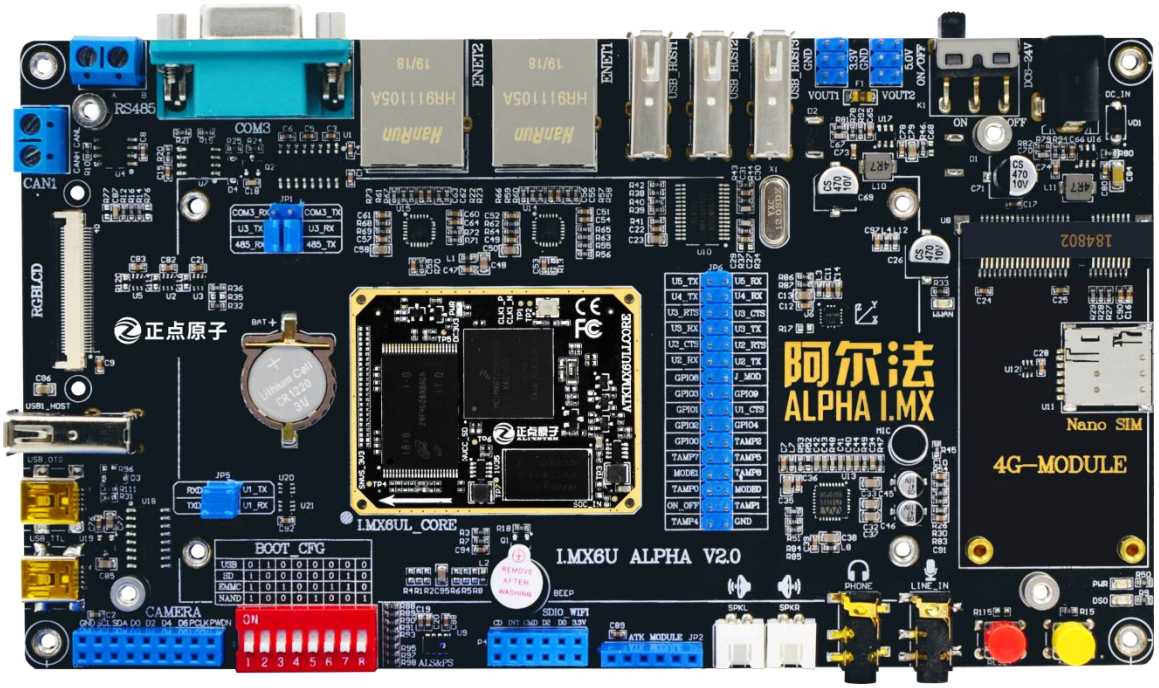
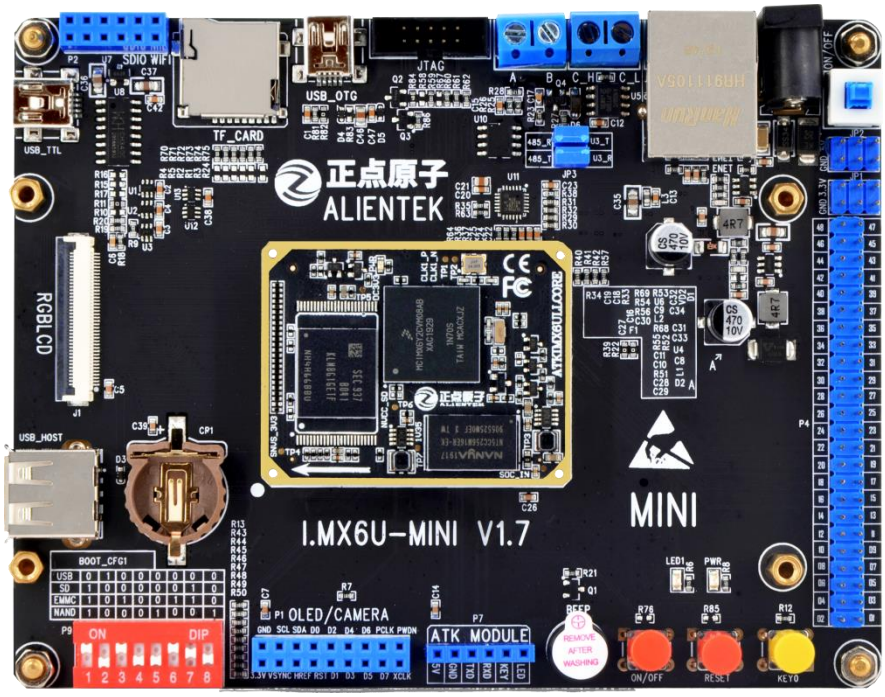
I.MX6U 开发板文件拷贝&固件分步更新手册 V1.2





正点原子广州市星翼电子科技有限公司

淘宝店铺 1：http://eboard.taobao.com

淘宝店铺 2：http://openedv.taobao.com

技术支持论坛 (开源电子网) ：www.openedv.com

原子哥在线教学：www.yuanzige.com

官方网站：www.alientek.com

最新资料下载链接：http://www.openedv.com/posts/list/13912.htm

E-mail: 389063473@qq.com **QQ:** 389063473

咨询电话：020-38271790

传真号码：020-36773971

团队：正点原子团队

正点原子，做最全面、最优秀的嵌入式开发平台软硬件供应商。

友 情 提 示

如果您想及时免费获取“正点原子”最新资料，敬请关注正点原子微信公众平台，我们将及时给您发布最新消息和重要资料。

关注方法：

(1)微信“扫一扫”，扫描右侧二维码，添加关注

(2)微信→添加朋友→公众号→输入“正点原子”→关注

(3)微信→添加朋友→输入“alientek_stm32” →关注

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	2020.1.7
V1.1	更正： 1.更正 2.2.1 小节中，uboot 向 emmc 的 boot 分区烧写的错误，由 mmcblk1boot0 改为/dev/mmcblk1boot0。	正点原子 linux 团队	正点原子 linux 团队	2020.7.2
V1.2	重新优化了排版、添加了分区表格	正点原子 linux 团队	正点原子 linux 团队	2020.11.23

目录

前言

正点原子 Linux 用户，在开发过程中，新手不知道怎么拷贝文件，所以就出了这份文档，可以说拷贝是新手必备的技能吧，后面还加了如何单步去更新开发板的固件。

这里必须注意：这里使用的是出厂的镜像，出厂的文件系统，编译的时候是使用正点原子修改过的源码。如果是教程的源码，请自行参考！

第一章 开发板拷贝篇

大家知道，在 Windows 这种图形界面的系统拷贝文件，直接用鼠标点击或者拖动文件就可以了。对的，在 Ubuntu 这种含图形界面的 Linux 系统也是可以这样做的。但是一般的 Linux 系统，除了别人去写图形界面，否它只有一个黑色的终端呈现在用户面前。“几乎”让人摸不着头脑！下面就介绍我们在学习或者开发时经常用到的文件拷贝方法。传文件的方法有很多种，下面介绍以下常用的几种。

1.1 Linux 下通过 U 盘或者 SD 卡拷贝文件

1.1.1 Linux 开发板通过 U 盘或者 SD 卡拷贝文件

准备一个 FAT32 格式的 U 盘，注意这里不能用 NTFS 格式的 U 盘！

开发板上电，打开串口终端，如果不知道怎么打开串口终端的用户，请看【正点原子】I.MX6U 用户快速体验 V1.x.pdf 里面第二章 ALIENTEK I.MX6U 使用前准备的内容，安装串口软件。

如下准备一个 U 盘（FAT32 格式，不能用 NTFS 格式的）如下图，拍个照片给大伙看看。(可用 Micro SD 卡（TF 卡），用读卡器插上也是可以的)。

图 1.1.1-1 FAT32 格式的 U 盘

开发板上电启动系统后，在串口终端里输入指令 df 查看当前挂载的内容。

```
df
```

```
root@ATK-IMX6U:~# df
Filesystem     1K-blocks    Used Available Use% Mounted on
/dev/root        7244864  441184   6429000   7% /
devtmpfs          187640      76    187564   1% /dev
tmpfs                 40       0        40   0% /mnt/.psplash
tmpfs             253436     180    253256   1% /run
tmpfs             253436     168    253268   1% /var/volatile
/dev/mmcblk1p1    129039    6897    122143   6% /run/media/mmcblk1p1
root@ATK-IMX6U:~#
```

图 1.1.1-2 插 U 盘前的挂载内容

插上 U 盘如下图所示。

图 1.1.1-3 开发板 USB 接口接入 U 盘

开发板串口终端打印信息如下，可以看到 U 盘的实际大小、格式与 U 盘的节点（sda）等一些信息。

```
root@ATK-IMX6U:~# [ 2059.332601] usb 2-1.1: new high-speed USB device number 3 using ci_hdrc
[ 2059.449595] usb-storage 2-1.1:1.0: USB Mass Storage device detected
[ 2059.459416] scsi host0: usb-storage 2-1.1:1.0
[ 2060.464182] scsi 0:0:0:0: Direct-Access     Generic  STORAGE DEVICE   1532 PQ: 0 ANSI: 6
[ 2060.773382] sd 0:0:0:0: [sda] 31116288 512-byte logical blocks: (15.9 GB/14.8 GiB)
[ 2060.781852] sd 0:0:0:0: [sda] Write Protect is off
[ 2060.788509] sd 0:0:0:0: [sda] Write cache: disabled, read cache: enabled, doesn't support DPO or FUA
[ 2060.818167]  sda: sda1
[ 2060.833169] sd 0:0:0:0: [sda] Attached SCSI removable disk
```

图 1.1.1-4 识别到 U 盘的信息

再使用 df 指令查看挂载的节点，可以看到/dev/sda1 挂载的目录为/run/media/sda1。因为我的 U 盘只有一个分区，所以只挂载了一个 sda1 分区，如果你的 SD 卡有两个分区，挂载的目录有两个分别是 sda1 或者 sda2。如果你还有别一个 U 盘，再插到开发板，它的挂载节点会是 sdb*了，以此类推。

```
df
```

```
root@ATK-IMX6U:~# df
Filesystem     1K-blocks    Used Available Use% Mounted on
/dev/root        7244864  441184   6429000   7% /
devtmpfs          187640      76    187564   1% /dev
tmpfs                 40       0        40   0% /mnt/.psplash
tmpfs             253436     192    253244   1% /run
tmpfs             253436     172    253264   1% /var/volatile
/dev/mmcblk1p1    129039    6897    122143   6% /run/media/mmcblk1p1
/dev/sda1       15541408      32  15541376   1% /run/media/sda1
root@ATK-IMX6U:~#
```

图 1.1.1-5 U 盘挂载节点和挂载目录等信息

所以我们只要进入/run/media/sda1 下去拷贝文件就可以了，直接用 cd 指令进入/run/media/sda1 目录下拷贝文件，使用 ls 指令可查看 U 盘里的内容。

在下图中，已经进入了/run/media/sda1 目录下。可以看到我的 U 盘下有很多文件，其中有一些有“？？？？”的文件，是因为文件系统字符终端不显示中文。

```
cd /run/media/sda1
ls
```

```
root@ATK-IMX6U:~# cd /run/media/sda1
root@ATK-IMX6U:/run/media/sda1# ls
'????.txt'   Makefile   'System Volume Information'   can-utils-2020.02.04.tar.gz
root@ATK-IMX6U:/run/media/sda1#
```

图 1.1.1-6 查看 U 盘下的内容

本次新建一个文件，然后将这个文件拷贝到家目录（/home/root）目录下。

使用 touch 指令创建一个 test 文件，然后把这个 test 文件拷贝到/home/root 目录下，如下图，已经通过 touch 创建了一个 test 文件。

```
touch test
ls
```

```
root@ATK-IMX6U:/run/media/sda1# touch test
root@ATK-IMX6U:/run/media/sda1# ls
'????.txt'   'System Volume Information'   test
Makefile     can-utils-2020.02.04.tar.gz
root@ATK-IMX6U:/run/media/sda1#
```

图 1.1.1-7 新建 test 文件

使用 cp 指令拷贝这个 test 复制一份到/home/root 目录下，或者你也可以使用 mv（移动/重命名）指令把它移动到/home/root 目录下。下面只演示 cp 指令(拷贝文件夹用 cp)。拷贝过去了，用 ls 查看/home/root 目录下有 test 这个文件，说明拷贝成功！

```
cp test /home/root/
ls /home/root
```

```
root@ATK-IMX6U:/run/media/sda1# cp test /home/root/
root@ATK-IMX6U:/run/media/sda1# ls /home/root
driver  shell  test
root@ATK-IMX6U:/run/media/sda1#
```

图 1.1.1-8 拷贝 test 文件到/home/root 目录下

问题又来了！怎么退出 U 盘？暴力做法直接拔出？为了数据写入写出完整，你需要在终端命令下执行指令 sync 来同步数据。

```
sync
```

```
root@ATK-IMX6U:/run/media/sda1# sync
root@ATK-IMX6U:/run/media/sda1#
```

图 1.1.1-9 同步数据

退出 U 盘时你需要退出这个挂载的目录，然后用 umount 指令去卸载这个挂载的目录就可以退出了！

使用 cd - 指令返回到上一次目录，再使用 umount /run/media/sda1/ 卸载这个 sda1 目录。然后用 df 指令查看 sda1 这个目录已经不存在了，表明已经卸载了，U 盘可以正常拔出了！

```
cd -
umount /run/media/sda1
df
```

```
root@ATK-IMX6U:/run/media/sda1# cd -
/home/root
root@ATK-IMX6U:~# umount /run/media/sda1
root@ATK-IMX6U:~# df
Filesystem     1K-blocks   Used Available Use% Mounted on
/dev/root        7244864 441184   6429000   7% /
devtmpfs          187640     76    187564   1% /dev
tmpfs                 40      0        40   0% /mnt/.psplash
tmpfs             253436    192    253244   1% /run
tmpfs             253436    176    253260   1% /var/volatile
/dev/mmcblk1p1    129039   6897    122143   6% /run/media/mmcblk1p1
root@ATK-IMX6U:~#
```

图 1.1.1-10 卸载 U 盘

1.1.2 Ubuntu 下通过 U 盘拷贝文件

本文使用的 Ubuntu 版本是 16.04 和 VMware® Workstation 15.5 Pro，其他版本的可能图标有点差异！使用 1.1.1 小节的 U 盘 16GB 的，先把鼠标点击到虚拟机，再把 U 盘直接插到 PC 电脑上的 USB 接口。U 盘插入的过程中可能会提示是否连接到虚拟机，选是就可以了。作者默认已经选是了，并且不让它再它再提示。

如下图，已经连接到了 Ubuntu 上。并且 Ubuntu 会弹出一个文件管理窗口，显示的正是 U 盘的内容。用鼠标点击拖动文件的拷贝方式在这里我就不说了，和 windows 的差不多。我们要用指令 cp/mv 来拷贝/移动。

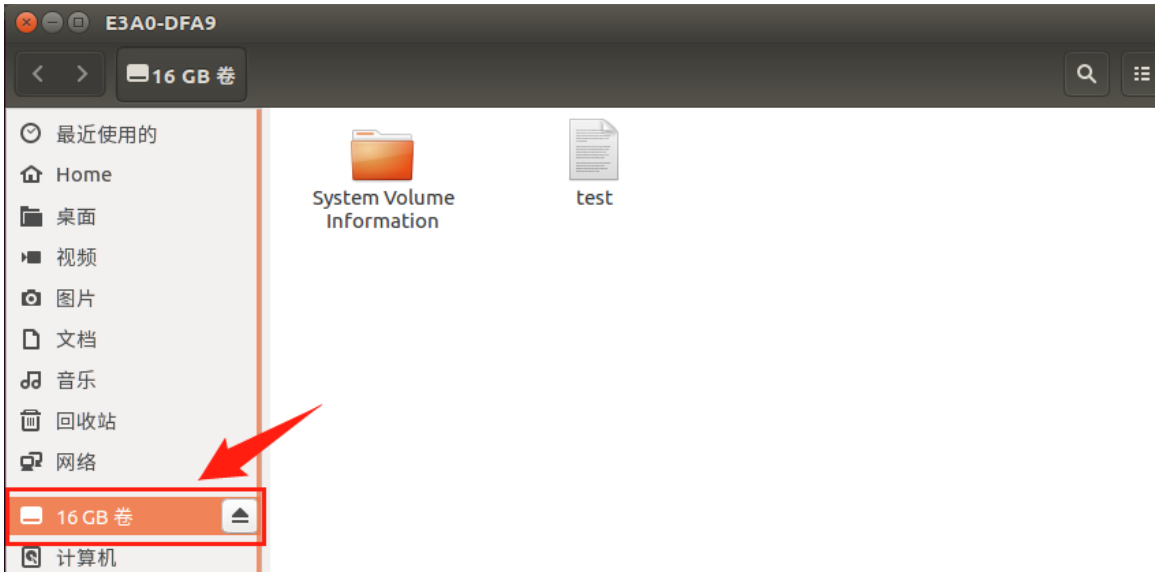


图 1.1.2-1 U 盘接入 Ubuntu 上所显示的文件管理窗口

同理，和 1.1 小节一样，先用 df 指令来查看 U 盘挂载的目录。

```
df
```

```
alientek@ubuntu16:~$ df
文件系统           1K-块      已用      可用 已用% 挂载点
udev             4041816         0   4041816    0% /dev
tmpfs             814472     19928    794544    3% /run
/dev/sda1      307486840 79869964 211974356   28% /
tmpfs            4072340       252   4072088    1% /dev/shm
tmpfs               5120         0      5120    0% /run/lock
tmpfs            4072340         0   4072340    0% /sys/fs/cgroup
tmpfs             814472        52    814420    1% /run/user/1000
/dev/sdb1       15549952       128  15549824    1% /media/alientek/E3A0-DFA9
alientek@ubuntu16:~$
```

图 1.1.2-2 使用 df 指令查看 U 盘挂载的节点与目录等信息

直接双击就可以选中这个目录的路径了，然后右键选择复制，就可以复制它的路径名了。

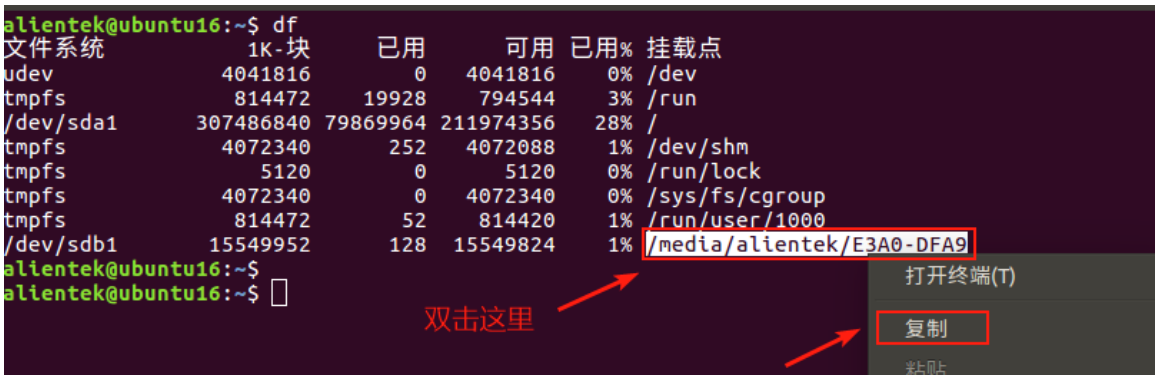


图 1.1.2-3 双击复制路径名

输入 cd 指令，然后按 ctrl+shift+v 快捷键去粘贴这个路径。再用 ls 指令查看当前目录下的文件。其他文件打了码，可以看到 test 这个文件，是 1.1.1 小节从开发板文件系统里测试拷贝出来的。现在要把它拷贝到 Ubuntu 里。

```
cd  + SD 卡的挂载路径
ls
```

```
alientek@ubuntu16:~$ cd /media/alientek/E3A0-DFA9
alientek@ubuntu16:/media/alientek/E3A0-DFA9$ ls
System Volume Information  test
alientek@ubuntu16:/media/alientek/E3A0-DFA9$
```

图 1.1.2-4 查看 test 文件

同样地，也是使用 cp/mv 指令进行拷贝或者移动文件即可(拷贝文件夹是 cp -r 文件夹名)。普通用户拷贝到“/home/”这样的目录需要权限，请在前面加上 sudo，再输入密码，这样就可以拷贝文件了！

```
sudo cp test /home/
```

```
alientek@ubuntu16:/media/alientek/E3A0-DFA9$ sudo cp test /home/
[sudo] alientek 的密码:
alientek@ubuntu16:/media/alientek/E3A0-DFA9$ ls /home/
alientek  test
alientek@ubuntu16:/media/alientek/E3A0-DFA9$
```

输入密码

图 1.1.2-5 拷贝 test 文件到/home 目录下

1.2 开发板通过 scp 指令拷贝文件

初学者很以为只能用 u 盘去给开发板拷贝文件，既然有网口，那么在想是否可以通过网络来传输文件呢，答案是肯定的。下面介绍常用的网络传输指令 scp。

1.2.1 开发板与 Ubuntu 在同一路由器/交换机拷贝文件

在路由器能联网的情况下，开发板与主机（ubuntu 或者 windows）都连在同一路由器，或者同一网段内的网络环境下。Windows 要与开发板传文件，请安装 winodws Git 软件。这里就不多说了，与 linux 的传输指令是一样的。

开发板上电启动，插上网线，在串口终端下输入指令 ifconfig 来查看开发板自动获取的 ip 地址。开发板上电，插网线，使用 ifconfig 指令查看网络的 ip。如下图 192.168.1.28 就是开发板的 ip 地址，记下来。

```
ifconfig
```

```
root@ATK-IMX6U:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 8a:c4:ba:48:1d:f9
          inet addr:192.168.1.181  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::88c4:baff:fe48:1df9/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:281 errors:0 dropped:0 overruns:0 frame:0
          TX packets:84 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:26797 (26.1 KiB)  TX bytes:14168 (13.8 KiB)
```

图 1.2.1-1 在串口终端查看开发板的 ip

在 ubuntu 虚拟机上也同样的使用 ifconfig 来查看 ubuntu 的 ip 地址。

```
ifconfig
```

```
alientek@ubuntu16:/media/alientek/E3A0-DFA9$ ifconfig
ens33     Link encap:以太网  硬件地址 00:0c:29:1c:5b:53
          inet 地址:192.168.1.194  广播:192.168.1.255  掩码:255.255.255.0
          inet6 地址: fe80::34eb:50dd:e4c3:98e/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  跃点数:1
          接收数据包:262055 错误:0 丢弃:0 过载:0 帧数:0
          发送数据包:5706 错误:0 丢弃:0 过载:0 载波:0
          碰撞:0 发送队列长度:1000
          接收字节:212480962 (212.4 MB)  发送字节:454942 (454.9 KB)

lo        Link encap:本地环回
          inet 地址:127.0.0.1  掩码:255.0.0.0
          inet6 地址: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  跃点数:1
          接收数据包:264 错误:0 丢弃:0 过载:0 帧数:0
          发送数据包:264 错误:0 丢弃:0 过载:0 载波:0
          碰撞:0 发送队列长度:1000
          接收字节:21741 (21.7 KB)  发送字节:21741 (21.7 KB)

alientek@ubuntu16:/media/alientek/E3A0-DFA9$
```

图 1.2.1-2 查看 Ubuntu 上的 ip，判断是否在同一网段

例如，在 home 目录下有个 test 文件我们要把这个 test 文件传到开发板的/home/root 目录（我们一般传文件都是传文件到家目录（/home/root）的）。指令格式如下

拷贝文件

```
scp 文件 用户名@ip 地址:路径
```

拷贝文件夹

```
scp -r 文件夹 用户名@ip 地址:路径
```

例：scp test root@192.168.1.181:/home/root

指令格式分析:

test	要传输的文件
root	为用户名，开发板默认的就是 root 用户，拥有最高权限
@	一个符号
192.168.1.181	开发板 ip
:	这里要加一个英文字符的“:”，不要忘记了！
/home/root	要传输到开发板的路径

```
scp test root@192.168.1.181:/home/root
```

```
alientek@ubuntu16:/media/alientek/E3A0-DFA9$ scp test root@192.168.1.181:/home/root
The authenticity of host '192.168.1.181 (192.168.1.181)' can't be established.
RSA key fingerprint is SHA256:mbOOV7N6ynrJgHXO6pDLuI8Z8kV6YrWtVbGb9odzkW0.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.1.181' (RSA) to the list of known hosts.
test                                          100%    0     0.0KB/s   00:00
alientek@ubuntu16:/media/alientek/E3A0-DFA9$
```

图 1.2.1-3 将 test 文件使用 scp 指令传到开发板

在开发板/home/root 可以看到 test.c 文件已经通过网络传输到开发板了。

```
ls
```

```
root@ATK-IMX6U:~#
root@ATK-IMX6U:~# ls
driver  shell  test
root@ATK-IMX6U:~#
```

图 1.2.1-4 在开发板/home/root 下查看 test 文件

1.2.2 开发板与 Ubuntu 直连拷贝文件

在上一小节中，用户需要在有网的情况下才能拷贝文件。假若用户没有网络，只有一根网线怎么办？下面作者为只有一根网线或者路由器不能上网但是又想拷贝文件的方法！

开发板上电，电脑与开发板用一根网线直接连接，或者电脑与开发板同连一个路由器的 lan 口处。

此时电脑上出现一个感叹号，表示无法联网，如下图。如果是一个红 x，表示硬件断开。请检查硬件是否连接正常！

图 1.2.2-1 开发板直连电脑

我们需要为电脑（pc 机）设置静态 ip，下面介绍静态 ip 的方法，如下图。作者用的是 windows7 环境。

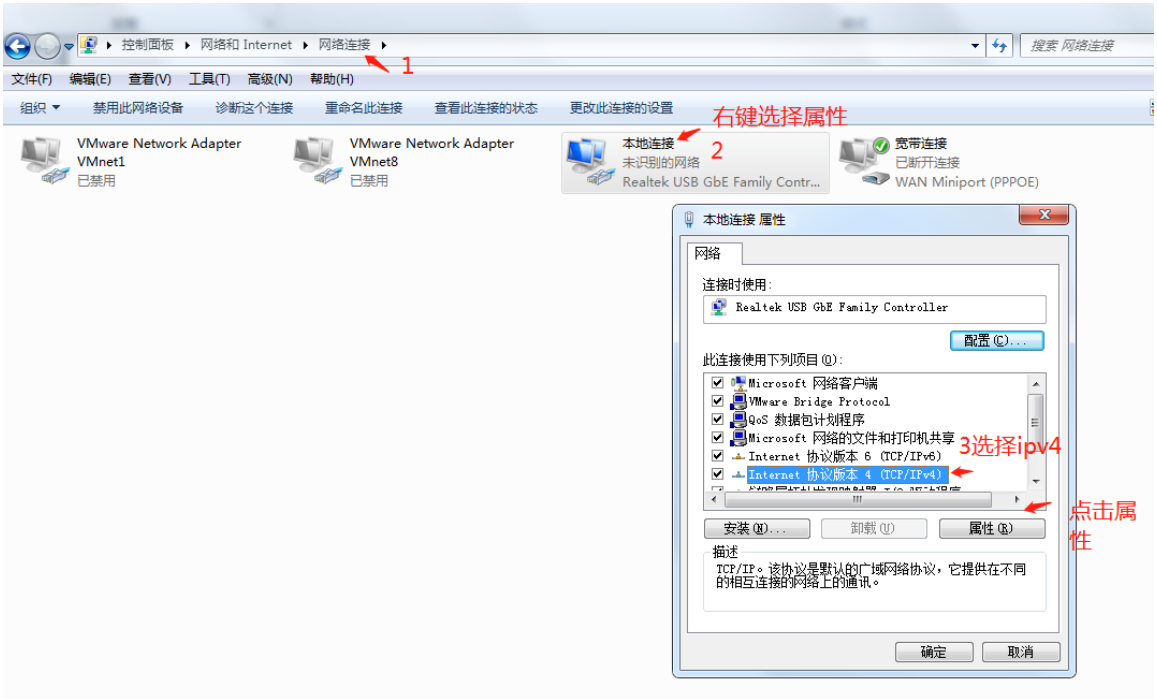


图 1.2.2-2 在本地连接属性里设置属性

设置静态 ip 地址，如下图，点击使用下面的 IP 地址，按如下图设置 ip 地址，子网掩码，默认网关。

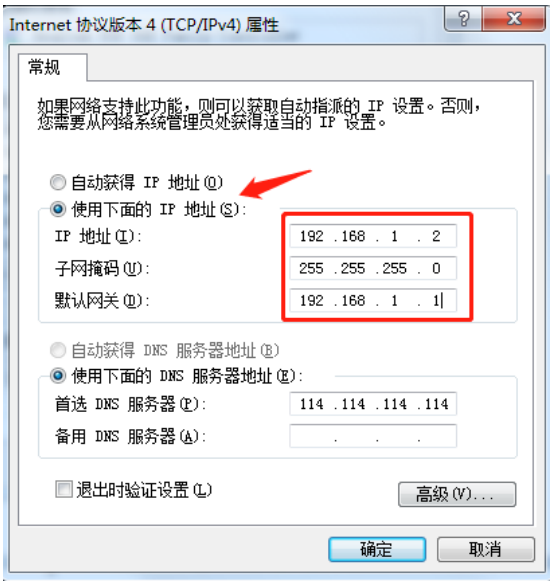


图 1.2.2-3 设置静态 IP

再到 ubuntu 虚拟机查看，此时，网络已经无法连接，所以显示无网络连接的符号或者是正在尝试连接的符号。

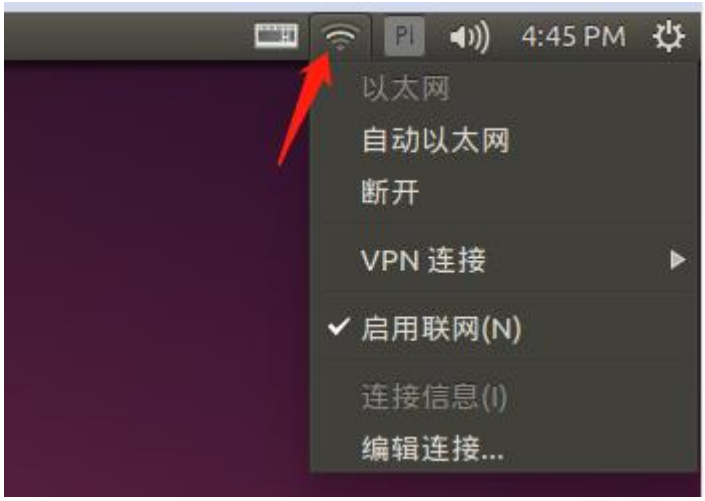


图 1.2.2-4 虚拟机正在尝试连接网络

请把虚拟机设置成桥接模式，按下图这样设置。

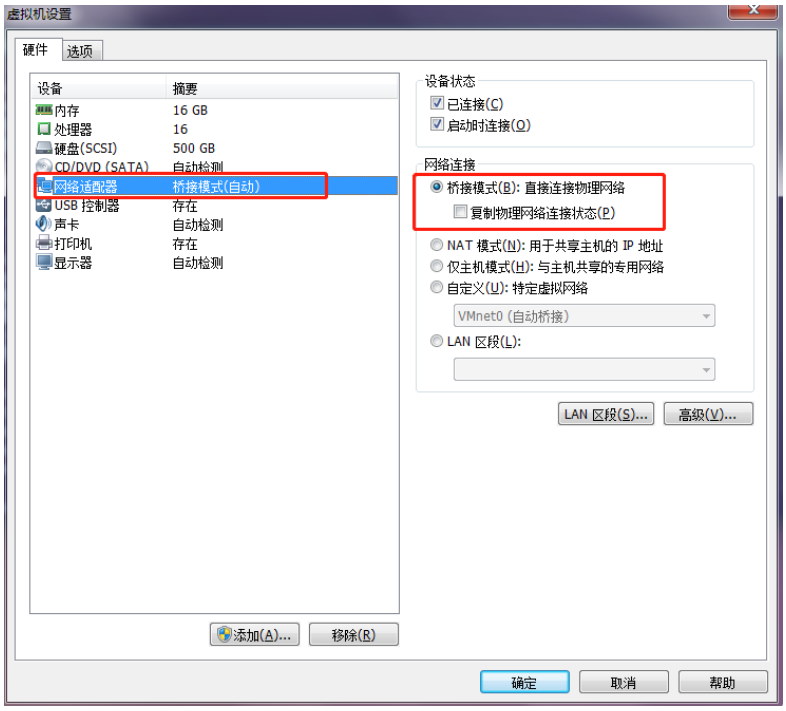


图 1.2.2-5 Ubuntu 虚拟机设置成桥接模式

在 ubuntu 虚拟机右上角设置启用联网，如下图。

图 1.2.2-6 再点击启动联网

设置 ubuntu 的静态 ip，如下图，sudo ifconfig eth0 192.168.1.3，掩码 ubuntu 它自己会补上。

```
sudo ifconfig eth0 192.168.1.3
ifconfig
```

```
alientek@ubuntu:~/test$ sudo ifconfig  eth0 192.168.1.3
[sudo] password for alientek:
alientek@ubuntu:~/test$ ifconfig
eth0      Link encap:以太网  硬件地址 00:0c:29:c5:42:e8
          inet 地址:192.168.1.3  广播:192.168.1.255  掩码:255.255.255.0
          inet6 地址: fe80::20c:29ff:fec5:42e8/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  跃点数:1
          接收数据包:13162381 错误:0 丢弃:1 过载:0 帧数:0
          发送数据包:1551693 错误:0 丢弃:0 过载:0 载波:0
          碰撞:0 发送队列长度:1000
          接收字节:1641535541 (1.6 GB)  发送字节:2013851479 (2.0 GB)

lo        Link encap:本地环回
          inet 地址:127.0.0.1  掩码:255.0.0.0
          inet6 地址: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  跃点数:1
          接收数据包:1648 错误:0 丢弃:0 过载:0 帧数:0
          发送数据包:1648 错误:0 丢弃:0 过载:0 载波:0
          碰撞:0 发送队列长度:1
          接收字节:193258 (193.2 KB)  发送字节:193258 (193.2 KB)

alientek@ubuntu:~/test$
```

图 1.2.2-7 设置虚拟机的静态 ip

再在开发板设置开发板的静态 ip。输入指令 ifconfig eth1 192.168.1.4 。作者插网络到 eth1 处了。所以设置了 eth1 的静态 ip。如下图。

```
ifconfig eth1 192.168.1.4
```

```
root@ATK-IMX6U:~# ifconfig eth1 192.168.1.4
root@ATK-IMX6U:~# ifconfig
eth0      Link encap:Ethernet  HWaddr b6:00:c7:26:8c:0f
          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)

eth1      Link encap:Ethernet  HWaddr 36:b0:55:cc:d9:36
          inet addr:192.168.1.4  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::34b0:55ff:fecc:d936/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST DYNAMIC  MTU:1500  Metric:1
          RX packets:1033 errors:0 dropped:12 overruns:0 frame:0
          TX packets:77 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:90025 (87.9 KiB)  TX bytes:13355 (13.0 KiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:2 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:140 (140.0 B)  TX bytes:140 (140.0 B)

root@ATK-IMX6U:~#
```

图 1.2.2-8 设置开发板的静态 ip

设置完就可以与开发板进行文件传送了。如下图，和 1.2.1 小节一样，使用相同的指令传送相同的文件。（注：电脑设置了静态 ip 后会不能上网，请设置回自动获取 ip 即可。）

```
scp test.c root@192.168.1.4:/home/root
```

```
alientek@ubuntu:~/test$ sudo ifconfig eth0 192.168.1.3
alientek@ubuntu:~/test$ ifconfig
eth0      Link encap:以太网  硬件地址 00:0c:29:c5:42:e8
          inet 地址:192.168.1.3  广播:192.168.1.255  掩码:255.255.255.0
          inet6 地址: fe80::20c:29ff:fec5:42e8/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  跃点数:1
          接收数据包:13163134 错误:0 丢弃:1 过载:0 帧数:0
          发送数据包:1552076 错误:0 丢弃:0 过载:0 载波:0
          碰撞:0 发送队列长度:1000
          接收字节:1641617862 (1.6 GB)  发送字节:2013918249 (2.0 GB)

lo        Link encap:本地环回
          inet 地址:127.0.0.1  掩码:255.0.0.0
          inet6 地址: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  跃点数:1
          接收数据包:1718 错误:0 丢弃:0 过载:0 帧数:0
          发送数据包:1718 错误:0 丢弃:0 过载:0 载波:0
          碰撞:0 发送队列长度:1
          接收字节:198012 (198.0 KB)  发送字节:198012 (198.0 KB)

alientek@ubuntu:~/test$ scp test.c root@192.168.1.4:/home/root
The authenticity of host '192.168.1.4 (192.168.1.4)' can't be established.
RSA key fingerprint is ce:53:55:e7:a1:23:e0:cf:82:ed:7d:bb:37:dc:b3:d9.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.1.4' (RSA) to the list of known hosts.
test.c                                                        100%   64     0.1KB/s   00:00
alientek@ubuntu:~/test$
```

图 1.2.2-9 使用 scp 指令向开发板发送文件

1.3 开发板使用 MobaXterm 与 Windows 互传文件

使用前提：开发板与 PC 机用网线连接在同一路由器上，路由器能上网。

请注意，这里使用的是出厂的文件系统，支持 ssh 协议。默认开发板文件系统不支持 FTP 传输，其他文件系统请确认是否支持 ssh 协议。

【正点原子】I.MX6U 嵌入式 Linux 驱动开发指南 V1.x 第 4.9 小节已经安装过 MobaXterm，使用 ifconfig 在 MobaXterm 查看开发板的 ip，如下图，记下开发板的 ip 为 192.168.1.222。

```
ifconfig
```

```
root@ATK-IMX6U:~# ifconfig
eth0      Link encap:Ethernet  HWaddr f6:a7:85:f3:de:fb
          inet addr:192.168.1.222  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::f4a7:85ff:fef3:defb/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:3317 errors:0 dropped:12 overruns:0 frame:0
          TX packets:84 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:228833 (223.4 KiB)  TX bytes:12374 (12.0 KiB)

eth1      Link encap:Ethernet  HWaddr 0a:aa:10:8e:90:2d
          UP BROADCAST MULTICAST DYNAMIC  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)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:14 errors:0 dropped:0 overruns:0 frame:0
          TX packets:14 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:1932 (1.8 KiB)  TX bytes:1932 (1.8 KiB)

root@ATK-IMX6U:~#
```

图 1.2.2-1 查看开发板的 ip

然后我们在 MobaXterm 选择 SSH。

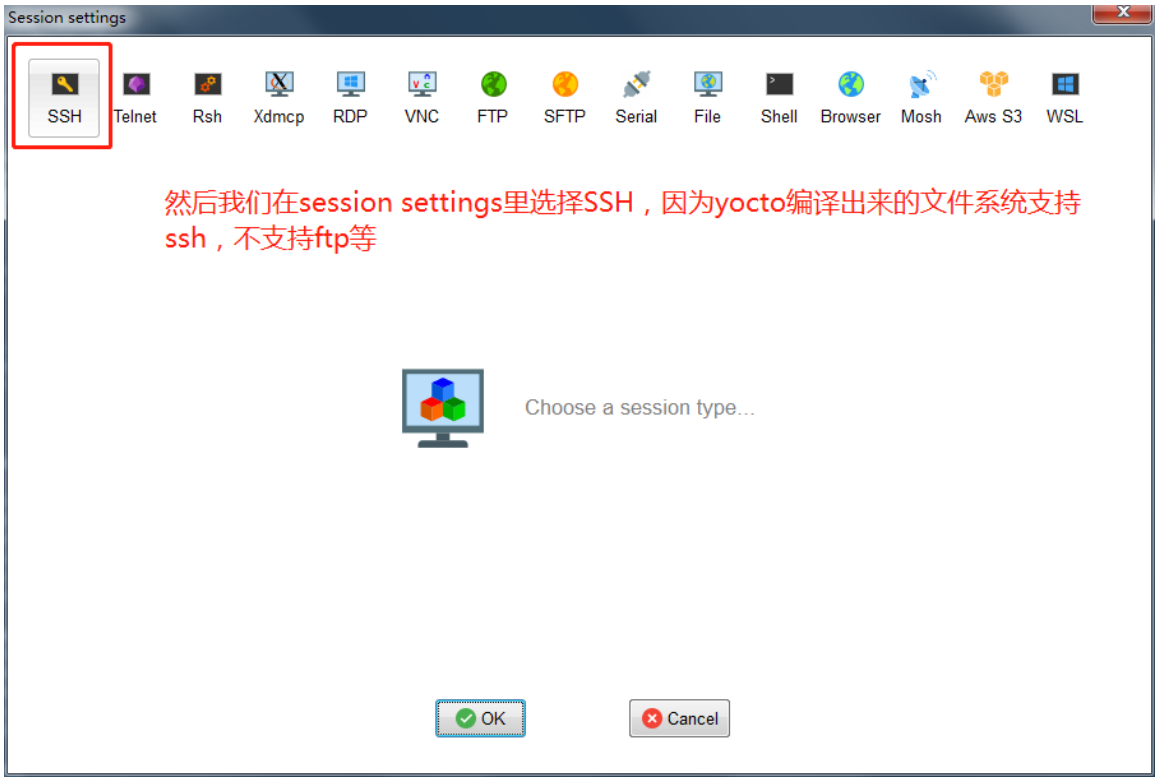


图 1.2.2-2 选择 SSH

按如下步骤使用 ssh 去连接开发板。

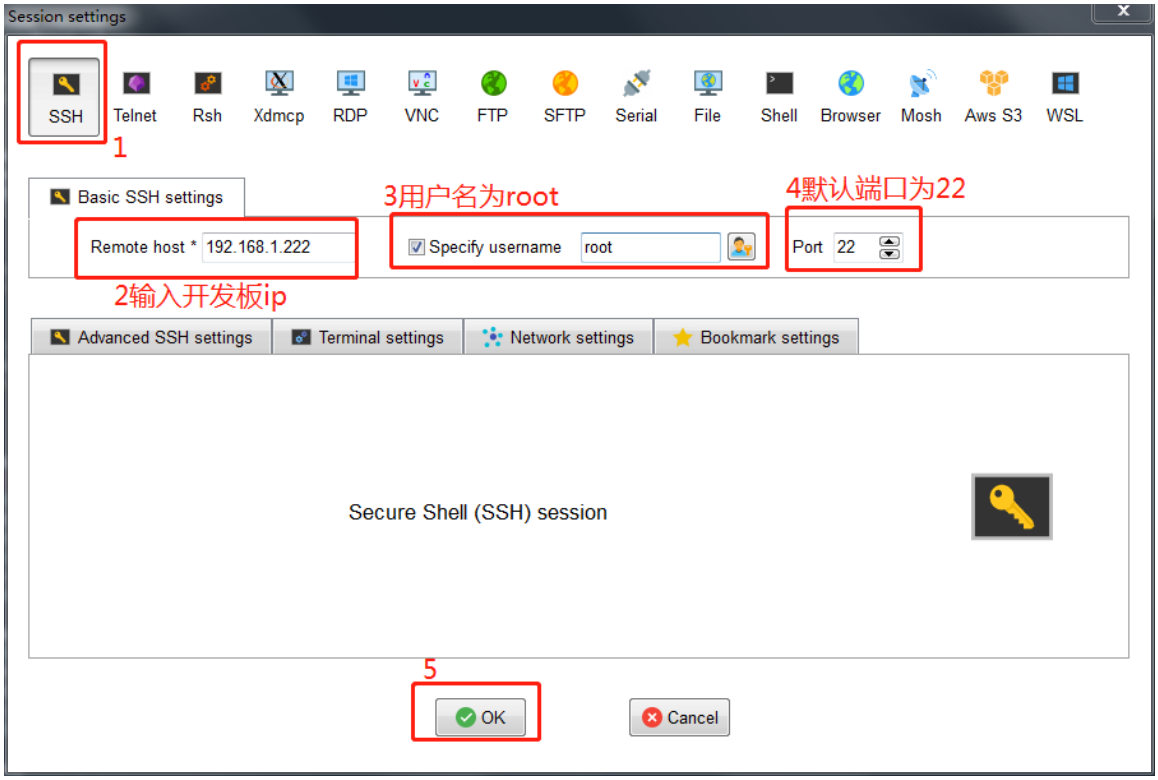


图 1.2.2-3 输入用户名、ip 连接开发板

按如下说明进行 windows 与开发板进行文件传输就可以了，左边的是开发板默认访问的目录，我们可以看到开发板上/home/root 目录下的文件。要想进行文件传输，直接拖拽文件即可。

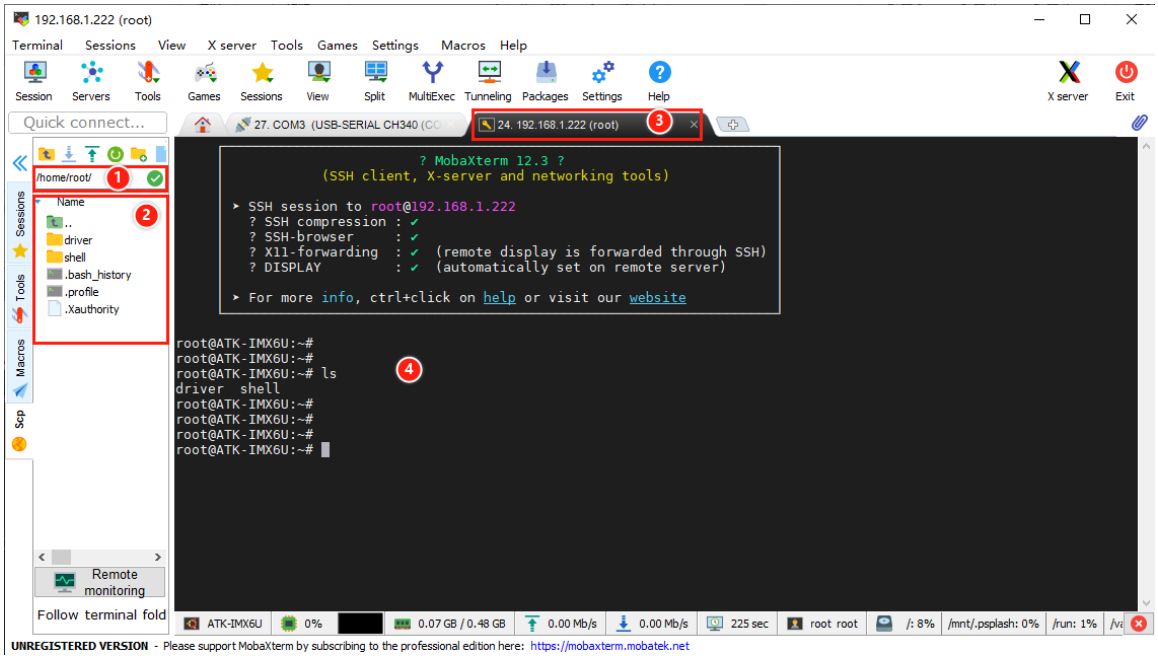


图 1.2.2-4 在串口终端查看开发板的目录

终端界面简析：

① 访问的目录。

② 开发板上的文件，可以把 Windows 的文件拖进来，实现与开发板文件的传输。

③ 开发板 SSH 信息。

④ 命令终端，可以直接在此处输入命令。

第二章 I.MX6U 更新固件

在【正点原子】I.MX6U 用户快速体验 V1.x.pdf 的第二章 2.2 小节我们已经成功固化了 Linux 文件系统，系统也已经成功启动了。但有时候我们开发过程中，尝试自己编译 uboot、内核与文件系统，编译后我们只需要更新 uboot、设备树和文件系统中的某一个，特别是在开发过程中可能要频繁更新，为了节省开发时间，我们可以采用单步更新的方法来更新我们需要更新的部分，而不必花更长的时间去重新制作一张 SD 系统启动卡或者使用脚本固化系统到 eMMC 或者 NAND FLASH 中。

这里说明一下，有些用户会使用 tftp 去更新 uboot、内核等。对于新手来说，这种方式难度还是有点大，不直观。所以这里我们考虑到新手的程度，就不说网络更新了！另外使用 nfs 从网络加载 zImage 和设备树，挂载文件系统的方法，请自行替换相应的文件进行更新就好了。

Windows 和开发板文件互传的方法可以参考本文档上一章的 SSH，Windows 和 Ubuntu 文件互传可以参考《【正点原子】I.MX6U 嵌入式 Linux 驱动开发指南 V1.5.pdf》里的 4.1 节 Ubuntu 和 Windows 文件互传。

关于光盘路径的说明：正点原子阿尔法开发板和 MINI 开发板资料盘都是通用的，可以在正点原子论坛上下载。地址：http://www.openedv.com/docs/boards/arm-linux/zdyz-i.mx6ull.html。本文档在编写时，所使用的的文件都在开发板光盘 A-基础资料\8、开发板系统镜像目录下，uboot 文件夹为 uboot-imx-2016.03-2.1.0-gd3f0479-v1.4，linux 内核文件夹为 linux-imx-4.1.15-2.1.0-g45a21e3-v1.4。后续可能会更新这些文件、修改文件夹名，基本路径不变，用户在操作时以具体路径、文件名为准。

2.1 Nand Flash 更新固件

如果用户使用的是 NandFlash 的核心板，正点原子出厂都有把固件固化到 NandFlash 中。可以直接从 Nand Flash 启动或者 SD 卡启动进入系统去更新这固化在 Nand Flash 的系统固件。

2.1.1 更新 uboot 到 Nand Flash

我们将正点原子网盘路径下的 u-boot-imx6ull-14x14-ddr256-nand.imx 更新到 Nand Flash。光盘路径：开发板光盘 A-基础资料\8、系统镜像\1、出厂系统镜像\1、u-boot 镜像\uboot-imx-2016.03-2.1.0-gd3f0479-v1.4\u-boot-imx6ull-14x14-ddr256-nand.imx

把 u-boot-imx6ull-14x14-ddr256-nand.imx 拷贝到开发板文件系统的/home/root 目录中（这个其实是个 u-boot.imx 文件，大家可以根据自己的习惯拷贝到其他目录，为后面更新 uboot 做准备）。可以参考【正点原子】I.MX6U 用户快速体验 V1.x.pdf 第 2.2.1.1 小节查看 u-boot-imx6ull-14x14-ddr256-nand.imx 命名的含义

```
root@ATK-IMX6U:~#
root@ATK-IMX6U:~# pwd
/home/root
root@ATK-IMX6U:~# ls
driver  shell  u-boot-imx6ull-14x14-ddr256-nand.imx
root@ATK-IMX6U:~#
```

图 2.1.1-1 将 Nand Flash 用的 uboot 文件拷贝到开发板

查看 Nand Flash 分区情况：

```
cat /proc/mtd
```

```
root@ATK-IMX6U:~# cat /proc/mtd
dev:    size   erasesize  name
mtd0: 00400000 00020000 "u-boot"
mtd1: 00020000 00020000 "env"
mtd2: 00100000 00020000 "logo"
mtd3: 00100000 00020000 "dtb"
mtd4: 00800000 00020000 "kernel"
mtd5: 1f1e0000 00020000 "rootfs"
```

图 2.1.1-2 查看 Nand Flash 的分区情况

使用指令 cat /proc/mtd 可以查看 mtd 中保存的系统磁盘分区信息。由以上的输出信息可知：

分区名	分区类型
设备 mtd0	u-boot 分区
mtd1	env（环境变量）分区
mtd2	logo 分区
mtd3	dtb（设备树文件）分区
mtd4	kernel（内核文件）分区
mtd5	rootfs（文件系统）分区

表 2.1-1 Nand 磁盘分区表

size 是对应 mtd 分区的最大字节数空间，erasesize 是对应分区的最小擦除字节数空间（以块为单位），例如：u-boot 分区，size=00400000(表示 16 进制 0x400000，单位为 Byte)=4MB，erasesize=00020000=128KB。

烧写前使用 flash_erase 指令先擦除 u-boot 对应的分区 mtd0，可以输入 flash_erase --help 指令来查看其用法，如果不在 MTD_DEVICE 后面加上 <start offset> 和<block count>，直接执行 flash_erase /dev/mtd0 的话将会报错。

```
flash_erase --help
flash_erase /dev/mtd0 0 0
```

```
root@ATK-IMX6U:~# flash_erase --help
Usage: flash_erase [options] MTD_DEVICE <start offset> <block count>
Erase blocks of the specified MTD device.
Specify a count of 0 to erase to end of device.

  -j, --jffs2       format the device for jffs2
  -N, --noskipbad   don't skip bad blocks
  -u, --unlock      unlock sectors before erasing
  -q, --quiet       do not display progress messages
      --silent      same as --quiet
      --help        display this help and exit
      --version     output version information and exit
root@ATK-IMX6U:~# flash_erase /dev/mtd0 0 0
Erasing 128 Kibyte @ 3e0000 -- 100 % complete
root@ATK-IMX6U:~#
```

图 2.1.1-3 擦除 mtd0 分区

```
kobs-ng init -x -v --chip_0_device_path=/dev/mtd0 u-boot-imx6ull-14x14-ddr256-nand.imx
sync
```

```
root@ATK-IMX6U:~# kobs-ng init -x -v --chip_0_device_path=/dev/mtd0 u-boot-imx6ull-14x14-ddr256-nand.imx
MTD CONFIG:
  chip_0_device_path = "/dev/mtd0"
  chip_1_device_path = "(null)"
  search_exponent = 2
  data_setup_time = 80
  data_hold_time = 60
  address_setup_time = 25
  data_sample_time = 6
  row_address_size = 3
  column_address_size = 2
  read_command_code1 = 0
  read_command_code2 = 48
  boot_stream_major_version = 1
  boot_stream_minor_version = 0
  boot_stream_sub_version = 0
  ncb_version = 3
  boot_stream_1_address = 0
  boot_stream_2_address = 0
        -- We add the 1k-padding to the uboot.
.tmp_kobs_ng: verifying using key '00000000000000000000000000000000'
.tmp_kobs_ng: is a valid bootstream for key '00000000000000000000000000000000'
mtd: use new bch layout raw access mode
mtd: opening: "/dev/mtd0"
NFC geometry :
        ECC Strength       : 4
        Page Size in Bytes : 2084
        Metadata size      : 10
        ECC Chunk Size in byte : 512
        ECC Chunk count        : 4
        Block Mark Byte Offset : 2018
        Block Mark Bit Offset  : 4
==================================================
mtd: opened '/dev/mtd0' - '(null)'
mtd: max_boot_stream_size_in_bytes = 1572864
```

此处过长，省略剩下的信息

图 2.1.1-4 烧写 uboot 到 mtd0 分区

以上使用 kobs-ng 烧录命令来更新 uboot，在此次操作中已经将 uboot 文件放在了文件系统的/home/root 下（即当前执行指令的目录），所以指令的后面省去了当前的路径/home/root，写成了 u-boot-imx6ull-14x14-ddr256-nand.imx。在执行以上的烧写 uboot 的指令的时候，根据自己 uboot 文件放置的路径情况来修改，如以上的指令将路径补全以后为：

```
kobs-ng init -x -v --chip_0_device_path=/dev/mtd0    /home/root/u-boot-imx6ull-14x14-ddr256-nand.imx
```

更新完 uboot 文件以后，最好执行一遍 sync 命令来将缓存内容同步到 NAND FLASH 中再重启系统，千万不要直接关机或者按 RESET 键来重启，否则有可能缓存还未同步导致文件的丢失，烧写失败。

```
sync
```

```
mtd: erasing @0:0x2e0000-0x300000
mtd: We write one page for save guard. *
root@ATK-IMX6U:~#
root@ATK-IMX6U:~# sync
root@ATK-IMX6U:~#
```

图 2.1.1-5 sync 同步

2.1.2 更新设备树到 Nand Flash

首先将光盘里的与 Nand Flash 相关设备树拷贝到开发板中。

光盘路径：开发板光盘 A-基础资料\8、系统镜像\1、出厂系统镜像\2、kernel 镜像\linux-imx-4.1.15-2.1.0-gb78e551-v1.4\

图 2.1.2-1 Nand Flash 相关的设备树

由前面的查看分区指令可知 mtd1 是 env（环境变量）分区，mtd2 是 logo 分区，mtd3 是 dtb（设备树文件）分区，按照前面更新 uboot 的步骤，如果我们要更新设备树的话，先将对应的设备树文件拷贝到文件系统的/home/root 分区，然后擦除对应的分区再进行烧写。

```
root@ATK-IMX6U:~# ls
driver                              imx6ull-14x14-nand-7-1024x600-c.dtb  shell
imx6ull-14x14-nand-10.1-1280x800-c.dtb  imx6ull-14x14-nand-7-800x480-c.dtb   u-boot-imx6ull-14x14-ddr256-nand.imx
imx6ull-14x14-nand-4.3-480x272-c.dtb    imx6ull-14x14-nand-hdmi.dtb
imx6ull-14x14-nand-4.3-800x480-c.dtb    imx6ull-14x14-nand-vga.dtb
root@ATK-IMX6U:~#
```

图 2.1.2-2 已拷贝相关设备树到/home/root 目录下

如上图，为了文档的完整，本文中是将对应各个分辨率的设备树文件拷贝到了文件系统的/home/root 中。

擦除设备树对应的分区：

```
flash_erase /dev/mtd3 0 0
```

根据自己显示屏的尺寸和分辨率的情况，使用 nandwrite 指令将对应的设备树文件烧写到 mtd3 对应的地址中。

如 imx6ull-14x14-nand-4.3-480x272-c.dtb 对应的是显示屏为 4.3 寸、分辨率为 480*272 的设备树文件，在 mtd3 的偏移地址 0x0 处将此设备树的数据写入到了 mtd3 中。同理，imx6ull-14x14-nand-7-800x480-c.dtb 对应的是显示屏为 7 寸、分辨率为 800*480 的设备树文件，在 mtd3 的偏移地址 0x40000 处将该设备树的数据写入到了 mtd3 中。

同理，更新完设备树文件以后，最好执行一遍 sync 命令以后再重启系统。

```
nandwrite -p /dev/mtd3 /home/root/imx6ull-14x14-nand-4.3-480x272-c.dtb
nandwrite -s 0x20000 -p /dev/mtd3 /home/root/imx6ull-14x14-nand-4.3-800x480-c.dtb
nandwrite -s 0x40000 -p /dev/mtd3 /home/root/imx6ull-14x14-nand-7-800x480-c.dtb
nandwrite -s 0x60000 -p /dev/mtd3 /home/root/imx6ull-14x14-nand-7-1024x600-c.dtb
nandwrite -s 0x80000 -p /dev/mtd3 /home/root/imx6ull-14x14-nand-10.1-1280x800-c.dtb
nandwrite -s 0xa0000 -p /dev/mtd3 /home/root/imx6ull-14x14-nand-hdmi.dtb
```

```
nandwrite -s 0xc0000 -p /dev/mtd3 /home/root/imx6ull-14x14-nand-vga.dtb
sync
```

```
root@ATK-IMX6U:~# flash_erase /dev/mtd3 0 0
Erasing 128 Kibyte @ e0000 -- 100 % complete
root@ATK-IMX6U:~#
root@ATK-IMX6U:~# nandwrite -p /dev/mtd3 /home/root/imx6ull-14x14-nand-4.3-480x272-c.dtb
nandwrite -s 0x40000 -p /dev/mtd3 /home/root/imx6ull-14x14-nand-7-800x480-c.dtb
nandwrite -s 0x60000 -p /dev/mtd3 /home/root/imx6ull-14x14-nand-7-1024x600-c.dtb
nandwrite -s 0x80000 -p /dev/mtd3 /home/root/imx6ull-14x14-nand-10.1-1280x800-c.dtb
nandwrite -s 0xa0000 -p /dev/mtd3 /home/root/imx6ull-14x14-nand-hdmi.dtb
nandwrite -s 0xc0000 -p /dev/mtd3 /home/root/imx6ull-14x14-nand-vga.dtb
sync
Writing data to block 0 at offset 0x0
root@ATK-IMX6U:~# nandwrite -s 0x20000 -p /dev/mtd3 /home/root/imx6ull-14x14-nand-4.3-800x480-c.dtb
Writing data to block 1 at offset 0x20000
root@ATK-IMX6U:~# nandwrite -s 0x40000 -p /dev/mtd3 /home/root/imx6ull-14x14-nand-7-800x480-c.dtb
Writing data to block 2 at offset 0x40000
root@ATK-IMX6U:~# nandwrite -s 0x60000 -p /dev/mtd3 /home/root/imx6ull-14x14-nand-7-1024x600-c.dtb
Writing data to block 3 at offset 0x60000
root@ATK-IMX6U:~# nandwrite -s 0x80000 -p /dev/mtd3 /home/root/imx6ull-14x14-nand-10.1-1280x800-c.dtb
Writing data to block 4 at offset 0x80000
root@ATK-IMX6U:~# nandwrite -s 0xa0000 -p /dev/mtd3 /home/root/imx6ull-14x14-nand-hdmi.dtb
Writing data to block 5 at offset 0xa0000
root@ATK-IMX6U:~# nandwrite -s 0xc0000 -p /dev/mtd3 /home/root/imx6ull-14x14-nand-vga.dtb
Writing data to block 6 at offset 0xc0000
root@ATK-IMX6U:~# sync
```

图 2.1.2-3 烧写各个设备树，注意设备树要对应相应的偏移地址

2.1.3 更新内核到 Nand Flash

首先将光盘里的 zImage 文件拷贝到开发板中。

光盘路径：开发板光盘 A-基础资料\8、系统镜像\1、出厂系统镜像\2、kernel 镜像\linux-imx-4.1.15-2.1.0-gb78e551-v1.4\zImage

执行如下指令，擦除对应的内核文件分区再进行内核文件的烧写，再重启系统。

```
flash_erase /dev/mtd4 0 0
nandwrite -p /dev/mtd4 /home/root/zImage
sync
```

```
root@ATK-IMX6U:~# pwd
/home/root
root@ATK-IMX6U:~# ls
driver                              imx6ull-14x14-nand-7-1024x600-c.dtb  shell
imx6ull-14x14-nand-10.1-1280x800-c.dtb  imx6ull-14x14-nand-7-800x480-c.dtb   u-boot-imx6ull-14x14-ddr256-nand.imx
imx6ull-14x14-nand-4.3-480x272-c.dtb    imx6ull-14x14-nand-hdmi.dtb          zImage
imx6ull-14x14-nand-4.3-800x480-c.dtb    imx6ull-14x14-nand-vga.dtb
root@ATK-IMX6U:~#
root@ATK-IMX6U:~# flash_erase /dev/mtd4 0 0
Erasing 128 Kibyte @ 7e0000 -- 100 % complete
root@ATK-IMX6U:~# nandwrite -p /dev/mtd4 /home/root/zImage
Writing data to block 0 at offset 0x0
Writing data to block 1 at offset 0x20000
Writing data to block 2 at offset 0x40000
Writing data to block 3 at offset 0x60000
Writing data to block 4 at offset 0x80000
Writing data to block 5 at offset 0xa0000
Writing data to block 6 at offset 0xc0000
Writing data to block 7 at offset 0xe0000
Writing data to block 8 at offset 0x100000
Writing data to block 9 at offset 0x120000
Writing data to block 10 at offset 0x140000
```

图 2.1.3-1 先擦除内核分区再烧写内核

2.1.4 更新文件系统到 Nand Flash

需要从 SD 卡系统卡启动才能更新文件系统到 Nand Flash 中，请参照【正点原子】I.MX6U 用户快速体验 V1.x.pdf 的第二章 2.2.1.1 小节固化系统到 SD 卡，从 SD 卡启动！注意 Nand Flash

只有 512MB，不能从 Nand Flash 去启动文件系统，正在运行的系统不能把自己文件系统分区给格式化！。

将光盘中的文件系统压缩包和模块压缩包拷贝到文件系统的/home/root 目录下。

文件系统包路径：开发板光盘 A-基础资料\8、系统镜像\1、出厂系统镜像\3、文件系统\rootfs.tar.bz2

内核模块包路径：开发板光盘 A-基础资料\8、系统镜像\1、出厂系统镜像\2、kernel 镜像\linux-imx-4.1.15-2.1.0-gb78e551-v1.4\ modules.tar.bz2

将上述两个文件拷贝到开发板后，如下图所示，rootfs.tar.bz2 是文件系统压缩包，modules.tar.bz2 是模块压缩包。

```
root@ATK-IMX6U:~# pwd
/home/root
root@ATK-IMX6U:~# ls
driver  modules.tar.bz2  rootfs.tar.bz2  shell
root@ATK-IMX6U:~#
```

图 2.1.4-1 拷贝文件系统和内核模块到开发板

执行以下指令格式化文件系统分区、挂载文件系统分区。

```
flash_erase /dev/mtd5 0 0
ubiformat /dev/mtd5
ubiattach /dev/ubi_ctrl -m 5
ubimkvol /dev/ubi0 -Nrootfs -m
mkdir -p /mnt/mtd5
mount -t ubifs ubi0:rootfs /mnt/mtd5
```

以上使用了 UBI 相关的一些指令来烧写 UBIFS 格式的文件系统， FLASH 常用的文件系统有 JFFS2、YAFFS2、LOGFS 和 UBIFS，而 UBIFS 专门为了解决 MTD 设备所遇到的瓶颈而设计的，其在设计与性能上均较 YAFFS2、JFFS2 更适合 NAND Flash。关于更详细的 UBIFS 格式文件系统和 UBI 工具的介绍可自行查找相关资料，这里只是做简单的描述。

ubiformat /dev/mtd5 作用是格式 mtd5 分区；ubiattach /dev/ubi_ctrl -m 5 指将/dev/mtd5 分区关联到 UBI 上。

ubimkvol /dev/ubi0 -Nrootfs -m 的作用是为新创建的分区，设置大小以及标识名。

mkdir -p /mnt/mtd5 用于创建一个旧时目录/mnt/mtd5，即创建用户挂载文件系统的地方。

mount -t ubifs ubi0:rootfs /mnt/mtd5 是指按照 UBI 的分区名进行挂载 UBI 分区到新建的目录/mnt/mtd5 中。

```
root@ATK-IMX6U:~# flash_erase /dev/mtd5 0 0
Erasing 128 Kibyte @ 1f140000 -- 99 % complete flash_erase: Skipping bad block at 1f160000
flash_erase: Skipping bad block at 1f180000
flash_erase: Skipping bad block at 1f1a0000
flash_erase: Skipping bad block at 1f1c0000
Erasing 128 Kibyte @ 1f1c0000 -- 100 % complete
root@ATK-IMX6U:~# ubiformat /dev/mtd5
ubiformat: mtd5 (nand), size 522059776 bytes (497.9 MiB), 3983 eraseblocks of 131072 bytes (128.0 KiB), min. I/O size 2048 bytes
libscan: scanning eraseblock 3982 -- 100 % complete
ubiformat: 3979 eraseblocks are supposedly empty
ubiformat: 4 bad eraseblocks found, numbers: 3979, 3980, 3981, 3982
ubiformat: formatting eraseblock 3982 -- 100 % complete
root@ATK-IMX6U:~# ubiattach /dev/ubi_ctrl -m 5
UBI device number 0, total 3979 LEBs (505237504 bytes, 481.8 MiB), available 3899 LEBs (495079424 bytes, 472.1 MiB), LEB size 126976 bytes (
124.0 KiB)
root@ATK-IMX6U:~# ubimkvol /dev/ubi0 -Nrootfs -m
Set volume size to 495079424
Volume ID 0, size 3899 LEBs (495079424 bytes, 472.1 MiB), LEB size 126976 bytes (124.0 KiB), dynamic, name "rootfs", alignment 1
root@ATK-IMX6U:~# mkdir -p /mnt/mtd5
root@ATK-IMX6U:~# mount -t ubifs ubi0:rootfs /mnt/mtd5
root@ATK-IMX6U:~#
```

图 2.1.4-2 格式化文件系统分区，然后挂载文件系统分区

将文件系统解压到/mnt/mtd5/中，如下图是解压过程的部分截图：

```
tar jxvfm rootfs.tar.bz2 -C /mnt/mtd5/
```

```
var/spool/at/jobs/
var/spool/at/jobs/.SEQ
var/spool/at/spool/
var/spool/mail/
var/volatile/
var/tmp
var/backups/
var/cache/
var/cache/fontconfig/
var/cache/fontconfig/3830d5c3ddfd5cd38a049b759396e72e-le32d8.cache-6
var/cache/fontconfig/CACHEDIR.TAG
var/cache/fontconfig/6ba42ae0000f58711b5caaf10d690066-le32d8.cache-6
var/cache/fontconfig/94322f4d3cdf1bd54794b691285ca062-le32d8.cache-6
var/cache/ldconfig/
var/cache/ldconfig/aux-cache
root@ATK-IMX6U:~#
```

图 2.1.4-3 解压文件系统到 Nand Flash 文件系统分区

以上指令是将 rootfs.tar.bz2 解压到/mnt/mtd5/中，jxvfm 中 j 指有 bz2 属性的解压，x 指解压缩，v 指将解压缩的过程展示出来，如果不想展示解压缩的过程可以将 v 去掉，f 指指定要操作的文件名，m 指保留文件不被覆盖，-C 指变更解压的目标目录，默认是当前目录，后面紧跟的是要解压缩到/mnt/mtd5/这个目录中。

解压完文件系统以后，如果有改动过内核模块选项，编译好内核以后还需要编译内核模块，所以更新了内核以后，我们还需要更新一下内核模块（如果没有更改过内核模块选项的，此步可以跳过）。 还需要解压模块到/mnt/mtd5/lib/modules 中。在设备驱动开发中会将某些驱动程序以模块的形式来编译，所以我们还需要将这些编译好的模块解压到对应的分区中以支持相应的功能。

```
tar jxvfm modules.tar.bz2 -C /mnt/mtd5/lib/modules
sync
```

```
root@ATK-IMX6U:~# tar jxvfm modules.tar.bz2 -C /mnt/mtd5/lib/modules
./
./4.1.15-gb8ddbbc/
./4.1.15-gb8ddbbc/modules.devname
./4.1.15-gb8ddbbc/modules.alias
./4.1.15-gb8ddbbc/modules.dep.bin
./4.1.15-gb8ddbbc/modules.symbols.bin
./4.1.15-gb8ddbbc/modules.order
./4.1.15-gb8ddbbc/modules.builtin.bin
./4.1.15-gb8ddbbc/kernel/
./4.1.15-gb8ddbbc/kernel/drivers/
./4.1.15-gb8ddbbc/kernel/drivers/input/
./4.1.15-gb8ddbbc/kernel/drivers/input/evbug.ko
./4.1.15-gb8ddbbc/kernel/drivers/input/mouse/
./4.1.15-gb8ddbbc/kernel/drivers/input/mouse/psmouse.ko
./4.1.15-gb8ddbbc/kernel/drivers/input/serio/
./4.1.15-gb8ddbbc/kernel/drivers/input/serio/serport.ko
./4.1.15-gb8ddbbc/kernel/drivers/video/
./4.1.15-gb8ddbbc/kernel/drivers/video/fbdev/
./4.1.15-gb8ddbbc/kernel/drivers/video/fbdev/mxc/
./4.1.15-gb8ddbbc/kernel/drivers/video/fbdev/mxc/mxc_dcic.ko
./4.1.15-gb8ddbbc/kernel/drivers/media/
```

图 2.1.4-4 解压内核模块到 Nand Flash 文件系统的 lib/modules 目录

更新完内核模块以后，我们需要卸载这个目录，然后再删除这个目录，注意，顺序不能变。

```
umount /mnt/mtd5
rm -rf /mnt/mtd5
ubidetach -p /dev/mtd5
```

```
sync
```

ubidetach 是 ubiattach 相反的操作，即将/dev/mtd5 设备从 UBI 设备上去掉关联，在使用此命令之前先将/mnt/mtd5 卸载掉。执行以上指令以后再进行一次 sync 同步。

```
root@ATK-IMX6U:~# umount /mnt/mtd5
root@ATK-IMX6U:~# rm -rf /mnt/mtd5
root@ATK-IMX6U:~# ubidetach -p /dev/mtd5
root@ATK-IMX6U:~# sync
root@ATK-IMX6U:~#
```

图 2.1.4-5 卸载目录再删除文件系统目录的临时目录

至此，我们已经将文件系统更新到了 Nand Flash 中了，开发板拨码开关选择从 Nand Flash 启动，查看系统是可以成功启动了的。

2.2 更新 eMMC 固件

如果用户使用的是 eMMC 的核心板，正点原子出厂都有把固件固化到 eMMC 中。我们需要从 eMMC 启动或者从 SD 卡启动替换自己固件。这里简要总结下 eMMC 分区表。

分区名	作用
mmcblk1boot0	uboot 分区
mmcblk1p1	内核分区（bootfs 分区）
mmcblk1p2	文件系统分区（rootfs 分区）

表 2.2-1 eMMC 分区

2.2.1 更新 uboot 到 eMMC

从 eMMC 启动系统，或者从 SD 卡启动系统来更新 uboot 到 eMMC 中。

光盘路径：开发板光盘 A-基础资料\8、系统镜像\1、出厂系统镜像\1、u-boot 镜像\uboot-imx-2016.03-2.1.0-gd3f0479-v1.4\u-boot-imx6ull-14x14-ddr512-emmc.imx

将该路径下的 u-boot-imx6ull-14x14-ddr512-emmc.imx 文件拷贝到文件系统的/home/root 中。（本文作者是拷贝到了/home/root，如下图，下图的 u-boot-imx6ull-14x14-ddr512-emmc.imx 实际上是一个 u-boot.imx 文件，也可以根据自己的习惯拷贝到其它目录），为后面更新 uboot 做准备。

注：可以参考【正点原子】I.MX6U 用户快速体验 V1.x.pdf 第 2.2.1.1 小节查看 u-boot-imx6ull-14x14-ddr512-emmc.imx 命名的含义。

```
root@ATK-IMX6U:~# pwd
/home/root
root@ATK-IMX6U:~# ls
driver  shell  u-boot-imx6ull-14x14-ddr512-emmc.imx
root@ATK-IMX6U:~#
```

图 2.2.1-1 拷贝 eMMC 所用的 uboot 到文件系统/home/root 目录下

执行下面的指令，先使能 emmc 启动分区，才能进行烧写。

```
echo 0 > /sys/block/mmcblk1boot0/force_ro
```

开始把当前目录下的烧写至 eMMC 的启动分区

```
dd if=u-boot-imx6ull-14x14-ddr512-emmc.imx of=/dev/mmcblk1boot0 bs=1024 seek=1 conv=fsync
```

烧写完成后，关闭要烧写的启动分区

```
echo 1 >/sys/block/mmcblk1boot0/force_ro
```

```
root@ATK-IMX6U:~# echo 0 > /sys/block/mmcblk1boot0/force_ro
root@ATK-IMX6U:~# dd if=u-boot-imx6ull-14x14-ddr512-emmc.imx of=/dev/mmcblk1boot0 bs=1024 seek=1 conv=fsync
371+0 records in
371+0 records out
379904 bytes (380 kB, 371 KiB) copied, 0.0188707 s, 20.1 MB/s
root@ATK-IMX6U:~# echo 1 >/sys/block/mmcblk1boot0/force_ro
root@ATK-IMX6U:~#
```

图 2.2.1-2 烧写 uboot 到 eMMC 的启动分区

最后执行以下指令使能启动分区。

```
mmc bootpart enable 1 1 /dev/mmcblk1
```

```
root@ATK-IMX6U:~# mmc bootpart enable 1 1 /dev/mmcblk1
root@ATK-IMX6U:~#
```

图 2.2.1-3 使能启动分区

这个指令的原型为：

mmc bootpart enable <boot_partition> <send_ack> <device>

分析：

boot_partition 表示 emmc boot 分区，指令里的第一个字符 1 表示 boot_partition 为 1，即 emmc boot 分区 1。

send_ack 表示是否确认应答，0 代表不发送确认应答，1 为在引导操作中发送确认应答。也就是指令中的第二个字符 1。

device 表示设备，指令中为/dev/mmcblk1，即 emmc 设备。

2.2.2 更新设备树到 eMMC

使用 ls 指令查看 eMMC 的 boot 分区设备树所在的目录，如下图，有各种屏对应的设备树，根据个人的屏的使用对应的设备树即可，如果没有屏，默认使用的是 imx6ull-14x14-emmc-4.3-480x272-c.dtb 这个设备树，更新这个设备树即可。

```
ls /run/media/mmcblk1p1
```

```
root@ATK-IMX6U:~# ls /run/media/mmcblk1p1
imx6ull-14x14-nand-10.1-1280x800-c.dtb  imx6ull-14x14-nand-7-1024x600-c.dtb  imx6ull-14x14-nand-vga.dtb
imx6ull-14x14-nand-4.3-480x272-c.dtb    imx6ull-14x14-nand-7-800x480-c.dtb   zImage
imx6ull-14x14-nand-4.3-800x480-c.dtb    imx6ull-14x14-nand-hdmi.dtb
root@ATK-IMX6U:~#
```

图 2.2.2-1 查看 eMMC 出厂的设备树所在的目录

我们可以先拷贝光盘下的设备树文件到开发板中，再拷贝到开发板对应的分区。

光盘路径：开发板光盘 A-基础资料\8、系统镜像\1、出厂系统镜像\2、kernel 镜像\linux-imx-4.1.15-2.1.0-gb78e551-v1.4

把全部的 eMMC 所有的设备树全部拷贝到/run/media/mmcblk1p1 目录下，把下图所有的 eMMC 所使用的设备树拷贝到 eMMC 的 boot 分区（挂载目录为/run/media/mmcblk1p1）。

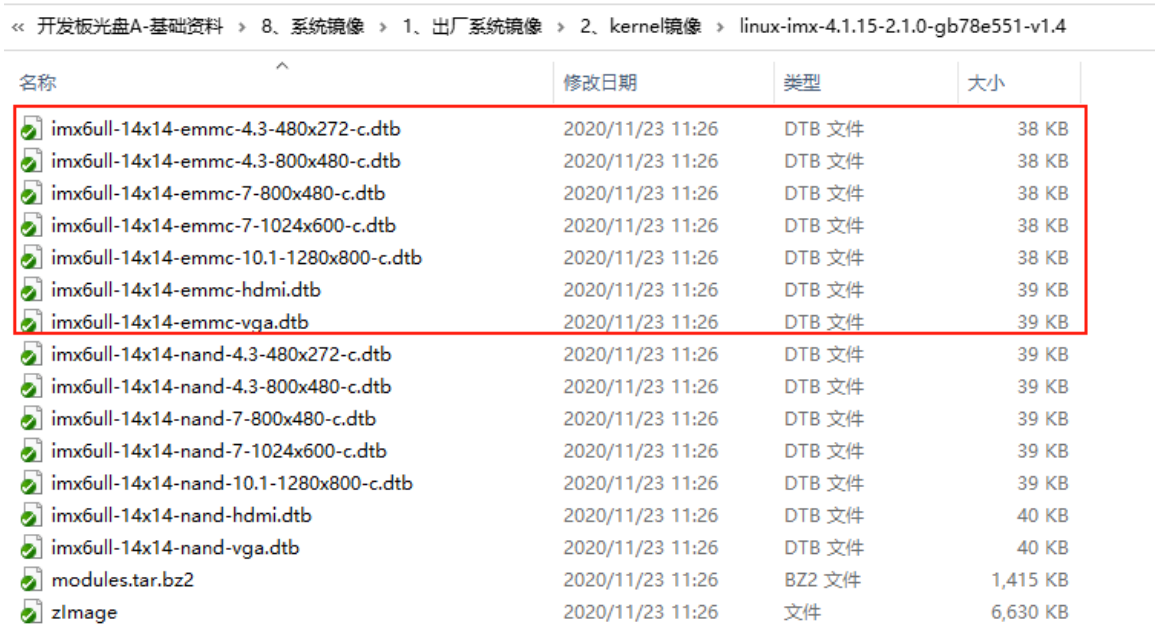


图 2.2.2-2 eMMC 所用的各种分辨率的设备树

把这些设备树文件拷贝到开发板的/home/root 目录下，如下图所示。

```
root@ATK-IMX6U:~# pwd
/home/root
root@ATK-IMX6U:~# ls
driver                                imx6ull-14x14-emmc-7-1024x600-c.dtb  shell
imx6ull-14x14-emmc-10.1-1280x800-c.dtb  imx6ull-14x14-emmc-7-800x480-c.dtb   u-boot-imx6ull-14x14-ddr512-emmc.imx
imx6ull-14x14-emmc-4.3-480x272-c.dtb    imx6ull-14x14-emmc-hdmi.dtb
imx6ull-14x14-emmc-4.3-800x480-c.dtb    imx6ull-14x14-emmc-vga.dtb
```

图 2.2.2-3 拷贝设备树到开发板

把需要更新的设备树拷贝到开发板/run/media/mmcblk1p1 目录下替换掉，执行 ls -l 指令可以查看设备树更新时间来检查是否更新成功，以更新 imx6ull-14x14-emmc-4.3-480x272-c.dtb 为例，如下所示。

```
cp imx6ull-14x14-emmc-4.3-480x272-c.dtb /run/media/mmcblk1p1
ls
```

```
root@ATK-IMX6U:~# cp imx6ull-14x14-emmc-4.3-480x272-c.dtb /run/media/mmcblk1p1/
root@ATK-IMX6U:~# ls /run/media/mmcblk1p1/imx6ull-14x14-emmc-4.3-480x272-c.dtb -l
-rwxrwx--- 1 root disk 38823 Nov 10 11:34 /run/media/mmcblk1p1/imx6ull-14x14-emmc-4.3-480x272-c.dtb
root@ATK-IMX6U:~#
```

图 2.2.2-4 更新设备树

2.2.3 更新内核到 eMMC

我们可以先拷贝光盘下的 zImage 文件拷贝到开发板中，再拷贝到开发板对应的分区。

光盘路径：开发板光盘 A-基础资料\8、系统镜像\1、出厂系统镜像\2、kernel 镜像\linux-imx-4.1.15-2.1.0-gb78e551-v1.4\zImage

将该路径下的 zImage 文件拷贝到开发板/home/root 中，再替换掉开发板的/run/media/mmcblk1p1 中的 zImage 文件即可。

执行以下命令的前提是文件系统/home/root 目录下有 zImage 文件。

cp zImage /run/media/mmcblk1p1/

ls /run/media/mmcblk1p1/zImage -l

```
root@ATK-IMX6U:~# cp zImage /run/media/mmcblk1p1/
root@ATK-IMX6U:~# ls /run/media/mmcblk1p1/zImage -l
-rwxrwx--- 1 root disk 6786368 Nov 10 11:38 /run/media/mmcblk1p1/zImage
root@ATK-IMX6U:~#
```

图 2.2.3-1 更新内核到 eMMC

2.2.4 更新内核模块到 eMMC

我们可以先拷贝光盘下的内核模块包拷贝到开发板中，再解压到开发板对应的目录。

光盘路径：开发板光盘 A-基础资料\8、系统镜像\1、出厂系统镜像\2、kernel 镜像\linux-imx-4.1.15-2.1.0-gb78e551-v1.4\ modules.tar.bz2

将该路径下的内核模块包 modules.tar.bz2 拷贝到开发板/home/root 中，再用 tar 指令解压内核模块到开发板/lib/modules 目录。

```
root@ATK-IMX6U:~# ls
driver                                imx6ull-14x14-emmc-7-1024x600-c.dtb  modules.tar.bz2
imx6ull-14x14-emmc-10.1-1280x800-c.dtb  imx6ull-14x14-emmc-7-800x480-c.dtb   shell
imx6ull-14x14-emmc-4.3-480x272-c.dtb    imx6ull-14x14-emmc-hdmi.dtb          u-boot-imx6ull-14x14-ddr512-emmc.imx
imx6ull-14x14-emmc-4.3-800x480-c.dtb    imx6ull-14x14-emmc-vga.dtb           zImage
root@ATK-IMX6U:~#
```

图 2.2.4-1 将内核模块包传到开发板

执行以下命令的前提是文件系统/home/root 目录下 modules 压缩包。

tar xf modules.tar.bz2 -C /lib/modules/

```
root@ATK-IMX6U:~# pwd
/home/root
root@ATK-IMX6U:~# tar xf modules.tar.bz2 -C /lib/modules/
root@ATK-IMX6U:~#
```

图 2.2.4-2 解压内核模块到/lib/modules 目录

2.2.5 更新文件系统到 eMMC

注意：不能用 eMMC 启动来更新 eMMC 分区里的文件系统。

要使用 SD 卡启动来更新 eMMC 的文件系统！正在运行的一个系统不能把自己给格式化。还需要注意的是在 2.2.3 小节里，因为内核模块是在 eMMC 的第二个分区（rootfs 分区）里的，更新文件系统时会把这个分区里的东西全部删除，请更新文件系统后，自行按照 2.2.4 小节按照步骤来更新内核模块即可。

我们可以先拷贝光盘下的文件系统压缩包拷贝到开发板中（SD 卡启动），再拷贝到开发板对应的分区。

光盘路径：开发板光盘 A-基础资料\8、系统镜像\1、出厂系统镜像\3、文件系统\ rootfs.tar.bz2

将该路径下的 rootfs.tar.bz2 拷贝到开发板/home/root 中，然后执行以下指令清空 eMMC 的第二个分区（rootfs 分区）文件系统分区。

rm -rf /run/media/mmcblk1p2/*

```
root@ATK-IMX6U:~# rm -rf /run/media/mmcblk1p2/*
root@ATK-IMX6U:~#
```

图 2.2.5-1 删除 eMMC 的文件系统分区所挂载的目录下所有内容

然后就可以直接将 rootfs.tar.bz2 文件系统解压至 eMMC 的文件系统分区目录就可以了。执行下面的指令。（注:解压文件系统时间比较长，请耐心等待）

```
tar rootfs.tar.bz2 -C /run/media/mmcblk1p2/
```

```
var/cache/
var/cache/fontconfig/
var/cache/fontconfig/3830d5c3ddfd5cd38a049b759396e72e-le32d8.cache-6
var/cache/fontconfig/CACHEDIR.TAG
var/cache/fontconfig/6ba42ae0000f58711b5caaf10d690066-le32d8.cache-6
var/cache/fontconfig/94322f4d3cdf1bd54794b691285ca062-le32d8.cache-6
var/cache/ldconfig/
var/cache/ldconfig/aux-cache
root@ATK-IMX6U:~#
root@ATK-IMX6U:~# sync
root@ATK-IMX6U:~#
```

图 2.2.5-2 解压文件系统到 eMMC

解压完成后，执行一次 sync 指令，同步一下数据，可防止数据未完全写入。

```
sync
```

2.3 SD 卡更新固件

SD 卡启动卡，更新固件比较简单，我们可以直接在 ubuntu 上操作。**首先确保你已经制作了一张 SD 卡系统启动卡**，可以按【正点原子】I.MX6U 用户快速体验.pdf 制作 SD 卡启动卡。

2.3.1 更新 uboot 到 SD 卡

- 如果核心板所带的存储类型是 eMMC。

将 SD 启动卡连接到 ubuntu 虚拟机，把光盘里的 u-boot-imx6ull-14x14-ddr512-emmc.imx 拷贝到 Ubuntu 中，为后面更新 uboot 做准备。

光盘路径：开发板光盘 A-基础资料\8、系统镜像\1、出厂系统镜像\1、u-boot 镜像\uboot-imx-2016.03-2.1.0-gd3f0479-v1.4\

（备注：可以参考【正点原子】I.MX6U 用户快速体验 V1.x.pdf 第 2.2.1.1 小节查看 u-boot-imx6ull-14x14-ddr512-emmc.imx 命名的含义）

- 如果核心板所带的存储类型是 Nand Flash。

将 SD 启动卡连接到 ubuntu 虚拟机，把光盘里的 u-boot-imx6ull-14x14-ddr256-nand-sd.imx 拷贝到 Ubuntu 中，为后面更新 uboot 做准备。（注意是带有 sd 关键词的 uboot！！！）

光盘路径：开发板光盘 A-基础资料\8、系统镜像\1、出厂系统镜像\1、u-boot 镜像\uboot-imx-2016.03-2.1.0-gd3f0479-v1.4

（备注：可以参考【正点原子】I.MX6U 用户快速体验 V1.x.pdf 第 2.2.1.1 小节查看 u-boot-imx6ull-14x14-ddr256-nand-sd.imx 命名的含义）

本次示范的核心板所带的存储类型为 eMMC。如下图，本文已经复制 u-boot-imx6ull-14x14-ddr512-emmc.imx 到虚拟机目录了。（sd_update 是作者自行创建的）

```
alientek@ubuntu16:~/sd_update$ ls
u-boot-imx6ull-14x14-ddr512-emmc.imx
alientek@ubuntu16:~/sd_update$
```

图 2.3.1-1 将 uboot 拷贝到 Ubuntu

在 ubuntu 上插上连接 SD 卡，使用 fdisk 指令查看 SD 卡挂载的节点。如下图，可以看到 SD 卡所挂载的节点为/dev/sdb。

```
sudo fdisk -l
```

```
alientek@ubuntu16:~/sd_update$ sudo fdisk -l
Disk /dev/sda: 299 GiB, 321048805376 bytes, 627048448 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: 0xd5fbae32

设备       启动     Start    末尾    扇区  Size Id 类型
/dev/sda1  *         2048 625047551 625045504  298G 83 Linux
/dev/sda2       625049598 627046399   1996802  975M  5 扩展
/dev/sda5       625049600 627046399   1996800  975M 82 Linux 交换 / Solaris



Disk /dev/sdb: 14.9 GiB, 15931539456 bytes, 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
Disklabel type: dos
Disk identifier: 0x28add5cc

设备       启动  Start   末尾   扇区  Size Id 类型
/dev/sdb1        20480   282623   262144  128M  c W95 FAT32 (LBA)
/dev/sdb2       282624 31116287 30833664 14.7G 83 Linux
alientek@ubuntu16:~/sd_update$
```

图 2.3.1-2 查看 SD 卡挂载节点

在当前目录下使用 dd 指令将 u-boot-imx6ull-14x14-ddr512-emmc.imx 烧写到 SD 卡中，执行下面的指令。

```
sudo dd if=u-boot-imx6ull-14x14-ddr512-emmc.imx of=/dev/sdb bs=1024 seek=1 conv=fsync
```

```
alientek@ubuntu16:~/sd_update$ sudo dd if=u-boot-imx6ull-14x14-ddr512-emmc.imx of=/dev/
sdb bs=1024 seek=1 conv=fsync
记录了371+0 的读入
记录了371+0 的写出
379904 bytes (380 kB, 371 KiB) copied, 0.0634953 s, 6.0 MB/s
alientek@ubuntu16:~/sd_update$
```

图 2.3.1-3 使用 dd 指令烧写 uboot 到 SD 卡 1K 地址处

至此更新 uboot 成功，需要注意的是：如果你的 SD 卡曾经保存过其他 uboot 环境变量，你需要在 SD 卡时恢复一次环境变量再保存，才能使用新的环境变量启动！

2.3.2 更新设备树到 SD 卡

将 SD 启动卡连接到 ubuntu 虚拟机，把光盘里的设备树拷贝到 Ubuntu 中，为后面更新设备树做准备。（可以根据自己需要的来更新设备树，不一定全部都更新）

光盘路径：开发板光盘 A-基础资料\8、系统镜像\1、出厂系统镜像\2、kernel 镜像\linux-imx-4.1.15-2.1.0-gb78e551-v1.4

```
alientek@ubuntu16:~/sd_update$ ls
fsl-image-qt5-v1.3.tar.bz2              imx6ull-14x14-nand-4.3-480x272-c.dtb
imx6ull-14x14-emmc-10.1-1280x800-c.dtb  imx6ull-14x14-nand-4.3-800x480-c.dtb
imx6ull-14x14-emmc-4.3-480x272-c.dtb    imx6ull-14x14-nand-7-1024x600-c.dtb
imx6ull-14x14-emmc-4.3-800x480-c.dtb    imx6ull-14x14-nand-7-800x480-c.dtb
imx6ull-14x14-emmc-7-1024x600-c.dtb     imx6ull-14x14-nand-hdmi.dtb
imx6ull-14x14-emmc-7-800x480-c.dtb      modules.tar.bz2
imx6ull-14x14-emmc-hdmi.dtb             u-boot-imx6ull-14x14-ddr512-emmc.imx
imx6ull-14x14-emmc-vga.dtb              zImage
imx6ull-14x14-nand-10.1-1280x800-c.dtb
alientek@ubuntu16:~/sd_update$
```

图 2.3.2-1 拷贝设备树到 Ubuntu

SD 卡启动卡接入 Ubuntu 后，会弹出文件管理器，如下图所示，在 boot 分区可看到 SD 卡中的设备树文件，我们可以直接像在 Windows 一样直接把 Ubuntu 中的设备树拖过去更新，也可以执行 sudo cp 命令来更新。

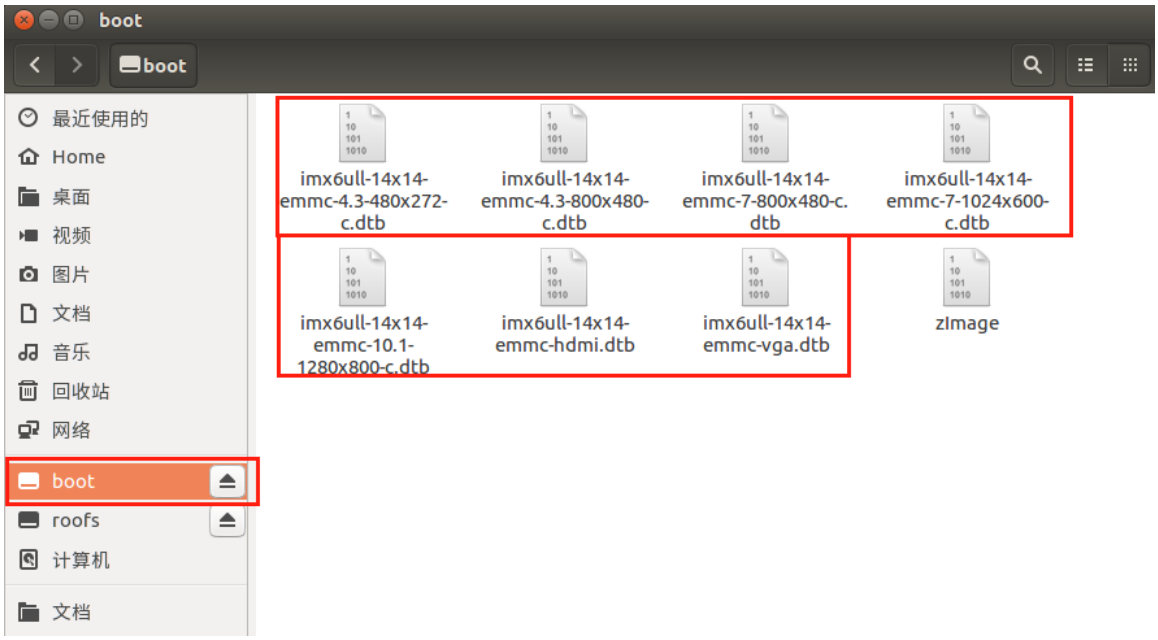


图 2.3.2-2 SD 卡的 boot 分区下的设备树文件

执行 df 指令来查看 SD 卡启动卡的各个分区，这里 SD 卡中设备树所在的分区路径是/media/alientek/boot，使用 sudo cp 指令即可拷贝设备树进行更新。

```
alientek@ubuntu16:~/sd_update$ df
文件系统           1K-块     已用      可用 已用% 挂载点
udev             4041816        0   4041816    0% /dev
tmpfs             814472    30208    784264    4% /run
/dev/sda1      307486840 79871900 211972420   28% /
tmpfs            4072340      212   4072128    1% /dev/shm
tmpfs               5120        0      5120    0% /run/lock
tmpfs            4072340        0   4072340    0% /sys/fs/cgroup
tmpfs             814472       52    814420    1% /run/user/1000
/dev/sdb1         129039     6897    122143    6% /media/alientek/boot
/dev/sdb2       15043544   462320  13810384    4% /media/alientek/roofs
alientek@ubuntu16:~/sd_update$ sudo cp imx6ull-14x14-emmc-4.3-800x480-c.dtb /media/alientek/roofs/
[sudo] alientek 的密码：
alientek@ubuntu16:~/sd_update$
```

图 2.3.2-3 更新设备树

2.3.3 更新内核到 SD 卡

将 SD 启动卡连接到 ubuntu 虚拟机，把光盘里的 zImage 文件拷贝到 Ubuntu 中，为后面更新内核做准备。

光盘路径：开发板光盘 A-基础资料\8、系统镜像\1、出厂系统镜像\2、kernel 镜像\linux-imx-4.1.15-2.1.0-gb78e551-v1.4\zImage

SD 卡启动卡接入 Ubuntu 后，会弹出文件管理器，如下图所示，在 boot 分区可看到 SD 卡中的 zImage 文件，我们可以直接像在 Windows 一样直接把 Ubuntu 中的 zImage 文件拖过去更新，也可以执行 sudo cp 命令来更新。

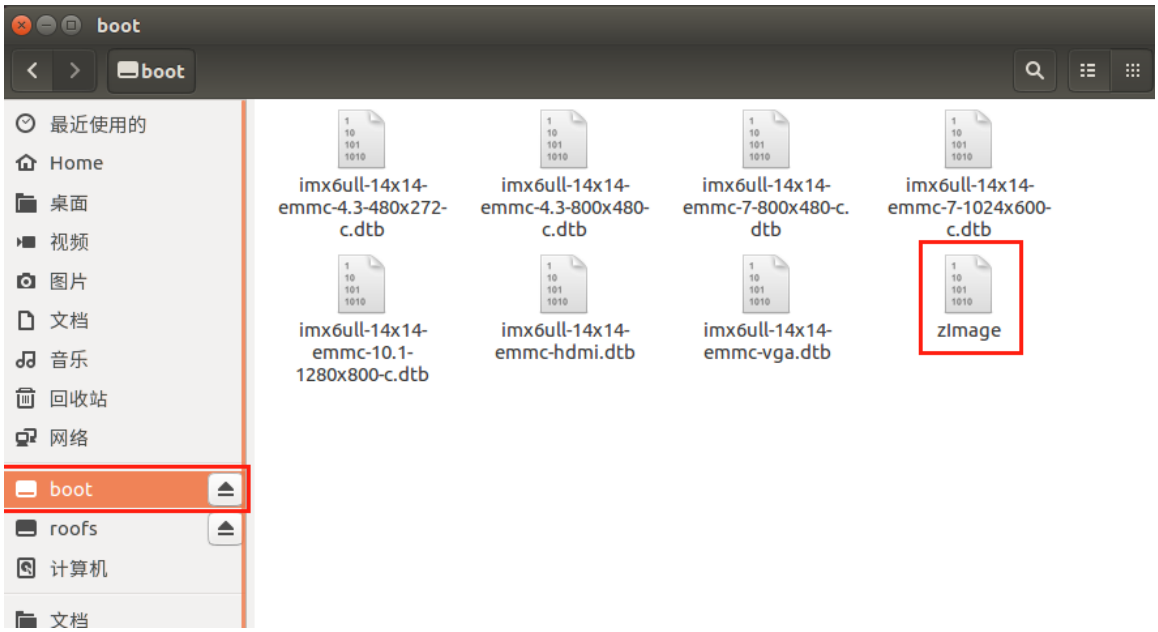


图 2.3.3-1 SD 卡的 boot 分区下的内核 zImage

2.3.4 更新内核模块到 SD 卡

同理，只需要将光盘的 modules.tar.bz2 拷贝到 Ubuntu，解压更新到 SD 卡的 rootfs 分区下的/lib/modules 目录即可。

光盘路径：开发板光盘 A-基础资料\8、系统镜像\1、出厂系统镜像\2、kernel 镜像\linux-imx-4.1.15-2.1.0-gb78e551-v1.4\ modules.tar.bz2

```
sudo tar vxjf modules.tar.bz2 -C /media/alientek/roofs/lib/modules/
```

```
alientek@ubuntu16:~/sd_update$ sudo tar vxjf modules.tar.bz2 -C /media/alientek/roofs/lib/modules/
[sudo] alientek 的密码:
./
./4.1.15-gb8ddbbc/
./4.1.15-gb8ddbbc/modules.devname
./4.1.15-gb8ddbbc/modules.alias
./4.1.15-gb8ddbbc/modules.dep.bin
./4.1.15-gb8ddbbc/modules.symbols.bin
./4.1.15-gb8ddbbc/modules.order
./4.1.15-gb8ddbbc/modules.builtin.bin
./4.1.15-gb8ddbbc/kernel/
```

图 2.3.4-1 更新内核模块

2.3.5 更新文件系统到 SD 卡

把光盘里的 rootfs.tar.bz2 文件拷贝到 Ubuntu 中，为后面更新文件系统做准备。

光盘路径：开发板光盘 A-基础资料\8、系统镜像\1、出厂系统镜像\3、文件系统\ rootfs.tar.bz2

```
alientek@ubuntu16:~/sd_update$ ls
fsl-image-qt5-v1.3.tar.bz2              imx6ull-14x14-nand-4.3-480x272-c.dtb
imx6ull-14x14-emmc-10.1-1280x800-c.dtb  imx6ull-14x14-nand-4.3-800x480-c.dtb
imx6ull-14x14-emmc-4.3-480x272-c.dtb    imx6ull-14x14-nand-7-1024x600-c.dtb
imx6ull-14x14-emmc-4.3-800x480-c.dtb    imx6ull-14x14-nand-7-800x480-c.dtb
imx6ull-14x14-emmc-7-1024x600-c.dtb     imx6ull-14x14-nand-hdmi.dtb
imx6ull-14x14-emmc-7-800x480-c.dtb      modules.tar.bz2
imx6ull-14x14-emmc-hdmi.dtb             rootfs.tar.bz2
imx6ull-14x14-emmc-vga.dtb              u-boot-imx6ull-14x14-ddr512-emmc.imx
imx6ull-14x14-nand-10.1-1280x800-c.dtb  zImage
alientek@ubuntu16:~/sd_update$
```

图 2.3.5-1 拷贝文件系统包到 Ubuntu

使用 df 指令查看 SD 卡的挂载路径，如下图，本文的挂载路径为/media/alientek/rootfs ，请根据个人 SD 卡挂载的目录具体填写

```
alientek@ubuntu16:~/sd_update$ df
文件系统           1K-块      已用      可用 已用% 挂载点
udev             4041816         0   4041816    0% /dev
tmpfs             814472      9748    804724    2% /run
/dev/sda1      307486840  79869964 211974356   28% /
tmpfs            4072340       252   4072088    1% /dev/shm
tmpfs               5120         0      5120    0% /run/lock
tmpfs            4072340         0   4072340    0% /sys/fs/cgroup
tmpfs             814472        64    814408    1% /run/user/1000
/dev/sdb1         129039      6897    122143    6% /media/alientek/boot
/dev/sdb2       15043544    531744  13740960    4% /media/alientek/rootfs
alientek@ubuntu16:~/sd_update$
```

图 2.3.5-2 查看 SD 卡文件系统挂载的目录

首先先删除 rootfs 分区下的根文件系统，执行如下指令。

```
sudo rm -rf /media/alientek/rootfs/*
ls /media/alientek/rootfs/
```

```
alientek@ubuntu16:~/sd_update$ sudo rm -rf /media/alientek/rootfs/*
alientek@ubuntu16:~/sd_update$ ls /media/alientek/rootfs/
alientek@ubuntu16:~/sd_update$
```

图 2.3.5-3 删除 SD 卡的 rootfs 分区下的文件系统

然后将 rootfs.tar.bz2 解压至 SD 卡根文件系统分区即可，执行下面的指令。

```
sudo tar xf rootfs.tar.bz2 -C /media/alientek/rootfs/
ls /media/alientek/rootfs/
sync
```