1. Burn Ubuntu 16.04 to nand flash

1、Unzip the compressed package

```
$ tar -xvf    2G-IOT_linux_nand_boot.tar.gz
$ cd 2G-IOT_linux_nand_boot/
$ tree -L 1
.
├── 7th
└── script

2 directories, 0 files
```

7th contains the burning script and the prepared system image, script is some scripts for making the root file system

2、Burn to nand flash

Host environment: ubuntu14.04
Hardware preparation: 2G-iot development board, Android usb data cable

The compressed package has provided pdl1.bin, pdl2.bin, uboot, ubi.img needed for nand startup
If you need to compile by yourself, please refer to Chapter 2 nand startup image production

- **Install python serial package**

```
sudo apt-get install python3-pip
sudo pip3 install pyserial
```

- **Burn**

The DIP switch is set to 1234up 5678 down state, the jumper cap is connected to the nand start position
Press and hold the power button while the Android data cable is connected to the board and the PC. Release the power button for 3~5 seconds
Execute the following commands to complete the burning of the nand version system

```
cd 7th
sudo python3 opi2g_nand_write.py -p /dev/ttyACM0 --format-flash --pdl1 pdl1.bin
--pdl2 pdl2.bin bootloader:u-boot.rda nandroot:ubi.img
```

The following message appears, indicating that the programming is complete

```
→  7th sudo python3 opi2g_nand_write.py -p /dev/ttyACM0 --format-flash --pdl1 pd
l1.bin --pdl2 pdl2.bin bootloader:u-boot.rda nandroot:ubi.img
[sudo] password for csy:
Opening /dev/ttyACM0...
Sending partition pdl1 (len 32416) to 0x00100100
Sending partition pdl2 (len 449304) to 0x80008000
Partition table: mtdparts=rda_nand:2M(bootloader),510M(nandroot)
Formatting flash memory...
Partition table: mtdparts=rda_nand:2M(bootloader),510M(nandroot)
Sending partition bootloader (len 523372) to 0x00000000
Sending partition nandroot (len 401604608) to 0x00000000
Done
```

*U-boot.rda is uboot that supports nand startup, and it will go to the /boot folder of the root file system
Find the boot-nand.cmd file, and load ramdisk, kernel, modem according to this file

*ubi.img is a file system in ubi format

The kernel is placed in the /boot folder of the file system. If you need to replace the kernel, please replace the zImage generated by the compilation with the zImage in the /boot folder.
boot-nand.cmd boot-nand.scr modem.bin uInitrd zImage

2. the production of nand boot image

1、Unzip the compressed package

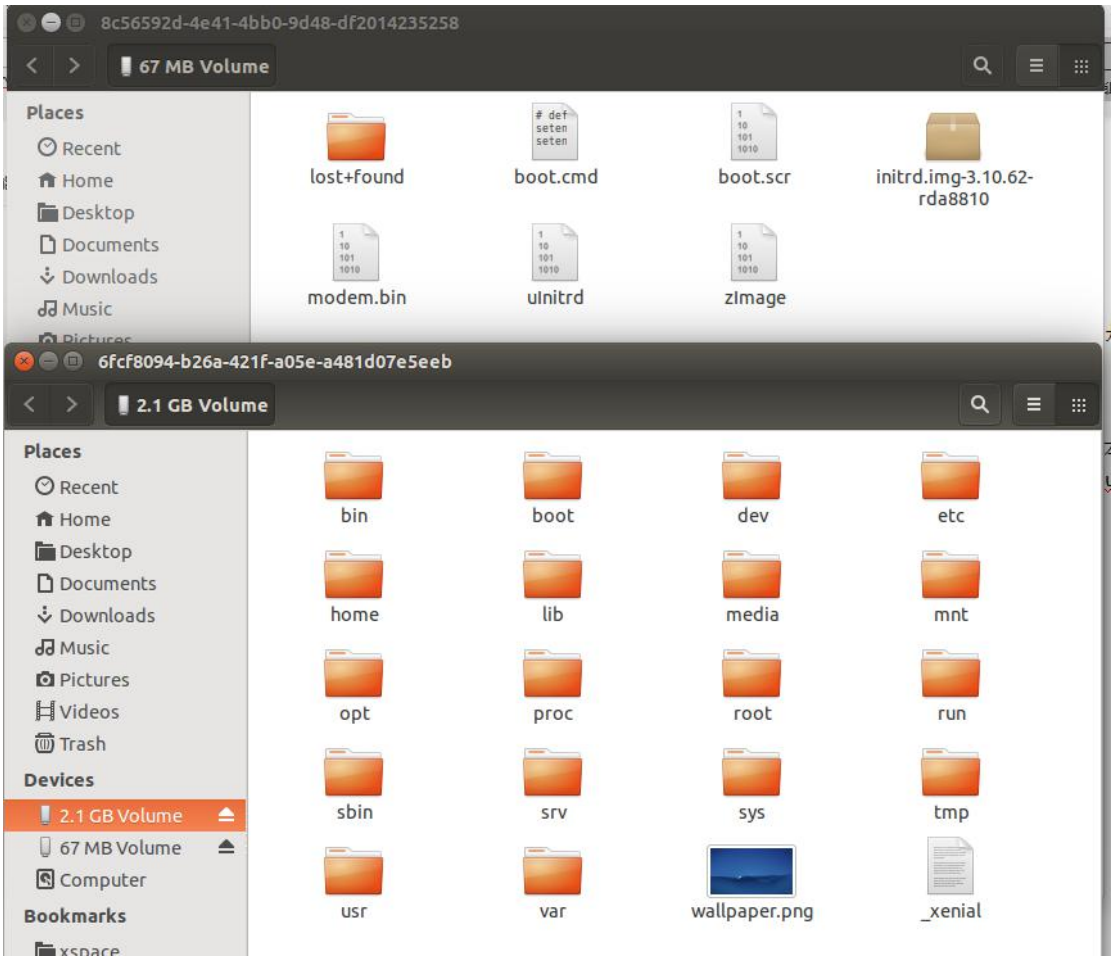
```
$ tar -xvf    2G-IOT_linux_nand_boot.tar.gz
$ cd 2G-IOT_linux_nand_boot/
$ tree -L 1
.
├── 7th
└── script

2 directories, 0 files
```

The 7th contains the burning script and the prepared system image. The script is some scripts for making the root file system.

2、Making the root file system

The method is to make a copy on the basis of the existing image on the official website.
First, you need a tf card with the official website image (debian or ubuntu) burned.

Two partitions will be recognized on the host.

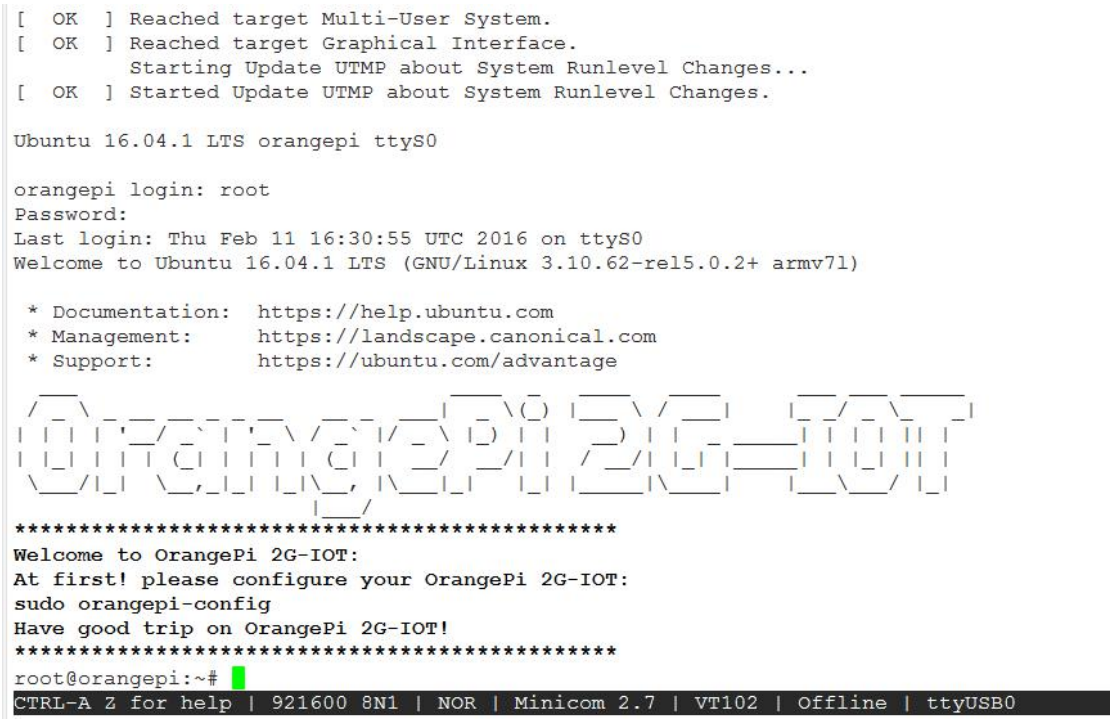



```
ls /media/csy/
6fcf8094-b26a-421f-a05e-a481d07e5eeb    8c56592d-4e41-4bb0-9d48-df2014235258
```

Copy the script folder of the compressed package to the file system

```
sudo cp -rfp script/        \
/media/csy/6fcf8094-b26a-421f-a05e-a481d07e5eeb/root/
```

Insert the TF card into the 2G-iot slot, start the board login



- **Extended file system partition**

```
root@orangepi:~# OrangePi_Resize_rootfs.sh
```

- **Back up the entire root file system**

```
root@orangepi:~# ls
script
root@orangepi:~# cd script/
root@orangepi:~/script# ./mknandfs.sh
```

Need to wait for a period of time, after completion, the nandroot folder will be generated

```
root@orangepi:~/script# ls nandroot/
bin   dev  home  media  opt   root  sbin  sys  usr  wallpaper.png
boot  etc  lib   mnt    proc  run   srv   tmp  var  _xenial
```

- **Copy the necessary files to boot**

```
root@orangepi:~/script# rm -rf nandroot/boot/*
root@orangepi:~/script# cp boot-nand.cmd ./nandroot/boot/
root@orangepi:~/script# mkimage -C none -A arm -T script -d      \
```

```
./nandroot/boot/boot-nand.cmd ./nandroot/boot/boot-nand.scr

root@orangepi:~/script# mount /dev/mmcblk0p1 /mnt
root@orangepi:~/script# cp /mnt/uInitrd ./nandroot/boot/
root@orangepi:~/script# cp /mnt/zImage ./nandroot/boot/
root@orangepi:~/script# cp /mnt/modem.bin ./nandroot/boot/
```

At this point, the file system is ready, the next step is to generate a ubifs format image

- **Generate ubifs image**

```
root@orangepi:~/script# apt-get install mtd-utils
root@orangepi:~/script# ./mkubifs.sh
...
root@orangepi:~/script# ./mkubi.sh
```

ubi.img is ready, we can copy ubi.img to the host.
Shut down the board, pull out the tf card, connect it to the host with a card reader, execute the following command on the host to copy ubi.img

```
sudo cp /media/csy/6fcf8094-b26a-421f-a05e-a481d07e5eeb/root/script/ubi.img ./
```

3、Bootloader compilation

- **Download**

```
$ git clone https://github.com/aib/u-boot-RDA8810.git
```

- **Specify the compiler path**

The linux source code compiler is placed in the toolchain directory.

```
$ cd OrangePiRDA
$ ls toolchain/
arm-linux-gnueabi    bin    lib    libexec    README.md
```

Take my system as an example, run the following command to import the compiler path into the environment variable, please fill in according to the actual path.

```
$ export PATH=/xspace/OrangePi/RDA/SRC/OrangePiRDA/toolchain/bin:$PATH
```

- **Compile PDL binary files**

```
$ cd u-boot-RDA8810
$ git checkout nand-boot
$ make CROSS_COMPILE=arm-linux-gnueabi- clean rda8810_config
$ make CROSS_COMPILE=arm-linux-gnueabi- pdl=1 PDL
```

Please save the generated pdl1.bin and pdl2.bin to other directories, because the following compilation process will delete these two files

- **Compile bootloader**

```
$ make CROSS_COMPILE=arm-linux-gnueabi- clean rda8810_config
$ make CROSS_COMPILE=arm-linux-gnueabi-
```

U-boot.rda is generated in the current directory

At this point, all the images required for burning are ready.