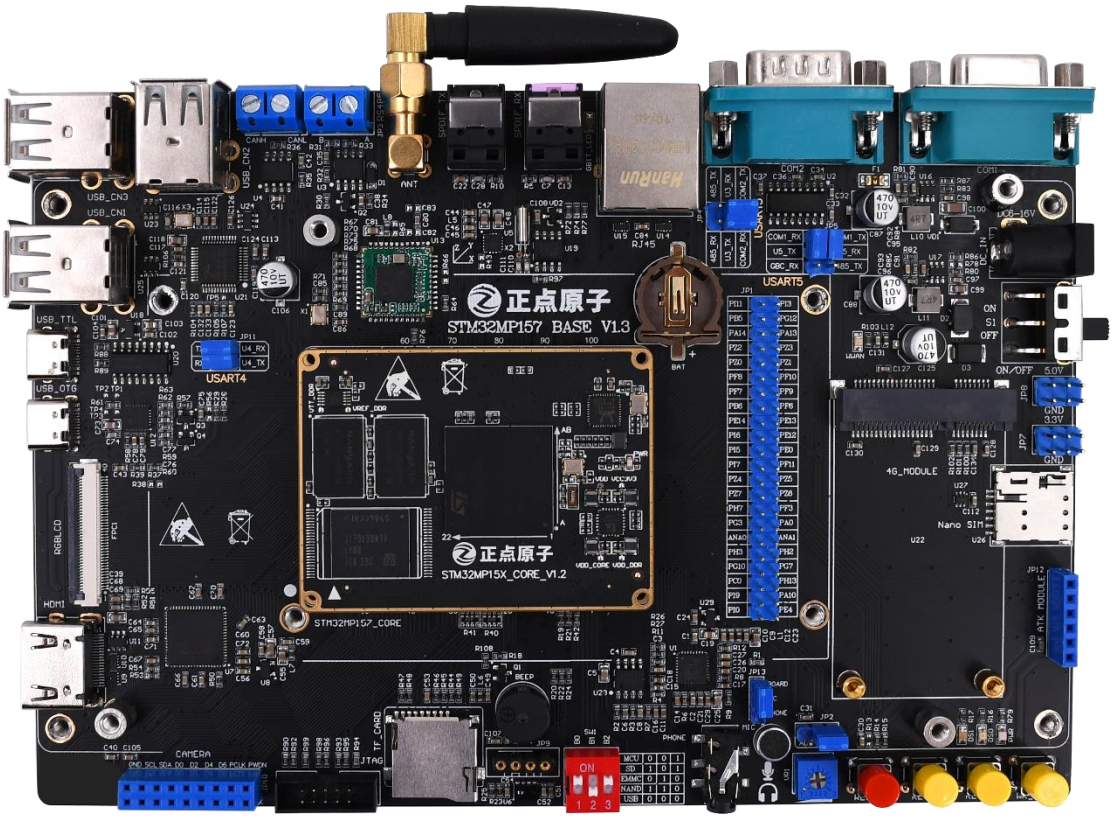
STM32MP157 文件传输及固件更新手册 V1.0

-正点原子 STM32MP157



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

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	2020.11.05

目录

前言

本文档主要用于引导正点原子 Linux 用户如何进行开发板与电脑端的文件互传，以及如何更新开发板的固件。

注意事项：本文档使用的是出厂的镜像、出厂的文件系统，正点原子已经提供在开发板光盘 A-基础资料\8、系统镜像\2、出厂系统镜像。如果是教程的源码或其他的源码，请自行参考！文档中的操作环境基于 Windows10 系统、正点原子提供的 Ubuntu18.04 系统。

第一章 Linux 开发板文件拷贝篇

我们在使用正点原子 STM32MP157 开发平台进行学习的时候，会使用到 Windows 和 Linux 两种操作系统。在拷贝文件方面，Windows 系统直接使用鼠标拖动文件或者使用快捷键就可以了；带有图形界面的 Linux 系统也可以像 Windows 系统那样拷贝文件，但更多时候使用的是终端命令行来拷贝文件。下面就介绍几种我们在学习或者开发时经常使用到的文件拷贝方法。

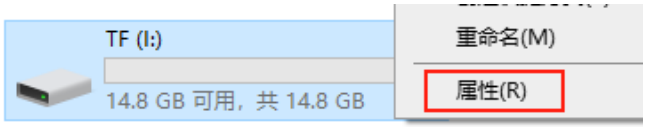
1.1 Linux 通过 U 盘或者 SD 卡拷贝文件

1.1.1 Linux 开发板通过 U 盘/SD 卡拷贝文件

硬件准备：STM32MP157 开发板（出厂系统）、U 盘（FAT 格式，不能用 NTFS 格式）。如果没有 U 盘，可以使用 TF 卡接上读卡器，TF 卡全称 Trans-flash Card，2004 年正式更名为 Micro SD Card，所以有时候会习惯性的称为 SD 卡，以下称 SD 卡的，都是指 TF 卡。这里我使用的是 SD 卡+读卡器，相当于一个 U 盘，如下图所示。

1.1.1-1 SD 卡和读卡器

如何确认 U 盘格式：将 U 盘挂载到 Windows 系统中，打开此电脑，找到 U 盘设备，右键选择属性。这里我的 SD 卡设备名是 TF，以自己实际情况为准。

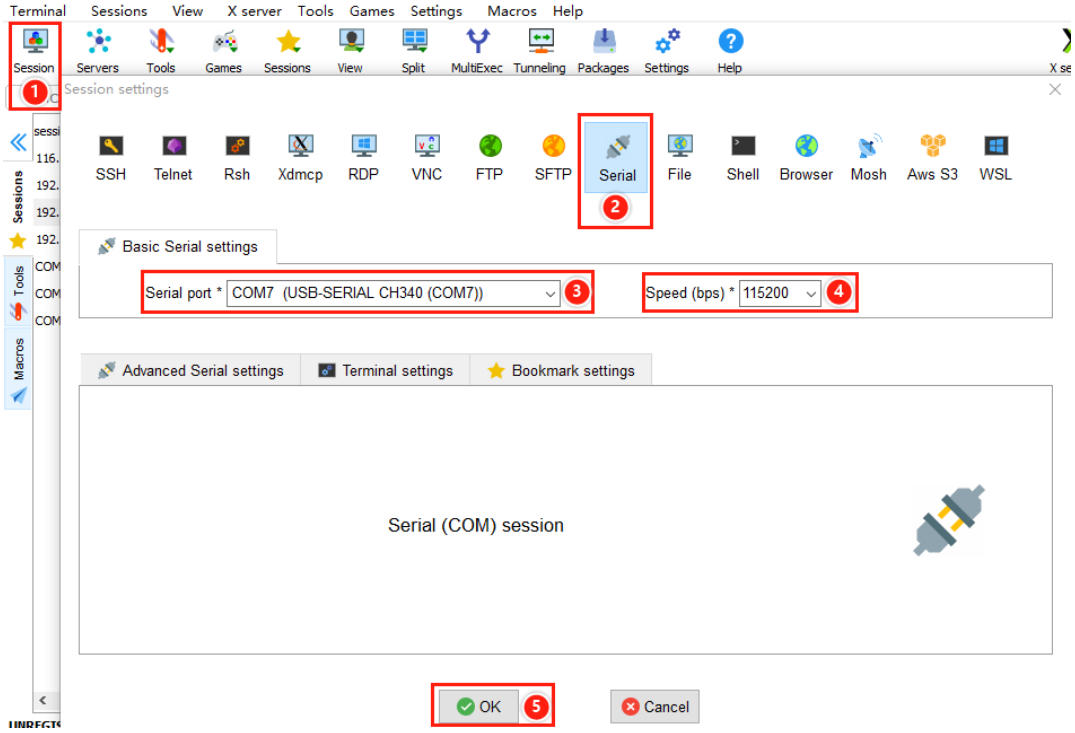


1.1.1-2 查看 SD 卡属性

在 常规 选项卡中可以看到 文件系统：FAT32。表示 U 盘的格式为 FAT32。

1.1.1-3 查看 SD 卡格式

开发板拨码开关选择 EMMC 模式，板子 USB_TTL 接口接到 Windows 系统电脑的 USB 端。如果没有安装 CH340 的驱动则需要手动安装它，正点原子提供的软件包里（开发板光盘 A-基础资料\3、软件）有这个驱动和串口终端软件。这里使用的串口终端是 MobaXterm，串口终端设置步骤如下。



1.1.1-4 串口终端设置

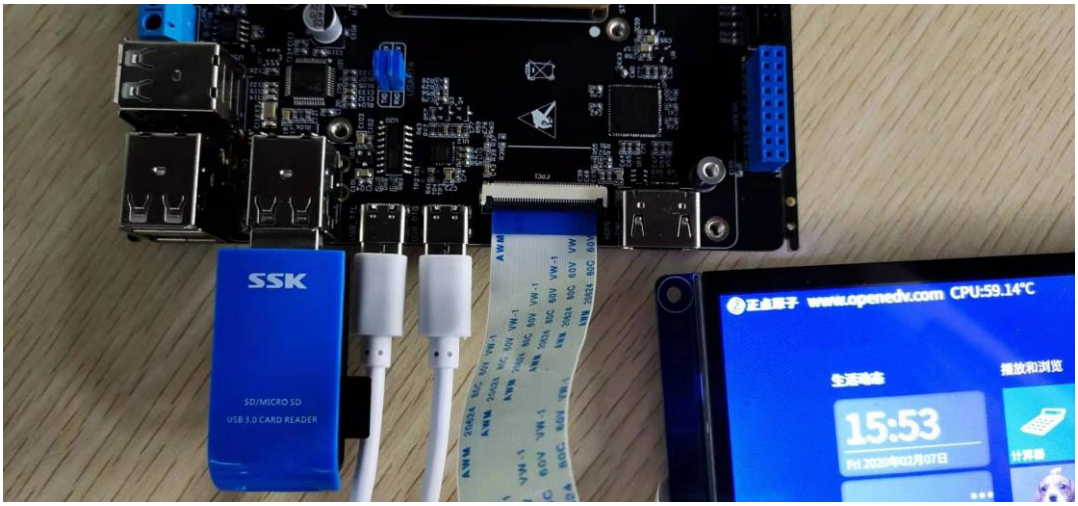
开发板上电启动系统后，在串口终端里输入指令 df 查看当前挂载的内容。

```
df
```

```
root@ATK-stm32mp1:~# df
Filesystem     1K-blocks    Used Available Use% Mounted on
devtmpfs          377328       0    377328   0% /dev
/dev/mmcblk2p3   7433416  814332   6290800  12% /
tmpfs             445152      64    445088   1% /dev/shm
tmpfs             445152    9076    436076   3% /run
tmpfs             445152       0    445152   0% /sys/fs/cgroup
tmpfs             445152       0    445152   0% /tmp
tmpfs             445152     140    445012   1% /var/volatile
/dev/mmcblk2p2     59365   13084     41695  24% /run/media/mmcblk2p2
tmpfs              89028       4     89024   1% /run/user/0
```

1.1.1-5 插入 SD 卡前的挂载内容

将带有 SD 卡的读卡器接入开发板的 USB 接口，如图所示。



1.1.1-6 开发板的 USB 接口接上读卡器

此时，开发板串口终端打印信息如下，可以看到 SD 卡的实际大小、节点等信息，这里我 SD 卡设备的挂载节点为 sda。

```
root@ATK-stm32mp1:~# [   55.837605] usb 2-1.5: new high-speed USB device number 8 using ehci-pla
tform
[   55.892949] usb-storage 2-1.5:1.0: USB Mass Storage device detected
[   55.928052] scsi host0: usb-storage 2-1.5:1.0
[   56.539492] usbcore: registered new interface driver uas
[   56.969012] scsi 0:0:0:0: Direct-Access     Generic  STORAGE DEVICE   1532 PQ: 0 ANSI: 6
[   56.979891] sd 0:0:0:0: Attached scsi generic sg0 type 0
[   57.269846] sd 0:0:0:0: [sda] 31116288 512-byte logical blocks: (15.9 GB/14.8 GiB)
[   57.279812] sd 0:0:0:0: [sda] Write Protect is off
[   57.288862] sd 0:0:0:0: [sda] Write cache: disabled, read cache: enabled, doesn't support DPO
 or FUA
[   57.378038]  sda: sda1
[   57.384968] sd 0:0:0:0: [sda] Attached SCSI removable disk
```

1.1.1-7 识别到 SD 卡的大小信息

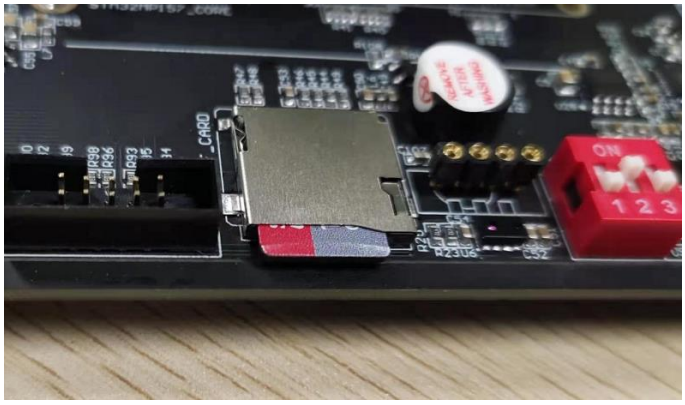
再使用 df 指令查看挂载的节点，可以看到/dev/sda1 挂载的目录为/run/media/sda1。这里我的 SD 卡只有一个分区，所以只挂载了一个 sda1 分区，如果你的 SD 卡有两个分区，挂载的目录有两个分别是 sda1 和 sda2。如果你还有一个 U 盘或 SD 卡，再接到开发板的 USB 接口，它的挂载节点会是 sdb*了，以此类推。

```
df
```

```
root@ATK-stm32mp1:~# df
Filesystem     1K-blocks    Used Available Use% Mounted on
devtmpfs          377328       0    377328   0% /dev
/dev/mmcblk2p3   7433416  814440   6290692  12% /
tmpfs             445152      64    445088   1% /dev/shm
tmpfs             445152    9100    436052   3% /run
tmpfs             445152       0    445152   0% /sys/fs/cgroup
tmpfs             445152       0    445152   0% /tmp
tmpfs             445152     140    445012   1% /var/volatile
/dev/mmcblk2p2     59365   13084     41695  24% /run/media/mmcblk2p2
tmpfs              89028       4     89024   1% /run/user/0
/dev/sda1       15541408      32  15541376   1% /run/media/sda1
root@ATK-stm32mp1:~#
```

1.1.1-8 SD 卡挂载的节点与挂载的目录等信息

开发板上有一个 SD 卡的卡槽，我们也可以把 SD 卡接入这个卡槽，这里我再用另一张 SD 卡接入。



1.1.1-9 SD 卡接入开发板卡槽

在串口终端可以看到接入 SD 卡后的信息。可以看 SD 卡槽的 SD 卡设备的挂载节点是 mmcblk1。

```
root@ATK-stm32mp1:~# [ 7606.349141] mmc1: host does not support reading read-only switch, assumi
ng write-enable
[ 7606.364718] mmc1: new high speed SDHC card at address aaaa
[ 7606.381200] mmcblk1: mmc1:aaaa SC16G 14.8 GiB
[ 7606.391067]  mmcblk1: p1
```

1.1.1-10 SD 卡接入卡槽后串口打印信息

再使用 df 指令查看挂载的节点

df

```
root@ATK-stm32mp1:~# df
Filesystem     1K-blocks    Used Available Use% Mounted on
devtmpfs          377328       0    377328   0% /dev
/dev/mmcblk2p3   7433416  814496   6290636  12% /
tmpfs             445152      64    445088   1% /dev/shm
tmpfs             445152    9112    436040   3% /run
tmpfs             445152       0    445152   0% /sys/fs/cgroup
tmpfs             445152       0    445152   0% /tmp
tmpfs             445152     156    444996   1% /var/volatile
/dev/mmcblk2p2     59365   13084     41695  24% /run/media/mmcblk2p2
tmpfs              89028       4     89024   1% /run/user/0
/dev/sda1       15541408      32  15541376   1% /run/media/sda1
/dev/mmcblk1p1  15549952     128  15549824   1% /run/media/mmcblk1p1
root@ATK-stm32mp1:~#
```

1.1.1-11 SD 卡接入卡槽后挂载的节点与挂载的目录等信息

可以看到 SD 卡接到卡槽后，挂载的节点是/dev/mmcblk1p1，在 USB 端接入的节点是/dev/sda1。mmcblk1 或者 sda 就是 SD 卡设备，mmcblk1p1 或者 sda1 就是 SD 卡的第一个分区，这里我的 SD 卡只有一个分区。

我们进入 SD 卡挂载的目录，即进入/run/media/sda1 下去新建、拷贝文件。直接使用 cd 命令进入/run/media/sda1 目录下。使用 ls 指令即可查看读卡器 SD 卡里的内容，如下图所示。

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

```
root@ATK-stm32mp1:~# cd /run/media/sda1/
root@ATK-stm32mp1:/run/media/sda1# ls
ATK  OK.txt  System Volume Information  ????.txt
root@ATK-stm32mp1:/run/media/sda1#
```

1.1.1-12 查看读卡器 SD 卡下的内容

可以看到，在我的读卡器 SD 卡里有几个文件，其中有 ???? 的文件，因为文件系统字符终端部显示中文。

这里新建一个文件，然后将这个文件拷贝到家目录下（/home/root）。

使用 touch 指令创建一个 test 文件，使用 mkdir 指令创建一个 test-lib 文件夹，然后把这个 test 文件和 test-lib 文件夹拷贝到/home/root 目录下，如下图所示。

```
touch test
mkdir test-lib
```

```
root@ATK-stm32mp1:/run/media/sda1# touch test
root@ATK-stm32mp1:/run/media/sda1# mkdir test-lib
root@ATK-stm32mp1:/run/media/sda1# ls
ATK  OK.txt  System Volume Information  test  test-lib  ????.txt
root@ATK-stm32mp1:/run/media/sda1#
```

1.1.1-13 新建 test 文件和 test-lib 文件夹

使用 cp 指令拷贝 test 文件到/home/root 目录下，也可以使用 mv 指令（移动/重命名），把它移动到/home/root 目录下。这里使用 cp 指令（拷贝文件夹用 cp -r）。拷贝过去后，使用 ls 指令查看/home/root 目录下有 test 这个文件，说明拷贝成功。

```
cp test /home/root
cp -r test-lib /home/root
ls /home/root
```

```
root@ATK-stm32mp1:/run/media/sda1# cp test /home/root
root@ATK-stm32mp1:/run/media/sda1# cp -r test-lib /home/root
root@ATK-stm32mp1:/run/media/sda1# ls /home/root/
helloworld.mp3  shell  test  test-lib
```

1.1.1-14 拷贝 test 文件和 test-lib 文件夹到/home/root 目录下

在退出 SD 卡或者 U 盘前，为了数据写入写出完整，需要在终端命令下执行 sync 指令来同步数据。

```
sync
```

```
root@ATK-stm32mp1:/run/media/sda1# sync
```

1.1.1-15 同步数据

退出 SD 卡或者 U 盘时，需要先退出这个挂载的目录（就是不要在挂载的目录下去卸载目录），然后使用 umount 指令去卸载这个挂载的目录就可以退出了。

使用 cd ~ 执行返回到家目录，再使用 umount /run/media/sda1 卸载这个 sda1 目录。然后使用 df 指令查看 sda1 是否已经不在了。

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

```
root@ATK-stm32mp1:/run/media/sda1# cd ~
root@ATK-stm32mp1:~# umount /run/media/sda1
root@ATK-stm32mp1:~# df
Filesystem     1K-blocks    Used Available Use% Mounted on
devtmpfs          377328       0    377328   0% /dev
/dev/mmcblk2p3   7433416  813352   6291780  12% /
tmpfs             445152      64    445088   1% /dev/shm
tmpfs             445152    9104    436048   3% /run
tmpfs             445152       0    445152   0% /sys/fs/cgroup
tmpfs             445152       0    445152   0% /tmp
tmpfs             445152     140    445012   1% /var/volatile
/dev/mmcblk2p2     59365   13084     41695  24% /run/media/mmcblk2p2
/dev/mmcblk1p1  15549952     128  15549824   1% /run/media/mmcblk1p1
tmpfs              89028       4     89024   1% /run/user/0
```

1.1.1-16 卸载 SD 卡读卡器

可以看到 sda1 这个目录已经不存在了，表面已经卸载了，SD 卡读卡器或 U 盘就可以正常取下。

卸载 SD 卡槽上的 SD 卡也是一样的道理，执行以下指令卸载卡槽上的 SD 卡。

```
umount /run/media/ mmcblk1p1
df
```

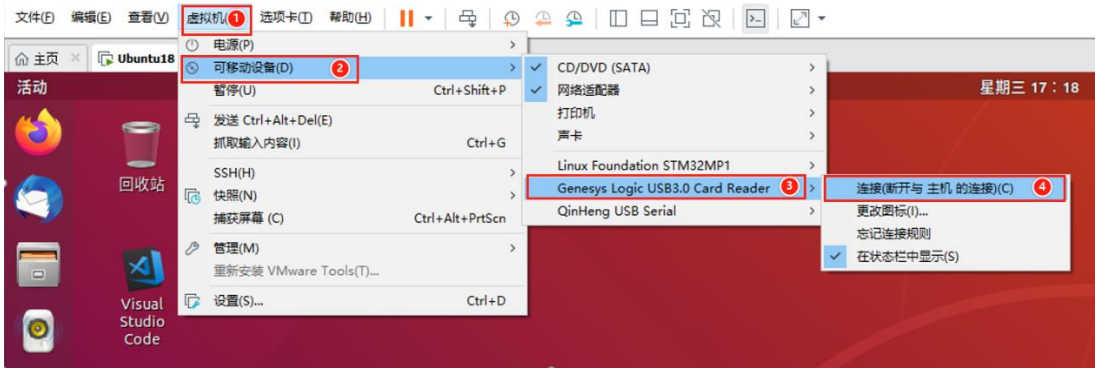
```
root@ATK-stm32mp1:~# umount /run/media/mmcblk1p1/
root@ATK-stm32mp1:~# df
Filesystem     1K-blocks    Used Available Use% Mounted on
devtmpfs          377328       0    377328   0% /dev
/dev/mmcblk2p3   7433416  813360   6291772  12% /
tmpfs             445152      64    445088   1% /dev/shm
tmpfs             445152    9100    436052   3% /run
tmpfs             445152       0    445152   0% /sys/fs/cgroup
tmpfs             445152       0    445152   0% /tmp
tmpfs             445152     144    445008   1% /var/volatile
/dev/mmcblk2p2     59365   13084     41695  24% /run/media/mmcblk2p2
tmpfs              89028       4     89024   1% /run/user/0
```

1.1.1-17 卸载卡槽上的 SD 卡

1.1.2 Ubuntu 下通过 U 盘/SD 卡拷贝文件

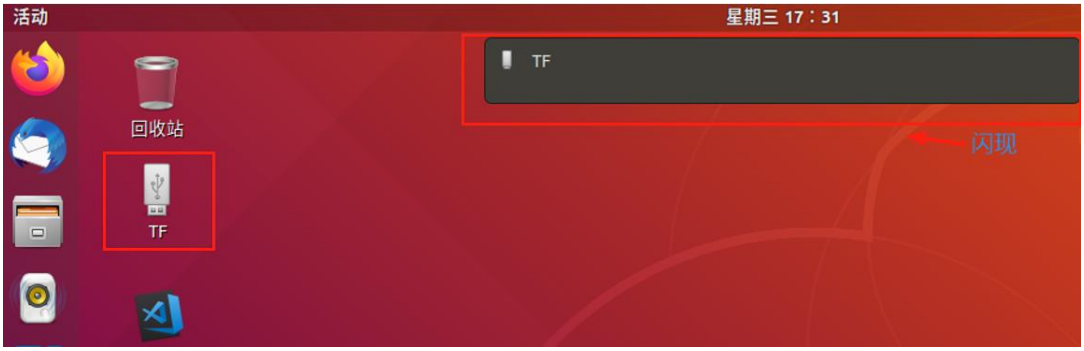
本文档使用的 Ubuntu 版本是 Ubuntu18.04，虚拟机软件是 VMware® Workstation 15 Pro，这些可以在正点原子的资料盘中可以找到，其他版本的可能图标有所差异。这里使用 1.1.1 小节的 SD 卡+读卡器的组合，相当于一个 U 盘。

将 SD 卡接入读卡器，读卡器接入电脑的 USB 接口上，在这个过程中，如果电脑上开着虚拟机，可能会提示是否连接到虚拟机，选择是就可以了。如果没有弹出提示，我们也可以手动挂载到虚拟机中。选择虚拟机菜单栏的 虚拟机 选项卡，依次选择 可移动设备 -> USB3.0 Card Reader -> 连接。



1.1.2-1 手动将 SD 卡连接到虚拟机

SD 卡接入虚拟机 Ubuntu 后，会在 Ubuntu 桌面闪现一个提示框，并且会在桌面出现一个图标，这里的 TF 是我 SD 卡的设备名字，如下图所示。

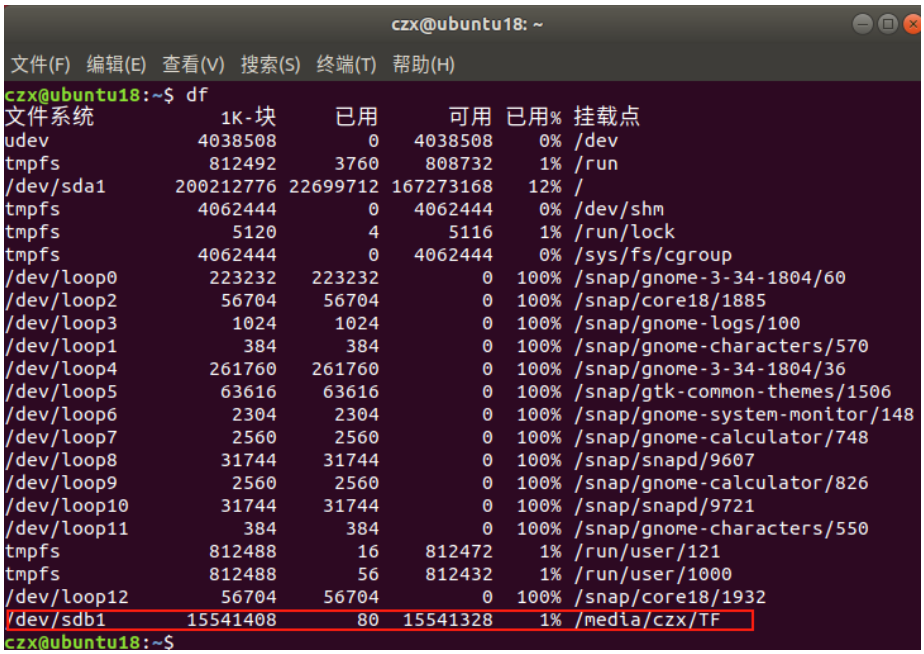


1.1.2-2 桌面出现设备图标

双击这个设备图标就打开了文件管理窗口，显示的就是 SD 卡里的内容。使用鼠标就可以拖动文件拷贝到 Ubuntu 桌面上，也可以右键进行拷贝粘贴，和 Windows 的方法差不多。这里我们主要讲下在终端使用指令来拷贝/移动。其中，cp 指令用于拷贝，mv 指令用于移动。

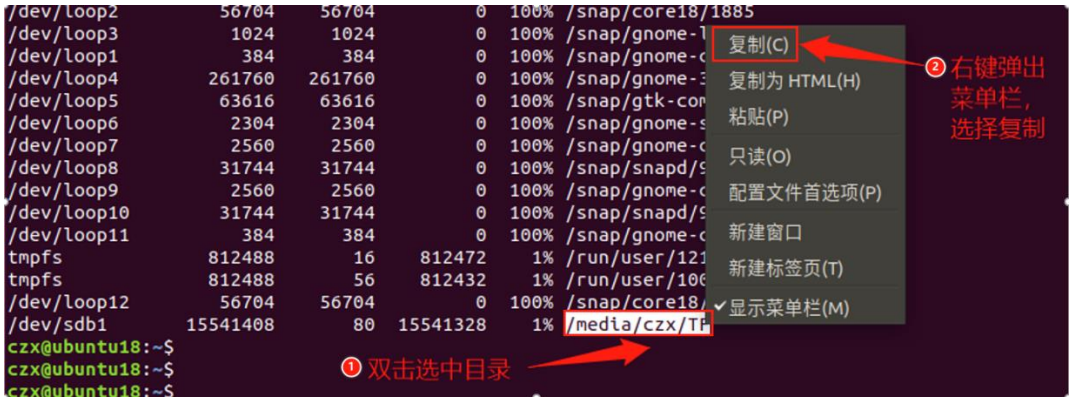
在 Ubuntu 桌面按下 Ctrl+Alt+T 快捷键，弹出终端命令行，或者鼠标右键也可以选择打开终端。使用 df 指令来查看 SD 卡挂载的目录。

```
df
```



1.1.2-3 使用 df 指令查看 SD 卡挂载的节点与目录等信息

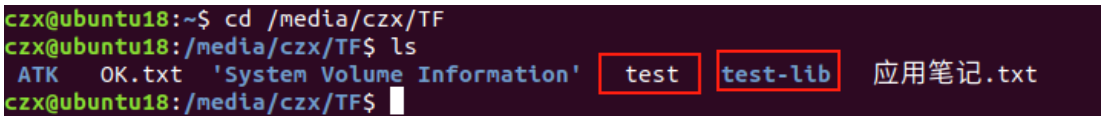
我们直接双击就可以选中这个目录的路径，然后右键选择复制，就复制了这个路径名。



1.1.2-4 双击复制路径名

输入 cd 指令，然后按 Ctrl+Shift+V 快捷键粘贴这个路径。再用 ls 指令查看当前目录下的文件。可以看到 test 文件和 test-lib 文件夹，这是在 1.1.1 小节中从开发板文件系统里创建拷贝出来的，现在把它拷贝到 Ubuntu 里。

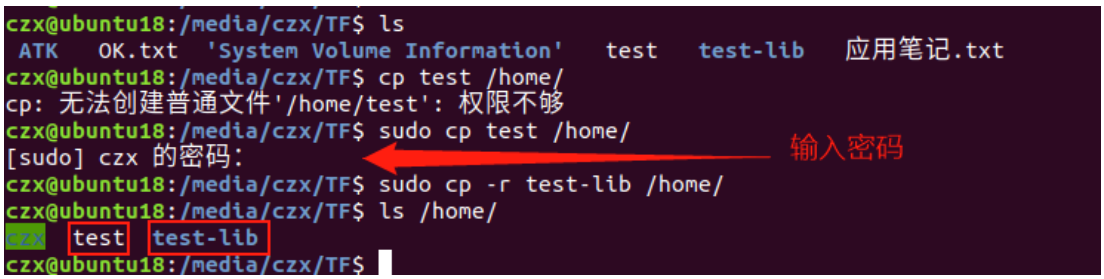
```
cd SD 卡的挂载路径
ls
```



1.1.2-5 查看 test 文件和 test-lib 文件夹

使用 cp/mv 指令进行拷贝或者移动文件或文件夹，普通用户拷贝/移动到 /home/ 这样的目录需要权限，需要在指令前面加上 sudo，再输入密码，即可拷贝/移动文件。

```
sudo cp test /home/
sudo cp -r test-lib /home/
```



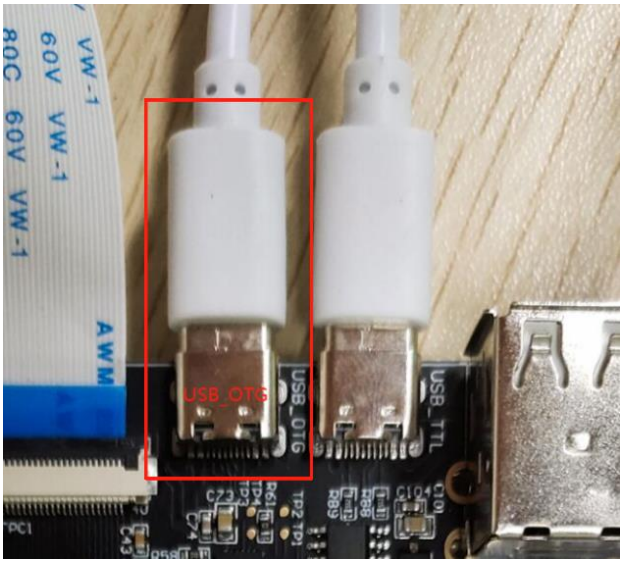
1.1.2-6 拷贝 test 文件和 test-lib 文件夹到/home 目录下

1.2 开发板通过 scp 指令拷贝文件

开发板除了有 USB 接口、SD 卡槽，还有网口，我们可以通过网络来传输文件。SMT32MP157 底板上的 USB_OTG 接口，可当做 USB 网络使用（局域网），我们也可以通过这个局域网来传输文件。网络传输就会经常用到 scp 指令，下面我们就介绍写如何使用 scp 指令来进行网络传输。

1.2.1 使用 OTG 网络拷贝文件

我们使用一根原子提供的 USB Type-C 连接线将开发板 USB_OTG 接口和电脑 USB 接口连接在一起。如下图所示。



1.2.1-1 USB_OTG 连接电脑

连接好后启动开发板，USB_OTG 会在系统启动后生成一个 usb0 网络节点，我们可以使用 ifconfig 指令查看。

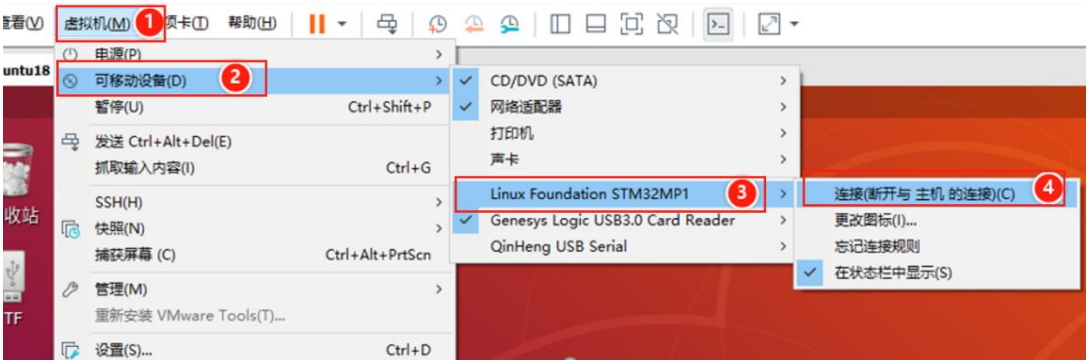
```
root@ATK-stm32mp1:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 6A:61:83:58:18:CA
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
          Interrupt:55 Base address:0xa000

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:80 errors:0 dropped:0 overruns:0 frame:0
          TX packets:80 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:6080 (5.9 KiB)  TX bytes:6080 (5.9 KiB)

usb0      Link encap:Ethernet  HWaddr 82:E7:22:49:61:8B
          inet addr:192.168.7.1  Bcast:192.168.7.255  Mask:255.255.255.0
          inet6 addr: fe80::80e7:22ff:fe49:618b/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:434 errors:0 dropped:0 overruns:0 frame:0
          TX packets:55 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:77519 (75.7 KiB)  TX bytes:11264 (11.0 KiB)
```

1.2.1-2 生成的 usb0 网络节点

打开虚拟机 Ubuntu 系统，按照如下步骤操作，将电脑识别的可移动设备连接到虚拟机上。



1.2.1-3 将 USB_OTG 设备连接到 Ubuntu 虚拟机

在 Ubuntu 终端和开发板串口终端各执行一次 ifconfig 指令，可以看到 Ubuntu 上生成了一个网络节点，此时开发板 usb0 网络节点获得了一个 IP，它和刚刚 Ubuntu 生成的网络节点在同一个网段，所以它们构成了一个局域网，可以通过网络进行通信或者传输文件等操作。

```
czx@ubuntu18:~$ ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.199  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::e1fd:ef0a:940:f249  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:a2:81:bc  txqueuelen 1000  (以太网)
        RX packets 326184  bytes 93183638 (93.1 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 20272  bytes 1807319 (1.8 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

enx90c0512c66e0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.7.254  netmask 255.255.255.0  broadcast 192.168.7.255
        inet6 fe80::fcc4:c522:b008:fe7e  prefixlen 64  scopeid 0x20<link>
        ether 90:c0:51:2c:66:e0  txqueuelen 1000  (以太网)
        RX packets 20  bytes 2996 (2.9 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 19  bytes 3778 (3.7 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

1.2.1-4 Ubuntu 系统获取的 USB_OTG 的 IP 地址

```
root@ATK-stm32mp1:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 6A:61:83:58:18:CA
          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)
          Interrupt:55 Base address:0xa000

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:368 errors:0 dropped:0 overruns:0 frame:0
          TX packets:368 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:22904 (22.3 KiB)  TX bytes:22904 (22.3 KiB)

usb0      Link encap:Ethernet  HWaddr 46:B0:89:F8:5D:A1
          inet addr:192.168.7.1  Bcast:192.168.7.255  Mask:255.255.255.0
          inet6 addr: fe80::44b0:89ff:fef8:5da1/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:790 errors:0 dropped:0 overruns:0 frame:0
          TX packets:93 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:110939 (108.3 KiB)  TX bytes:23150 (22.6 KiB)
```

1.2.1-5 查看开发板获取的 USB_OTG 网络 IP 地址

这里我的 Ubuntu 下有个 test 文件夹，里面有个 test.c 文件，我们现在把这个 test.c 文件传到开发板的/home/root 目录。格式指令如下：

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
```

示例：

```
scp test.c root@192.168.7.1:/home/root
```

指令格式分析：

1) scp：scp 指令
2) test.c：要传输的文件
3) root：用户名，开发板默认最高权限用户 root
4) @：地址符
5) 192.168.7.1：开发板 IP
6) ：注意这里有个英文冒号
7) /home/root：要传输到开发板的路径

```
czx@ubuntu18:~/test$ scp test.c root@192.168.7.1:/home/root
The authenticity of host '192.168.7.1 (192.168.7.1)' can't be established.
RSA key fingerprint is SHA256:EoSdGlXrpoqM3NDO421osN033UJIgcsPupkaUQNIZ6g.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.7.1' (RSA) to the list of known hosts.
test.c                                        100%    0     0.0KB/s   00:00
czx@ubuntu18:~/test$
```

出现这个提示，输入yes再回车，不能直接按回车。

1.2.1-6 将 test 文件使用 scp 指令传到开发板

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

```
root@ATK-stm32mp1:~# ls
shell  test  test.c  test-lib
root@ATK-stm32mp1:~#
```

1.2.1-7 在开发板/home/root 目录下查看传过来的 test.c 文件

1.2.2 开发板与 Ubuntu 在同一路由器下拷贝文件

USB_OTG 的网络功能需要系统启动后才开启，有时候我们想在 uboot 阶段使用网络，这时候就要用网线和路由器让开发板和电脑组成局域网。在路由器能上网的情况下，开发板和主机都接在同一个路由器，或者在同一网段的网络环境，并确认相互能 ping 通。

开发板 eMMC 启动系统，插上网线，连接到路由器。在串口终端使用 ifconfig 指令来查看开发板自动获取的 IP，如图所示，这里我的开发板 IP 是 192.168.1.126。

```
root@ATK-stm32mp1:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:04:9F:04:D2:35
          inet addr:192.168.1.126  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::204:9fff:fe04:d235/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1066 errors:0 dropped:48 overruns:0 frame:0
          TX packets:92 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:96883 (94.6 KiB)  TX bytes:12220 (11.9 KiB)
          Interrupt:55 Base address:0xa000

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:368 errors:0 dropped:0 overruns:0 frame:0
          TX packets:368 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:22904 (22.3 KiB)  TX bytes:22904 (22.3 KiB)
```

1.2.2-1 查看开发板 IP

在虚拟机 Ubuntu 下使用 ifconfig 指令查看自己虚拟机的 IP，确保虚拟机的 IP 和开发板的 IP 在同一网段下。如图所示，这里我的虚拟机 IP 是 192.168.1.199，我的网段就是 192.168.1.x（1 < x < 255）。

```
czx@ubuntu18:~$ ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.199  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::e1fd:ef0a:940:f249  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:a2:81:bc  txqueuelen 1000  (以太网)
        RX packets 325937  bytes 346608826 (346.6 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 7316  bytes 584266 (584.2 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (本地环回)
        RX packets 1185  bytes 76692 (76.6 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1185  bytes 76692 (76.6 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

1.2.2-2 查看虚拟机 IP

Ubuntu 执行 sudo ufw disable 关闭防火墙，开发板和 Ubuntu 互 ping，成功后就表示网络通信正常。

```
czx@ubuntu18:~$ ping 192.168.1.126
PING 192.168.1.126 (192.168.1.126) 56(84) bytes of data.
64 bytes from 192.168.1.126: icmp_seq=1 ttl=64 time=1.16 ms
64 bytes from 192.168.1.126: icmp_seq=2 ttl=64 time=0.502 ms
64 bytes from 192.168.1.126: icmp_seq=3 ttl=64 time=0.517 ms
```

```
root@ATK-stm32mp1:~# ping 192.168.1.199
PING 192.168.1.199 (192.168.1.199) 56(84) bytes of data.
64 bytes from 192.168.1.199: icmp_seq=1 ttl=64 time=0.460 ms
64 bytes from 192.168.1.199: icmp_seq=2 ttl=64 time=0.413 ms
64 bytes from 192.168.1.199: icmp_seq=3 ttl=64 time=0.467 ms
64 bytes from 192.168.1.199: icmp_seq=4 ttl=64 time=0.426 ms
64 bytes from 192.168.1.199: icmp_seq=5 ttl=64 time=0.591 ms
```

1.2.2-3 Ubuntu 和开发板互 ping

验证网络通信正常后，按下 Ctrl + C 键即可停止。此时开发板和 Ubuntu 形成了一个局域网，可以按照 1.2.1 小节的方法来执行 scp 指令进行文件传输。

1.3 开发板使用 MobaXterm 与 Windows 互传文件

使用前提：开发板与电脑用网线接在同一路由器上，路由器能上网。

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。

这里用到的串口终端是 MobaXterm，使用 ifconfig 指令查看开发板的 IP，这里我的开发板 IP 是 192.168.1.186，如下图所示。

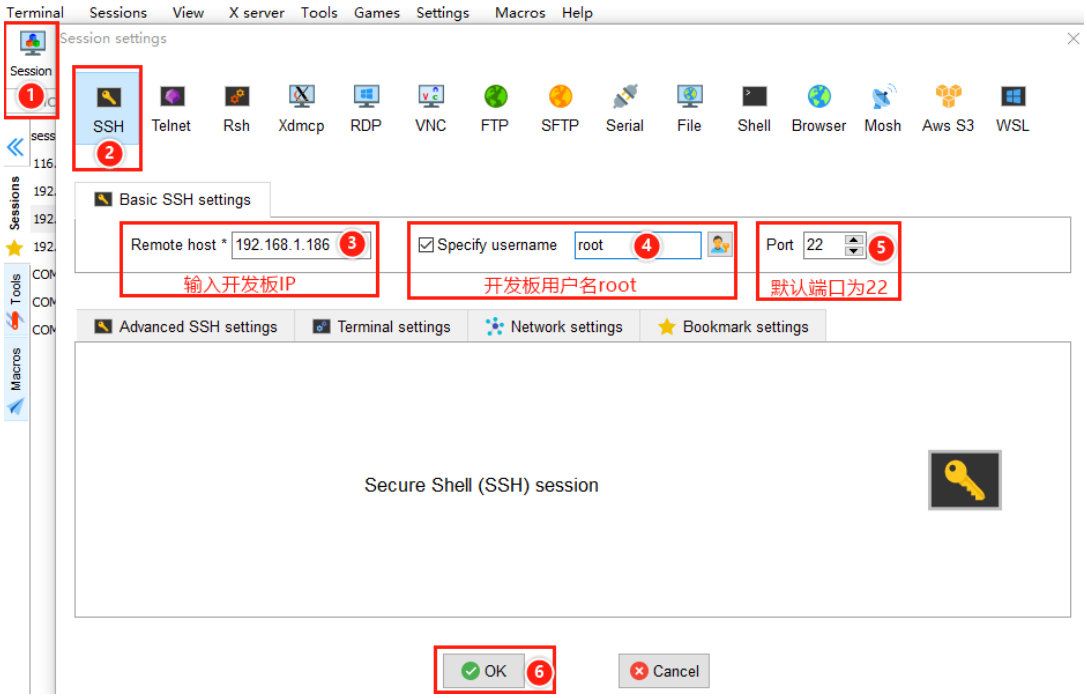
```
ifconfig
```

```
root@ATK-stm32mp1:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 52:B1:68:00:D5:B2
          inet addr:192.168.1.186  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::50b1:68ff:fe00:d5b2/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:12667 errors:0 dropped:769 overruns:0 frame:0
          TX packets:706 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1123123 (1.0 MiB)  TX bytes:124302 (121.3 KiB)
          Interrupt:55 Base address:0xa000

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:368 errors:0 dropped:0 overruns:0 frame:0
          TX packets:368 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:22904 (22.3 KiB)  TX bytes:22904 (22.3 KiB)
```

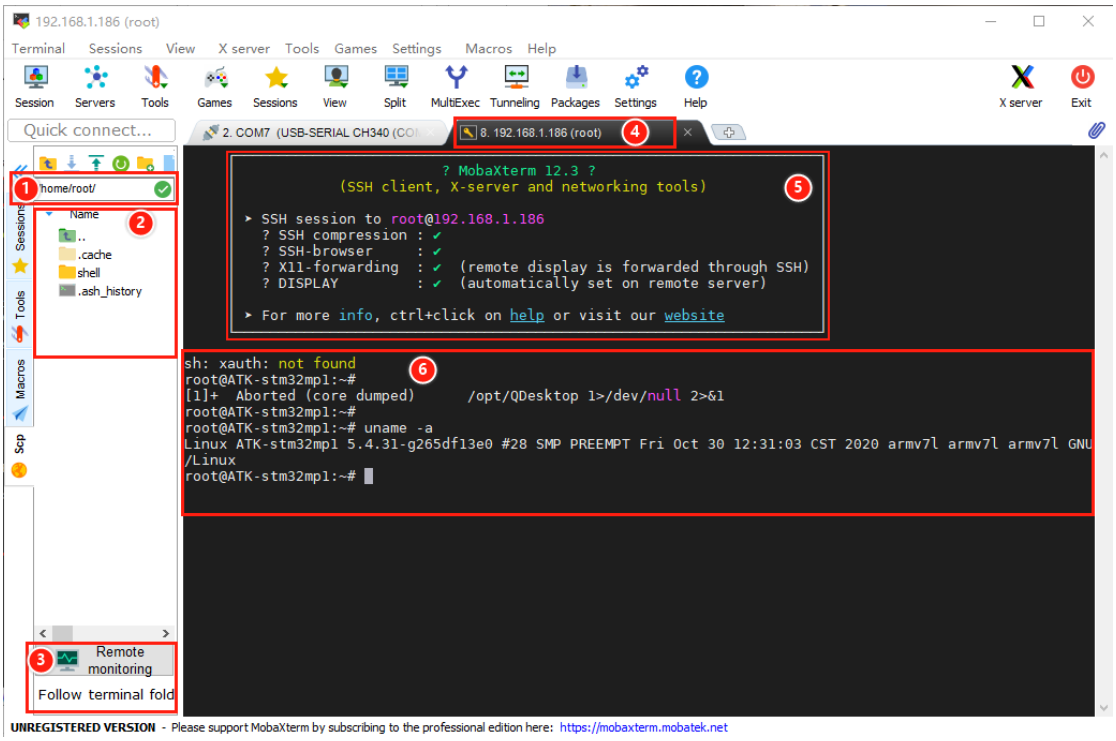
1.3-1 查看开发板 IP

按以下步骤在 MobaXterm 中选择并设置 SSH，使用 SSH 连接开发板。开发板的出厂系统是由 yocto 编译出来的，支持 SSH。



1.3-2 设置 SSH 连接开发板

SSH 登陆开发板后界面如下，左边是开发板默认访问的目录，可以看到开发板/home/root 目录下的文件。我们可以直接拖拽文件，进行文件传输。主界面是开发板文件系统，可以想前面的串口终端一样输入指令，进行指令相关的操作。

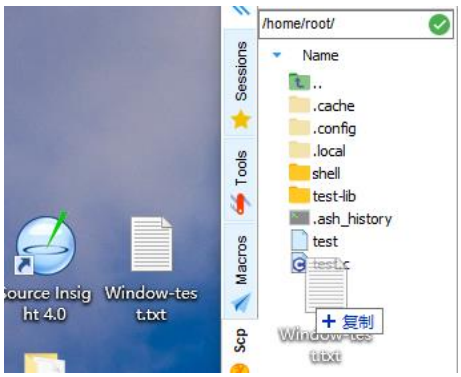


1.3-3 SSH 登陆开发板后的界面

SSH 界面简要分析：

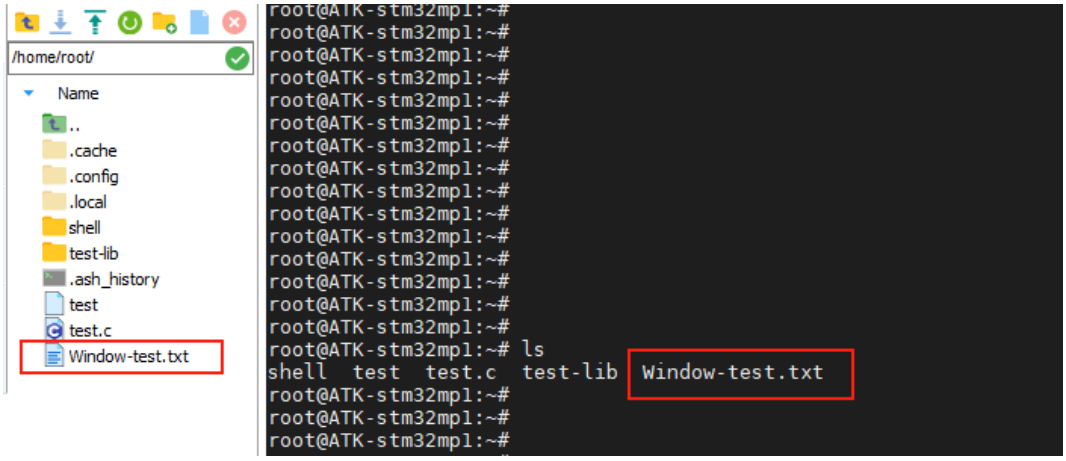
1) 默认访问的目录。
2) 开发板文件管理器。我们可以直接将文件拖到这里来实现 PC 与开发板文件传输。
3) 远程监控。我们拷贝文件进来时，这里会显示文件传输进度；点击开启远程监控，还能查看开发板的实时运行情况。
4) SSH 终端名。由开发板 IP 和用户名组成，这里可能不会随着 IP 的修改而更新。
5) SSH 终端信息。这里会显示具体的 SSH 信息，此处显示的 IP 是随系统更新的。
6) SSH 命令行界面。在这里可以输指令，例如这里输入了 uname -a 指令。

这里演示下直接把 Windows 桌面上的一个文件拖进开发板 test-lib 目录中。同理，开发板上的文件也可以拖到 Windows 系统中。



1.3-4 将 Windows 下的文件拖进开发板

然后在开发板使用 ls 指令查看该文件是否放进了开发板/home/root 目录中，在 SSH 界面的左边目录中我们也可以看到这个文件已经拷贝进开发板/home/root 目录下了。



1.3-5 验证 Windows 和开发板 SSH 互传

第二章 STM32MP157 更新固件篇

在【正点原子】STM32MP157 快速体验 V1.x.pdf 中，我们可以学到如何使用烧录工具或者 SD 卡固化文件系统，但我们在开发的过程中往往会自己尝试编译 uboot、内核设备树和文件系统，编译后我们只需要更新 uboot、设备树、文件系统中的某一个，特别是在开发的过程中要频繁更新。

这里我们学习下如何单独更新某个文件，就不用每次更新都重新制作一张 SD 系统启动卡或者使用脚本固化到 eMMC 中，带大家把光盘里的 uboot、内核、设备树等相关文件更新到 eMMC 或 SD 卡中。前提是 eMMC 和 SD 卡中有启动系统，可以按照【正点原子】STM32MP157 快速体验 V1.x.pdf 里烧录系统的方法来固化启动 eMMC 和 SD 卡的启动系统。

因为 eMMC 和 SD 卡中分区较多，这里简要总结下各分区，方便查找。（注：这里 eMMC 和 SD 卡的文件系统是出厂系统）

存储类型	分区名	作用
SD 卡	sdb1	TF-A 分区，存放 tf-a 文件
	sdb2	备份 TF-A 分区，备份 tf-a 文件
	sdb3	U-BOOT 分区，存放着 uboot
	sdb4	bootfs 分区，存放内核启动的文件
	sdb5	rootfs 分区，存放文件系统
eMMC	mmcblk2boot0	TF-A 分区，存放 tf-a 文件
	mmcblk2boot1	备份 TF-A 分区，备份 tf-a 文件
	mmcblk2p1	U-BOOT 分区，存放着 uboot
	mmcblk2p2	bootfs 分区，存放内核启动的文件
	mmcblk2p3	rootfs 分区，存放文件系统

2.1 eMMC 更新固件

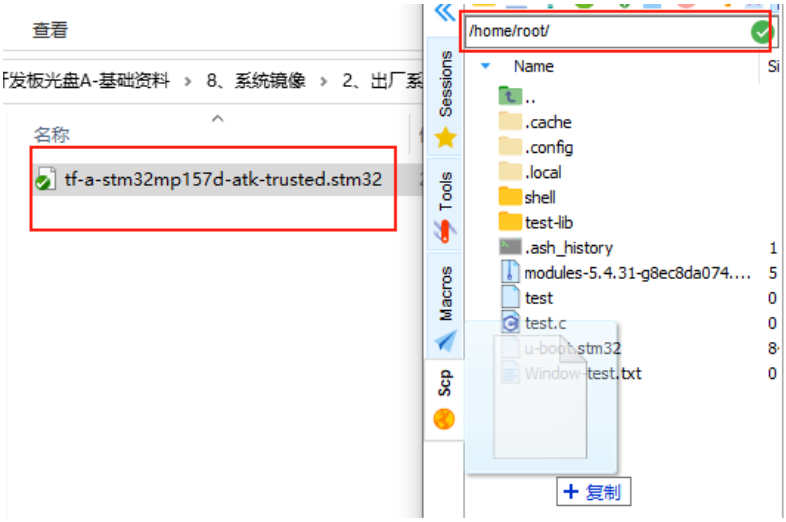
正点原子出厂都有把固件固化到 eMMC 中，我们从 eMMC 启动或者从 SD 卡启动替换自己的固件。

2.1.1 更新 tf-a 到 eMMC

从 eMMC 启动系统开发板，更新 tf-a 到 eMMC 中。

光盘路径：开发板光盘 A-基础资料\8、系统镜像\2、出厂系统镜像\4、tf-a-mp1-2.2-g463d4d8-v1.0\

把该路径下的 tf-a-stm32mp157d-atk-trusted.stm32 拷贝到开发板的/home/root 目录下，为后面的更新 tf-a 做准备。



2.1.1-1 拷贝光盘的 tf-a 文件到开发板/home/root 目录

```
root@ATK-stm32mp1:~# ls
modules-5.4.31-g8ec8da074.tar.bz2  test     test-lib                        u-boot.stm32
shell                              test.c   tf-a-stm32mp157d-atk-trusted.stm32  Window-test.txt
root@ATK-stm32mp1:~#
```

2.1.1-2 tf-a 文件在开发板/home/root 目录

执行下面的指令，先使能 eMMC 启动分区，才能进行烧写。

```
echo 0 > /sys/class/block/mmcblk2boot0/force_ro
echo 0 > /sys/class/block/mmcblk2boot1/force_ro
```

把当前目录下的 tf-a-stm32mp157d-atk-trusted.stm32 烧写到 eMMC 的 TF-A 分区和备份 TF-A 分区。

```
dd if=tf-a-stm32mp157d-atk-trusted.stm32 of=/dev/mmcblk2boot0 conv=fsync
dd if=tf-a-stm32mp157d-atk-trusted.stm32 of=/dev/mmcblk2boot1 conv=fsync
```

烧写完成后，关闭要烧写的启动分区。

```
echo 1 > /sys/class/block/mmcblk2boot0/force_ro
echo 1 > /sys/class/block/mmcblk2boot1/force_ro
```

```
root@ATK-stm32mp1:~# echo 0 > /sys/class/block/mmcblk2boot0/force_ro
root@ATK-stm32mp1:~# echo 0 > /sys/class/block/mmcblk2boot0/force_ro
root@ATK-stm32mp1:~# echo 0 > /sys/class/block/mmcblk2boot1/force_ro
root@ATK-stm32mp1:~# dd if=tf-a-stm32mp157d-atk-trusted.stm32 of=/dev/mmcblk2boot0 conv=fsync
472+1 records in
472+1 records out
241984 bytes (242 kB, 236 KiB) copied, 0.0697889 s, 3.5 MB/s
root@ATK-stm32mp1:~# dd if=tf-a-stm32mp157d-atk-trusted.stm32 of=/dev/mmcblk2boot1 conv=fsync
472+1 records in
472+1 records out
241984 bytes (242 kB, 236 KiB) copied, 0.0733856 s, 3.3 MB/s
root@ATK-stm32mp1:~# echo 1 > /sys/class/block/mmcblk2boot0/force_ro
root@ATK-stm32mp1:~# echo 1 > /sys/class/block/mmcblk2boot1/force_ro
```

2.1.1-3 烧录 tf-a 到 eMMC

2.1.2 更新 uboot 到 eMMC

从 eMMC 启动系统，或从 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\2、出厂系统镜像\5、uboot-mp1-2020.01-gdb8d2374-v1.0\

把该路径下的 u-boot.stm32 文件拷贝到开发板的/home/root 目录下，为后面的更新 uboot 做准备。

```
root@ATK-stm32mp1:~# ls
shell  test  test.c  test-lib  u-boot.stm32  Window-test.txt
root@ATK-stm32mp1:~#
```

2.1.2-1 拷贝 eMMC 所用的 uboot 到文件系统/home/root 目录下

把当前目录下的 uboot 烧写到 eMMC 的 uboot 分区。

```
dd if=u-boot.stm32 of=/dev/mmcblk2p1 conv=fsync
```

```
root@ATK-stm32mp1:~# dd if=u-boot.stm32 of=/dev/mmcblk2p1 conv=fsync
1691+1 records in
1691+1 records out
866124 bytes (866 kB, 846 KiB) copied, 0.224404 s, 3.9 MB/s
root@ATK-stm32mp1:~#
```

2.1.2-2 烧写 uboot 到 eMMC 的 uboot 分区

2.1.3 更新设备树到 eMMC

使用 ls 指令查看 eMMC 的 uboot 分区设备树所在的目录，该目录是/run/media/mmcblk2p2。

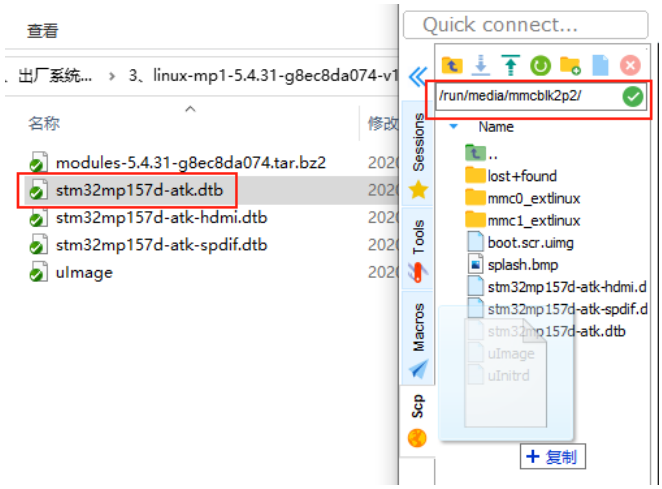
```
ls /run/media/mmcblk2p2/
```

```
root@ATK-stm32mp1:~# ls /run/media/mmcblk2p2/
boot.scr.uimg  mmc0_extlinux  splash.bmp            stm32mp157d-atk-hdmi.dtb  uImage
lost+found     mmc1_extlinux  stm32mp157d-atk.dtb   stm32mp157d-atk-spdif.dtb uInitrd
root@ATK-stm32mp1:~#
```

2.1.3-1 查看 eMMC 出厂的设备树所在的目录

把光盘里的设备树文件 stm32mp157d-atk.dtb、stm32mp157d-atk-hdmi.dtb、stm32mp157d-atk-spdif.dtb 拷贝到开发板/run/media/mmcblk2p2 目录下。

光盘路径：开发板光盘 A-基础资料\8、系统镜像\2、出厂系统镜像\3、linux-mp1-5.4.31-g8ec8da074-v1.0\



2.1.3-2 拷贝设备树到开发板/run/media/mmcblk2p2 目录

拷贝完成后，可以通过执行以下指令查看设备树文件修改时间是否已经更新，从而确定设备树已经更新成功。

```
ls /run/media/mmcblk2p2/stm32mp157d-atk.dtb -l
```

```
root@ATK-stm32mp1:~# ls /run/media/mmcblk2p2/stm32mp157d-atk.dtb -l
-rw-r--r-- 1 root root 74594 Oct 29 15:46 /run/media/mmcblk2p2/stm32mp157d-atk.dtb
root@ATK-stm32mp1:~#
```

2.1.3-3 确认设备树已更新

2.1.4 更新内核到 eMMC

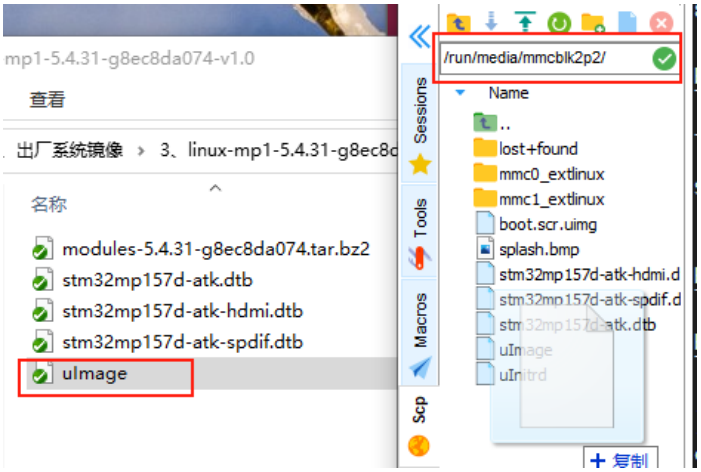
在2.1.3小节中，我们在开发板的/run/media/mmcblk2p2目录下还看到一个uImage文件，这个就是内核文件，我们现在来更新内核。首先查看下开发板eMMC里的uImage的信息。

```
root@ATK-stm32mp1:~# ls /run/media/mmcblk2p2/uImage -l
-rw-r--r-- 1 root root 8126336 Oct 19 13:28 /run/media/mmcblk2p2/uImage
root@ATK-stm32mp1:~#
```

2.1.4-1 eMMC中的uImage文件信息

接下来把光盘里的uImage文件拷贝到开发板的/run/media/mmcblk2p2目录中，这和2.1.3小节的方法一样。

光盘路径：开发板光盘 A-基础资料\8、系统镜像\2、出厂系统镜像\3、linux-mp1-5.4.31-g8ec8da074-v1.0\



2.1.4-2 拷贝光盘里的 uImage 到开发板/run/media/mmcblk2p2 目录

拷贝完成后，可以通过执行以下指令查看 uImage 文件修改时间是否已经更新，从而确定 uImage 已经更新成功。

```
ls /run/media/mmcblk2p2/uImage -l
```

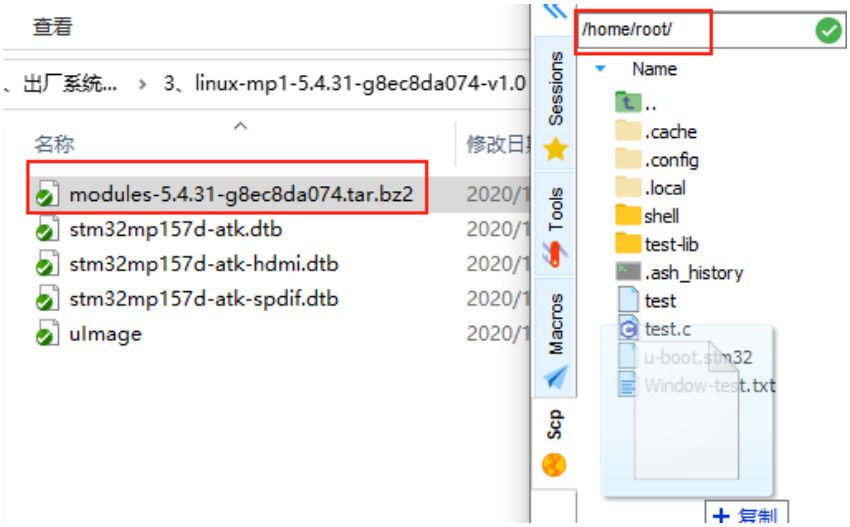
```
root@ATK-stm32mp1:~# ls /run/media/mmcblk2p2/uImage  -l
-rw-r--r-- 1 root root 8126336 Oct 19 13:28 /run/media/mmcblk2p2/uImage
root@ATK-stm32mp1:~# ls /run/media/mmcblk2p2/uImage  -l
-rw-r--r-- 1 root root 8126336 Oct 29 16:04 /run/media/mmcblk2p2/uImage
root@ATK-stm32mp1:~#
```

2.1.4-3 确认 uImage 已更新

除了更新 uImage，我们有时候还会更新内核模块。这里我们将光盘里的内核模块拷贝到 eMMC 文件系统目录下。

光盘路径：开发板光盘 A-基础资料\8、系统镜像\2、出厂系统镜像\3、linux-mp1-5.4.31-g8ec8da074-v1.0\

将该路径下的 modules-5.4.31-g8ec8da074.tar.bz2 内核模块压缩包拷贝到开发板的/home/root 目录即可。



2.1.4-4 将内核模块压缩包拷贝到开发板/home/root 目录

在开发板/home/root 目录下使用 tar 指令，将 modules-5.4.31-g8ec8da074.tar.bz2 解压到开发板/lib/modules 目录下即可，开发板启动时会去这个目录找对应的内核模块来加载的。

```
tar vxf modules-5.4.31-g8ec8da074.tar.bz2 -C /lib/modules/
```

```
root@ATK-stm32mp1:~# tar vxf modules-5.4.31-g8ec8da074.tar.bz2 -C /lib/modules/
5.4.31-g8ec8da074/
5.4.31-g8ec8da074/modules.builtin.bin
5.4.31-g8ec8da074/modules.builtin.modinfo
5.4.31-g8ec8da074/modules.alias.bin
5.4.31-g8ec8da074/modules.dep
5.4.31-g8ec8da074/modules.symbols
5.4.31-g8ec8da074/kernel/
5.4.31-g8ec8da074/kernel/crypto/
5.4.31-g8ec8da074/kernel/crypto/algif_skcipher.ko
5.4.31-g8ec8da074/kernel/crypto/authencesn.ko
5.4.31-g8ec8da074/kernel/crypto/echainiv.ko
```

显示解压的内容，内容较多，只截取部分

2.1.4-5 解压内核模块到开发板/lib/modules 目录

2.1.5 更新文件系统到 eMMC

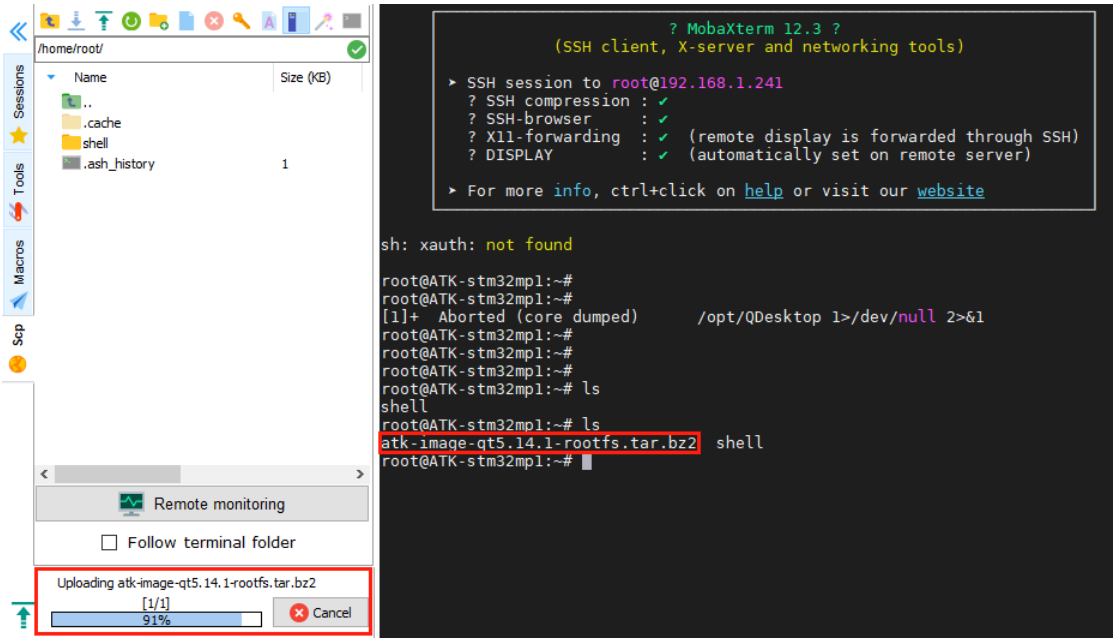
注意：不能用 eMMC 启动开发板来更新 eMMC 分区里的文件系统，正在运行的一个系统不能把自己给格式化。这里要使用 SD 卡启动开发板来更新 eMMC 的文件系统，制作 SD 卡启动的方法可以参考【正点原子】STM32MP157 快速体验 V1.x.pdf 的 2.3.3 小节。

因为内核模块是在 eMMC 的文件系统分区（rootfs 分区）里的，更新文件系统时会把这个分区里的文件全部删除，所以在更新文件系统后，请自行安装 2.1.4 小节的步骤来更新内核模块。

从 SD 卡启动开发板，将光盘里的 atk-image-qt5.14.1-rootfs.tar.bz2 拷贝到开发板文件系统中。

光盘路径：开发板光盘 A-基础资料\8、系统镜像\2、出厂系统镜像\6、Qt5.14.1 文件系统\

文件较大，拷贝需要一定时间，要等 atk-image-qt5.14.1-rootfs.tar.bz2 全部拷贝到开发板的/home/root 目录后才能继续操作。



2.1.5-1 拷贝文件系统压缩包到开发板的/home/root 目录

拷贝完成后，我们直接删除 eMMC 的文件系统分区（mmcblk2p3），执行以下指令。

```
rm -rf /run/media/mmcblk2p3/*
```

```
root@ATK-stm32mp1:~# rm -rf /run/media/mmcblk2p3/*
root@ATK-stm32mp1:~#
```

2.1.5-2 删除 eMMC 的文件系统分区所挂载的目录下的全部内容

eMMC 的文件系统分区清空后，我们就可以直接将 atk-image-qt5.14.1-rootfs.tar.bz2 文件系统直接解压到 eMMC 的文件系统分区目录中。执行以下指令进行解压。（注：文件系统较大，解压时间比较长，请耐心等待）

```
tar vxf atk-image-qt5.14.1-rootfs.tar.bz2 -C /run/media/mmcblk2p3/
```

```
tar: ./usr/lib/systemd/user: time stamp 2020-10-30 05:20:56 is 22939725.401348049 s in the future
tar: ./usr/lib/systemd: time stamp 2020-10-30 05:20:56 is 22939725.400704383 s in the future
tar: ./usr/lib: time stamp 2020-10-30 05:21:03 is 22939732.400215716 s in the future
tar: ./usr/bin: time stamp 2020-10-30 05:20:41 is 22939710.399665966 s in the future
tar: ./usr/share/dbus-1/services: time stamp 2020-10-30 05:20:41 is 22939710.399217466 s in the futu
re
tar: ./usr/share/dbus-1: time stamp 2020-10-30 05:20:41 is 22939710.398773799 s in the future
tar: ./usr/share: time stamp 2020-10-30 05:20:44 is 22939713.398336508 s in the future
tar: ./usr: time stamp 2020-10-30 05:21:03 is 22939732.397924841 s in the future
tar: .: time stamp 2020-10-30 05:21:11 is 22939740.397509758 s in the future
root@ATK-stm32mp1:~#
```

2.1.5-3 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。这样我们 eMMC 里的文件系统就更新了。

```
sync
```

```
tar: ./usr/share: time stamp 2020-10-30 05:20:44 is 22939713.398336508 s in the future
tar: ./usr: time stamp 2020-10-30 05:21:03 is 22939732.397924841 s in the future
tar: .: time stamp 2020-10-30 05:21:11 is 22939740.397509758 s in the future
root@ATK-stm32mp1:~#
root@ATK-stm32mp1:~#
root@ATK-stm32mp1:~# sync
root@ATK-stm32mp1:~#
```

2.1.5-4 执行 sync 同步数据

执行 ls 指令可以查看刚刚烧录的文件系统。

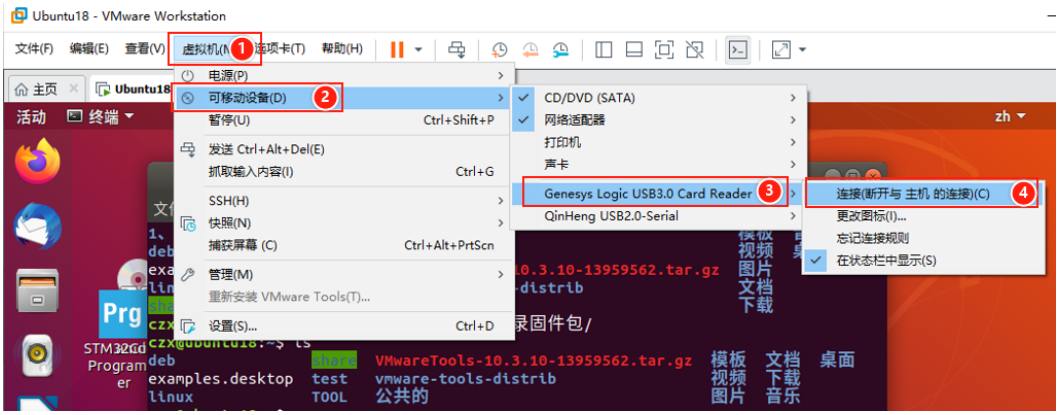
```
ls /run/media/mmcblk2p3
```

```
root@ATK-stm32mp1:~# ls /run/media/mmcblk2p3
bin   dev  home  lost+found  mnt  proc  sbin  tmp  var
boot  etc  lib   media       opt  run   sys   usr  vendor
root@ATK-stm32mp1:~#
```

2.1.5-5 查看 eMMC 里的文件系统

2.2 SD 卡更新固件

我们直接在 Ubuntu 上进行 SD 卡更新固件，首先我们需要一张 STM32MP157 的 SD 卡出厂系统启动卡，可以参考【正点原子】STM32MP157 快速体验 V1.x.pdf 的 2.4.1.2 小节来制作一张 SD 卡启动卡。将 SD 启动卡按以下步骤接到 Ubuntu 中，如图所示。



2.2-1 SD 卡接入 Ubuntu

在 Ubuntu 接入 SD 系统启动卡后，使用 fdisk 指令查看 SD 卡挂载的节点，如下图所示。

```
sudo fdisk -l
```

```
czx@ubuntu18:~$ sudo fdisk -l
[sudo] czx 的密码:
```

输入密码

2.2-2 fdisk 指令查看 SD 卡挂载的节点

可以看到这里我的 SD 卡挂载节点为/dev/sdb。请以自己实际的 SD 卡挂载节点为准。

```
Disk /dev/sdb: 14.9 GiB, 15931539456 字节，31116288 个扇区
单元：扇区 / 1 * 512 = 512 字节
扇区大小(逻辑/物理)：512 字节 / 512 字节
I/O 大小(最小/最佳)：512 字节 / 512 字节
磁盘标签类型：gpt
磁盘标识符：3EBA7C59-43B9-4809-9B19-6AD05F276FFA

设备          起点     末尾     扇区  大小 类型
/dev/sdb1       34      545      512  256K Linux 保留
/dev/sdb2      546     1057      512  256K Linux 保留
/dev/sdb3     1058     5153     4096    2M Linux 保留
/dev/sdb4     5154   136225   131072   64M Linux 文件系统
/dev/sdb5   136226 31116252 30980027 14.8G Linux 文件系统
```

2.2-3 SD 卡挂载的节点

在 Ubuntu 用户目录下新建一个 sd_update 文件夹用来存放等下更新要用到的文件。

```
mkdir sd_update
```

```
czx@ubuntu18:~$ mkdir sd_update
czx@ubuntu18:~$ ls
deb               sd_update  TOOL                                    公共的  图片  音乐
examples.desktop  share      VMwareTools-10.3.10-13959562.tar.gz     模板    文档  桌面
linux             test       vmware-tools-distrib                    视频    下载
```

2.2-4 创建 sd_update 文件夹

2.2.1 更新 tf-a 到 SD 卡

使用 filezilla 软件将光盘里的 tf-a-stm32mp157d-atk-trusted.stm32 文件拷贝到 Ubuntu 的用户目录/sd_update 文件夹里。

光盘路径：开发板光盘 A-基础资料\8、系统镜像\2、出厂系统镜像\4、tf-a-mp1-2.2-g463d4d8-v1.0\

执行 cd 指令到 sd_update 文件夹，再执行 ls 指令。

```
cd sd_update
ls
```

```
czx@ubuntu18:~$ cd sd_update/
czx@ubuntu18:~/sd_update$ ls
tf-a-stm32mp157d-atk-trusted.stm32
```

2.2.1-1 查看 tf-a 文件

在前面我们知道了此时 SD 卡挂载在/dev/sdb 节点上，在当前目录下（sd_update），使用 dd 指令将 tf-a-stm32mp157d-atk-trusted.stm32 烧写到 SD 卡中，执行下面的指令。

```
sudo dd if=tf-a-stm32mp157d-atk-trusted.stm32 of=/dev/sdb1 conv=fsync
sudo dd if=tf-a-stm32mp157d-atk-trusted.stm32 of=/dev/sdb2 conv=fsync
```

```
czx@ubuntu18:~/sd_update$ sudo dd if=tf-a-stm32mp157d-atk-trusted.stm32 of=/dev/sdb1 conv=fsync
记录了472+1 的读入
记录了472+1 的写出
241984 bytes (242 kB, 236 KiB) copied, 0.0484532 s, 5.0 MB/s
czx@ubuntu18:~/sd_update$ sudo dd if=tf-a-stm32mp157d-atk-trusted.stm32 of=/dev/sdb2 conv=fsync
记录了472+1 的读入
记录了472+1 的写出
241984 bytes (242 kB, 236 KiB) copied, 0.0585803 s, 4.1 MB/s
```

2.2.1-2 烧写 tf-a 文件到 SD 卡

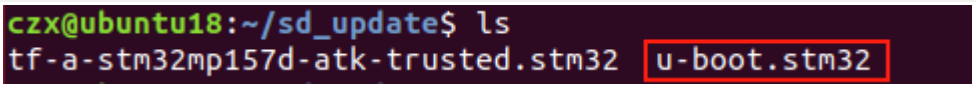
2.2.2 更新 uboot 到 SD 卡

使用 filezilla 软件将光盘里的 u-boot.stm32 文件拷贝到 Ubuntu 的用户目录/sd_update 文件夹里。

光盘路径：开发板光盘 A-基础资料\8、系统镜像\2、出厂系统镜像\5、uboot-mp1-2020.01-gdb8d2374-v1.0\

在 sd_update 文件夹执行 ls 指令查看是否存在 u-boot.stm32 文件。

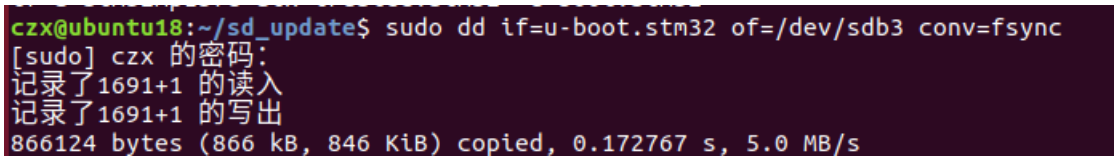
```
ls
```



2.2.2-1 查看准备好的 u-boot.stm32 文件

在前面我们知道了此时 SD 卡挂载在/dev/sdb 节点上，在当前目录下（sd_update），使用 dd 指令将 u-boot.stm32 烧写到 SD 卡中，执行下面的指令。

```
sudo dd if=u-boot.stm32 of=/dev/sdb3 conv=fsync
```



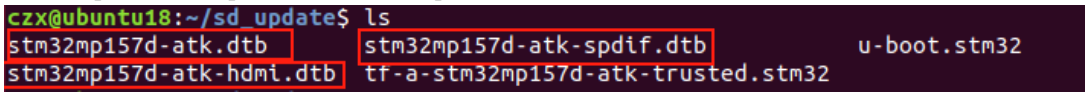
2.2.2-2 烧写 u-boot 到 SD 卡

2.2.3 更新设备树到 SD 卡

使用 filezilla 软件将光盘的 stm32mp157d-atk.dtb、stm32mp157d-atk-spdif.dtb、stm32mp157d-atk-hdmi.dtb 设备树文件拷贝到 Ubuntu 的用户目录/sd_update 文件夹里。

光盘路径：开发板光盘 A-基础资料\8、系统镜像\2、出厂系统镜像\ 3、linux-mp1-5.4.31-g265df13e0-v1.0\

在 sd_update 文件夹执行 ls 指令查看是否存在 stm32mp157d-atk.dtb、stm32mp157d-atk-spdif.dtb、stm32mp157d-atk-hdmi.dtb 设备树文件。



2.2.3-1 查看准备好的设备树文件

这里我们先查看下 SD 卡中设备树的相关信息，执行 ls -l 指令。这里我的用户名是 czx，大家根据自己情况执行命令。

```
ls /media/用户名/bootfs/ -l
```

```
czx@ubuntu18:~/sd_update$ ls /media/czx/bootfs/ -l
总用量 11800
-rwxr-xr-x 1 root root    2943 10月 13 11:15 boot.scr.uimg
drwx------ 2 root root    1024 10月 13 11:15 lost+found
drwxr-xr-x 2 root root    1024 10月 30 12:56 mmc0_extlinux
drwxr-xr-x 2 root root    1024 10月 19 15:41 mmc1_extlinux
-rw-r--r-- 1 root root   92670 10月 13 12:12 splash.bmp
-rw-r--r-- 1 root root   74594 10月 19 21:28 stm32mp157d-atk.dtb
-rw-r--r-- 1 root root   74014 10月 19 21:28 stm32mp157d-atk-hdmi.dtb
-rw-r--r-- 1 root root   74510 10月 19 21:28 stm32mp157d-atk-spdif.dtb
-rw-r--r-- 1 root root 8125872 10月 30 14:36 uImage
-rw-r--r-- 1 root root 3632241 10月 13 11:15 uInitrd
```

2.2.3-2 查看 SD 卡中设备树信息

执行 sudo cp 指令，依次复制准备好的设备树文件到 SD 卡中。

```
sudo cp stm32mp157d-atk.dtb /media/用户名/bootfs/
sudo cp stm32mp157d-atk-hdmi.dtb /media/用户名/bootfs/
sudo cp stm32mp157d-atk-spdif.dtb /media/用户名/bootfs/
```

```
czx@ubuntu18:~/sd_update$ sudo cp stm32mp157d-atk.dtb /media/czx/bootfs/
czx@ubuntu18:~/sd_update$ sudo cp stm32mp157d-atk-hdmi.dtb /media/czx/bootfs/
czx@ubuntu18:~/sd_update$ sudo cp stm32mp157d-atk-spdif.dtb /media/czx/bootfs/
```

2.2.3-3 拷贝设备树文件到 SD 卡

再次执行 ls -l 指令来查看设备树信息，可以看到设备树修改时间已经更新了。

```
ls /media/用户名/bootfs/ -l
```

```
czx@ubuntu18:~/sd_update$ ls /media/czx/bootfs/ -l
总用量 11800
-rwxr-xr-x 1 root root    2943 10月 13 11:15 boot.scr.uimg
drwx------ 2 root root    1024 10月 13 11:15 lost+found
drwxr-xr-x 2 root root    1024 10月 30 12:56 mmc0_extlinux
drwxr-xr-x 2 root root    1024 10月 19 15:41 mmc1_extlinux
-rw-r--r-- 1 root root   92670 10月 13 12:12 splash.bmp
-rw-r--r-- 1 root root   74594 10月 31 19:29 stm32mp157d-atk.dtb
-rw-r--r-- 1 root root   74014 10月 31 19:29 stm32mp157d-atk-hdmi.dtb
-rw-r--r-- 1 root root   74510 10月 31 19:30 stm32mp157d-atk-spdif.dtb
-rw-r--r-- 1 root root 8125872 10月 30 14:36 uImage
-rw-r--r-- 1 root root 3632241 10月 13 11:15 uInitrd
```

2.2.3-4 查看设备树是否修改

2.2.4 更新内核到 SD 卡

使用 filezilla 软件将光盘里的 uImage 内核文件拷贝到 Ubuntu 的用户目录/sd_update 文件夹里。

光盘路径：开发板光盘 A-基础资料\8、系统镜像\2、出厂系统镜像\ 3、linux-mp1-5.4.31-g265df13e0-v1.0\

在 sd_update 文件夹执行 ls 指令查看是否存在 uImage 内核文件。

```
czx@ubuntu18:~/sd_update$ ls
stm32mp157d-atk.dtb       stm32mp157d-atk-spdif.dtb          u-boot.stm32
stm32mp157d-atk-hdmi.dtb  tf-a-stm32mp157d-atk-trusted.stm32 uImage
```

2.2.4-1 查看准备好的 uImage 文件

执行 sudo cp 指令，复制准备好的 uImage 文件到 SD 卡中。

```
sudo cp uImage /media/用户名/bootfs/
```

```
czx@ubuntu18:~/sd_update$ sudo cp uImage /media/czx/bootfs
[sudo] czx 的密码：
czx@ubuntu18:~/sd_update$
```

2.2.4-2 拷贝 uImage 到 SD 卡

同样，执行 ls -l 指令来查看 uImage 文件是否修改。

```
ls /media/用户名/bootfs/ -l
```

```
czx@ubuntu18:~/sd_update$ ls /media/czx/bootfs/ -l
总用量 11800
-rwxr-xr-x 1 root root    2943 10月 13 11:15 boot.scr.uimg
drwx------ 2 root root    1024 10月 13 11:15 lost+found
drwxr-xr-x 2 root root    1024 10月 30 12:56 mmc0_extlinux
drwxr-xr-x 2 root root    1024 10月 19 15:41 mmc1_extlinux
-rw-r--r-- 1 root root   92670 10月 13 12:12 splash.bmp
-rw-r--r-- 1 root root   74594 10月 31 19:33 stm32mp157d-atk.dtb
-rw-r--r-- 1 root root   74014 10月 31 19:33 stm32mp157d-atk-hdmi.dtb
-rw-r--r-- 1 root root   74510 10月 31 19:33 stm32mp157d-atk-spdif.dtb
-rw-r--r-- 1 root root 8125872 10月 31 19:41 uImage
-rw-r--r-- 1 root root 3632241 10月 13 11:15 uInitrd
```

2.2.4-3 查看设备树是否修改

2.2.5 更新文件系统到 SD 卡

使用 filezilla 软件将光盘里的 atk-image-qt5.14.1-rootfs.tar.bz2 文件系统压缩包拷贝到 Ubuntu 的用户目录/sd_update 文件夹里。

光盘路径：开发板光盘 A-基础资料\8、系统镜像\2、出厂系统镜像\ 6、Qt5.14.1 文件系统\

在 sd_update 文件夹执行 ls 指令查看是否存在 atk-image-qt5.14.1-rootfs.tar.bz2 文件系统压缩包。

```
czx@ubuntu18:~/sd_update$ ls
atk-image-qt5.14.1-rootfs.tar.bz2  stm32mp157d-atk-spdif.dtb          uImage
stm32mp157d-atk.dtb                tf-a-stm32mp157d-atk-trusted.stm32
stm32mp157d-atk-hdmi.dtb           u-boot.stm32
czx@ubuntu18:~/sd_update$
```

2.2.5-1 查看准备好的文件系统压缩包

使用 df 指令查看 SD 卡文件系统的挂载路径，如图所示。这里我的挂载路径为/media/czx/rootfs，请根据个人 SD 卡挂载的实际目录来填写。（czx 是我虚拟机用户名）

```
czx@ubuntu18:~$ df
文件系统           1K-块     已用      可用 已用% 挂载点
udev             4038504        0   4038504    0% /dev
tmpfs             812492     3740    808752    1% /run
/dev/sda1      200212776 21907644 168065236   12% /
tmpfs            4062440        0   4062440    0% /dev/shm
tmpfs               5120        4      5116    1% /run/lock
tmpfs            4062440        0   4062440    0% /sys/fs/cgroup
/dev/loop1         56704    56704         0  100% /snap/core18/1885
/dev/loop0         31744    31744         0  100% /snap/snapd/9721
/dev/loop3           384      384         0  100% /snap/gnome-characters/570
/dev/loop4        261760   261760         0  100% /snap/gnome-3-34-1804/36
/dev/loop5          2560     2560         0  100% /snap/gnome-calculator/748
/dev/loop2         31744    31744         0  100% /snap/snapd/9607
/dev/loop7         63616    63616         0  100% /snap/gtk-common-themes/1506
/dev/loop8          2560     2560         0  100% /snap/gnome-calculator/826
/dev/loop10         1024     1024         0  100% /snap/gnome-logs/100
/dev/loop9          2304     2304         0  100% /snap/gnome-system-monitor/148
/dev/loop6        223232   223232         0  100% /snap/gnome-3-34-1804/60
/dev/loop11          384      384         0  100% /snap/gnome-characters/550
/dev/loop12        56704    56704         0  100% /snap/core18/1932
tmpfs             812488       16    812472    1% /run/user/121
tmpfs             812488       28    812460    1% /run/user/1000
/dev/sr0         2142112  2142112         0  100% /media/czx/Ubuntu 18.04.5 LTS
amd64
/dev/sdb4          59365    13084     41695   24% /media/czx/bootfs
/dev/sdb5         999320   802384    128124   87% /media/czx/rootfs
czx@ubuntu18:~$
```

2.2.5-2 df 指令查看 SD 卡文件系统挂载目录

首先删除 SD 卡文件系统分区下的全部文件，执行以下指令。

```
sudo rm -rf /media/用户名/rootfs/*
```

```
czx@ubuntu18:~$ sudo rm -rf /media/czx/rootfs/*
[sudo] czx 的密码：
czx@ubuntu18:~$
```

2.2.5-3 删除 SD 卡的文件系统分区下的文件系统

然后将 atk-image-qt5.14.1-rootfs.tar.bz2 文件系统压缩包解压至 SD 卡根文件系统分区即可，执行以下指令。文件系统较大，解压需要时间，请耐心等待。

```
sudo tar xf atk-image-qt5.14.1-rootfs.tar.bz2 -C /media/czx/rootfs/
```

```
czx@ubuntu18:~/sd_update$ sudo tar xf atk-image-qt5.14.1-rootfs.tar.bz2 -C /media/czx/rootfs/
czx@ubuntu18:~/sd_update$
```

2.2.5-4 解压文件系统到 SD 卡文件系统分区

解压完后执行 sync 指令来同步数据，防止数据未完全写入。

```
sync
```

```
czx@ubuntu18:~/sd_update$ sync
czx@ubuntu18:~/sd_update$
```

2.2.5-5 同步数据

未完待续，持续更新中