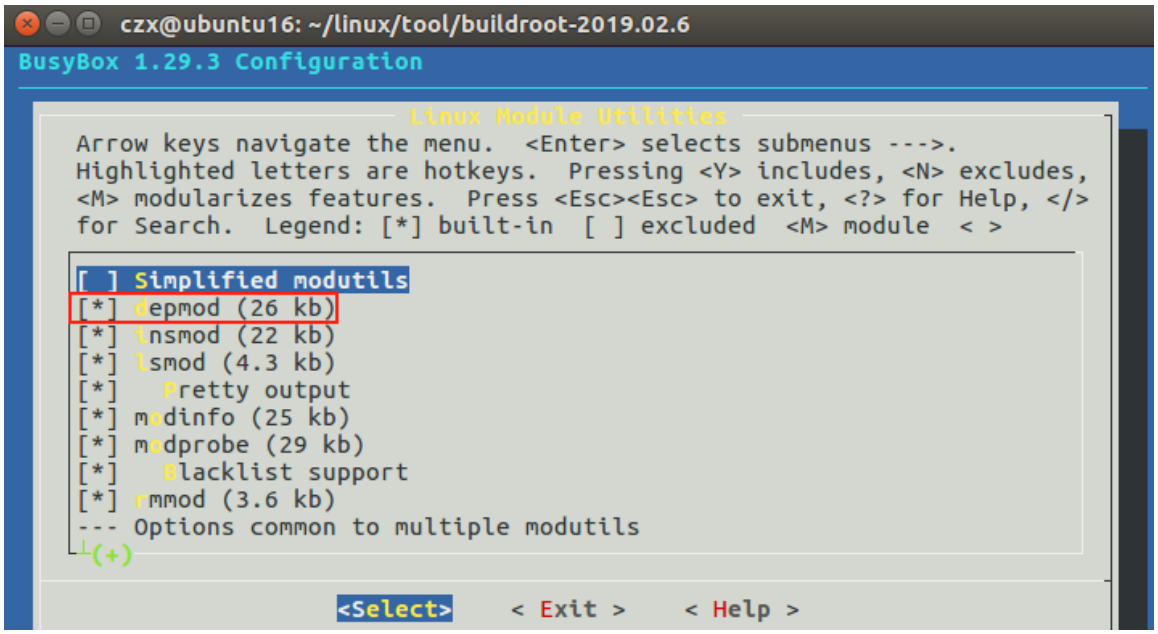


图 8.1.8-3 勾选 depmod

设置好后，执行 sudo make，编译生成 rootfs.tar，用这个系统启动开发板 Linux。

8.1.9 modprobe: FATAL: Module dtsled.ko not found in directory /lib/modules/4.1.15

```
root@ATK-IMX6U:/lib/modules/4.1.15# ls
8188eu.ko  ft5x06.ko    leddriver.ko       modules.builtin.bin  modules.order        source
8189fs.ko  icm20608.ko  modules.alias      modules.dep          modules.softdep
build      kernel       modules.alias.bin  modules.dep.bin      modules.symbols
dtsled.ko  key.ko       modules.builtin    modules.devname      modules.symbols.bin
root@ATK-IMX6U:/lib/modules/4.1.15# lsmod
Module                  Size  Used by
root@ATK-IMX6U:/lib/modules/4.1.15# depmod
root@ATK-IMX6U:/lib/modules/4.1.15# modprobe dtsled.ko
modprobe: FATAL: Module dtsled.ko not found in directory /lib/modules/4.1.15
root@ATK-IMX6U:/lib/modules/4.1.15# modprobe dtsled
alphaled node find!
compatible = atkalpha-led
status = okay
reg data:
0X20C406C 0X4 0X20E0068 0X4 0X20E02F4 0X4 0X209C000 0X4 0X209C004 0X4
dtsled major=249,minor=0
root@ATK-IMX6U:/lib/modules/4.1.15# lsmod
Module                  Size  Used by
dtsled                  2238  0
root@ATK-IMX6U:/lib/modules/4.1.15# rmmod dtsled.ko
root@ATK-IMX6U:/lib/modules/4.1.15# lsmod
Module                  Size  Used by
root@ATK-IMX6U:/lib/modules/4.1.15# insmod dtsled
insmod: ERROR: could not load module dtsled: No such file or directory
root@ATK-IMX6U:/lib/modules/4.1.15# insmod dtsled.ko
alphaled node find!
compatible = atkalpha-led
status = okay
reg data:
0X20C406C 0X4 0X20E0068 0X4 0X20E02F4 0X4 0X209C000 0X4 0X209C004 0X4
dtsled major=249,minor=0
root@ATK-IMX6U:/lib/modules/4.1.15#
root@ATK-IMX6U:/lib/modules/4.1.15# modprobe -r dtsled.ko
modprobe: FATAL: Module dtsled.ko not found.
root@ATK-IMX6U:/lib/modules/4.1.15# modprobe -r dtsled
root@ATK-IMX6U:/lib/modules/4.1.15#
```

图 8.1.9-1 报错信息

可以查看上面的操作过程，使用 modprobe 来加载驱动模块的时候报错找不到 dtsled.ko 这个文件，但是 dtsled.ko 这个文件在/lib/modules/4.1.15 下是有的，当写为 modprobe dtsled 以后模块加载成功。用 insmod dtsled.ko 加载成功，用 insmod dtsled 加载不成功。为什么会这样？什么时候要加.ko 什么时候不加.ko 呢？其实这个是不同文件系统下 modprobe 的用法，有的文件系统下可以加.ko，有的文件系统下就不加.ko，那么在测试的时候，报错找不到模块的话，就多试试加与不加.ko。

如下图是正点原子出厂的文件系统，查看 modprobe 的用法：

```
modprobe [options] [-i] [-b] modulename
```

这里说明命令 modprobe 后加的是模块的名字，dtsled.ko 这个中的 dtsled 就是模块的名字

```
root@ATK-IMX6U:/lib/modules/4.1.15# modprobe -h
Usage:
        modprobe [options] [-i] [-b] modulename
        modprobe [options] -a [-i] [-b] modulename [modulename...]
        modprobe [options] -r [-i] modulename
        modprobe [options] -r -a [-i] modulename [modulename...]
        modprobe [options] -c
        modprobe [options] --dump-modversions filename
Management Options:
        -a, --all                   Consider every non-argument to
                                    be a module name to be inserted
                                    or removed (-r)
        -r, --remove                Remove modules instead of inserting
            --remove-dependencies   Also remove modules depending on it
        -R, --resolve-alias         Only lookup and print alias and exit
            --first-time            Fail if module already inserted or removed
        -i, --ignore-install        Ignore install commands
        -i, --ignore-remove         Ignore remove commands
        -b, --use-blacklist         Apply blacklist to resolved alias.
        -f, --force                 Force module insertion or removal.
                                    Implies --force-modversions and
                                      --force-vermagic
```

图 8.1.9-2 modprobe -h

如下图是用 busybox 编译的文件系统，查看 modprobe 的用法：

Usage: modprobe [-alrqvsDb] MODULE [SYMBOL=VALUE]...

这里可以加后缀.ko 或者不加。具体信息可以上 busybox 官网查询。

```
/lib/modules/4.1.15 # modprobe -h
modprobe: invalid option -- 'h'
BusyBox v1.29.0 (2019-09-24 19:00:38 CST) multi-call binary.

Usage: modprobe [-alrqvsDb] MODULE [SYMBOL=VALUE]...

        -a      Load multiple MODULEs
        -l      List (MODULE is a pattern)
        -r      Remove MODULE (stacks) or do autoclean
        -q      Quiet
        -v      Verbose
        -s      Log to syslog
        -D      Show dependencies
        -b      Apply blacklist to module names too
/lib/modules/4.1.15 #
```

图 8.1.9-3 modprobe -h

用 insmod 加载模块的话，后面要加模块的路径，如果模块.ko 文件在当前目录下的话，当前路径的./可以省略不写。

```
root@ATK-IMX6U:~#
root@ATK-IMX6U:~# insmod dtsled.ko
insmod: ERROR: could not load module dtsled.ko: No such file or directory
root@ATK-IMX6U:~# insmod /lib/modules/4.1.15/dtsled.ko
alphaled node find!
compatible = atkalpha-led
status = okay
reg data:
0X20C406C 0X4 0X20E0068 0X4 0X20E02F4 0X4 0X209C000 0X4 0X209C004 0X4
dtsled major=249,minor=0
root@ATK-IMX6U:~# lsmod
Module                  Size  Used by
dtsled                  2238  0
root@ATK-IMX6U:~#
root@ATK-IMX6U:~# rmmod dtsled.ko
root@ATK-IMX6U:~# lsmod
Module                  Size  Used by
root@ATK-IMX6U:~#
```

图 8.1.9-4 insmod 加载

8.1.10 modprobe: ERROR: could not insert 'gpioled': Exec format error

已经执行 depmod 了，modprob gpioled 报错：

```
gpioled: no symbol version for module_layout
modprobe: ERROR: could not insert 'gpioled': Exec format error
```

```
root@ALIENTEK-IMX6U:/lib/modules/4.1.15# modprobe gpioled.ko
modprobe: FATAL: Module gpioled.ko not found in directory /lib/modules/4.1.15
root@ALIENTEK-IMX6U:/lib/modules/4.1.15# modprobe gpioled
gpioled: no symbol version for module_layout
modprobe: ERROR: could not insert 'gpioled': Exec format error
root@ALIENTEK-IMX6U:/lib/modules/4.1.15# chmod 777 gpioled.ko
root@ALIENTEK-IMX6U:/lib/modules/4.1.15# modprobe gpioled
gpioled: no symbol version for module_layout
modprobe: ERROR: could not insert 'gpioled': Exec format error
root@ALIENTEK-IMX6U:/lib/modules/4.1.15# ls
build          gpioled.ko   ledApp             modules.builtin.bin  modules.order        source
chrdevbase.ko  imx6uirq.ko  modules.alias      modules.dep          modules.softdep
chrdevbaseApp  imx6uirqApp  modules.alias.bin  modules.dep.bin      modules.symbols
dtsled.ko      kernel       modules.builtin    modules.devname      modules.symbols.bin
root@ALIENTEK-IMX6U:/lib/modules/4.1.15# modprobe gpioled
gpioled node find!
led-gpio num = 3
gpioled major=249,minor=0
root@ALIENTEK-IMX6U:/lib/modules/4.1.15# #
```

图 8.1.10-1 报错信息

最可能的问题是，拷贝的 gpioled.ko 文件有问题，重新拷贝一次就好了。

8.1.11 lsmod: can't open '/proc/modules'

按照教程制作的根文件系统 busybox，运行驱动失败。

```
/lib/modules/4.1.15 # depmod
/lib/modules/4.1.15 # modprobe chrdevbase.ko
modprobe: module 'chrdevbase.ko' not found
/lib/modules/4.1.15 # ls
chrdevbase.ko   chrdevbaseApp   modules.dep.bb
/lib/modules/4.1.15 # depmod
/lib/modules/4.1.15 # ls
chrdevbase.ko   chrdevbaseApp   modules.dep.bb
/lib/modules/4.1.15 # insmod chrdevbase.ko
chrdevbase init!
/lib/modules/4.1.15 # lsmod
lsmod: can't open '/proc/modules': No such file or directory
```

图 8.1.11-1 报错信息

原因：根文件系统没有完善，缺少/etc/init.d/rcS 文件、/etc/fstab 文件、/etc/inittab 文件

解决：按照教程 38.4.2 创建/etc/fstab 文件

注意：是修改文件系统 rootfs 的/etc/fstab 文件，不要修改到 Ubuntu 的/etc/fstab 文件，否则 Ubuntu 会崩溃，启动不了。

8.2 实验外设问题

8.2.1 使用教程设备树驱动实验点灯失败

使用原子教程做好的设备树，需要把心跳灯节点屏蔽掉。

具体操作：打开文件 arch/arm/boot/dts/imx6ull-alientek-emmc.dts，找到 dtsleds 节点。

```
/* zuozhongkai dts leds linux自带led驱动的设备树*/
dtsleds {
    compatible = "gpio-leds";

    led0 {
        label = "red";
        gpios = <&gpio1 3 GPIO_ACTIVE_LOW>;
        linux,default-trigger = "heartbeat";
        default-state = "on";
    };
```

图 8.2.1-1 dtsled 节点

将此节点修改为如下，修改 default-state 值为 off，添加 status 值为 disable

```
/* zuozhongkai dts leds linux自带led驱动的设备树*/
dtsleds {
    compatible = "gpio-leds";

    led0 {
        label = "red";
        gpios = <&gpio1 3 GPIO_ACTIVE_LOW>;
        linux,default-trigger = "heartbeat";
        default-state = "off";
        status = "disable";
    };
```

图 8.2.1-2 修改好的节点

保存修改好的文件，重新编译设备树放到板子上运行。

8.2.2 pinctrl 和 和 gpio 实验 gpioled.ko 灯无法点亮

方法：检查设备树，设备树部分将如下两句注释掉，再重新编译设备树

```
pinctrl_enet1: enet1grp {
        fsl,pins = <
                MX6UL_PAD_ENET1_RX_EN__ENET1_RX_EN      0x1b0b0
                MX6UL_PAD_ENET1_RX_ER__ENET1_RX_ER      0x1b0b0
                MX6UL_PAD_ENET1_RX_DATA0__ENET1_RDATA00 0x1b0b0
                MX6UL_PAD_ENET1_RX_DATA1__ENET1_RDATA01 0x1b0b0
                MX6UL_PAD_ENET1_TX_EN__ENET1_TX_EN      0x1b0b0
                MX6UL_PAD_ENET1_TX_DATA0__ENET1_TDATA00 0x1b0b0
                MX6UL_PAD_ENET1_TX_DATA1__ENET1_TDATA01 0x1b0b0
                MX6UL_PAD_ENET1_TX_CLK__ENET1_REF_CLK1  0x4001b031
        /*      MX6ULL_PAD_SNVS_TAMPER7__GPIO5_IO07 0X10B0*/
                MX6UL_PAD_SNVS_TAMPER7__GPIO5_IO07      0X10B0
        >;
};

pinctrl_enet2: enet2grp {
        fsl,pins = <
                MX6UL_PAD_GPIO1_IO07__ENET2_MDC         0x1b0b0
                MX6UL_PAD_GPIO1_IO06__ENET2_MDIO        0x1b0b0
                MX6UL_PAD_ENET2_RX_EN__ENET2_RX_EN      0x1b0b0
                MX6UL_PAD_ENET2_RX_ER__ENET2_RX_ER      0x1b0b0
                MX6UL_PAD_ENET2_RX_DATA0__ENET2_RDATA00 0x1b0b0
                MX6UL_PAD_ENET2_RX_DATA1__ENET2_RDATA01 0x1b0b0
                MX6UL_PAD_ENET2_TX_EN__ENET2_TX_EN      0x1b0b0
                MX6UL_PAD_ENET2_TX_DATA0__ENET2_TDATA00 0x1b0b0
                MX6UL_PAD_ENET2_TX_DATA1__ENET2_TDATA01 0x1b0b0
                MX6UL_PAD_ENET2_TX_CLK__ENET2_REF_CLK2  0x4001b031
        /*      MX6ULL_PAD_SNVS_TAMPER8__GPIO5_IO08     0X10B0*/
                MX6UL_PAD_SNVS_TAMPER8__GPIO5_IO08      0X10B0
        >;
};

pinctrl_flexcan1: flexcan1grp{
```

图 8.2.2-1 屏蔽掉复用管脚

8.2.3 驱动实验 IO 口占用问题

①操作中断那一章报错

```
root@ALIENTEK-IMX6U:/lib/modules/4.1.15# insmod imx6uirq.ko
key0:gpio=18, irqnum=50
```

genirq: Flags mismatch irq 50. 00000003 (KEY0) vs. 00000083 (GPIO Key Enter)
irq 50 request failed!

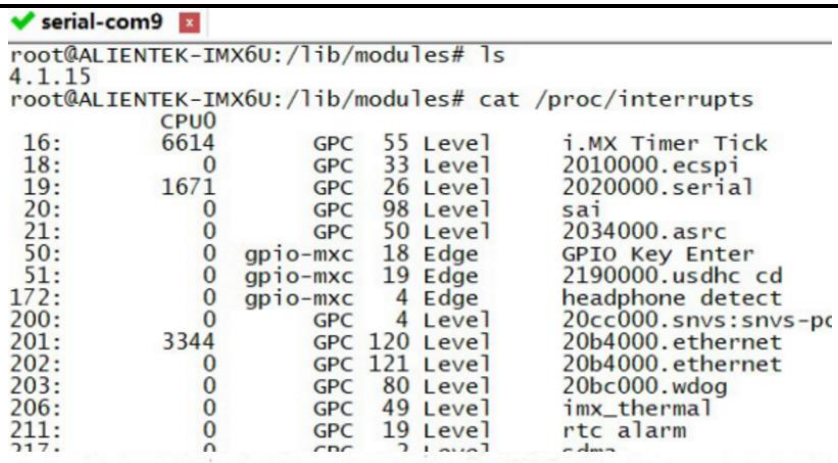


图 8.2.3-1 报错信息

如上图，这里有个 50 占用了，检查设备树下有哪个地方配置了 label 为 GPIO Key Enter 的地方，找到后注释掉：

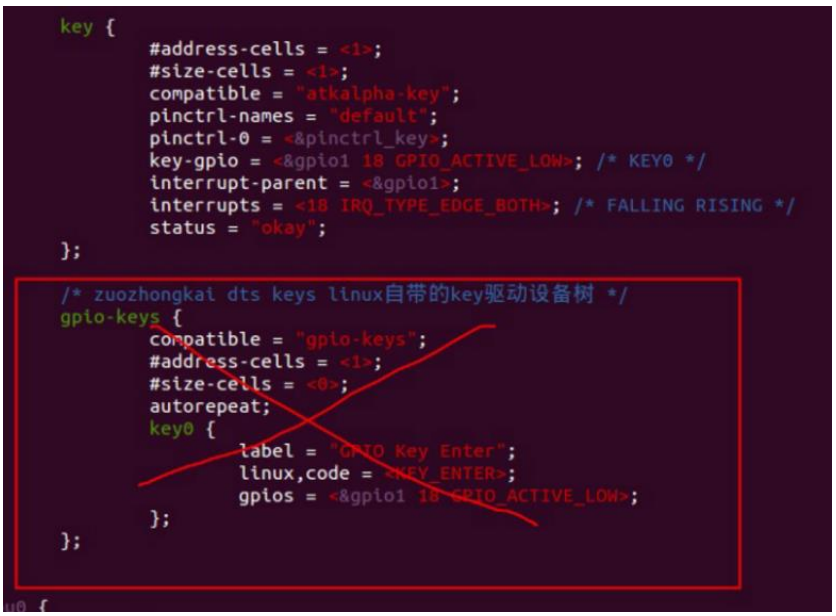


图 8.2.3-2 注释掉被占用的节点

8.2.4 SPI 驱动实验

加载完驱动后提示信息：

ICM20608 ID=0X0

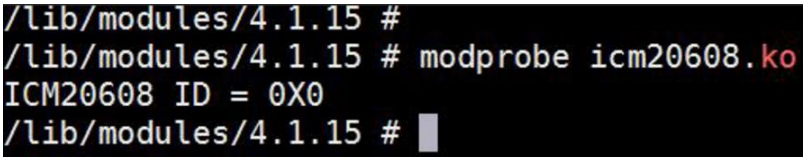


图 8.2.4-1 提示信息

```
/lib/modules/4.1.15 # ./icm20608App /dev/icm20608

原始值:
gx = 0, gy = 0, gz = 0
ax = 0, ay = 0, az = 0
temp = 0
实际值:act gx = 0.00? S, act gy = 0.00? S, act gz = 0.0
act ax = 0.00g, act ay = 0.00g, act az = 0.00g
act temp = 24.92癈

原始值:
gx = 0, gy = 0, gz = 0
ax = 0, ay = 0, az = 0
temp = 0
实际值:act gx = 0.00? S, act gy = 0.00? S, act gz = 0.0
act ax = 0.00g, act ay = 0.00g, act az = 0.00g
act temp = 24.92癈
```

图 8.2.4-2 传感器值为 0

解决：

SPI 的 IO 口和 UART2 的 IO 口复用，将 UART2 相关的注释掉，再重新编译设备树就可以。

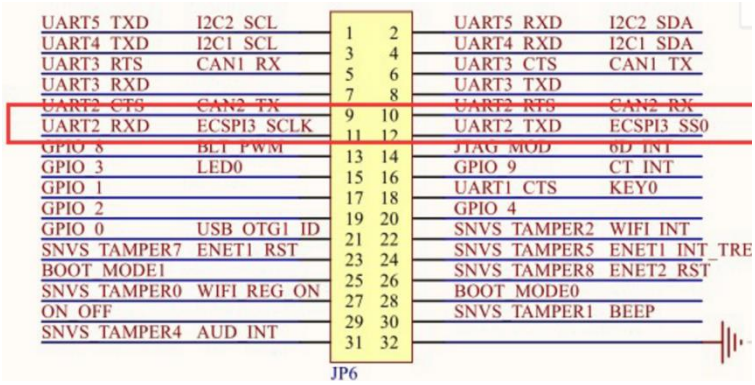


图 8.2.4-3 核心板原理图

```
pinctrl_uart1: uart1grp {
        fsl,pins = <
                MX6UL_PAD_UART1_TX_DATA__UART1_DCE_TX 0x1b0b1
                MX6UL_PAD_UART1_RX_DATA__UART1_DCE_RX 0x1b0b1
        >;
};

pinctrl_uart2: uart2grp {
        fsl,pins = <
        /*      MX6UL_PAD_UART2_TX_DATA__UART2_DCE_TX   0x1b0b1
                MX6UL_PAD_UART2_RX_DATA__UART2_DCE_RX   0x1b0b1*/
                MX6UL_PAD_UART3_RX_DATA__UART2_DCE_RTS  0x1b0b1
                MX6UL_PAD_UART3_TX_DATA__UART2_DCE_CTS  0x1b0b1
        >;
};

pinctrl_uart2dte: uart2dtegrp {
        fsl,pins = <
                MX6UL_PAD_UART2_TX_DATA__UART2_DTE_RX   0x1b0b1
                MX6UL_PAD_UART2_RX_DATA__UART2_DTE_TX   0x1b0b1
```

图 8.2.4-4 注释掉复用功能

8.2.5 音频报错 Playing WAVE 'test.wav' : Signed 16 bit Little Endian, Rate 44100 Hz, Stereo

报错信息：

```
Playing WAVE 'test.wav' : Signed 16 bit Little Endian, Rate 44100 Hz, Stereo
random: nonblocking pool is initialized
aplay: pcm_write:2061: write error: Input/output error
```

```
/music # aplay test.wav
Playing WAVE 'test.wav' : Signed 16 bit Little Endian, Rate 44100 Hz, Stereo
random: nonblocking pool is initialized
aplay: pcm_write:2061: write error: Input/output error
```

图 8.2.5-1 报错信息

原因是没有执行下面的指令：

1、使用 amixer 设置声卡

第一次使用声卡之前一定要先使用 amixer 设置声卡，打开耳机和喇叭，并且设置喇叭和机音量，输入如下命令：

```
amixer sset Headphone 100,100
amixer sset Speaker 120,120
amixer sset 'Right Output Mixer PCM' on
amixer sset 'Left Output Mixer PCM' on
```

2、使用 aplay 播放 WAV 格式音乐

8.2.6 音频报错 alsactl: state_lock:125: file /var/lib/alsa/asound.state lock error: No such file or directory

```
alsabat          alsactl          alsatplg
alsabat-test.sh  alsaloop         alsaucm
/ # alsactl -f /var/lib/alsa/asound.state store
alsactl: state_lock:125: file /var/lib/alsa/asound.state lock error: No such file or director
/ #
/ # alsactl -f /var/lib/alsa/asound.state store
alsactl: state_lock:125: file /var/lib/alsa/asound.state lock error: No such file or director
/ # cd /var/lib/alsa/
/var/lib/alsa # ls
/var/lib/alsa # vim asound.state
-/bin/sh: vim: not found
/var/lib/alsa # vi asound.state
/var/lib/alsa # cd
/ #
/ #
/ #
/ # alsactl -f /var/lib/alsa/asound.state store
alsactl: state_lock:125: file /var/lib/alsa/asound.state lock error: No such file or director
/ # alsactl -f /var/lib/alsa/asound.state
alsactl: Specify command...
/ # alsactl -f /var/lib/alsa/asound.state store
alsactl: state_lock:125: file /var/lib/alsa/asound.state lock error: No such file or director
/ #
```

图 8.2.6-1 报错信息

解决办法：在根文件系统 etc/profile 下添加这句话

```
vi etc/profile
export ALSA_CONFIG_PATH=/usr/share/arm-alsa/alsa.conf
```

```
#!/bin/sh

LD_LIBRARY_PATH=/lib:/usr/lib:$LD_LIBRARY_PATH
export LD_LIBRARY_PATH

export TERM=vt100
export TERMINFO=/usr/share/terminfo
export ALSA_CONFIG_PATH=/usr/share/arm-alsa/alsa.conf
```

图 8.2.6-2 修改 etc/profile

然后开发板重启，手动在/var/lib/alsa 新建文件 asound.state，然后再执行指令。

```
alsactl -f asound.state store
```

就可以将配置的信息保存了。

```
/var/lib/alsa # vi asound.state
/var/lib/alsa # alsactl -f asound.state
alsactl: Specify command...
/var/lib/alsa # alsactl -f asound.state store
/var/lib/alsa # ls
asound.state
/var/lib/alsa # vi asound.state
/var/lib/alsa #
```

图 8.2.6-3 测试结果

8.2.7 音频实验报错/bin/amixer:line 2:syntax error:unexpected")"

执行音频实验移植的 amixer 报错：

```
/bin/amixer:line 2:syntax error:unexpected")"
/#/bin/amixer:line 1:ELF:not found
```

```
/ # source /etc/profile
/ # ls
?           dev         hello.c     mnt         sbin
bin         etc         lib         proc        sys
brightness  hello       linuxrc     root        tmp
/ # amixer --help
/bin/amixer: line 2: syntax error: unexpected ")"
/ # /bin/amixer: line 1: ELF: not found

/ # source /etc/profile
/ # amixer --help
/bin/amixer: line 2: syntax error: unexpected ")"
/ # /bin/amixer: line 1: ELF: not found
```

图 8.2.7-1 报错信息

这种报错很可能是因为编译 amixer 的时候用了 ubuntu 自带的 X86 架构的编译器来编译的，所以编译后出来的 amixer 是 X86 架构的不是 ARM 架构的，所以放到 ARM 架构的开发板后运行就报错了。可以用 file amixer 指令在 ubuntu 上查看这个 amixer 的格式：

```
lmq@lmq-virtual-machine:~/linux/IMX6ULL/tool/alsa-utils/bin$ ls
aconnect  alsaloop  alsaucm  amixer  aplaymidi  arecordmidi  aseqnet  iecset
alsabat   alsatplg  amidi   aplay   arecord    aseqdump     axfer    speaker-test
lmq@lmq-virtual-machine:~/linux/IMX6ULL/tool/alsa-utils/bin$ file amixer
amixer: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/l, for GNU/Linux 2.6.32,
BuildID[sha1]=2a2e69effbc5882719e4e257 54843bd1a3d664a, not stripped
lmq@lmq-virtual-machine:~/linux/IMX6UL tool/alsa-utils/bin$
向到该虚拟机，请将鼠标指针移入其中或按 Ctrl+G。
```

清理工程了重新配置和编译，在编译前先执行 source /etc/profile 以后再编译，编译指令也可以将 make 换成

```
make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf-
```

8.2.8 音频实验报错 alsactl: state_lock:125: file /var/lib/alsa/asound.state lock error: No such file or directory

执行 alsactl -f /var/lib/alsa/asound.state store 报错：

```
alsactl: state_lock:125: file /var/lib/alsa/asound.state lock error: No such file or directory
```

```
alsabat          alsactl          alsatplg
alsabat-test.sh  alsaloop         alsaucm
/ # alsactl -f /var/lib/alsa/asound.state store
alsactl: state_lock:125: file /var/lib/alsa/asound.state lock error: No such file or directory
/ #
/ # alsactl -f /var/lib/alsa/asound.state store
alsactl: state_lock:125: file /var/lib/alsa/asound.state lock error: No such file or directory
/ # cd /var/lib/alsa/
/var/lib/alsa # ls
/var/lib/alsa # vim asound.state
-/bin/sh: vim: not found
/var/lib/alsa # vi asound.state
/var/lib/alsa # cd
/ #
/ #
/ #
/ # alsactl -f /var/lib/alsa/asound.state store
alsactl: state_lock:125: file /var/lib/alsa/asound.state lock error: No such file or directory
/ # alsactl -f /var/lib/alsa/asound.state
alsactl: Specify command...
/ # alsactl -f /var/lib/alsa/asound.state store
alsactl: state_lock:125: file /var/lib/alsa/asound.state lock error: No such file or directory
/ #
```

图 8.2.8-1 报错信息

可以尝试在 etc/profile 下添加这句话

vi etc/profile

export ALSA_CONFIG_PATH=/usr/share/arm-alsa/alsa.conf

```
#!/bin/sh

LD_LIBRARY_PATH=/lib:/usr/lib:$LD_LIBRARY_PATH
export LD_LIBRARY_PATH

export TERM=vt100
export TERMINFO=/usr/share/terminfo
export ALSA_CONFIG_PATH=/usr/share/arm-alsa/alsa.conf
~
```

图 8.2.8-2 添加 ALSA 内容

然后开发板重启，手动在/var/lib/alsa 新建文件 asound.state，然后再执行指令就可以将配置的信息保存了：

alsactl -f asound.state store

如果还是有错的话，可以尝试执行

alsactl -f var/lib/alsa/asound.state store

以上指令就是把 alsactl -f /var/lib/alsa/asound.state store 中的/var/改为 var/，即去掉了/

```
/var/lib/alsa # vi asound.state
/var/lib/alsa # alsactl -f asound.state
alsactl: Specify command...
/var/lib/alsa # alsactl -f asound.state store
/var/lib/alsa # ls
asound.state
/var/lib/alsa # vi asound.state
/var/lib/alsa #
```

图 8.2.8-3 保存修改好的配置

8.2.9 音频测试报错 Playing WAVE 'test.wav' : Signed 16 bit Little Endian, Rate 44100 Hz, Stereo

执行 aplay test.wav 后报错：

```
Playing WAVE 'test.wav' : Signed 16 bit Little Endian, Rate 44100 Hz, Stereo
```

random: nonblocking pool is initialized

aplay: pcm_write:2061: write error: Input/output error

```
/music # aplay test.wav
Playing WAVE 'test.wav' : Signed 16 bit Little Endian, Rate 44100 Hz, Stereo
random: nonblocking pool is initialized
aplay: pcm_write:2061: write error: Input/output error
```

图 8.2.9-1 报错信息

上面报错的原因是没有执行下面的指令：

1、使用 amixer 设置声卡

第一次使用声卡之前一定要先使用 amixer 设置声卡，打开耳机和喇叭，并且设置喇叭和耳机音量，输入如下命令：

```
amixer sset Headphone 100,100
amixer sset Speaker 120,120
amixer sset 'Right Output Mixer PCM' on
amixer sset 'Left Output Mixer PCM' on
```

图 8.2.9-2 教程设置声卡

8.2.10 WIFI 实验使用 udhcpc 能获取到 DHCP 自动分配的 IP，但是该 IP 却没有设置到网卡上

没有打印 Setting IP address 192.168.1.126 on wlan0

```
          RX bytes:525258 (512.9 KiB)  TX bytes:1827 (1.7 KiB)

/ # udhcpc -i wlan0
udhcpc: started, v1.29.0
udhcpc: sending discover
udhcpc: sending select for 192.168.1.57
udhcpc: lease of 192.168.1.57 obtained, lease time 86400
/ # ifconfig wlan0
wlan0     Link encap:Ethernet  HWaddr 00:13:EF:F1:12:DF
          inet6 addr: fe80::213:efff:fef1:12df/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:6047 errors:0 dropped:329 overruns:0 frame:0
          TX packets:13 errors:0 dropped:3 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:585219 (571.5 KiB)  TX bytes:2567 (2.5 KiB)

/ #
```

图 8.2.10-1 没有 IP

解决办法：

在开发板文件系统的/usr/share/下新建一个文件夹 udhcpc

```
mkdir /usr/share/udhcpc
```

```
/usr/share # ls
man         tabset      terminfo
/usr/share # mkdir udhcpc
/usr/share # ls
man         tabset      terminfo    udhcpc
/usr/share # ls udhcpc/
```

图 8.2.10-2 新建 udhcpc 文件夹

在当初编译 busybox 文件系统的包中找到 busybox-1.29.0/examples/udhcp/simple.script 文件，并将这个 simple.script 文件拷贝到开发板文件系统的 udhcpc 目录下：

```
MY@IMX6ULL:~/nfs/busybox-1.29.0/examples/udhcp$ ls
sample.bound  sample.deconfig  sample.nak  sample.renew  sample.script  simple.script  udhcpd.conf
MY@IMX6ULL:~/nfs/busybox-1.29.0/examples/udhcp$ sudo cp simple.script /home/MY/NFS/myrootfs/usr/share/udhcpc
[sudo] password for MY:
MY@IMX6ULL:~/nfs/busybox-1.29.0/examples/udhcp$ pwd
/home/MY/nfs/busybox-1.29.0/examples/udhcp
```

图 8.2.10-3 拷贝

PS:我的文件系统在 NFS 的目录是/home/MY/NFS/myrootfs，开发板挂载的是 NFS 的文件系统目录目录，拷贝成功后将开发板下 simple.script 文件改名字为 default.script

```
/usr/share #
/usr/share # ls
man         tabset      terminfo
/usr/share # mkdir udhcpc
/usr/share # ls
man         tabset      terminfo    udhcpc
/usr/share # ls udhcpc/
/usr/share # ls
man         tabset      terminfo    udhcpc
/usr/share # cd udhcpc/
/usr/share/udhcpc # ls
simple.script
/usr/share/udhcpc # mv simple.script default.script
/usr/share/udhcpc # ls
default.script
/usr/share/udhcpc # pwd
/usr/share/udhcpc
/usr/share/udhcpc #
```

图 8.2.10-4 修改文件名

最后开发板启动再加载模块再进行一系列的设置，IP 或者成功。

8.2.11 USB WIFI 开启 AP 热点模式

正点原子最新出厂系统上支持 USB WIFI 开启热点模式，有一个 WIFI 脚本。我们打开出厂系统的 WiFi 脚本。

```
    echo -ne "interface=wlan0\nssid=alientek_softap\ndriver=rtl871xdrv\nchannel=6
auth_algs=1\nwpa=3\nwpa_passphrase=12345678\nwpa_key_mgmt=WPA-PSK\nwpa_pairwise=T
        a=$(ifconfig |grep -E "br0" | grep -v grep | awk '{print $0}')
        if [ -n "$a" ];then    .0
                brctl delif br0 $device
                brctl delif br0 $device
                ifconfig br0 down
        fi                                                  2}')
        rm -rf /var/lib/misc/*
        touch /var/lib/misc/udhcpd.leases
        ifconfig $device up
        sleep 2
        ifconfig $device 192.168.1.38 netmask 255.255.255.0
        udhcpd -fS /etc/udhcpd.conf     &
        hostapd $wifi_conf -B
        export WIFI_MODE=softap
fi

#station Mode（上网模式）up
```

图 8.2.11-1 出厂系统 wifi 脚本

可以看到脚本中使用了 hostapd 这个工具。我们将件系统目录下编译好的 /usr/bin/hostapd 拷贝到自己移植文件系统里，参考脚本里的写法就可以使用 USB WIFI 的 AP 热点模式。

8.2.12 ts_calibrate 报错 /bin/ts_calibrate: line 1: syntax error: unexpected word

/bin/ts_calibrate: line 1: syntax error: unexpected word (expecting ")")

```
/etc # ts_calibrate
/bin/ts_calibrate: line 1: syntax error: unexpected word (expecting ")")
/etc #
```

图 8.2.12-1 报错信息

解答：问题在于编译的时候使用的交叉编译器不对了，看可执行文件，文件是 x86 架构的，明显不对。在 ubuntu 上使能交叉编译器了再进行配置和编译。

执行 file ts_calibrate 查看这个可执行文件是 x86 架构的，不是 ARM 架构的，所以再开发板上运行报错。

```
MY@IMX6ULL:~/program/ts/tslib-1.21$ cd ../tslib
MY@IMX6ULL:~/program/ts/tslib$ ls
bin  etc  include  lib  share
MY@IMX6ULL:~/program/ts/tslib$ cd bin/
MY@IMX6ULL:~/program/ts/tslib/bin$ ls
ts_calibrate  ts_conf  ts_finddev  ts_harvest  ts_print  ts_print_mt  ts_print_raw  ts_test  ts_test_mt  ts_uinput  ts_verify
MY@IMX6ULL:~/program/ts/tslib/bin$ file ts_calibrate
ts_calibrate: ELF 64-bit LSB  executable, x86-64, version 1 (SYSV), dynamically linked (uses shared libs), for GNU/Linux 2.6.24, BuildID[
597b24f80607644be7e03308a8ef5241c6a, not stripped
MY@IMX6ULL:~/program/ts/tslib/bin$
```

图 8.2.12-2 查看文件架构

重新编译后，可以看出是 ARM 的了，重新拷贝到开发板上运行：

```
make[1]: Leaving directory `/home/MY/program/ts/tslib-1.21'
MY@IMX6ULL:~/program/ts/tslib-1.21$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/usr/local/arm/gcc-linaro-4.9.4-2017.01-x86_64_ar
ueabihf/bin
MY@IMX6ULL:~/program/ts/tslib-1.21$ file ../tslib/bin/ts_calibrate
../tslib/bin/ts_calibrate: ELF 32-bit LSB  executable, ARM, EABI5 version 1 (SYSV), dynamically linked (uses shared libs), for GNU/Linux 2
ldID[sha1]=ec862270d072e9393994cf86a4d2c38ba07504de, not stripped
MY@IMX6ULL:~/program/ts/tslib-1.21$
```

图 8.2.12-3 编译成 ARM 架构

PS：校准后，校准的数据会保留在/etc/pointercal 下。

```
/etc #
/etc # ts_calibrate
xres = 800, yres = 480
Took 1 samples...
Top left : X =   49 Y =   31
Took 1 samples...
Top right : X =  746 Y =   36
Took 1 samples...
Bot right : X =  741 Y =  422
Took 1 samples...
Bot left : X =   58 Y =  409
Took 1 samples...
Center : X =  395 Y =  230
-2.412537 1.014432 -0.005003
20.752930 -0.012880 0.994550
Calibration constants: -158108 66481 -327 1360064 -844 65178 65536
/etc # cat pointercal
```

图 8.2.12-4 保存数据

8.2.13 ts_setup: No such file or directory

目前遇见这个问题的情况有：拷贝文件的时候导致文件格式错误，重新拷贝就好了（推荐换一种拷贝方法试试），又或者是环境变量写错了地方或者环境变量配置不对，系统找不到环境变量，请检查/etc/profile 环境变量的配置，例如文件系统的/etc/profile 文件误写成/etc/provile 以后导致系统找不到环境变量。

8.2.14 触摸屏实验报错 tslib: Selected device is not a touchscreen (must support ABS event type)

```
lib: Selected device is not a touchscreen (must support ABS event
lib: Selected device is not a touchscreen (must support ABS event
lib: Selected device is not a touchscreen (must support ABS event
lib: Selected device is not a touchscreen (must support ABS event
lib: Selected device is not a touchscreen (must support ABS event
lib: Selected device is not a touchscreen (must support ABS event
lib: Selected device is not a touchscreen (must support ABS event
lib: Selected device is not a touchscreen (must support ABS event
lib: Selected device is not a touchscreen (must support ABS event
lib: Selected device is not a touchscreen (must support ABS event
lib: Selected device is not a touchscreen (must support ABS event
lib: Selected device is not a touchscreen (must support ABS event
lib: Selected device is not a touchscreen (must support ABS event
lib: Selected device is not a touchscreen (must support ABS event
lib: Selected device is not a touchscreen (must support ABS event
lib: Selected device is not a touchscreen (must support ABS event
lib: Selected device is not a touchscreen (must support ABS event
```

图 8.2.14-1 报错信息

出现此问题是因为没有成功移植触摸驱动，重新检查看是哪里漏了步骤，或者/etc/profile 下的环境变量没有设置成功，例如 TSLIB_TSDEVICE=/dev/input/eventX 的 X 写的不对的也会出现这个情况，这个 X 是 0 还是 1 或者是 2 要根据自己的实际情况来确定，例如，如果想知道是不是 2，可以执行如下指令，执行以后便用手指触摸屏幕，查看屏幕是否有上报数据：

```
hexdump /dev/input/event2
```

驱动加载成功以后就会生成/dev/input/eventX(X=1,2,3…)，比如本实验的多点电容触摸驱动就会在我所使用的 ALPHA 开发板平台下就会生成/dev/input/event2 这个文件，如图 64.4.2.2 所示：

```
/dev/input # ls
event0  event1  event2  js0     mice
/dev/input #
```

电容触摸屏

图 64.4.2.2 电容屏对应的 event 设备

不同的平台 event 序号不同，也可能是 event3，event4 等，一切以实际情况为准！输入如下命令查看 event2，也就是多点电容触摸屏上报的原始数据：

```
hexdump /dev/input/event2
```

现在用一根手指触摸屏幕的右上角，然后再抬起，理论坐标值为(1023,0)，但是由于触摸误差的原因，大概率不会是绝对的(1023,0)，应该是在此值附近的一个触摸坐标值，实际的上报数据如图 64.4.2.3 所示：

```
/lib/modules/4.1.15 # hexdump /dev/input/event2
0000000 02bb 0000 9459 0007 0003 002f 0000 0000
0000010 02bb 0000 9459 0007 0003 0039 0005 0000
0000020 02bb 0000 9459 0007 0003 0035 03ec 0000
0000030 02bb 0000 9459 0007 0003 0036 0017 0000
0000040 02bb 0000 9459 0007 0001 014a 0001 0000
0000050 02bb 0000 9459 0007 0003 0000 03ec 0000
0000060 02bb 0000 9459 0007 0003 0001 0017 0000
0000070 02bb 0000 9459 0007 0000 0000 0000 0000
0000080 02bb 0000 e5f8 0008 0003 0039 ffff ffff
0000090 02bb 0000 e5f8 0008 0001 014a 0000 0000
00000a0 02bb 0000 e5f8 0008 0000 0000 0000 0000
```

图 8.2.14-2 查看上报数据

8.2.15 4G 无法 ping 百度

问题：按照教程移植 4G 驱动，注册完 4G 后无法 ping 百度。

```
/etc/gosuncn # ifconfig
ppp0      Link encap:Point-to-Point Protocol
          inet addr:10.212.195.22  P-t-P:10.212.195.21  Mask:255.255.255.255
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:4 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:3
          RX bytes:68 (68.0 B)  TX bytes:54 (54.0 B)

/etc/gosuncn #
/etc/gosuncn #
/etc/gosuncn # ping www.baidu.com
ping: bad address 'www.baidu.com'
```

图 8.2.15-1 现象

解决：在 /etc/resolv.conf 文件中添加 nameserver 114.114.114.114

在 rootfs 中新建文件/etc/resolv.conf，然后在里面输入如下内容：

示例代码 38.5.4.1 resolv.conf 文件内容

```
1 nameserver 114.114.114.114
2 nameserver 192.168.1.1      自己的网关
```

图 8.2.15-2 添加内容

执行 sync 同步命令后就可以 ping 百度。

```
/etc # vi resolv.conf
/etc #
/etc #
/etc # sync
/etc #
/etc #
/etc #
/etc # ping www.baidu.com
PING www.baidu.com (39.156.66.14): 56 data bytes
64 bytes from 39.156.66.14: seq=0 ttl=49 time=223.457 ms
64 bytes from 39.156.66.14: seq=1 ttl=49 time=170.734 ms
64 bytes from 39.156.66.14: seq=2 ttl=49 time=691.101 ms
64 bytes from 39.156.66.14: seq=3 ttl=49 time=1451.218 ms
64 bytes from 39.156.66.14: seq=4 ttl=49 time=2134.833 ms
```

图 8.2.15-3 测试

8.2.16 IPv6: ADDRCONF(NETDEV_UP): wlan0: link is not ready

使用 USB_WIFI 模块时，ifconfig wlan0 up 报错。

```
/lib/modules/4.1.15 # depmod
/lib/modules/4.1.15 # modprobe 8192cu
usbcore: registered new interface driver rtl8192cu
/lib/modules/4.1.15 # modprobe 8192cu
/lib/modules/4.1.15 #
/lib/modules/4.1.15 #
/lib/modules/4.1.15 #
/lib/modules/4.1.15 #
/lib/modules/4.1.15 # modprobe 8192cu
/lib/modules/4.1.15 #
/lib/modules/4.1.15 #
/lib/modules/4.1.15 # ifconfig wlan0 up
IPv6: ADDRCONF(NETDEV_UP): wlan0: link is not ready
```

图 8.2.16-1 报错信息

解决：重新加载一次驱动，再执行 ifconfig wlan0 up 即可成功。

```
/lib/modules/4.1.15 #
/lib/modules/4.1.15 # modprobe 8192cu
/lib/modules/4.1.15 #
/lib/modules/4.1.15 #
/lib/modules/4.1.15 # ifconfig wlan0 up
IPv6: ADDRCONF(NETDEV_UP): wlan0: link is not ready
/lib/modules/4.1.15 # modprobe 8192cu
/lib/modules/4.1.15 # ifconfig wlan0 up
/lib/modules/4.1.15 # ^C
/lib/modules/4.1.15 #
```

图 8.2.16-2 重新加载驱动

8.2.17 RTL871X: ERROR sdio_chk_hci_resume((null)) err:-123

操作：按照开发指南教程验证 SDIO WIFI 驱动时，先将板子和 SDIO WIFI 模块连接，再执行加载驱动命令时报错：

RTL871X: ERROR sdio_chk_hci_resume((null)) err:-123

```
/lib/modules/4.1.15 # depmod
/lib/modules/4.1.15 #
/lib/modules/4.1.15 #
/lib/modules/4.1.15 # modprobe 8189fs.ko
modprobe: module '8189fs.ko' not found
/lib/modules/4.1.15 # modprobe 8189fs
RTL871X: module init start
RTL871X: rtl8189fs v4.3.24.8_22657.20170607
sched: RT throttling activated
RTL871X: ERROR sdio_chk_hci_resume((null)) err:-123
RTL871X: ERROR sdio_chk_hci_resume((null)) err:-123
RTL871X: ERROR sdio_chk_hci_resume((null)) err:-123
```

图 8.2.17-1 报错信息

解决：先取下 SDIO WIFI 模块，执行加载驱动命令。

```
/lib/modules/4.1.15 # depmod
/lib/modules/4.1.15 # modprobe 8189fs
RTL871X: module init start
RTL871X: rtl8189fs v4.3.24.8_22657.20170607
random: nonblocking pool is initialized
RTL871X: module init ret=0
```

图 8.2.17-2 加载驱动

执行加载命令成功后，再接入 SDIO WIFI 模块，此时会有相关打印信息。

```
/lib/modules/4.1.15 # sdhci-esdhc-imx 2190000.usdhc: card claims to support voltag
es below defined range
mmc0: error -110 whilst initialising SDIO card
sdhci-esdhc-imx 2190000.usdhc: card claims to support voltages below defined range
mmc0: new high speed SDIO card at address 0001
RTL871X: HW EFUSE
RTL871X: hal_com_config_channel_plan chplan:0x20
====>_BlockWrite 109 i:20
====>_WriteFW 244
RTL871X: rtw_regsty_chk_target_tx_power_valid return _FALSE for band:0, path:0, rs
:0, t:-1
RTL871X: rtw_ndev_init(wlan0) if1 mac_addr=30:4a:26:b1:45:62
```

图 8.2.17-3 接入模块后打印信息

执行 ifconfig -a 即可看到 wlan0。

```
wlan0     Link encap:Ethernet  HWaddr 30:4A:26:B1:45:62
          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)
```

图 8.2.17-4 查看网卡

8.2.18 USB 驱动实验中，Mass Storage Gadget 无法选为 M

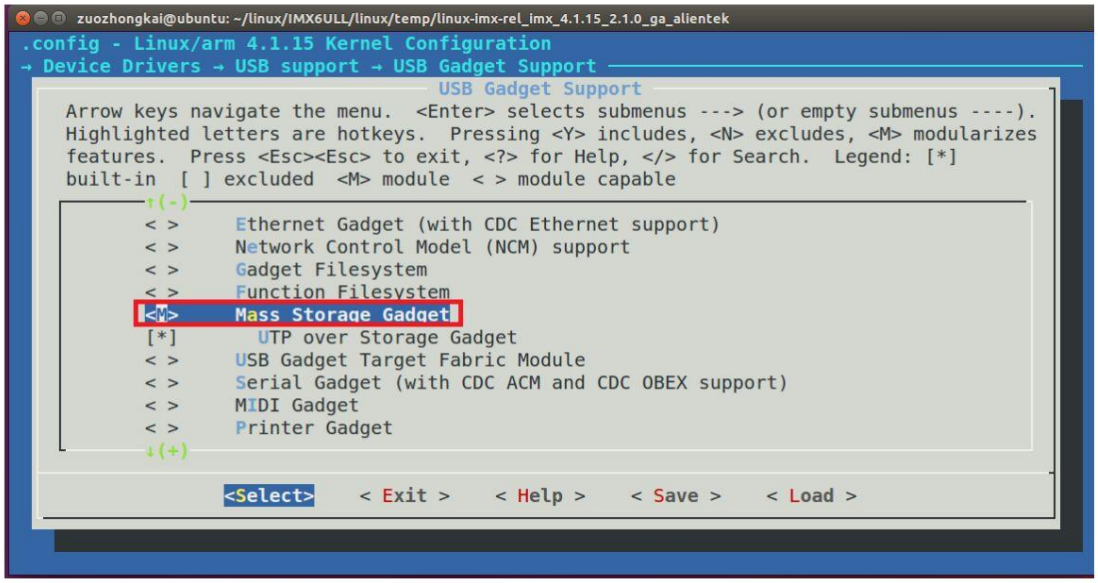


图 8.2.18-1 现象

解决：USB Gadget Drivers 要选为 M，才能接着实现 Mass Storage Gadget 为 M。

```
-> Device Drivers
    -> USB support (USB_SUPPORT [=y])
        -> USB Gadget Support (USB_GADGET [=y]
            -> [M]USB Gadget Drivers (<choice> [=m])      //选中 USB Gadget 驱动
                ->[M]Mass Storage Gadget                  //大容量存储
```

图 8.2.18-2 解决

8.3 实验工具包出错

8.3.1 ncurses 和 minicom 报错

在编译 ncurses 的时候报错。

```
gnatlink tour.ali -L../lib -lAdaCurses -L../../lib -lform -lmenu -lpanel -lncurses -Wl,-rpath,/home/MY/kehu/111
/ncurses-6.0/lib
/usr/bin/ld: ../lib/libAdaCurses.a(static_c_varargs_to_ada.o): Relocations in generic ELF (EM: 40)
../lib/libAdaCurses.a: error adding symbols: File in wrong format
collect2: ld returned 1 exit status
gnatlink: error when calling /usr/bin/gcc-4.6
gnatmake: *** link failed.
make[2]: *** [tour] Error 4
make[2]: Leaving directory `/home/MY/kehu/111/ncurses-6.0/Ada95/samples'
make[1]: *** [all] Error 2
make[1]: Leaving directory `/home/MY/kehu/111/ncurses-6.0/Ada95'
make: *** [all] Error 2
MY@IMX6ULL:~/kehu/111/ncurses-6.0$
```

图 8.3.1-1 报错信息

在编译 minicom 的时候报错。

```
window.c:2009:6: warning: assignment makes pointer from integer without a cast
   TS = tgetstr("ts", &_tptr);
      ^
window.c:2010:6: warning: assignment makes pointer from integer without a cast
   FS = tgetstr("fs", &_tptr);
      ^
window.c:2011:6: warning: assignment makes pointer from integer without a cast
   DS = tgetstr("ds", &_tptr);
      ^
window.c:2036:3: warning: implicit declaration of function ‘tgetflag’ [-Wimplicit-function-declaration]
   if (!tgetflag("hs") || !tgetflag("es") || !TS || !FS)
   ^
make[2]: *** [window.o] Error 1
make[2]: Leaving directory `/home/MY/kehu/111/minicom-2.7.1/src'
make[1]: *** [all-recursive] Error 1
make[1]: Leaving directory `/home/MY/kehu/111/minicom-2.7.1'
make: *** [all] Error 2
MY@IMX6ULL:~/kehu/111/minicom-2.7.1$
```

图 8.3.1-2 报错信息

这种一般是 ncurses 没配置好，所以会有各种错误，检查路径是否有写错的，配置指令可以参照如下的来写，特别是加上了这个选项--target=arm-linux-gnueabihf

解决办法：

```
./configure --prefix=/home/zuozhongkai/linux/IMX6ULL/tool/ncurses --host=arm-linux-gnueabihf --target=arm-linux-gnueabihf --with-shared --without-profile --disable-stripping --without-progs --with-manpages --without-tests
```

8.3.2 alsa-utils 移植报错 configure: error: No linkable libasound was found

```
checking for dlopen in -lc... no
checking for dlopen in -ldl... yes
checking for ALSA LDFLAGS...  -lasound -lm -ldl -lpthread
checking required libasound headers version... 1.0.27
checking for libasound headers version >= 1.0.27 (1.0.27)...
checking for libatopology (sound headers version > 1.1.9)...
checking for snd_ctl_open in -lasound... no
configure: error: No linkable libasound was found.
lmq@lmq-virtual-machine:~/linux/IMX6ULL/tool/alsa-utils-1.2.2
```

图 8.3.2-1 报错信息

遇见这种情况比较多的情况是，指令打错了，建议重新弄，还有就是在编译前先执行 source /etc/profile 来使能一下环境变量以后再进行编译，编译指令也可以将 make 换成

```
make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf-
```

另一种方法就是切换到 root 用户进行编译：

```
sudo -s
source /etc/profile
```

再按照教程执行配置指令：

```
alientek@ubuntu16:~/linux/IMX6ULL/tool/alsa-utils-1.2.2$ sudo -s
root@ubuntu16:~/linux/IMX6ULL/tool/alsa-utils-1.2.2# source /etc/profile
root@ubuntu16:~/linux/IMX6ULL/tool/alsa-utils-1.2.2# ./configure --host=arm-linux
-gnueabihf --prefix=/home/alientek/linux/IMX6ULL/tool/alsa-utils --with-alsa-inc-
prefix=/home/alientek/linux/IMX6ULL/tool/alsa-lib/include/ --with-alsa-prefix=/ho
me/alientek/linux/IMX6ULL/tool/alsa-lib/lib/ --disable-alsamixer --disable-xmlto
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for arm-linux-gnueabihf-strip... arm-linux-gnueabihf-strip
checking for a thread-safe mkdir -p... /bin/mkdir -p
checking for gawk... no
checking for mawk... mawk
checking whether make sets $(MAKE)... yes
checking whether make supports nested variables... yes
checking whether to enable maintainer-specific portions of Makefiles... yes
checking for a sed that does not truncate output... /bin/sed
checking whether NLS is requested... yes
```

图 8.3.2-2 编译

配置完成后，进行编译和安装，全部完成后，切换回用户账号。

```
make
sudo make install
su 用户名                    \\切换为原来的用户
```

8.4 其他错误

8.4.1 cc1:error:code model kernel does not support PIc mode

```
jiang@jiang:~/linux/Linux_Device/1_chrdevbase$ ls
1_chrdevbase.code-workspace  chrdevbaseApp.c  chrdevbase.c  Makefile
jiang@jiang:~/linux/Linux_Device/1_chrdevbase$ make
make -C /home/jiang/linux/alientek_linux M=/home/jiang/linux/Linux_Device/1_chrd
evbase modules
make[1]: 进入目录“/home/jiang/linux/alientek_linux”
  CC [M]  /home/jiang/linux/Linux_Device/1_chrdevbase/chrdevbase.o
cc1: error: code model kernel does not support PIC mode
make[2]: *** [scripts/Makefile.build:265：/home/jiang/linux/Linux_Device/1_chrde
vbase/chrdevbase.o] 错误 1
make[1]: *** [Makefile:1384：_module_/home/jiang/linux/Linux_Device/1_chrdevbase
] 错误 2
make[1]: 离开目录“/home/jiang/linux/alientek_linux”
make: *** [Makefile:8：kernel_modules] 错误 2
jiang@jiang:~/linux/Linux_Device/1_chrdevbase$
```

图 8.4.1-1 报错信息

解决办法：

上面编译报错的那个，试试在内核源码根目录下的 Makefile 下加上这句话，保存退出了重新编译就不报错了（PS：这是因为用的是 开发板光盘 A-基础资料\1、例程源码\3、正点原子 Uboot 和 Linux 出厂源码 来做这个实验才会有这个报错的，如果按照驱动教程使用对应的内核源码，就不会有这个问题的）。

```
KBUILD_CFLAGS += $(CFLAGS_EXTRA) -fno-pie
```

打开内核源码根目录的 Makefile 文件，可以发现是已经有 KBUILD_CFLAGS 变量了，那么直接在这个变量后面追加-fno-pie 就可以了，如下图：

```
392         -Iarch/$(hdr-arch)/include/generated/uapi \
393         -Iarch/$(hdr-arch)/include/generated \
394         $(if $(KBUILD_SRC), -I$(srctree)/include) \
395         -Iinclude \
396         $(USERINCLUDE)
397
398 KBUILD_CPPFLAGS := -D__KERNEL__
399
400 KBUILD_CFLAGS   := -Wall -Wundef -Wstrict-prototypes -Wno-trigraphs \
401            -fno-strict-aliasing -fno-common \
402            -Werror-implicit-function-declaration \
403            -Wno-format-security \
404            -std=gnu89 -fno-pie
405 KBUILD_AFLAGS_KERNEL :=
406 KBUILD_CFLAGS_KERNEL :=
407 KBUILD_AFLAGS   := -D__ASSEMBLY__
408 KBUILD_AFLAGS_MODULE  := -DMODULE
409 KBUILD_CFLAGS_MODULE  := -DMODULE
410 KBUILD_LDFLAGS_MODULE := -T $(srctree)/scripts/module-common.lds
```

图 8.4.1-2 追加

8.4.2 编译第 13 个驱动例程 irq 失败

报错信息：

131:49: error: ‘name’ undeclared (first use in this function)
memset(imx6uirq.irqkeydesc[i].name, 0, sizeof(name)); /* 缓冲区清零 */

```
alientek@ubuntu16:~/alpha/alientek-zzk/demo/Linux-demo/13_irq$ make
make -C /home/alientek/linux/IMX6ULL/linux/temp/linux-imx-rel_imx_4.1.15_2.1.0_ga_alientek M=/home/alientek/alpha/alientek-zzk/demo/Linux-demo/13_irq modules
make[1]: Entering directory '/home/alientek/linux/IMX6ULL/linux/temp/linux-imx-rel_imx_4.1.15_2.1.0_ga_alientek'
  CC [M]  /home/alientek/alpha/alientek-zzk/demo/Linux-demo/13_irq/imx6uirq.o
In file included from include/linux/string.h:17:0,
                 from include/linux/dynamic_debug.h:111,
                 from include/linux/printk.h:275,
                 from include/linux/kernel.h:13,
                 from /home/alientek/alpha/alientek-zzk/demo/Linux-demo/13_irq/imx6uirq.c:2:
/home/alientek/alpha/alientek-zzk/demo/Linux-demo/13_irq/imx6uirq.c: In function ‘keyio_init’:
/home/alientek/alpha/alientek-zzk/demo/Linux-demo/13_irq/imx6uirq.c:131:49: error: ‘name’ undeclared (first use in this function)
   memset(imx6uirq.irqkeydesc[i].name, 0, sizeof(name)); /* 缓冲区清零 */
                                                 ^
./arch/arm/include/asm/string.h:31:34: note: in definition of macro ‘memset’
   void *__p = (p); size_t __n = n;   \
                                 ^
/home/alientek/alpha/alientek-zzk/demo/Linux-demo/13_irq/imx6uirq.c:131:49: note: each undeclared identifier is reported only once for each function it appears in
   memset(imx6uirq.irqkeydesc[i].name, 0, sizeof(name)); /* 缓冲区清零 */
                                                 ^
./arch/arm/include/asm/string.h:31:34: note: in definition of macro ‘memset’
   void *__p = (p); size_t __n = n;   \
                                 ^
scripts/Makefile.build:263: recipe for target '/home/alientek/alpha/alientek-zzk/demo/Linux-demo/13_irq/imx6uirq.o' failed
make[2]: *** [/home/alientek/alpha/alientek-zzk/demo/Linux-demo/13_irq/imx6uirq.o] Error 1
Makefile:1384: recipe for target '_module_/home/alientek/alpha/alientek-zzk/demo/Linux-demo/13_irq' failed
make[1]: *** [_module_/home/alientek/alpha/alientek-zzk/demo/Linux-demo/13_irq] Error 2
make[1]: Leaving directory '/home/alientek/linux/IMX6ULL/linux/temp/linux-imx-rel_imx_4.1.15_2.1.0_ga_alientek'
Makefile:9: recipe for target 'kernel_modules' failed
make: *** [kernel_modules] Error 2
```

图 8.4.2-1 报错信息

解决方法：

源码中 memset(imx6uirq.irqkeydesc[i].name, 0, sizeof(name)); /* 缓冲区清零 */ 有误。

```
    /* 初始化key所使用的IO，并且设置成中断模式 */
    for (i = 0; i < KEY_NUM; i++) {
        memset(imx6uirq.irqkeydesc[i].name, 0, sizeof(name));   /* 缓冲区清零 */
        sprintf(imx6uirq.irqkeydesc[i].name, "KEY%d", i);       /* 组合名字 */
        gpio_request(imx6uirq.irqkeydesc[i].gpio, imx6uirq.irqkeydesc[i].name);
        gpio_direction_input(imx6uirq.irqkeydesc[i].gpio);
        imx6uirq.irqkeydesc[i].irqnum = irq_of_parse_and_map(imx6uirq.nd, i);
#if 0
        imx6uirq.irqkeydesc[i].irqnum = gpio_to_irq(imx6uirq.irqkeydesc[i].gpio);
#endif
```

图 8.4.2-2 修改前的源码

将其修改为

```
memset(imx6uirq.irqkeydesc[i].name, 0, sizeof(imx6uirq.irqkeydesc[i].name));   /* 缓冲区清零 */
```

```
    /* 初始化key所使用的IO，并且设置成中断模式 */
    for (i = 0; i < KEY_NUM; i++) {
        memset(imx6uirq.irqkeydesc[i].name, 0, sizeof(imx6uirq.irqkeydesc[i].name));   /* 缓冲区清零 */
        sprintf(imx6uirq.irqkeydesc[i].name, "KEY%d", i);       /* 组合名字 */
        gpio_request(imx6uirq.irqkeydesc[i].gpio, imx6uirq.irqkeydesc[i].name);
        gpio_direction_input(imx6uirq.irqkeydesc[i].gpio);
        imx6uirq.irqkeydesc[i].irqnum = irq_of_parse_and_map(imx6uirq.nd, i);
#if 0
        imx6uirq.irqkeydesc[i].irqnum = gpio_to_irq(imx6uirq.irqkeydesc[i].gpio);
#endif
```

图 8.4.2-3 修改完的源码

8.4.3 Permission denied

执行例程出现问题 Permission denied

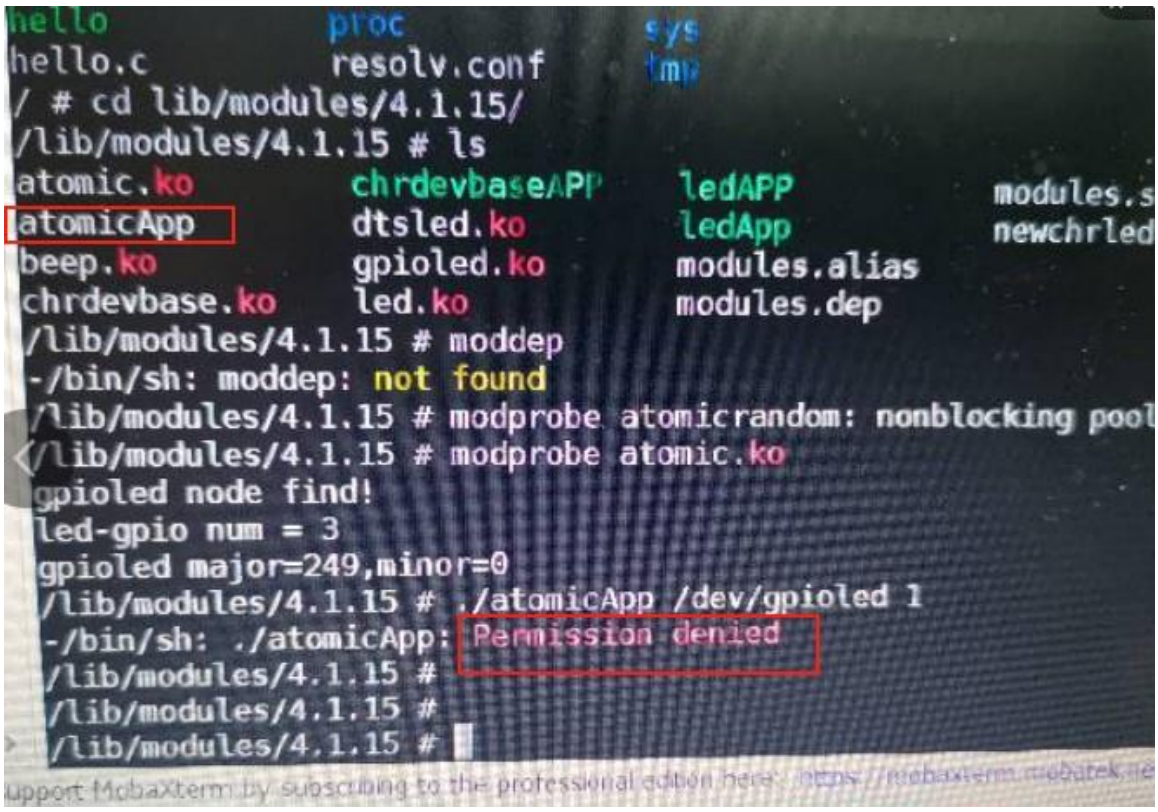


图 8.4.3-1 报错信息

原因：App 测试程序没有可执行权限，给到 777 最高权限即可。

8.5 加载模块的方法

如果以上的情况都无法解决（排除程序问题），那么最好还是按步骤编译内核和设备树→编译内核模块→再安装内核模块，最后加载内核模块进行验证。

```
make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- zImage
make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- imx6ull-alientek-emmc.dtb
make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- modules
make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- modules_install INSTALL_MOD_PATH=后面加上文件系统的路径
```

以上指令也可以写成两条来执行

make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf-

make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- modules_install INSTALL_MOD_PATH=后面加上文件系统的路径

文件系统的路径，这里说明一下，如果文件系统在 ubuntu 的 NFS 目录下，那么这个目录就写 ubuntu 上 NFS 文件系统的目录，如果是要安装到 SD 系统启动卡中的文件系统分区，那么将 SD 系统启动卡插入到 ubuntu 并挂载，这个目录就是/media/MY/roofs。以下分情况进行讲解。

8.5.1 SD 卡的文件系统

make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- modules_install INSTALL_MOD_PATH=/media/MY/roofs

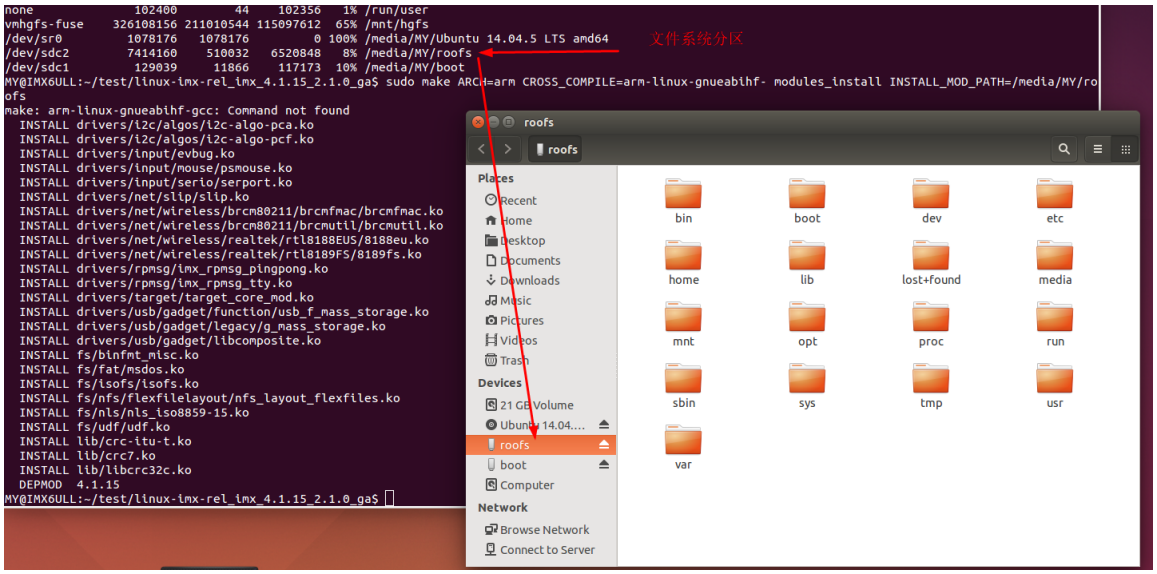


图 8.5.1-1 SD 卡加载模块

8.5.2 核心板 EMMC 或者 NAND FLASH 里的文件系统

如果是 EMMC 或者 NAND FLASH 里的文件系统，那么可以在 ubuntu 上新建一个目录，再将内核模块安装到这个目录下，然后再将这个目录里的文件打包。

如下图，我在内核源码下新建了一个目录 mymodules，执行指令将内核模块安装到这个目录下，然后看到在 mymodules/lib/modules/4.1.15/下生成一些文件（ubuntu 下最好切换到 root 用户下来操作）

mkdir mymodules

make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- modules_install INSTALL_MOD_PATH=mymodules

```
root@IMX6ULL:/home/MY/test/linux-imx-rel_imx_4.1.15_2.1.0_ga# mkdir mymodules
root@IMX6ULL:/home/MY/test/linux-imx-rel_imx_4.1.15_2.1.0_ga# ls
arch     CREDITS        drivers   include  jj       kernel       Makefile         modules.order   net     REPORTING-BUGS  security    tools  vmlinux
block    crypto         firmware  init     Kbuild   lib          mm               Module.symvers  p.dts   samples         sound       usr    vmlinux.o
COPYING  Documentation  fs        ipc      Kconfig  MAINTAINERS  modules.builtin  mymodules       README  scripts         System.map  virt
root@IMX6ULL:/home/MY/test/linux-imx-rel_imx_4.1.15_2.1.0_ga# make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- modules_install INSTALL_MOD_PATH=mymodule
s
  INSTALL drivers/i2c/algos/i2c-algo-pca.ko
  INSTALL drivers/i2c/algos/i2c-algo-pcf.ko
  INSTALL drivers/input/evbug.ko
  INSTALL drivers/input/mouse/psmouse.ko
  INSTALL drivers/input/serio/serport.ko
  INSTALL drivers/net/slip/slip.ko
  INSTALL drivers/net/wireless/brcm80211/brcmfmac/brcmfmac.ko
  INSTALL drivers/net/wireless/brcm80211/brcmutil/brcmutil.ko
  INSTALL drivers/net/wireless/realtek/rtl8188EUS/8188eu.ko
  INSTALL drivers/net/wireless/realtek/rtl8189FS/8189fs.ko
  INSTALL drivers/rpmsg/imx_rpmsg_pingpong.ko
  INSTALL drivers/rpmsg/imx_rpmsg_tty.ko
  INSTALL drivers/target/target_core_mod.ko
  INSTALL drivers/usb/gadget/function/usb_f_mass_storage.ko
  INSTALL drivers/usb/gadget/legacy/g_mass_storage.ko
  INSTALL drivers/usb/gadget/libcomposite.ko
  INSTALL fs/binfmt_misc.ko
  INSTALL fs/fat/msdos.ko
  INSTALL fs/isofs/isofs.ko
  INSTALL fs/nfs/flexfilelayout/nfs_layout_flexfiles.ko
  INSTALL fs/nls/nls_iso8859-15.ko
  INSTALL fs/udf/udf.ko
  INSTALL lib/crc-itu-t.ko
  INSTALL lib/crc7.ko
  INSTALL lib/libcrc32c.ko
  DEPMOD  4.1.15
root@IMX6ULL:/home/MY/test/linux-imx-rel_imx_4.1.15_2.1.0_ga#
root@IMX6ULL:/home/MY/test/linux-imx-rel_imx_4.1.15_2.1.0_ga# ls mymodules/lib/modules/4.1.15/
build   modules.alias      modules.builtin      modules.dep      modules.devname  modules.softdep  modules.symbols.bin
kernel  modules.alias.bin  modules.builtin.bin  modules.dep.bin  modules.order    modules.symbols  source
root@IMX6ULL:/home/MY/test/linux-imx-rel_imx_4.1.15_2.1.0_ga#
```

图 8.5.2-1 安装模块到文件夹

进行打包，注意这里打包的指令和是在哪个目录下打包的！！！：

```
tar -vcjf modules.tar.bz2 *
```

```
root@IMX6ULL:/home/MY/test/linux-imx-rel_imx_4.1.15_2.1.0_ga/mymodules/lib/modules# ls
4.1.15
root@IMX6ULL:/home/MY/test/linux-imx-rel_imx_4.1.15_2.1.0_ga/mymodules/lib/modules# tar -vcjf modules.tar.bz2 *
4.1.15/
4.1.15/source
4.1.15/modules.devname
4.1.15/modules.order
4.1.15/modules.dep.bin
4.1.15/modules.builtin.bin
4.1.15/modules.alias
4.1.15/modules.builtin
4.1.15/modules.alias.bin
4.1.15/build
4.1.15/modules.symbols
4.1.15/modules.softdep
4.1.15/modules.dep
4.1.15/modules.symbols.bin
4.1.15/kernel/
4.1.15/kernel/fs/
4.1.15/kernel/fs/nfs/
```

图 8.5.2-2 打包模块

```
root@IMX6ULL:/home/MY/test/linux-imx-rel_imx_4.1.15_2.1.0_ga/mymodules/lib/modules# ls
4.1.15  modules.tar.bz2
root@IMX6ULL:/home/MY/test/linux-imx-rel_imx_4.1.15_2.1.0_ga/mymodules/lib/modules#
```

图 8.5.2-3 打包完的模块包

如上图，最后得到 modules.tar.bz2 这个压缩包。如果是 EMMC 版本的核心板，直接将这个压缩包解压在 EMMC 文件系统的/lib/下。如果是 NAND 版本的核心板，可以按照《【正点原子】I.MX6U 开发板与 pc 传输文件、单步更新固件参考手册 V1.0》这个文档来操作，将内核模块解压到/mnt/mtd5/lib/modules 下：

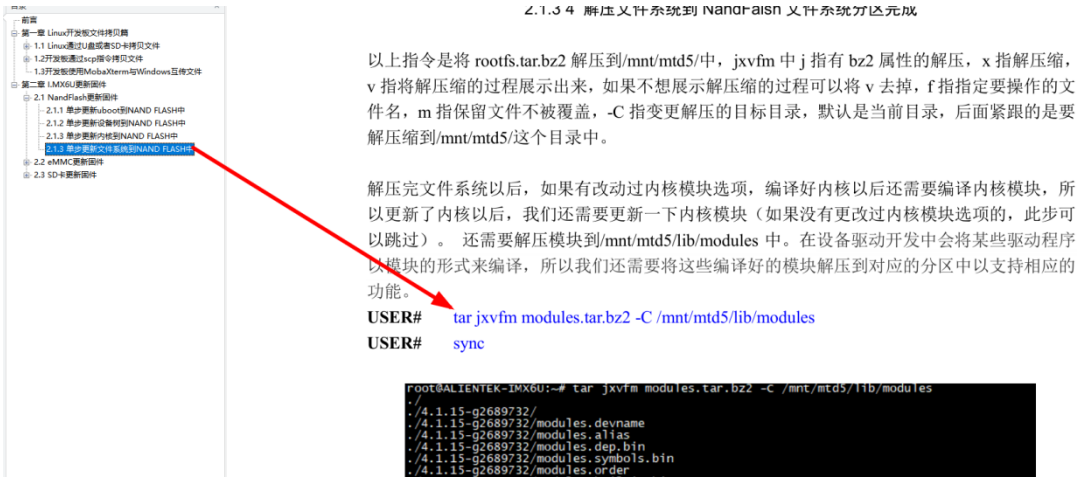


图 8.5.2-4 解压模块包到开发板文件系统

8.5.3 ubuntu 上 NFS 目录下的文件系统

例如下图，我的文件系统是在 NFS 下的，路径为/home/MY/NFS/myrootfs，那么安装模块的指令就是（最好切换到 root 用户下来操作）：

```
make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- modules_install INSTALL_MOD_PATH= /home/MY/NFS/myrootfs
```

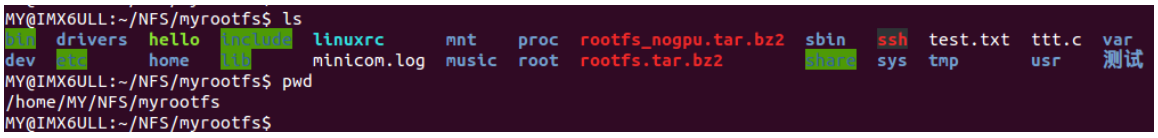


图 8.5.3-1 确定文件系统路径

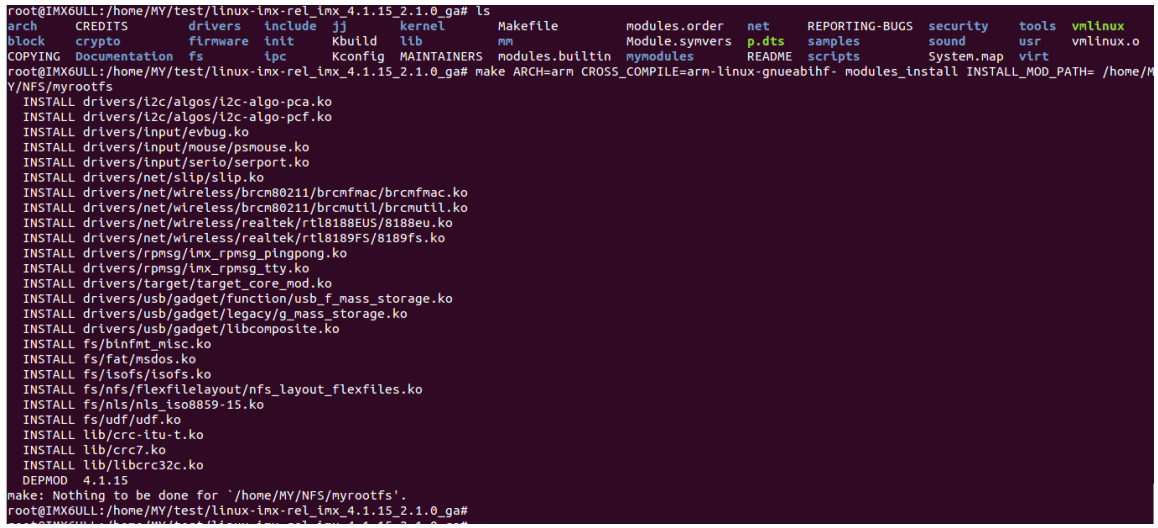


图 8.5.3-2 安装模块到文件系统

第九章 文件系统操作相关

9.1 出厂系统

9.1.1 如何关闭 QT 桌面

（1）临时关闭 QT 桌面程序

用出厂的系统进行触摸屏校准时，或者需要关闭 QT 桌面时，输入指令 `/etc/init.d/psplash.sh` 可退出桌面程序，然后再执行指令 `ts_calibrate` 即可进行触摸屏校准，校准以后屏幕还是黑的，显示两行英文，还是校准的那个界面，恢复 QT 桌面的办法可以重启开发板，或者不用重启开发板，执行以下打开 QT 桌面程序的指令就可以了：

```
/opt/qt5.5.1/apps/QDesktop/QDesktop &
```

（注：如果有校准文件的话，校准文件一般是在/etc/pointercal 下）

（2）永久关闭 QT 桌面程序

如果要永久关闭出厂文件系统的 QT 桌面，可以打开/etc/rc.local 文件，修改如下：

将/opt/qt5.5.1/apps/QDesktop/QDesktop >/dev/null 2>&1 &这句话用#号注释掉。

```
#!/bin/sh -e
#
# rc.local
#
# This script is executed at the end of each multiuser runlevel.
# Make sure that the script will "exit 0" on success or any other
# value on error.
#
# In order to enable or disable this script just change the execution
# bits.
#
# By default this script does nothing.


echo 30000 >  /proc/sys/vm/min_free_kbytes
echo "0" > /sys/class/graphics/fb0/blank
source /etc/profile
#/opt/qt5.5.1/apps/QDesktop/QDesktop >/dev/null 2>&1 &
exit 0
```

图 9.1.1-1 永久关闭 Qt

9.1.2 使用双网口 ping 百度

前提：阿尔法板烧录出厂系统，开发板直连路由器，或者已经搭建好网络环境。

要分别给 eth0,eth1 设置 IP 和网关，这里以自启动为例，打开出厂系统/etc/rc.local，添加如下指令，IP 可以根据实际情况设置。

```
ifconfig eth0 192.168.1.166 netmask 255.255.255.0
ifconfig eth1 192.168.1.103 netmask 255.255.255.0
route add default gw 192.168.1.1 eth0
route add default gw 192.168.1.1 eth1
echo "nameserver 114.114.114.114" > /etc/resolv.conf

source /etc/profile
/opt/QDesktop >/dev/null 2>&1 &
exit 0
```

图 9.1.2-1 设置 IP 和网关

使用 route 查看现在的网关是否一致。

```
root@ATK-IMX6U:~# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         192.168.1.1     0.0.0.0         UG    0      0        0 eth1
default         192.168.1.1     0.0.0.0         UG    0      0        0 eth0
public1.114dns. 192.168.1.1     255.255.255.255 UGH   0      0        0 eth0
public1.114dns. 192.168.1.1     255.255.255.255 UGH   0      0        0 eth1
192.168.1.0     *               255.255.255.0   U     0      0        0 eth0
192.168.1.0     *               255.255.255.0   U     0      0        0 eth1
192.168.1.1     *               255.255.255.255 UH    0      0        0 eth0
192.168.1.1     *               255.255.255.255 UH    0      0        0 eth1
root@ATK-IMX6U:~#
```

图 9.1.2-2 查看网关信息

执行 ifconfig 指令查看 IP。

```
root@ATK-IMX6U:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 8e:2a:c9:75:38:69
          inet addr:192.168.1.166  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::8c2a:c9ff:fe75:3869/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1971 errors:0 dropped:18 overruns:0 frame:0
          TX packets:380 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:171864 (167.8 KiB)  TX bytes:37632 (36.7 KiB)

eth1      Link encap:Ethernet  HWaddr 12:08:c6:ed:f2:57
          inet addr:192.168.1.103  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::1008:c6ff:feed:f257/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST DYNAMIC  MTU:1500  Metric:1
          RX packets:2371 errors:0 dropped:18 overruns:0 frame:0
          TX packets:139 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:223268 (218.0 KiB)  TX bytes:16979 (16.5 KiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
```

图 9.1.2-3 查看 IP

使用以下指令测试网口上网情况，注意对应的网口要接网线。

```
ping www.baidu.com -I eth0
ping www.baidu.com -I eth1
```

ctrl+d 可以退掉当前测试。

```
root@ATK-IMX6U:~# ping www.baidu.com -I eth0
PING www.a.shifen.com (14.215.177.38) from 192.168.1.166 eth0: 56(84) bytes of data.
64 bytes from 14.215.177.38: icmp_seq=1 ttl=56 time=10.0 ms
^C
--- www.a.shifen.com ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 10.043/10.043/10.043/0.000 ms
root@ATK-IMX6U:~# ping www.baidu.com -I eth1
PING www.a.shifen.com (14.215.177.39) from 192.168.1.103 eth1: 56(84) bytes of data.
64 bytes from 14.215.177.39: icmp_seq=1 ttl=56 time=6.40 ms
64 bytes from 14.215.177.39: icmp_seq=2 ttl=56 time=6.17 ms
^C
--- www.a.shifen.com ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1000ms
rtt min/avg/max/mdev = 6.176/6.290/6.404/0.114 ms
```

图 9.1.2-4 单步测试一个网口

```
root@ATK-IMX6U:/opt#
root@ATK-IMX6U:/opt#
root@ATK-IMX6U:/opt#
root@ATK-IMX6U:/opt#
root@ATK-IMX6U:/opt#
root@ATK-IMX6U:/opt# 64 bytes from 192.168.1.242: icmp_seq=16 ttl=64 time=0.322 ms

root@ATK-IMX6U:/opt#
root@ATK-IMX6U:/opt# ping 192.168.1.242 -I eth1 & 64 bytes from 192.168.1.242: icmp_seq=17 ttl=64 time7 -I eth0 64
 bytes from 192.168.1.242: icmp_seq=18 ttl=64 time=0.393 ms
64 bytes from 192.168.1.242: icmp_seq=19 ttl=64 time=0.413 ms
 &64 bytes from 192.168.1.242: icmp_seq=20 ttl=64 time=0.390 ms

[2] 698
PING 192.168.1.247 (192.168.1.247) from 192.168.1.115 eth0: 56(84) bytes of data.
64 bytes from 192.168.1.247: icmp_seq=1 ttl=64 time=0.326 ms
root@ATK-IMX6U:/opt# 64 bytes from 192.168.1.242: icmp_seq=21 ttl=64 time=0.383 ms
64 bytes from 192.168.1.247: icmp_seq=2 ttl=64 time=0.337 ms
64 bytes from 192.168.1.242: icmp_seq=22 ttl=64 time=0.396 ms
64 bytes from 192.168.1.247: icmp_seq=3 ttl=64 time=0.328 ms
64 bytes from 192.168.1.242: icmp_seq=23 ttl=64 time=0.354 ms
64 bytes from 192.168.1.247: icmp_seq=4 ttl=64 time=0.319 ms
64 bytes from 192.168.1.242: icmp_seq=24 ttl=64 time=0.367 ms
64 bytes from 192.168.1.247: icmp_seq=5 ttl=64 time=0.321 ms
64 bytes from 192.168.1.242: icmp_seq=25 ttl=64 time=0.405 ms
64 bytes from 192.168.1.247: icmp_seq=6 ttl=64 time=0.320 ms
64 bytes from 192.168.1.242: icmp_seq=26 ttl=64 time=0.387 ms
64 bytes from 192.168.1.247: icmp_seq=7 ttl=64 time=0.291 ms
64 bytes from 192.168.1.242: icmp_seq=27 ttl=64 time=0.381 ms
64 bytes from 192.168.1.247: icmp_seq=8 ttl=64 time=0.302 ms
64 bytes from 192.168.1.242: icmp_seq=28 ttl=64 time=0.391 ms
64 bytes from 192.168.1.247: icmp_seq=9 ttl=64 time=0.323 ms
64 bytes from 192.168.1.242: icmp_seq=29 ttl=64 time=0.380 ms
64 bytes from 192.168.1.247: icmp_seq=10 ttl=64 time=0.315 ms
64 bytes from 192.168.1.242: icmp_seq=30 ttl=64 time=0.396 ms
64 bytes from 192.168.1.247: icmp_seq=11 ttl=64 time=0.325 ms
64 bytes from 192.168.1.242: icmp_seq=31 ttl=64 time=0.395 ms
64 bytes from 192.168.1.247: icmp_seq=12 ttl=64 time=0.323 ms
64 bytes from 192.168.1.242: icmp_seq=32 ttl=64 time=0.417 ms
```

图 9.1.2-5 同时测试两个网口

9.1.3 NFS 挂载出厂系统后缺少驱动

问题：TFTP/NFS 挂载了出厂内核、设备树和文件系统后，某些驱动却不能使用，比如摄像头驱动，因为还没有加载驱动模块包。

打开 ucl2.xml 文件，可以看到烧录的具体命令。

ucl2.xml 文件所在路径：阿尔法 Linux 开发板（A 盘）-基础资料\5、开发工具\4、正点原子 MFG_TOOL 出厂固件烧录工具\mfgtool\Profiles\Linux\OS Firmware

图 9.1.3-1 ucl2.xml 文件

打开这个文件，搜索 burn rootfs

```
<!-- burn rootfs -->
<CMD state="Updater" type="push" body="pipe tar -jxv -C /mnt/mmcblk0p2" file="files/filesystem/rootfs.tar.bz2" ifdev="MX6ULL">Sending
and writting rootfs</CMD>
<CMD state="Updater" type="push" body="frf">Finishing rootfs write</CMD>
<CMD state="Updater" type="push" body="$ mkdir -p /mnt/mmcblk0p2/lib/modules">Mkdir -p /mnt/mmcblk0p2/lib/modules</CMD>
<CMD state="Updater" type="push" body="send" file="files/modules/modules.tar.bz2" ifdev="MX6ULL">Sending Modules file</CMD>
<CMD state="Updater" type="push" body="$ tar jxf $FILE -C /mnt/mmcblk0p2/lib/modules/" ifdev="MX6ULL">tar Modules file</CMD>
<CMD state="Updater" type="push" body="$ sleep 1">delay</CMD>
<CMD state="Updater" type="push" body="$ sync">Sync...</CMD>
<CMD state="Updater" type="push" body="$ umount /mnt/mmcblk0p2">Unmounting rootfs partition</CMD>
<CMD state="Updater" type="push" body="$ echo Update Complete!">Done</CMD>
</LIST>
```

图 9.1.3-2 搜索结果

可以看到，需要将 modules.tar.bz2 解压到 文件系统的 lib/modules 目录下。

烧录工具里 modules.tar.bz2 的路径：4、正点原子 MFG_TOOL 出厂固件烧录工具\mfgtool\Profiles\Linux\OS Firmware\files\modules

将它拷贝到 NFS 目录里文件系统的 lib/modules 下。

```
czx@ubuntu16:~/linux/nfs/rootfs-atk/lib$ cd modules/
czx@ubuntu16:~/linux/nfs/rootfs-atk/lib/modules$ ls
modules.tar.bz2
```

图 9.1.3-3 拷贝完 modules 包到 NFS 下的出厂系统

执行 tar jxf modules.tar.bz2 ，将驱动模块解压到 /lib/modules 目录下。

```
czx@ubuntu16:~/linux/nfs/rootfs-atk/lib/modules$ tar jxf modules.tar.bz2
czx@ubuntu16:~/linux/nfs/rootfs-atk/lib/modules$ ls
4.1.15-gb8ddbbc  modules.tar.bz2
```

图 9.1.3-4 解压驱动包

重新挂载出厂系统，可以看到 ov5640 驱动已经起来了。

```
Starting system log daemon...0
Starting kernel log daemon...0
 * Starting Avahi mDNS/DNS-SD Daemon: avahi-daemon                    [ ok ]
Starting Telephony daemon
Starting Linux NFC daemon
Starting crond: OK
Running local boot scripts (/etc/rc.local).

root@ATK-IMX6U:~# [   65.514928] ov5640: actual frame rate of XGA is 22.5fps
[   90.301386] ov5640: the actual frame rate of 1080P is 7.5fps

root@ATK-IMX6U:~#
```

图 9.1.3-5 挂载出厂系统启动

测试摄像头模块，这个需要刚刚添加模块压缩包的步骤才能进行。具体测试方法看用户快速体验手册的摄像头测试章节。

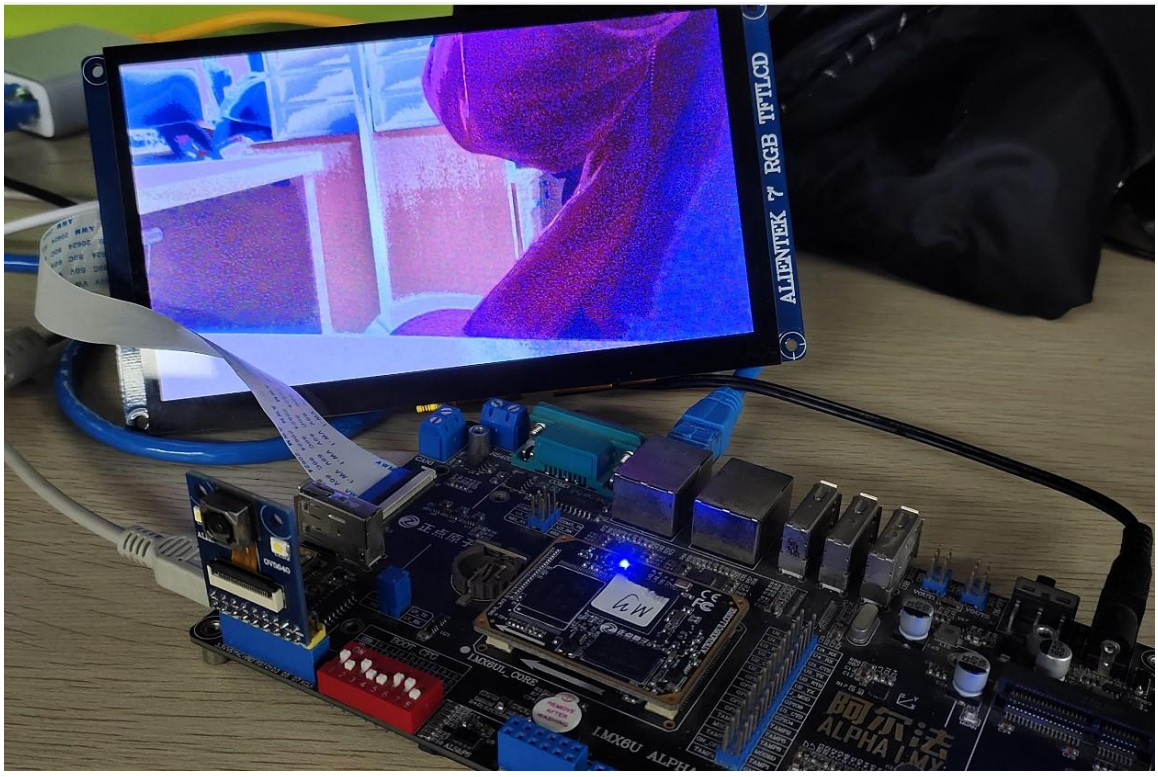


图 9.1.3-6 测试挂载出厂系统的摄像头

因为网口已接入网线用于 NFS 网络挂载，暂时没法测试上网功能。

9.2 开发板如何设置静态 IP 地址

如果文件系统/etc/init.d/下有 connman 文件，执行以下指令将 connman 命名为 connman2，这样做的目的是让系统找不到 connman 文件，关闭了 connmand 网络连接守护进程，如果不关闭，那么在后面的设置完成后会有两个 IP 地址。如果文件系统下没有这个文件，这步不必操作。

```
mv /etc/init.d/connman /etc/init.d/connman2
```

设置 IP 地址

IP 地址要根据当前网络拓扑关系来设置，例如需要设置开发板的 IP 地址是 192.168.1.110，以下列举设置开发板底板网口 ETH2（文件系统中对应的是 eth0）IP 地址的方法，选择其中一种就好：

9.2.1 设置永久静态 IP 地址（自启动执行命令）

在开发板文件系统中执行以下指令，执行完重启开发板

```
echo "ifconfig eth0 192.168.1.110 netmask 255.255.255.0" | tee -a /etc/init.d/rcS
echo "route add default gw 192.168.1.1" | tee -a /etc/init.d/rcS
```

以上指令中，| 是管道，管道经常用于拼接命令，tee 这个工具的作用就是把标准输出复制一份后扔到对应的文件里，-a 其实也就是>>即 shell 脚本中的重定向，表示追加的意思，追加就是在文件的后面写上要添加的指令，而不是覆盖源文件，>>和>不同的是，>是覆盖源文件（不能执行>哈，要不然原来的配置会被覆盖了）。以上指令也可以这样写：

```
echo "ifconfig eth0 192.168.1.110 netmask 255.255.255.0" | tee >> /etc/init.d/rcS
echo "route add default gw 192.168.1.1" | tee >> /etc/init.d/rcS
```

执行完以上指令后，执行 `cat -n /etc/init.d/rcS` 可以发现/etc/init.d/rcS 文件最后两行（第 17 和第 18 行）添加了以上两条指令，那么在开机启动后，这两条指令便会执行。

```
/ #
/ # echo "ifconfig eth0 192.168.1.110 netmask 255.255.255.0" | tee -a /etc/init.
d/rcS
ifconfig eth0 192.168.1.110 netmask 255.255.255.0
/ # echo "route add default gw 192.168.1.1" | tee -a /etc/init.d/rcS
route add default gw 192.168.1.1
/ #
/ # cat -n /etc/init.d/rcS
     1  #!/bin/sh
     2
     3  PATH=/sbin:/bin:/usr/sbin:/usr/bin:$PATH
     4  LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/lib:/usr/lib
     5  export PATH LD_LIBRARY_PATH
     6
     7  mount -a
     8  mkdir /dev/pts
     9  mount -t devpts devpts /dev/pts
    10
    11
    12  echo /sbin/mdev > /proc/sys/kernel/hotplug
    13  mdev -s
    14
    15
    16  # /hello &
    17  ifconfig eth0 192.168.1.110 netmask 255.255.255.0
    18  route add default gw 192.168.1.1
/ #
```

图 9.2.1-1 添加自启动命令

重启开发板，执行 ifconfig 指令或者 ip -a 查看 IP 地址修改成功。这种设置的好处在于操作简单，一次修改成功，下次再重启开发板 IP 地址不会变化，即永久生效。

```
/ #
/ # ifconfig
eth0      Link encap:Ethernet  HWaddr B6:56:74:93:9F:1E
          inet addr:192.168.1.110  Bcast:192.168.1.255  Mask:255.255.255.0
          UP 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)

/ # ip a
1: lo: <LOOPBACK> mtu 65536 qdisc noop
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether b6:56:74:93:9f:1e brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.110/24 brd 192.168.1.255 scope global eth0
       valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST> mtu 1500 qdisc noop qlen 1000
    link/ether 8e:df:fe:11:cd:ff brd ff:ff:ff:ff:ff:ff
4: sit0@NONE: <NOARP> mtu 1480 qdisc noop
    link/sit 0.0.0.0 brd 0.0.0.0
/ #
```

图 9.2.1-2 效果

9.2.2 设置永久静态 IP 地址（修改配置文件）

修改配置文件/etc/network/interfaces（这个配置文件在有的文件系统中有，有的文件系统下没有，没有的话，这种方法就不适合了）

原来的配置文件：

```
# /etc/network/interfaces -- configuration file for ifup(8), ifdown(8)

# The loopback interface
auto lo
iface lo inet loopback

# Wireless interfaces
iface wlan0 inet dhcp
        wireless_mode managed
        wireless_essid any
        wpa-driver wext
        wpa-conf /etc/wpa_supplicant.conf

iface atml0 inet dhcp

# Wired or wireless interfaces
auto eth0
iface eth0 inet dhcp
iface eth1 inet dhcp

# Ethernet/RNDIS gadget (g_ether)
# ... or on host side, usbnet and random hwaddr
iface usb0 inet static
        address 192.168.7.2
        netmask 255.255.255.0
        network 192.168.7.0
        gateway 192.168.7.1

# Bluetooth networking
iface bnep0 inet dhcp

~
~
- /etc/network/interfaces 1/31 3%
```

图 9.2.2-1 原配置文件

修改后如下：

```
# /etc/network/interfaces -- configuration file for ifup(8), ifdown(8)

# The loopback interface
auto lo
iface lo inet loopback

# Wireless interfaces
iface wlan0 inet dhcp
        wireless_mode managed
        wireless_essid any
        wpa-driver wext
        wpa-conf /etc/wpa_supplicant.conf

iface atml0 inet dhcp

# Wired or wireless interfaces
auto eth0
#iface eth0 inet dhcp
iface eth0 inet static
address 192.168.1.110
gateway 192.168.1.1
netmask 255.255.255.0

iface eth1 inet dhcp

# Ethernet/RNDIS gadget (g_ether)
# ... or on host side, usbnet and random hwaddr
iface usb0 inet static
        address 192.168.7.2
        netmask 255.255.255.0
        network 192.168.7.0
        gateway 192.168.7.1

- /etc/network/interfaces 1/36 2%
```

图 9.2.2-2 修改配置文件

修改完执行指令/etc/init.d/networking restart 重启网络配置即可，这种修改方法是改完后执行/etc/init.d/networking restart 就永久生效。

9.2.3 设置临时静态 IP 地址

在开发版文件系统中执行指令

```
ifconfig eth0 192.168.1.110 netmask 255.255.255.0
route add default gw 192.168.1.1
```

这种设置的好处在于，执行完指令不需重启开发板就生效，但是关机后下次再开机就没有了，即临时有效。

```
/ # ifconfig
/ #
/ # ifconfig eth0 192.168.1.110 netmask 255.255.255.0
fec 20b4000.ethernet eth0: Freescale FEC PHY driver [SMSC LAN8710/LAN8720] (mii_bus:phy_addr=20b4000.et
hernet:01, irq=-1)
IPv6: ADDRCONF(NETDEV_UP): eth0: link is not ready
/ #
/ # route add default gw 192.168.1.1
/ #
/ # ifconfig
eth0      Link encap:Ethernet  HWaddr 1A:C7:BC:2C:C8:AA
          inet addr:192.168.1.110  Bcast:192.168.1.255  Mask:255.255.255.0
          UP 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)
```

图 9.2.3-1 效果

9.3 开发板上运行可执行文件（程序）报错

9.3.1 文件不存在或者损坏

文件可能不存在或者已经损坏，进入对应的目录核实一下文件是否存在，不存在的文件执行的话是会报这个错的。

```
root@alientek_imx6ul:~# ls
test  tt
root@alientek_imx6ul:~# ./hello
-bash: ./hello: No such file or directory
root@alientek_imx6ul:~#
```

图 9.3.1-1 文件不存在

9.3.2 文件权限问题

文件权限问题，可以给文件设置权限再执行。（以下用笔者 hello 程序为例）
chmod 777 hello 或者 chmod +x hello
执行./hello

9.3.3 动态链接库问题

尝试重新编译，使用 arm-linux-gcc -static -o 来进行静态编译（即使用静态链接程序，不使用动态库），将编译后的程序放到板子上跑，如果运行成功，说明少了某些库。（可以观察，静态编译生成的文件比动态编译生成的文件要大的多）或者，可以用指令查询少了什么库，将少的那个库就从叉编译器 lib 下拷贝到开发板上，查询库依赖的指令是：

```
ldd 后面加可执行文件的名字
```

或者

```
arm-linux-readelf -a 后面加可执行文件的名字
```

9.3.4 编译器的版本问题

可能内核用 A 版本的编译器编译的，而可执行程序 hello 是用 B 版本的编译器来编译的，两个编译器版本不同的问题也可能会出现 not found 的问题。

9.3.5 系统位数的问题

如果是系统位数不同，板子是 32 位的系统，ubuntu 是 64 位的，那么 ubuntu 上要安装对应 32 位的库了再编译。

```
sudo apt-get update
sudo apt-get install lsb-core lib32stdc++6
```

9.4 Ubuntu 下执行某个可执行文件报错

报错信息

```
Cannot execute binary file: Exec format error
bash：./xxx：无法执行二进制文件
```

可能原因：

9.4.1 文件权限问题

可执行的文件权限的问题，不是可执行权限的文件无法执行，那么修改一下文件的权限就好，一般用指令：

```
chmod 777 后面加可执行文件的名字
```

9.4.2 系统版本问题

系统版本问题，例如可能那个可执行文件是在 64 位的系统上编译的，放到 32 位的系统下无法运行，可以自己拿源码在 32 位的系统编译好，得到适用于自己系统的可执行文件即可。

9.4.3 编译没链接

可能在编译的时候多加了个 c 参数，表示只编译不链接，从而无法执行，去掉 c 以后再编译就可以了：

使用错误的命令，如 gcc -c hello.c -o hello，这样得到的是.o 文件，而不是可执行程序，此时使用./hello 就会报上述错误！此时应改用 gcc hello.c -o hello 就可以啦。

```
MY@IMX6ULL:~/program/chr$ ls
11_key  13_irq  1_chrdevbase  2_led  3_newchrled  4_dtsled  5_gpioled  hello.c
MY@IMX6ULL:~/program/chr$ gcc -c hello.c -o hello
MY@IMX6ULL:~/program/chr$ ls
11_key  13_irq  1_chrdevbase  2_led  3_newchrled  4_dtsled  5_gpioled  hello  hello.c
MY@IMX6ULL:~/program/chr$ chmod 777 hello
MY@IMX6ULL:~/program/chr$ ./hello
bash: ./hello: cannot execute binary file: Exec format error
MY@IMX6ULL:~/program/chr$ gcc hello.c -o hello
MY@IMX6ULL:~/program/chr$ ./hello
hello world
MY@IMX6ULL:~/program/chr$
```

图 9.4.3-1 报错信息

9.5 文件系统配置问题

9.5.1 无法保存文件系统/etc/resolv.conf 配置

修改文件系统/etc/resolv.conf 文件，添加域名地址以后，保存退出，重启开发板以后再查看之前修改的/etc/resolv.conf 文件，发现之前修改的配置不见了，如何解决呢？可以在开机自启动文件中用 echo 指令来将域名地址添加到文件里，可以在文件系统的/etc/rc.local 里或者/etc/init.d/rcS 里加上下面的指令：

```
echo "nameserver 114.114.114.114">>/etc/resolv.conf
echo "nameserver 8.8.8.8">>/etc/resolv.conf
```

```
#!/bin/sh -e
#
# rc.local
#
# This script is executed at the end of each multiuser runlevel.
# Make sure that the script will "exit 0" on success or any other
# value on error.
#
# In order to enable or disable this script just change the execution
# bits.
#
# By default this script does nothing.


echo 30000 >  /proc/sys/vm/min_free_kbytes
echo "0" > /sys/class/graphics/fb0/blank
source /etc/profile
/opt/qt5.5.1/apps/QDesktop/QDesktop >/dev/null 2>&1 &

echo "nameserver 114.114.114.114">>/etc/resolv.conf
echo "nameserver 8.8.8.8">>/etc/resolv.conf

exit 0

- /etc/rc.local 1/24 4%
```

图 9.5.1-1 添加指令

上面两句就是将 nameserver 114.114.114.114 以及 nameserver 8.8.8.8 这两句话追加到文件/etc/resolv.conf 中，修改好以后，保存退出，重启开发板，可以看到文件/etc/resolv.conf 里多了这两句话，这样就不会修改完重启了这些配置就丢失了。

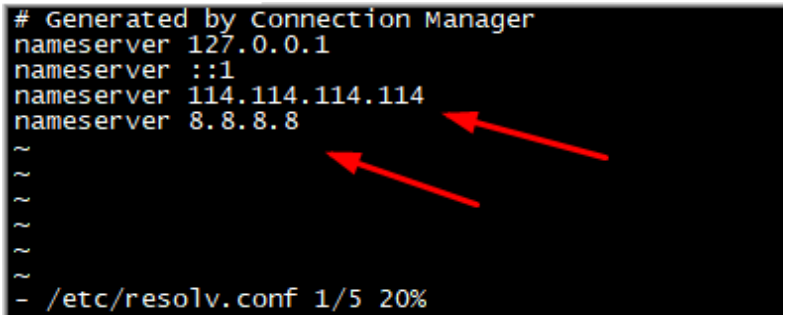


图 9.5.1-2 检查

9.5.2 proc/device-tree/: No such file or directory

报错信息：

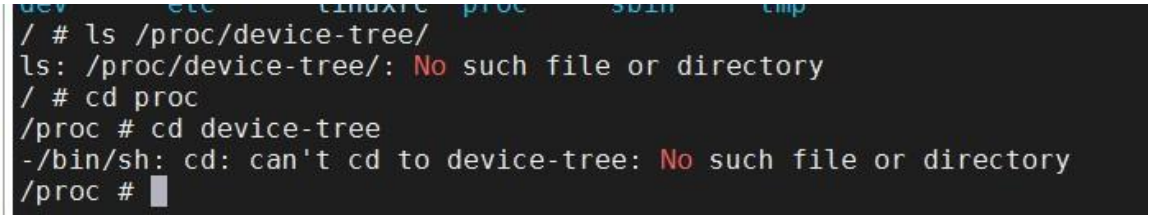


图 9.5.2-1 报错信息

解决思路：这种类似的问题经常出现在用户自己做的根文件系统上，根文件系统的/etc/fstab 写错。

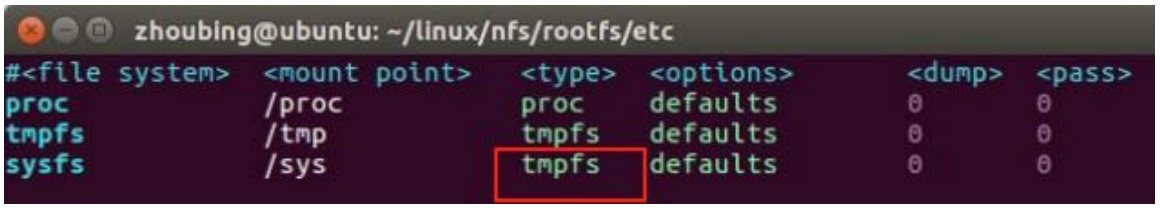


图 9.5.2-2 找到错误

应该按教程改成这样：

按照上述格式，在 fstab 文件中输入如下内容：

示例代码 38.4.2.1 /etc/fstab 文件

```
1 #<file system> <mount point>   <type>  <options>  <dump>  <pass>
2 proc            /proc          proc    defaults   0       0
3 tmpfs           /tmp           tmpfs   defaults   0       0
4 sysfs           /sys           sysfs   defaults   0       0
```

fstab 文件创建完成以后重新启动 Linux，结果如图 38.4.2.1 所示：

图 9.5.2-3 修改方法

9.5.3 mount: mounting ~ on failed: No such file or directory

操作：按照教程制作 busybox 根文件系统，最后挂载时报错：

mount: mounting ~ on failed: No such file or directory

```
[    8.978613] devtmpfs: mounted
[    8.982288] Freeing unused kernel memory: 524K (80b45000 - 80bc8000)
mount: mounting ~ on  failed: No such file or directory

Please press Enter to activate this console. Hello World!
Hello World!
```

图 9.5.3-1 报错信息

原因：fstab 文件多多写了个~符号

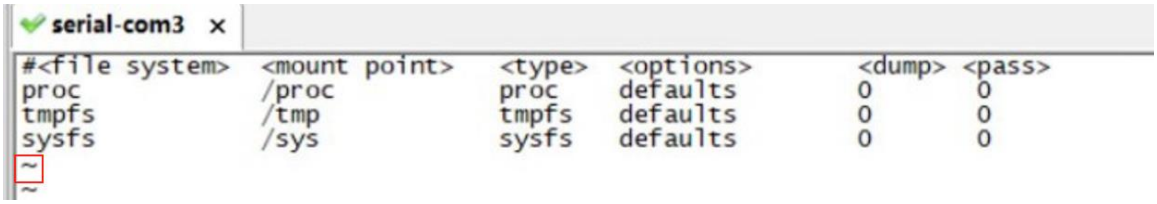


图 9.5.3-2 查找错误

这个在串口终端可能看不出，需要在 Ubuntu 下查看，或者用对比工具查看。

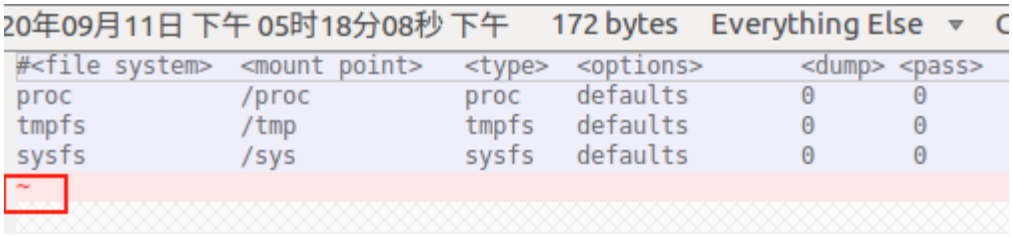


图 9.5.3-3 找到错误

9.5.4 教程系统开机自动配置声卡失败

报错信息：

```
Freeing unused kernel memory: 404K (809a1000 - 80a06000)
IPv6: eth0: IPv6 duplicate address fe80::204:9fff:fe04:d235 detected!
ALSA: Restoring mixer setting......

Please press Enter to activate this console. /sbin/alsactl: state_lock:125: file /
var/lib/alsa/asound.state lock error: No such file or directory
/sbin/alsactl: load_state:1683: Cannot open /var/lib/alsa/asound.state for reading
: No such file or directory
/sbin/alsactl: parse:1669: Unable to open file '/home/czx/linux/IMX6ULL/tool/alsa-
utils/share/alsa/init/00main': No such file or directory

/ #
```

图 9.5.4-1 报错信息

解决：

先按照开发指南 65.6 小节使用 amixer 设置声卡。

如果出现报错 alsactl: state_lock:125: file /var/lib/alsa/asound.state lock error: No such file or directory，则按照常见问题汇总 V1.1 的 7.15 小节办法设置声卡。

按照 7.15 小节的方法设置后，如果使用 alsactl -f asound.state store 命令保存配置信息成功，此时打开/etc/init.d/rcS 文件，在最后面追加如下内容：

```
if [ -f "/var/lib/alsa/asound.state" ]; then
        echo "ALSA: Restoring mixer setting....."
        /sbin/alsactl -f asound.state restore &
fi
```

注意第 3 行和开发指南教程上的有所区别。

保存文件，重启开发板，开发板会自动设置声卡。

```
Freeing unused kernel memory: 404K (809a1000 - 80a06000)
ALSA: Restoring mixer setting......

Please press Enter to activate this console. IPv6: eth0: IPv6 duplicate address fe
80::204:9fff:fe04:d235 detected!

/ #
/ #
```

图 9.5.4-2 验证

9.5.5 设置 Ubuntu-base 开机自启动

首先按教程做好一个 Ubuntu-base 系统烧录到开发板中（或者挂载）。

执行以下指令修改 getty@.service 文件。

```
sudo vi /lib/systemd/system/getty@.service
```

屏蔽掉文件中原先的 ExecStart，添加如下内容。

```
[Service]
ExecStart=
ExecStart=-/sbin/agetty -a root --noclear %I $TERM
//root 可以换成任何用户
```

```
# unit.
ConditionPathExists=/dev/tty0

[Service]
# the VT is cleared by TTYVTDisallocate
#ExecStart=-/sbin/agetty --noclear %I $TERM
[Service]
ExecStart=
ExecStart=-/sbin/agetty -a root --noclear %I $TERM

Type=idle
Restart=always
```

屏蔽掉原先的ExecStart

添加内容

图 9.5.5-1 屏蔽和添加

保存文件，重启开发板即可自动登陆 root 用户。

9.5.6 Ubuntu-base 没法用 sudo

报错信息：

```
chenzx@alientek_imx6ul:~$
chenzx@alientek_imx6ul:~$ sudo
sudo: unable to stat /etc/sudoers: Permission denied
sudo: no valid sudoers sources found, quitting
sudo: unable to initialize policy plugin
chenzx@alientek_imx6ul:~$ sudo fdisk -l
sudo: unable to stat /etc/sudoers: Permission denied
sudo: no valid sudoers sources found, quitting
sudo: unable to initialize policy plugin
```

图 9.5.6-1 报错信息

已经按照教程配置了/etc/sudoers，在虚拟机下也不行。

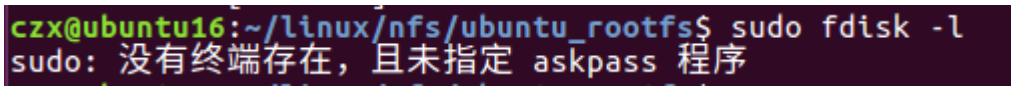


图 9.5.6-2 报错信息

原因：Ubuntu-base 文件系统根目录没有权限

解决：在串口终端进入 Ubuntu-base 根目录，赋予根目录权限。chmod 0755 /

关于虚拟机不能 sudo，重启虚拟机即可。

9.5.7 sudo: no tty present and no askpass program specified

按照教程制作 Ubuntu-base 文件系统，在执行完./mount.sh 后报错：

```
sudo: no tty present and no askpass program specified
```

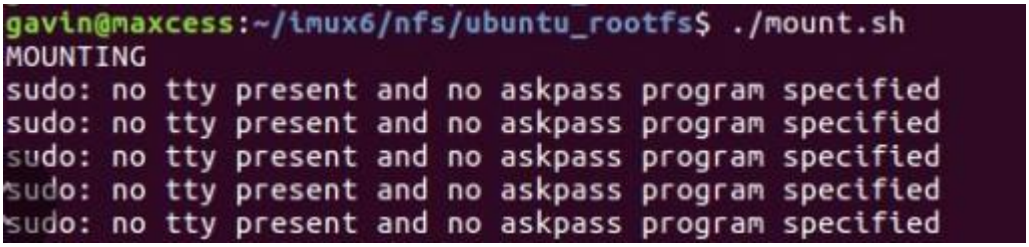


图 9.5.7-1 报错信息

原因：帐号并没有开启免密码

解决方法：

首先检查脚本有没有写错，没错就按下面修改。假设当前帐号为 abc，切换到 root 下，打开 sudoers。

```
vi /etc/sudoers
```

添加免密码。

```
abc ALL = NOPASSWD: ALL
```

9.5.8 Starting sshd:/var/empty/sshd must be owned by root and not group or world-writable

连接 ssh 时连接不上，报错：

```
Starting sshd:/var/empty/sshd must be owned by root and not group or world-writable.
```

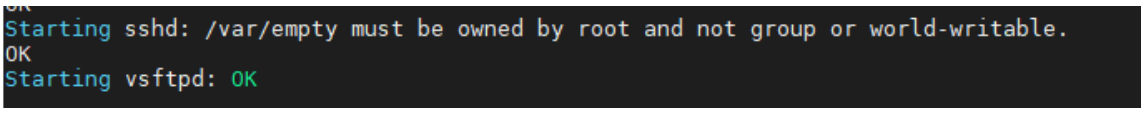


图 9.5.8-1 报错信息

解决方法：

```
cd /var/empty
mkdir sshd
```

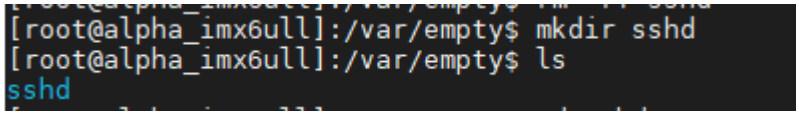


图 9.5.8-2 创建 sshd 文件

```
chmod 644 /var/empty/sshd
```

```
[root@alpha_imx6ull]:~$ chmod 644 /var/empty/sshd
[root@alpha_imx6ull]:~$ ls /var/empty/ -l
total 4
drw-r--r--    2 root     root          4096 Jan 10  2021 sshd
[root@alpha_imx6ull]:~$
```

图 9.5.8-3 赋予权限

/usr/sbin/sshd reasrt

```
[root@alpha_imx6ull]:/$ /usr/sbin/sshd restart
Extra argument restart.
[root@alpha_imx6ull]:/$
```

图 9.5.8-4 重启

复位重启。sshd 变 ok。

```
Starting postgresql: su: can't change directory to '/var/lib/pgsql'
pg_ctl: could not open PID file "/var/lib/pgsql/postmaster.pid": Permission denied
OK
Starting sshd: OK
Starting vsftpd: OK
```

图 9.5.8-5 完成

9.6 打包根文件系统问题

9.6.1 错误的打包方式

有些用户在学习打包文件系统的时候，会习惯在文件系统外面直接压缩 rootfs。这样会导致系统无法运行，因为烧写的时候，烧写脚本里是直接将打包好的文件系统解压的，如下图所示：

```
<!-- burn rootfs -->
<CMD state="Updater" type="push" body="pipe tar -jxv -C /mnt/mmcblk1p2" file="files/filesystem/rootfs.tar.bz2" ifdev="MX6ULL">
Sending and writting rootfs</CMD>
<CMD state="Updater" type="push" body="frf">Finishing rootfs write</CMD>
<CMD state="Updater" type="push" body="send" file="files/modules/modules.tar.bz2" ifdev="MX6ULL">Sending Modules file</CMD>
<CMD state="Updater" type="push" body="$ mkdir -p /mnt/mmcblk1p2/lib/modules">Mkdir -p /mnt/mmcblk1p2/lib/modules</CMD>
<CMD state="Updater" type="push" body="$ tar jxf $FILE -C /mnt/mmcblk1p2/lib/modules/" ifdev="MX6ULL">tar Modules file</CMD>
<CMD state="Updater" type="push" body="$ sleep 1">delay</CMD>
<CMD state="Updater" type="push" body="$ sync">Sync...</CMD>
```

图 9.6.1-1 烧写脚本

如果是在文件系统外面直接打包 rootfs 文件夹，则烧写的时候解压出来的一级目录是 rootfs（bin 等这些文件夹在二级目录），而不是 bin、home 等等这些文件夹，因而无法启动系统。

常见的报错信息：

devtmpfs: error mounting -2

Freeing unused kernel memory: 376K (8079a000 - 807f8000)

Kernel panic - not syncing: No working init found. Try passing init= option to kernel. See Linux Documentation/init.txt for guidance.

---[end Kernel panic - not syncing: No working init found. Try passing init= option to kernel. See Linux Documentation/init.txt for guidance.

9.6.2 正确的打包方式

在虚拟机中进入 rootfs 文件系统根目录下，即文件系统顶层目录，如下图所示：

```
czx@ubuntu16:~/linux/nfs$ cd rootfs/
czx@ubuntu16:~/linux/nfs/rootfs$ ls
bin  dev  etc  lib  linuxrc  mnt  proc  root  sbin  sys  tmp  usr
```

图 9.6.2-1 文件系统根目录

在根文件系统的顶层目录打包根文件系统，执行以下指令：

```
sudo tar -cjf rootfs.tar.bz2 ./*
```

```
czx@ubuntu16:~/linux/nfs/rootfs$ ls
bin  dev  etc  lib  linuxrc  mnt  proc  root  sbin  sys  tmp  usr
czx@ubuntu16:~/linux/nfs/rootfs$ sudo tar -cjf rootfs.tar.bz2 ./*
[sudo] czx 的密码：
czx@ubuntu16:~/linux/nfs/rootfs$ ls
bin  etc  linuxrc  proc  rootfs.tar.bz2  sys  usr
dev  lib  mnt      root  sbin            tmp
```

图 9.6.2-2 在根目录下打包

如果之前已经在文件系统外面打包了 rootfs 了，就需要解压出来重新打包，具体步骤如下：

1、 创建一个 rootfs 目录，将 rootfs.tar 压缩包放到该目录下，进入该目录。

2、 执行 `sudo tar -xf rootfs.tar`，解压。

3、 执行 `sudo rm rootfs.tar`，删除掉压缩包。

4、 执行 `sudo tar -cjf rootfs.tar.bz2 ./*`，重新打包。

第十章 Qt 问题记录

10.1 Qt 项目构建编译报错

10.1.1 error while loading shared libraries: libgstapp-0.10.so.0: cannot open shared 或打不开帮助模块

图 10.1.1-1 报错信息

解决办法，安装下面两个库就好：

```
libgstreamer0.10-dev
libgstreamer-plugins-base0.10-dev
```

10.1.2 cannot find -lGL

```
cannot find -lGL
collect2:error:ld returned 1 exit status
```

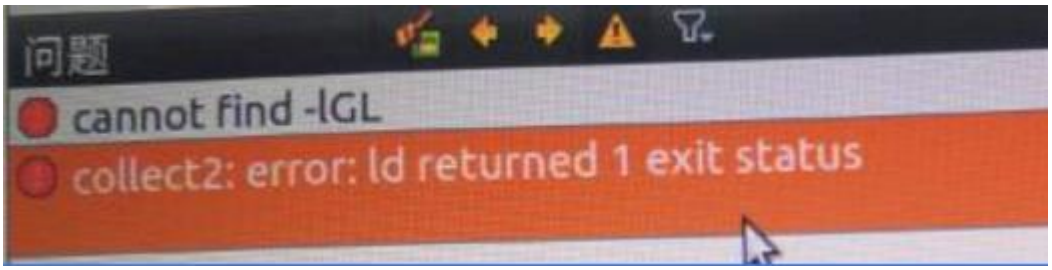


图 10.1.2-1 报错信息

解决办法，安装一个库：

```
sudo apt-get install libglu1-mesa-dev
```

10.1.3 qtcreator.sh:source:not found

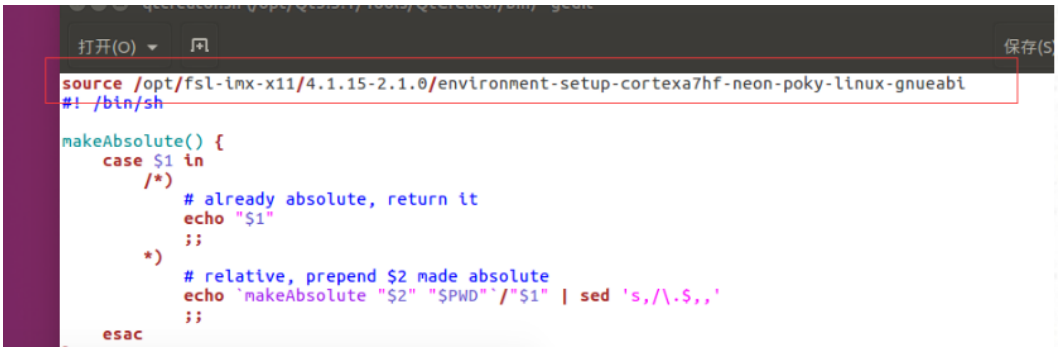


图 10.1.3-1 脚本内容

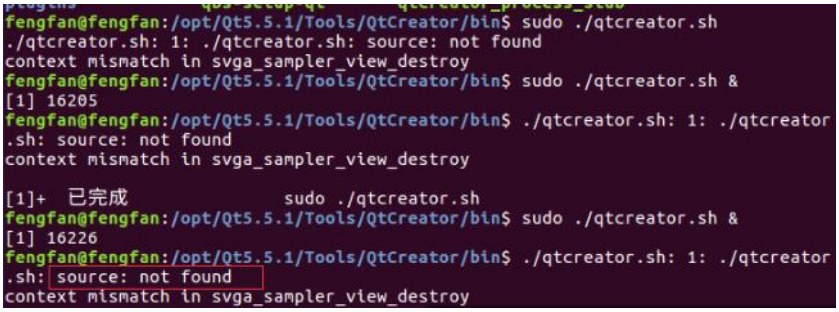


图 10.1.3-2 报错信息

有遇见用户自己搭建环境时在 root 下操作的，所以环境变量问题导致的，他切换到 root 用户下操作就解决了。

10.1.4 Cannot overwrite file /home/zdyz/.config/QtProject/qtcreator/devices.xml: Permission denied

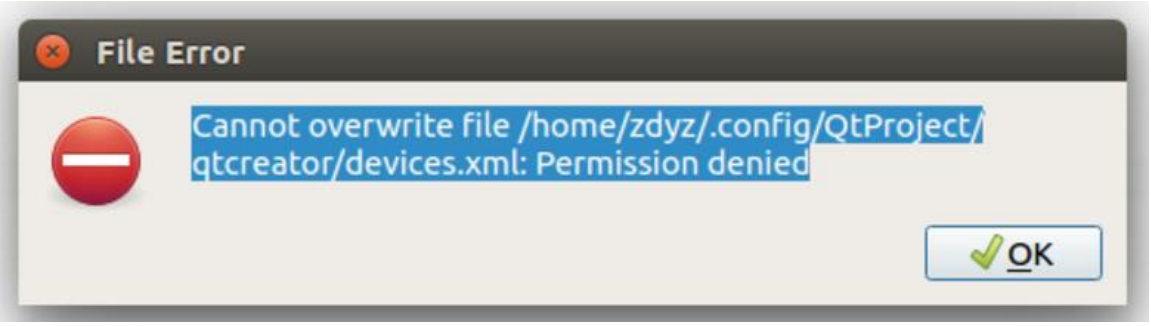


图 10.1.4-1 报错信息

权限问题，解决办法：

```
sudo chown -R zdyz:zdyz.config/
```

zdyz:zdyz 是我的当前的登录的用户名，改成你自己的用户名就可以了。

10.1.5 command not found

问题：构建 qt 项目时，报错 command not found

解决：

通过图标打开 qt，在构建 qt 项目时会报错。

需要通过脚本打开 qt，再构建 qt 项目。具体方法看 Qt 移植文档或者 Qt 环境搭建文档。

10.1.6 make: Nothing to be done for 'first'

问题：构建 QT 项目时，编译信息显示 make: Nothing to be done for 'first'

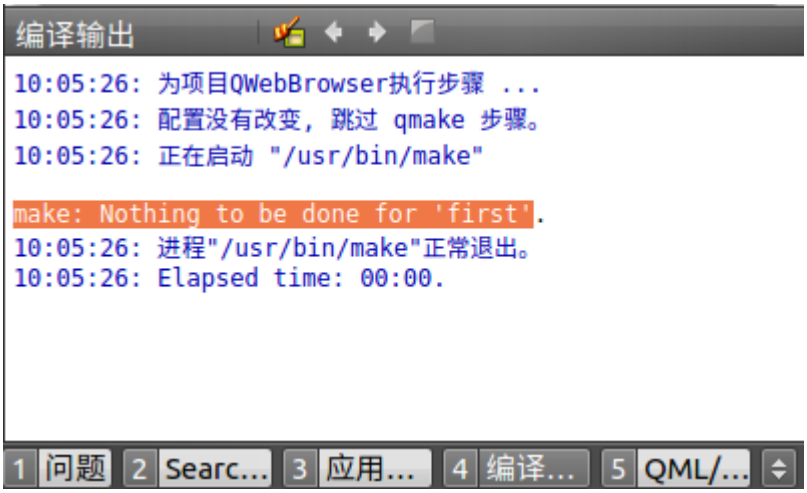


图 10.1.6-1 提示信息

原因：该错误原因是之前已经按照这个工程成功运行了多次，在没有任何修改下，程序“懒得”再运行了。可以在 CPP 文件中的随意空白位置添加个空格，或者随意做一些无关紧要的改动即可再次运行。

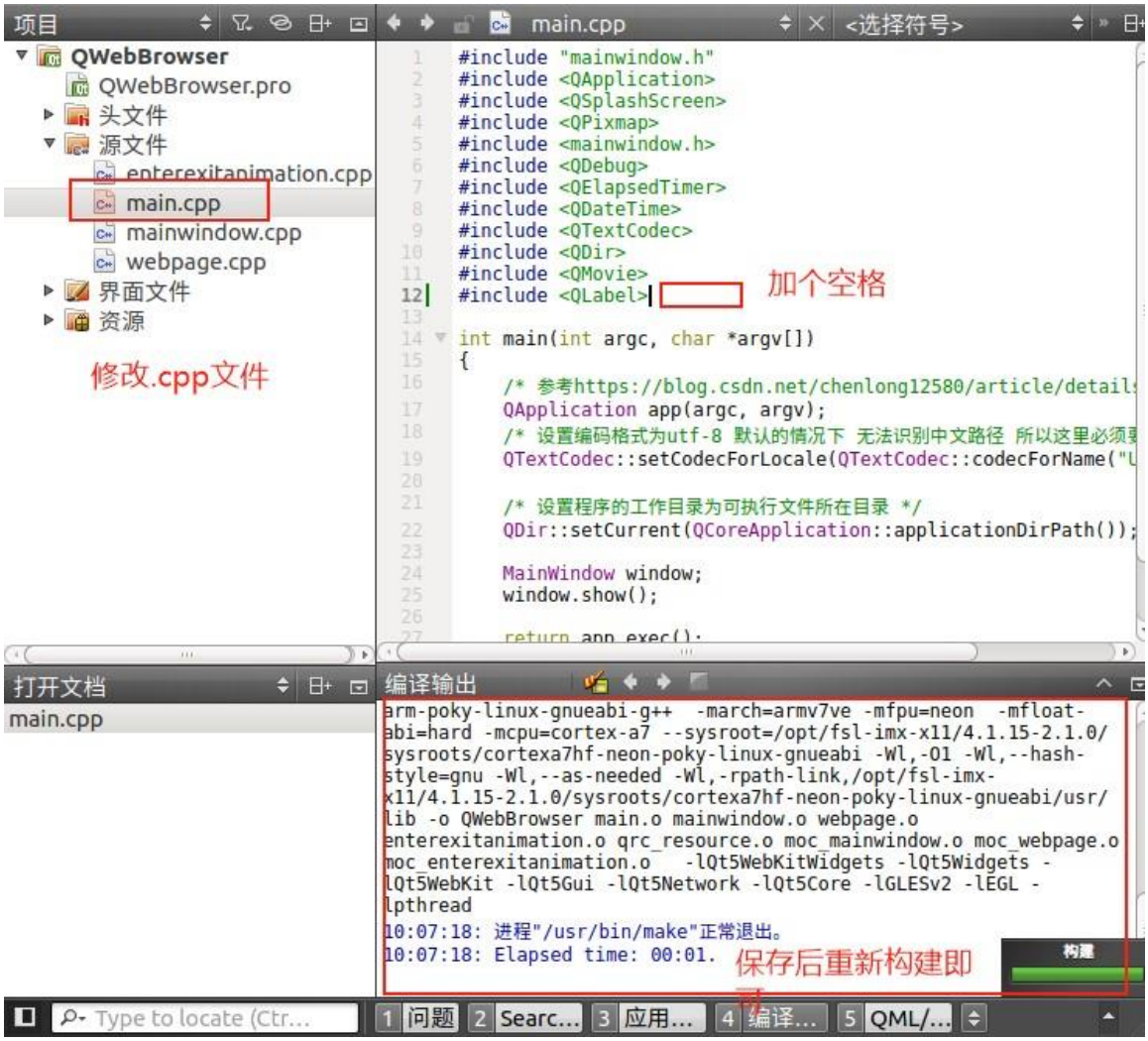


图 10.1.6-2 示例

10.1.7 execvp: /usr/local/arm/gcc-linaro-4.9.4-2017.01-x86_64_arm-linux-gnueabihf/bin: 权限不够

图 10.1.7-1 报错信息

解决方法 ： 使能交叉编译器

```
source  /usr/local/arm/gcc-linaro-4.9.4-2017.01-x86_64_arm-linux-gnueabihf/bin
```

10.2 运行 Qt 程序

10.2.1 运行 QT 程序提示 ts_open（）failed

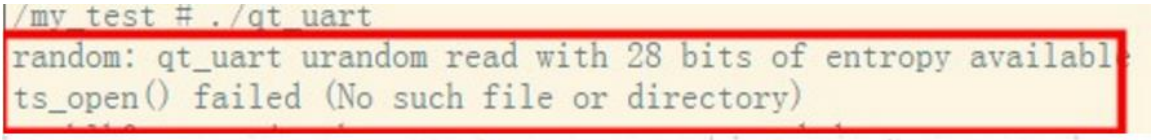


图 10.2.1-1 提示信息

出现 ts_open（）failed 信息可不用管它，这个是 tslib 的 debug 信息而已，其实程序是可以正常运行的。

10.2.2 libpng warning:iccp:known incorrect sRGB profile

运行 QT 程序提示 libpng warning:iccp:known incorrect sRGB profile，这个是 debug 信息是可以忽略的，程序依然可以正常运行。

10.2.3 error while loading shared libraries: libQt5WebKitWidgets.so.5: cannot open shared object file: No such file or directory

做 QT 编译环境搭建时，将 QWebBrowser 放到板子上执行遇到问题。

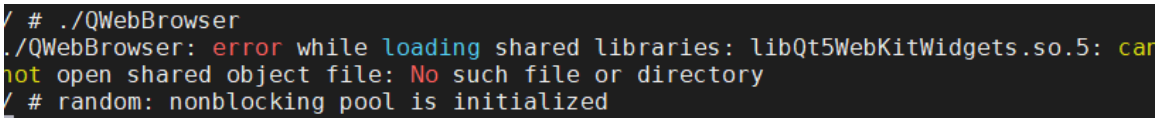


图 10.2.3-1 报错信息

解决：不能使用 busybox 根文件系统（busybox+移植的 Qt 不支持网络和音频等），需要使用出厂系统。

10.3 Qt 移植问题

10.3.1 报错/bin/sh: 1: python: not found

图 10.3.1-1 报错信息

解决办法，安装一个库：

```
sudo apt-get install swig python-dev python3-dev
```

10.3.2 在移植 QT 的库的时候，执行指令 ./autoconfigure.sh 报错 project.o:error adding symbols:File in wrong format

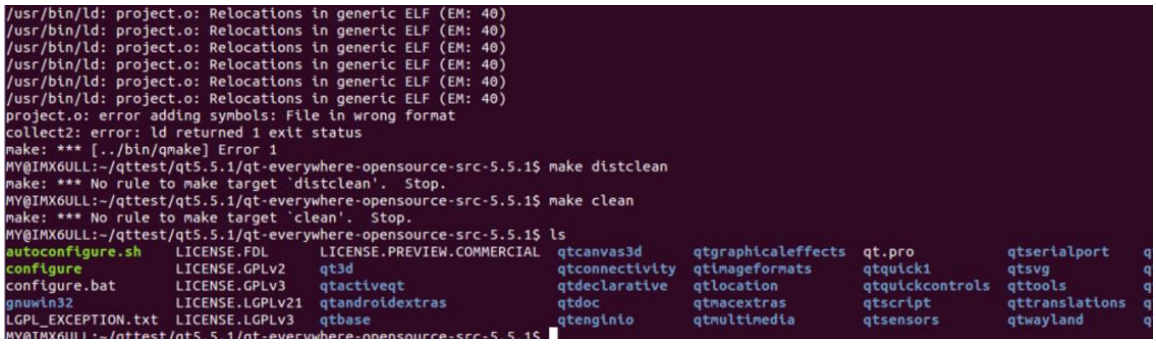


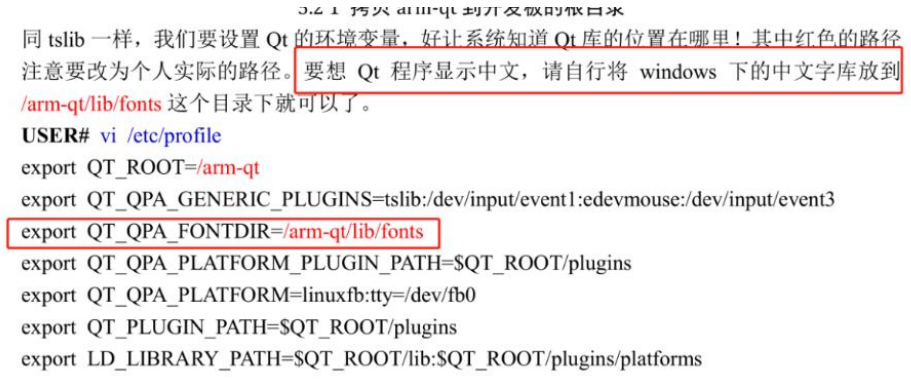
图 10.3.2-1 报错信息

解决办法：看提示说 project.o 这个文件格式有误了，用指令 find -name “project.o”搜索出这个文件所在的目录，然后进入那个目录下指令指令 make distclen 来清理之前的工程，再回到上图这个目录执行 ./autoconfigure.sh 重新配置，如下图，清理后重新配置不再报错：

图 10.3.2-2 解决

10.3.3 QT 设置字库在哪里设置?

QT 移植的文档里，字库就在/usr/share/fonts 这个目录下，您也可以根据自己的字库目录来配置：


同 tslib 一样，我们要设置 Qt 的环境变量，好让系统知道 Qt 库的位置在哪里！其中红色的路径注意要改为个人实际的路径。要想 Qt 程序显示中文，请自行将 windows 下的中文字库放到/arm-qt/lib/fonts 这个目录下就可以了。

```
USER# vi /etc/profile
export QT_ROOT=/arm-qt
export QT_QPA_GENERIC_PLUGINS=tslib:/dev/input/event1:edevmouse:/dev/input/event3
export QT_QPA_FONTDIR=/arm-qt/lib/fonts
export QT_QPA_PLATFORM_PLUGIN_PATH=$QT_ROOT/plugins
export QT_QPA_PLATFORM=linuxfb:tty=/dev/fb0
export QT_PLUGIN_PATH=$QT_ROOT/plugins
export LD_LIBRARY_PATH=$QT_ROOT/lib:$QT_ROOT/plugins/platforms
```

图 10.3.3-1 教程

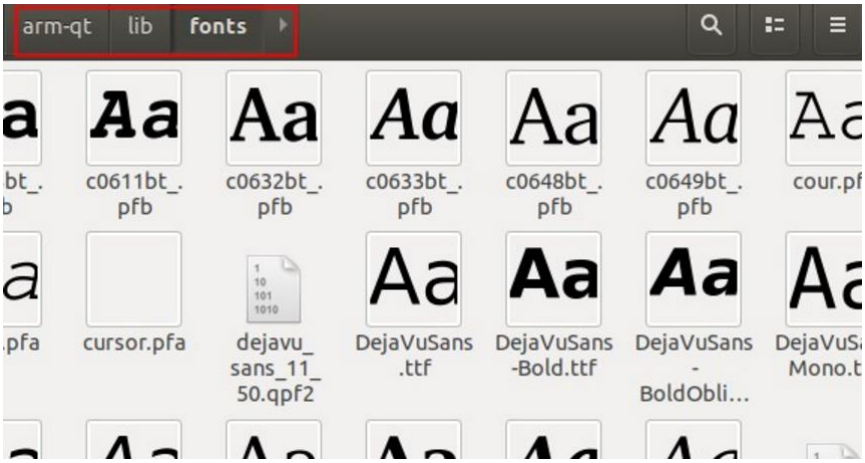


图 10.3.3-2 字库路径

10.3.4 QT 移植过程中出现 QSQLITE driver not loaded

参考论坛帖子：
http://www.openedv.com/forum.php?mod=viewthread&tid=311997&extra=page%3D1
根据帖子的来改，然后按照《【正点原子】I.MX6U Qt 移植》文档来进行操作。

10.3.5 编译 Qt 报错 Cannot run target compiler 'arm-linux-gnueabi-g++'

问题：编译 Qt5.12.9 报错 Cannot run target compiler 'arm-linux-gnueabi-g++'

```
alientek@ubuntu16:~/qt-everywhere-src-5.12.9$ source /etc/profile
alientek@ubuntu16:~/qt-everywhere-src-5.12.9$ ./autoconfigure.sh
+ cd qtbase
+ /home/alientek/qt-everywhere-src-5.12.9/qtbase/configure -top-level -prefix /home/alien
tek/qt-everywhere-src-5.12.9/arm-qt -opensource -confirm-license -release -strip -shared
-xplatform linux-arm-gnueabi-g++ -optimized-qmake -c++std c++11 --rpath=no -pch -skip qt3
d -skip qtactiveqt -skip qtandroidextras -skip qtcanvas3d -skip qtconnectivity -skip qtda
tavis3d -skip qtdoc -skip qtgamepad -skip qtlocation -skip qtmacextras -skip qtnetworkaut
h -skip qtpurchasing -skip qtremoteobjects -skip qtscript -skip qtscxml -skip qtsensors -
skip qtspeech -skip qtsvg -skip qttools -skip qttranslations -skip qtwayland -skip qtwebe
ngine -skip qtwebview -skip qtwinextras -skip qtx11extras -skip qtxmlpatterns -make libs
-make examples -nomake tools -nomake tests -gui -widgets -dbus-runtime --glib=no --iconv=
no --pcre=qt --zlib=qt -no-openssl --freetype=qt --harfbuzz=qt -no-opengl -linuxfb --xcb=
no -tslib --libpng=qt --libjpeg=qt --sqlite=qt -plugin-sql-sqlite -I/home/alientek/tslib-
1.21/arm-tslib/include -L/home/alientek/tslib-1.21/arm-tslib/lib -recheck-all
Creating qmake...
.Done.
Project ERROR: Cannot run target compiler 'arm-linux-gnueabi-g++'. Output:
====================
====================
Maybe you forgot to setup the environment?
```

图 10.3.5-1 报错信息

解决方法：检查 qtbase/mkspecs/linux-arm-gnueabi-g++/qmake.conf 文件配置错误。常见的问题如下图所示，注意 hf 问题。

```
# modifications to g++.conf
QMAKE_CC                = arm-linux-gnueabihf-gcc
QMAKE_CXX               = arm-linux-gnueabihf-g++
QMAKE_LINK              = arm-linux-gnueabihf-g++
QMAKE_LINK_SHLIB        = arm-linux-gnueabihf-g++

# modifications to linux.conf
QMAKE_AR                = arm-linux-gnueabihf-ar cqs
QMAKE_OBJCOPY           = arm-linux-gnueabihf-objcopy
QMAKE_NM                = arm-linux-gnueabihf-nm -P
QMAKE_STRIP             = arm-linux-gnueabihf-strip
load(qt_config)
```

图 10.3.5-2 注意此处有 hf-

10.3.6 编译 qt 找不到 C++

Ubuntu 20 或者新的 ubuntu 可能会出现这个问题，在移植 QT 时，说没有 C++环境。

```
  Multi-touch support ..... yes
Qt Multimedia:
  ALSA ..... no
  GStreamer 1.0 ..... no
  GStreamer 0.10 ..... no
  Video for Linux ..... yes
  OpenAL ..... no
  PulseAudio ..... no
  Resource Policy (libresourceqt5) ..... no
  Windows Audio Services ..... no
  DirectShow ..... no
  Windows Media Foundation ..... no

Note: Also available for Linux: linux-clang linux-icc

Note: -optimized-tools is not useful in -release mode.

WARNING: Cross compiling without sysroot. Disabling pkg-config

ERROR: C++11 <random> is required and is missing or failed to compile.

Check config.log for details.
yuwei@yuwei-virtual-machine:~/qt-everywhere-src-5.12.9$
```

图 10.3.6-1 报错信息

打开 qmake.conf，修改以下内容：

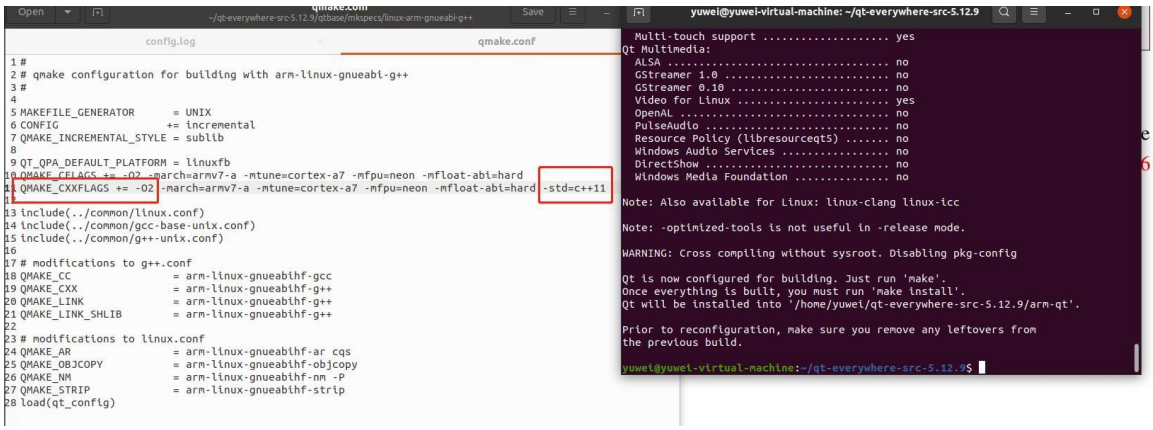


图 10.3.6-2 修改

10.4 开机进度条实验报错

10.4.1 ./autogen.sh: 9: ./autogen.sh: aclocal: not found

类似报错：

```
./autogen.sh: 9: ./autogen.sh: aclocal: not found
./autogen.sh: 10: ./autogen.sh: autoheader: not found
./autogen.sh: 11: ./autogen.sh: automake: not found
./autogen.sh: 12: ./autogen.sh: autoconf: not found
```

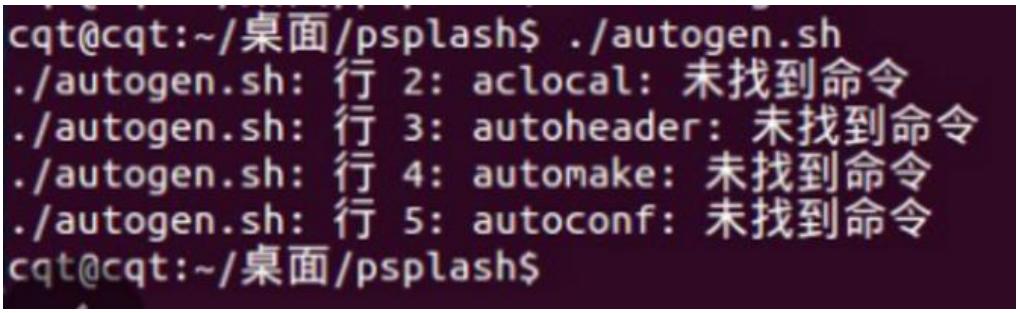


图 10.4.1-1 报错信息

解决办法，安装一个库：

```
sudo apt-get install automake
```

10.4.2 configure.ac:7: installing './compile'

类似报错：

```
configure.ac:7: installing './compile'
configure.ac:3: installing './install-sh'
configure.ac:3: installing './missing'
Makefile.am: installing './INSTALL'
Makefile.am: installing './depcomp'
```

原因：因为没执行这两个指令

```
./make-image-header.sh alientek.png POKY
```

./make-image-header.sh psplash-bar.png BAR

```
MY@IMX6ULL:~/kehu/pa/psplash$ chmod 777 autogen.sh
MY@IMX6ULL:~/kehu/pa/psplash$ ./autogen.sh
configure.ac:7: installing './compile'
configure.ac:3: installing './install-sh'
configure.ac:3: installing './missing'
Makefile.am: installing './INSTALL'
Makefile.am: installing './depcomp'
MY@IMX6ULL:~/kehu/pa/psplash$  ./make-image-header.sh alientek.png POKY
bash: ./make-image-header.sh alientek.png POKY: No such file or directory
MY@IMX6ULL:~/kehu/pa/psplash$ ./make-image-header.sh alientek.png POKY
MY@IMX6ULL:~/kehu/pa/psplash$ vi alientek-img.h
MY@IMX6ULL:~/kehu/pa/psplash$ ./make-image-header.sh psplash-bar.png BAR
MY@IMX6ULL:~/kehu/pa/psplash$ ls
aclocal.m4       autom4te.cache  configure     install-sh           NEWS               psplash-co
alientek-img.h   base-images     configure.ac  Makefile.am          psplash-bar-img.h  psplash-co
alientek.png     ChangeLog       COPYING       Makefile.in          psplash-bar.png    psplash-co
AUTHORS          compile         depcomp       make-image-header.sh psplash.c          psplash.do
autogen.sh       config.h.in     INSTALL       missing              psplash-colors.h   psplash-fb
MY@IMX6ULL:~/kehu/pa/psplash$ vi psplash-bar-img.h
MY@IMX6ULL:~/kehu/pa/psplash$ vi psplash.c
MY@IMX6ULL:~/kehu/pa/psplash$ ./autogen.sh
MY@IMX6ULL:~/kehu/pa/psplash$ pwd
```

图 10.4.2-1 执行指令

第十一章 一些常见问题（论坛有帖子）

11.1 开发板启动后打印 random: nonblocking pool is initialized，如何去掉这句话

参考论坛帖子：
http://www.openedv.com/forum.php?mod=viewthread&tid=314677&extra=

11.2 如何在内核中开启时间戳，如何开启启动打印时间

参考论坛帖子：
http://www.openedv.com/forum.php?mod=viewthread&tid=314674&extra=

11.3 如何将驱动模块静态编译到 Linux 内核

参考论坛帖子：
http://www.openedv.com/forum.php?mod=viewthread&tid=307986&extra=

11.4 阿尔法开发板（I.MX6ULL）修改调试串口

参考论坛帖子：
http://www.openedv.com/forum.php?mod=viewthread&tid=307440&extra=

11.5 为什么 bic sp, sp, #7 能够实现 8 字节对齐

参考论坛帖子:
http://www.openedv.com/forum.php?mod=viewthread&tid=305007&extra=

11.6 TFTP+NFS 环境搭建以及 TFTP 加载内核和设备数，NFS 挂载文件系统的方法（完整操作方法）

参考论坛帖子：
http://www.openedv.com/forum.php?mod=viewthread&tid=302907&extra=

11.7 裸机实验程序烧录到 TF 卡注意事项

参考论坛帖子：
http://www.openedv.com/forum.php?mod=viewthread&tid=302742&extra=

11.8 使用 mfgtool 上位机固化系统（OTG 方式）一些注意事项

参考论坛帖子：
http://www.openedv.com/forum.php?mod=viewthread&tid=302985&page=1&extra=#pid1025855

11.9 开发板文件系统自动登录 root 用户的方法

参考论坛帖子：
http://www.openedv.com/forum.php?mod=viewthread&tid=303142&extra=page%3D1

11.10 Ubuntu 开机时出现 file system with errors 怎么办

参考论坛帖子：
http://www.openedv.com/forum.php?mod=viewthread&tid=303235&page=1&extra=#pid1027559

11.11 开发板和电脑直连的设置方法（电脑不用 WIFI 上网）

参考视频：
https://beta.yuanzige.com/course/detail/50096
注意，如果虚拟机可以设置为桥接模式的话尽量设置为桥接模式。

11.12 开发板和电脑直连，电脑要用 WIFI（热点）上网

参考论坛帖子：
http://www.openedv.com/forum.php?mod=viewthread&tid=306536&extra=page%3D1

11.13 uboot 网络功能不能用，协商成功后出现 data abort.

这个是编译器版本的问题，解决办法是可以更换另一个版本的编译器
参考论坛帖子：
http://www.openedv.com/forum.php?mod=viewthread&tid=304604&extra=page%3D1

11.14 虚拟机目前提醒硬盘不够，如何添加一个磁盘

参考论坛帖子：
http://www.openedv.com/forum.php?mod=viewthread&tid=304537

11.15 如何直接使用正点原子修改后的 mfgtool 来烧录自己的 uboot、内核和设备树以及文件系统

参考论坛帖子：
http://www.openedv.com/forum.php?mod=viewthread&tid=305387&extra=page%3D2

11.16 裸机程序烧写到/dev/sda 了导致 ubuntu 起不来怎么办

参考论坛帖子：
http://www.openedv.com/thread-306631-1-1.html

11.17 NXP 官方 uboot 源码下载地址

官网链接：
https://source.codeaurora.org/external/imx/uboot-imx/
PS：可能以后这个链接会变。

11.18 NXP 官网内核源码下载链接

官网链接：
https://source.codeaurora.org/external/imx/linux-imx/refs/heads

PS：可能以后链接会变。

11.19 修改 Linux 内核开机启动 logo 方法

参考论坛帖子：
http://www.openedv.com/forum.php?mod=viewthread&tid=309374&extra=page%3D1

11.20 修改 uboot 开机 logo（以及将 logo 居中）

参考论坛帖子：
http://www.openedv.com/forum.php?mod=viewthread&tid=309886&highlight=logo

11.21 阿尔法开发板开机进度条怎么做的

参考论坛帖子：
http://www.openedv.com/forum.php?mod=viewthread&tid=306831&highlight=logo

11.22 ubuntu 登录后蓝屏修复方法

参考论坛帖子：
http://www.openedv.com/forum.php?mod=viewthread&tid=308838&extra=page%3D3

11.23 阿尔法开发板使用 USB 蓝牙分享

参考论坛帖子：
http://www.openedv.com/forum.php?mod=viewthread&tid=310531&extra=page%3D1

11.24 在 IMX6ULL 上使用 Qt sqlite（数据库）

参考论坛帖子：
http://www.openedv.com/forum.php?mod=viewthread&tid=311997&extra=page%3D1

11.25 ubuntu 如何更换源

参考网贴：
https://blog.csdn.net/xkwy100/article/details/80301156

11.26 VSCode 无法跳转到定义

参考网贴：
https://www.cnblogs.com/liuxianan/p/vscode-plugin-jump-completion-hover.html

11.27 阿尔法移植 OpenCV

参考论坛链接：
http://www.openedv.com/forum.php?mod=viewthread&tid=306103&extra=page%3D1

11.28 阿尔法移植 Debian 文件系统

参考论坛链接：

http://www.openedv.com/forum.php?mod=viewthread&tid=306211&extra=page%3D1

11.29 阿尔法旧的 QT 桌面程序

参考论坛链接：
http://www.openedv.com/forum.php?mod=viewthread&tid=307169&fromuid=195955

11.30 QT5.5.1 安装包下载地址

参考官网：
http://download.qt.io/new_archive/qt/5.5/5.5.1/

11.31 安装 samba 服务器

参考论坛链接：
http://www.openedv.com/forum.php?mod=viewthread&tid=320053&highlight=samba

11.32 使用 vsftpd 搭载 FTP 文件服务器

参考论坛链接：
http://www.openedv.com/forum.php?mod=viewthread&tid=303019&extra=page%3D1

11.33 一种对裸机烧写器改进的方法_imxdownload

参考论坛链接：
http://www.openedv.com/forum.php?mod=viewthread&tid=302808&extra=page%3D1

11.34 SecureCRT 注册机分享

参考论坛链接：
http://www.openedv.com/forum.php?mod=viewthread&tid=301350

11.35 虚拟机开机时出现 file system with errors

参考论坛链接：
http://www.openedv.com/forum.php?mod=viewthread&tid=303235&page=1&extra=#pid1027559

11.36 如何使用自己最新编译的 DTB 文件

参考论坛链接：
http://www.openedv.com/forum.php?mod=viewthread&tid=303491&page=1&extra=#pid1029414

11.37 vscode 在 ubuntu 的 terminal 中下划线不显示解决方案

提示有错，但实际上没有错。
https://www.cnblogs.com/huoqs/p/11833378.html

11.38 uboot 不支持 fatwrite 命令

参考论坛链接：
http://www.openedv.com/forum.php?mod=viewthread&tid=303418&highlight=fatwrite

11.39 VM TOOLS 安装

https://www.cnblogs.com/lixuejian/p/11758371.html

11.40 用 dpkg 来卸载 VSCode

用 dpkg 来安装就可以用 dpkg 来卸载。
https://blog.csdn.net/Jeffxu_lib/article/details/86606160

11.41 Uboot 终端命令行参数输入过长，无法完整输入

http://www.openedv.com/forum.php?mod=viewthread&tid=302206

11.42 UBOOT 网络功能不能用，协商成功后出现 data abort

这个是编译器版本的问题，解决办法是换编译器。
http://www.openedv.com/forum.php?mod=viewthread&tid=304604&extra=page%3D1

11.43 GT9147 裸机代码

http://www.openedv.com/forum.php?mod=viewthread&tid=304725&extra=page%3D1
http://www.openedv.com/forum.php?mod=viewthread&tid=304916&page=1&extra=#pid1040042
http://www.openedv.com/forum.php?mod=viewthread&tid=303882&extra=page%3D1
http://www.openedv.com/forum.php?mod=viewthread&tid=306559&page=1&extra=#pid1052576

11.44 gt9xx 触摸驱动调试

https://blog.csdn.net/qq_26943851/article/details/104417858

11.45 共享文件夹如何创建

https://www.cnblogs.com/huangjianxin/p/6343881.html?tdsourcetag=s_pctim_aiomsg

11.46 I2C 中假读 Dummy Read 的问题

https://stackoverflow.com/questions/49219086/i2c-what-is-a-dummy-read
https://community.nxp.com/thread/378405
这个链接里是说，该读取不是对 I2C 总线的真实读取，而是对 IC 模块中 IICD 寄存器的读取。

I2C 硬件模块包括一个状态机，该状态机将在每次写入或读取 lICD 寄存器时在总线上执行 IC 字节传输。当 I2C 地址值写入 IICD 寄存器时，发送第一个字节（12C 设备地址）。一旦将

其写出，则需要进行虚拟读取以启动下一个字节的传输。请注意，这是虚拟读取，因为 IICD 寄存器中还没有实际数据，因为硬件尚未真正完成传输。

11.47 NXP 官方 uboot 源码下载地址

https://source.codeaurora.org/external/imx/uboot-imx/
Linux 下载地址：
https://source.codeaurora.org/external/imx/linux-imx/refs/heads

11.48 阿尔法开发板（I.MX6ULL）修改调试串口

http://www.openedv.com/forum.php?mod=viewthread&tid=307440&extra=page%3D1

11.49 VSCode 学习链接

https://geek-docs.com/vscode/vscode-tutorials/vs-code-multi-root-workspace.html

11.50 IMX6ULL 开发板支持 TFT 系列小屏幕（SPI 接口的 ST7789 为例）

http://www.openedv.com/forum.php?mod=viewthread&tid=320312&extra=page%3D1

11.51 网口频繁 UP 和 DOWN

http://www.openedv.com/forum.php?mod=viewthread&tid=322548
http://www.openedv.com/forum.php?mod=viewthread&tid=307469

11.52 IMX6ULL eMMC 启动流程以及分区表分析

http://www.openedv.com/forum.php?mod=viewthread&tid=323802&extra=page%3D1

11.53 开发板多路串口 UART 配置

http://www.openedv.com/forum.php?mod=viewthread&tid=326944&extra=page%3D1

11.54 USB_OTG1 设置为串行下载后无法识别 USB

http://www.openedv.com/forum.php?mod=viewthread&tid=327011&extra=page%3D1

11.55 自制阿尔法开发板的 DHT11 驱动

http://www.openedv.com/forum.php?mod=viewthread&tid=307034&extra=page%3D2

11.56 最新移植的 ubuntu18 根文件系统用不了 sudo

http://www.openedv.com/forum.php?mod=viewthread&tid=326671&extra=page%3D2

11.57 阿尔法板子出厂系统输出 PWM 方波方法

http://www.openedv.com/forum.php?mod=viewthread&tid=321035&highlight=pwm