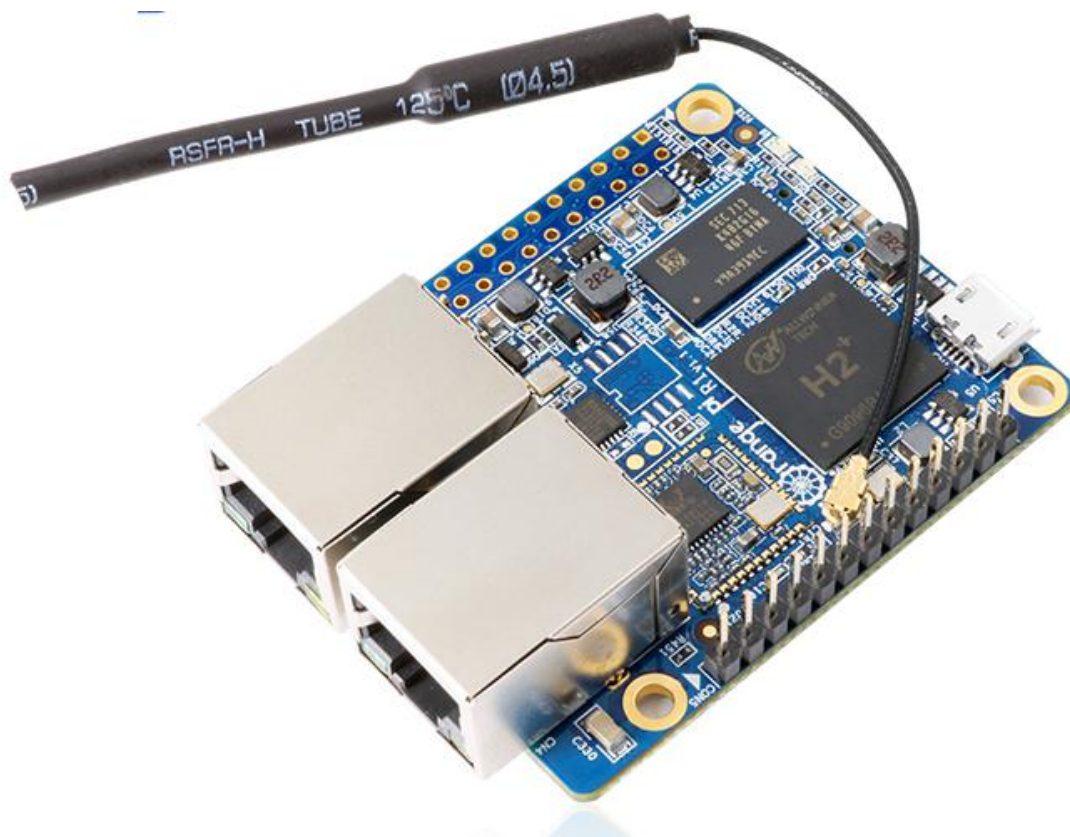




Orange Pi R1

用户手册



V1.1



目录

| | |
|------------------------------|----|
| 一、 Orange Pi R1 的基本特性..... | 1 |
| 1. 什么是 Orange Pi R1..... | 1 |
| 2. Orange Pi R1 的用途..... | 1 |
| 3. Orange Pi R1 是为谁设计的..... | 1 |
| 4. Orange Pi R1 的硬件特性..... | 1 |
| 5. GPIO 规格..... | 4 |
| 二、 开发板使用介绍..... | 6 |
| 1. 步骤 1: 准备需要的配件..... | 6 |
| 2. 步骤 2: 准备板子启动需要的 TF 卡..... | 6 |
| 3. 步骤 3: 启动你的香橙派开发板..... | 10 |
| 4. 步骤 4: 正确关闭你的香橙派开发板..... | 13 |
| 5. 其他设置..... | 13 |
| 6. 通用软件配置..... | 16 |
| 三、 Linux 内核源码编译..... | 26 |
| 1. 下载 Linux 源码..... | 27 |
| 2. 项目源码的编译..... | 27 |
| 3. 内核镜像文件和库的替换..... | 29 |
| 四、 Linux 内核源码编译..... | 31 |
| 1. JDK 的安装..... | 32 |
| 2. 安装平台支持软件..... | 32 |
| 3. 下载 Android 源码..... | 33 |
| 4. 编译工具链的安装..... | 33 |
| 5. Lichee 源码的编译..... | 33 |
| 6. Android 源码编译命令..... | 34 |
| 五、 使用工程配置化文件..... | 37 |



| | |
|----------------------------|----|
| 1. sys_config.fex 简介..... | 37 |
| 2. 例程..... | 37 |
| 六、 OrangePi 驱动程序开发..... | 40 |
| 1. 设备驱动和应用程序的编写..... | 41 |
| 2. 设备驱动的编译方法..... | 43 |
| 3. 交叉编译器编译应用程序..... | 45 |
| 4. 驱动和程序的运行方式..... | 47 |
| 七、 串口调试工具介绍..... | 48 |
| 1. 基于 Windows 平台的使用..... | 49 |
| 2. 基于 Linux 平台的使用..... | 52 |
| 八、 OrangePi R1 路由系统适配..... | 56 |
| 1. Quagga 的配置、编译和安装..... | 56 |
| 2. 搭建测试环境 一个小型的局域网..... | 57 |
| 3. 配置 OSPF 路由协议..... | 60 |
| 4. 配置 NAT..... | 61 |
| 5. 配置 DHCP..... | 62 |



一、Orange Pi R1 的基本特性

1. 什么是 Orange Pi R1

香橙派是一款开源的开发板卡片电脑，新一代的 arm 开发板，它可以运行 Android 4.4、Ubuntu、Debian 等操作系统，兼容树莓派。香橙派开发板电脑使用全志 H2 系统级芯片，同时拥有 256MB DDR3 内存。

2. Orange Pi R1 的用途

我们可以用它搭建：

- 一台计算机
- 一个无线网络服务器
- 游戏机
- 音乐播放器
- 高清视频播放器
- 扬声器
- Android
- Scratch
-

还有更多的其他功能，因为 Orange Pi R1 是开源的。

3. Orange Pi R1 是谁设计的

Orange Pi R1 不仅仅是一款消费品，同时也是给任何想用技术来进行创作创新的人设计的。它是一款非常简单、有趣、实用的工具，你可以用它去打造你身边的世界。

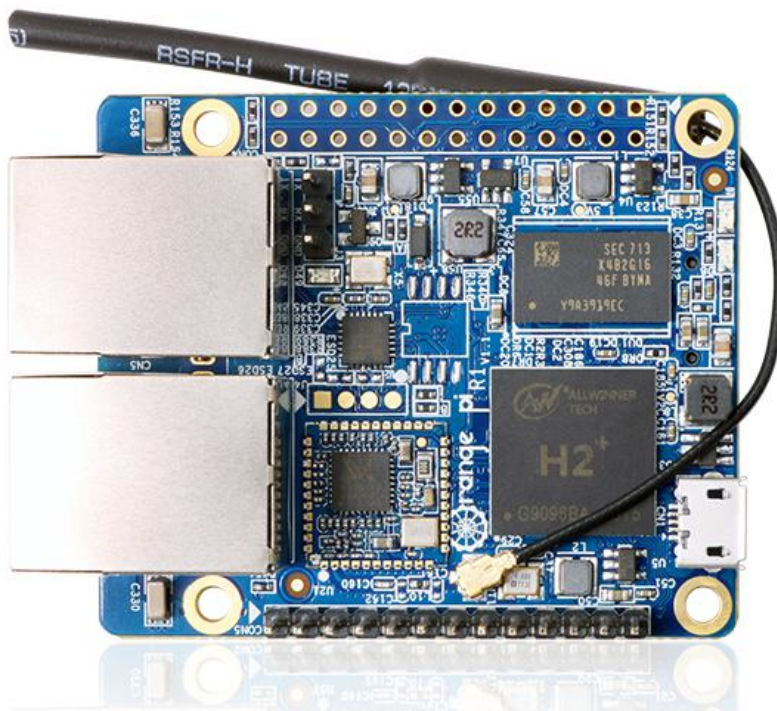
4. Orange Pi R1 的硬件特性



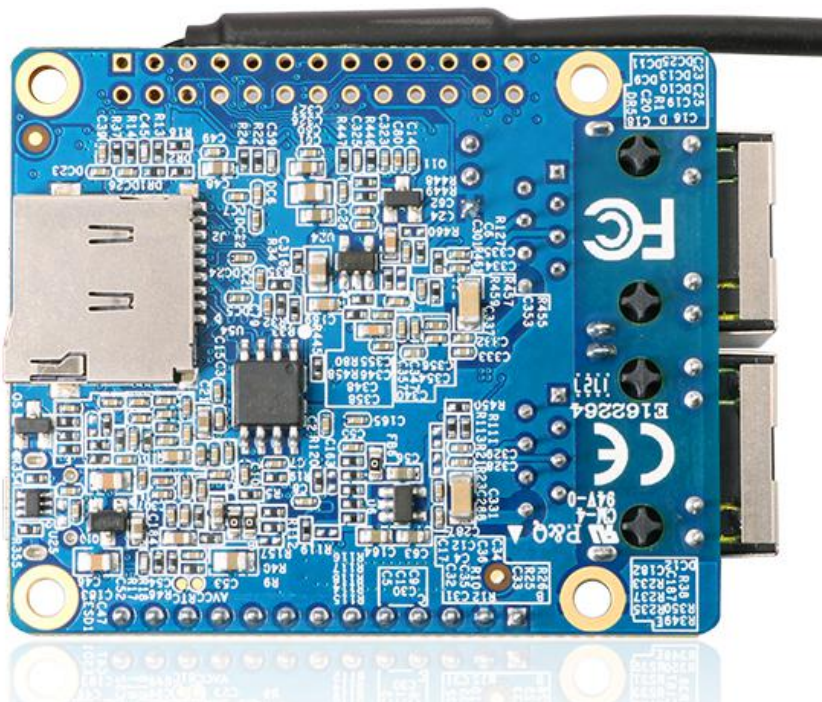
| 硬件特性介绍 | |
|----------------------------|--|
| CPU | 高性能全志 H2 芯片，4 核 32 位 Cortex-A7 |
| GPU | Mali400MP2 GPU |
| 内存 | 256MB DDR3（与 GPU 共享） |
| 板载存储 | TF 卡（最大 32GB） / spi Flash(16MB) |
| 板载 wifi | RTL8189ETV, IEEE 802.11 b/g/n |
| 板载网络 | 100M 以太网 1 RJ45（板载百兆） 100M 以太网 1 RJ45（RTL8152B） |
| 视频输出 | 支持 13pin 拓展板（AV-out） |
| 音频输出 | 支持 13pin 拓展板（cvbs） |
| 电源 | USB OTG 用作电源输入 |
| Low-level 外设 | 26pin 接头，兼容树莓派 13pin 接头（带 2 个 USB，IR，AUDIO） |
| GPIO(1x3) 口 | UART, ground. |
| LED 灯 | 电源指示灯和状态指示灯 |
| 支持的操作系统 | Android, Ubuntu, Debian 等操作系统, 兼容树莓派. |
| 外观规格介绍 | |
| 产品尺寸 | 45mm×60mm |
| 重量 | 48g |
| OrangePi™是深圳市迅龙软件有限公司的注册商标 | |



顶层视图:

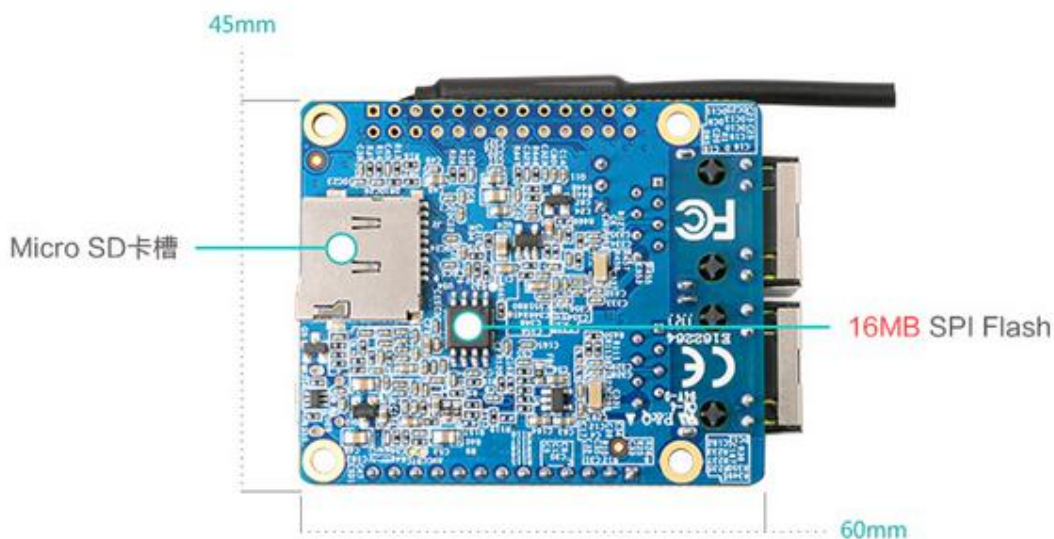
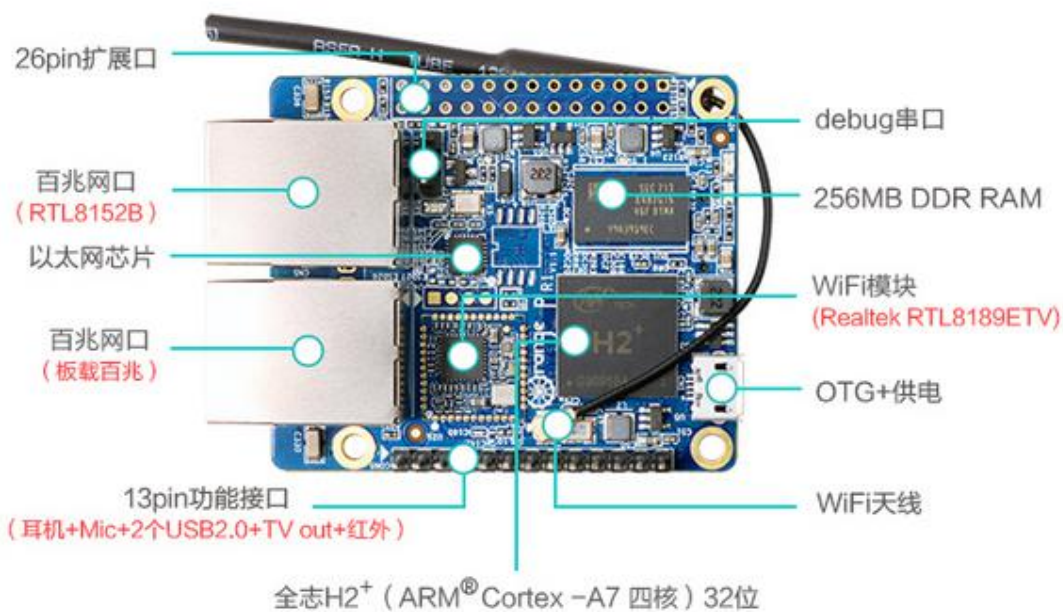


底层视图:





接口详情图:



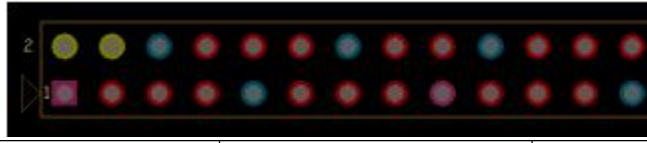
5. GPIO 规格

Orange Pi 26-pin GPIO

香橙派开发板有 26 pin GPIO 接头用来匹配 Raspberry Pi (树莓派) 的 Model A 和 Model B。



下图是香橙派开发板的 GPIO 引脚线：



| OrangePi_R1 (H2) | | |
|------------------|------------|------|
| CON3-P01 | VCC-3V3 | |
| CON3-P02 | VCC-5V | |
| CON3-P03 | TWI0-SDA | PA12 |
| CON3-P04 | VCC-5V | |
| CON3-P05 | TWI0-SCK | PA11 |
| CON3-P06 | GND | |
| CON3-P07 | PWM1 | PA6 |
| CON3-P08 | UART1_TX | PG6 |
| CON3-P09 | GND | |
| CON3-P10 | UART1_RX | PG7 |
| CON3-P11 | UART2_RX | PA1 |
| CON3-P12 | PA7 | PA7 |
| CON3-P13 | UART2_TX | PA0 |
| CON3-P14 | GND | |
| CON3-P15 | UART2_CTS | PA3 |
| CON3-P16 | TWI1-SDA | PA19 |
| CON3-P17 | VCC3V3-EXT | |
| CON3-P18 | TWI1-SCK | PA18 |
| CON3-P19 | SPI1_MOSI | PA15 |
| CON3-P20 | GND | |
| CON3-P21 | SPI1_MISO | PA16 |
| CON3-P22 | UART2_RTS | PA2 |
| CON3-P23 | SPI1_CLK | PA14 |
| CON3-P24 | SPI1_CS | PA13 |
| CON3-P25 | GND | |
| CON3-P26 | PA10 | PA10 |



二、开发板使用介绍

按照如下步骤，你可以在很短的时间内配置并使用你的香橙派开发板。启动你的香橙派开发板需要完成以下几步。

1. 步骤 1：准备需要的配件

第一次使用香橙派开发板，你需要准备如下一些配件(建议配合扩展板使用)：

| 编号 | 项目 | 最低要求及说明 |
|----|---------|---|
| 1 | TF卡 | 最小 8GB容量，class 10 级，建议使用品牌TF卡 |
| 3 | AV 视频线 | 如果没有 HDMI 显示器，可以使用 AV 视频线连接模拟显示设备 |
| 4 | 键盘鼠标 | 任何标准usb接口的键盘鼠标都可以。键盘和鼠标可能会需要较大的功率，所以可能需要使用一个 USB 集线器。 |
| 5 | 网线（可选） | 有线网络 |
| 6 | 电源适配器 | OTG 用作电源输入。 |
| 7 | 音频线（可选） | 你可以选择一个 3.5 mm 接口的音频线来体验立体音效。 |



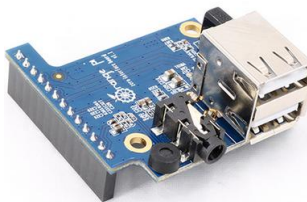
AV 视频线



TF 卡



电源适配器



扩展板

2. 步骤 2：准备板子启动需要的TF卡



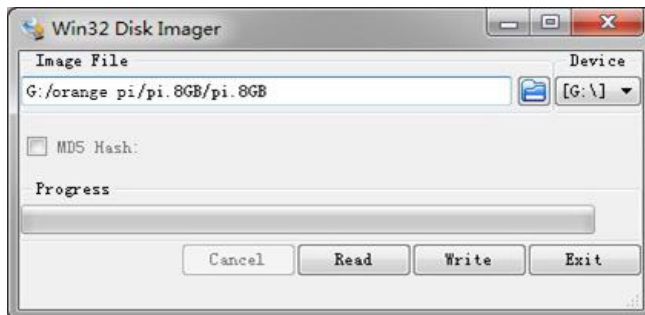
为了能够正常使用香橙派开发板，必须先在 TF 卡上安装操作系统。

1) 基于 Windows 平台将 linux 操作系统写入 TF 卡

- a. 把TF卡插入电脑中，TF卡的容量必须比操作系统镜像大，通常需要 8GB或更大的容量。
- b. 格式化TF卡
 - i 下载TF卡格式化工具，例如TF Formatter，下载地址
https://www.sdcard.org/downloads/formatter_4/eula_windows/
 - ii 解压下载的文件，并运行 setup.exe
 - iii 在“选项设置”选项里，设置“格式化类型”选项为快速格式化，“逻辑大小调整”选项为“开启(ON)”



- iv 确认插入的TF卡盘符和选择的盘符一致
 - v 点击“格式化”按钮
- c. 从下载页面下载操作系统镜像文件，页面地址如下：
<http://www.orangepi.cn/downloadresourcescn/>
- d. 解压下载的文件（除Android系统外的系统可用该方法来烧写，Android系统需要用其他的模式来烧写）
- e. 右键单击下载的文件，选择“解压文件”写入镜像文件到TF卡
 - i 下载镜像写入工具，例如 Win32Diskimager，下载页面：
<http://sourceforge.net/projects/win32diskimager/files/Archive/>
 - ii 选择已经解压的镜像文件路径





- iii 点击 “Write” 按钮，耐心等待镜像写入
- iv 镜像写入完成后，点击 “Exit” 按钮

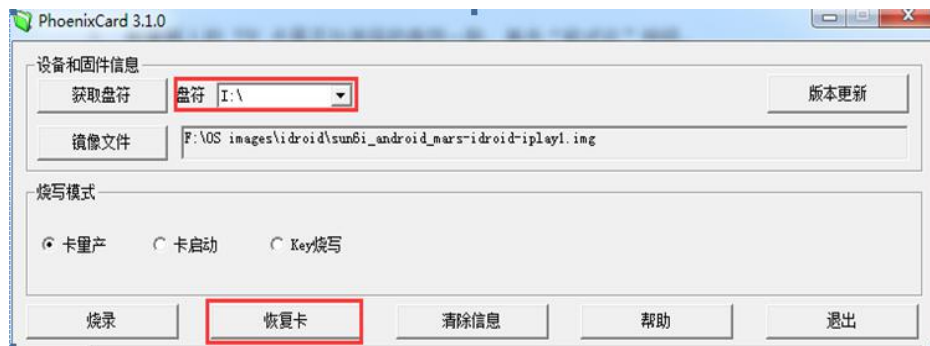
2) 基于 Linux 平台将 linux 操作系统写入 TF 卡

- a. 把 TF 卡插入电脑，TF卡的容量必须比操作系统镜像大，通常需要 4GB或更大
- b. 格式化 TF 卡
 - i 运行 `fdisk -l` 命令确认 TF 卡的盘符
 - ii 运行 `umount /dev/sdx` 卸载 TF 卡的所有分区
 - iii 运行 `sudo fdisk /dev/sdx` 命令. 使用 `o` 命令去删除 TF 卡的所有分区，然后使用 `n` 命令去添加一个新的分区，最后使用 `w` 命令保存退出
 - iv 运行 `sudo mkfs.vfat /dev/sdx1` 命令去格式化刚生成的 TF 卡分区为 FAT32 格式(根据你的 TF 卡盘符来替换 x)
- c. 从下载页面下载操作系统镜像文件，页面地址如下：
<http://www.orangepi.cn/downloadresourcescn/>
- d. 解压下载的文件右键单击下载的文件，选择 “解压文件”
- e. 写入镜像文件到 TF 卡
 - i 运行 `sudo fdisk -l` 命令确认TF卡的盘符
 - ii 确认镜像文件的hash key或者是md5 和下载页面提供的一致（可选）
`shasum [path]/[imagename]`
这将会输出一长串数字，应该和你下载的镜像页面的“SHA-1” 那一行匹配
 - iii 运行 `umount /dev/sdxx` 命令卸载TF卡的所有分区
 - iv 运行 `sudo dd bs=4M if=[path]/[imagename] of=/dev/sdx` 命令去写入镜像文件，耐心等待镜像写入。你可以使用 `sudo pkill -USR1 -n -x dd` 命令去查看烧写进度。

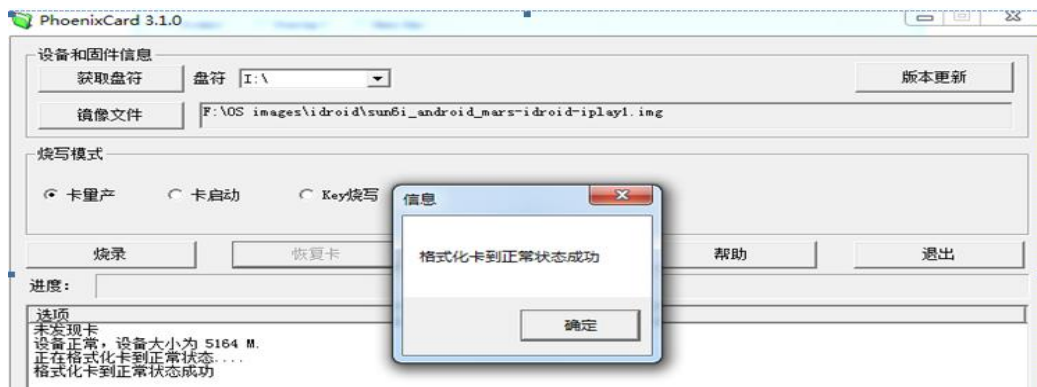
3) 使用 PhoenixCard 工具将 Android 系统镜像写入 TF 卡

Android 系统镜像文件不能在 Linux 下使用 `dd` 命令或者在 Window 用 Win32 Diskimager 工具来写入 TF 卡。需要使用工具 PhoenixCard 来写入。

- a. 下载 Android 系统和 PhoenixCard 烧写工具
PhoenixCard 从下面网页中下载：
<http://pan.baidu.com/share/link?shareid=2785461830&uk=1077680202>
Android 系统从下面的网页中下载：
<http://www.orangepi.cn/downloadresourcescn/>
- b. 格式化 TF 卡

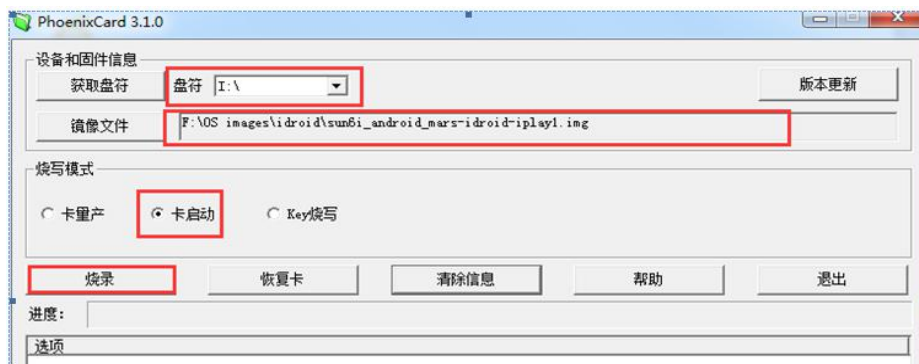


- c. 检查插入的 TF 卡是否与选择的盘符一致，单击“恢复卡”按钮，开始格式 TF

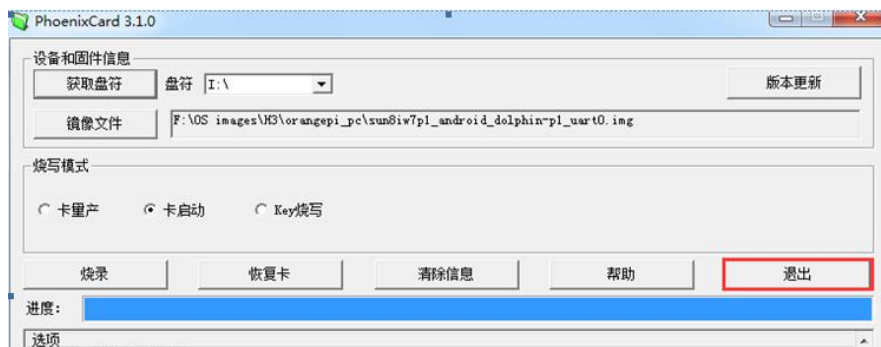


格式化 TF 卡成功后，单击“确定”按钮。

- d. 然后开始将 Android 系统写入 TF 卡，请注意下图红色标记的地方。



点击“烧录”，开始写入 TF 卡，等待烧录完成。





Android 系统成功烧写完成后。单击“退出”按钮。

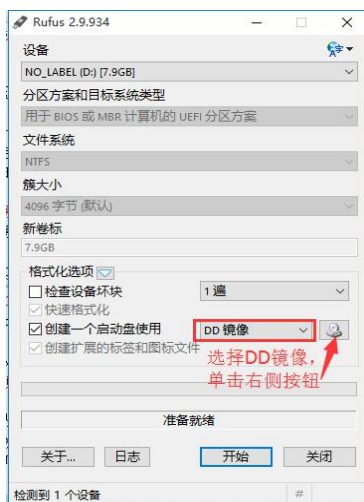
4) Armbian 镜像写入 TF 卡

- a. TF 卡插入电脑中，TF 卡的容量必须比操作系统镜像大，通常需要 8GB。下载操作系统镜像文件，镜像下载页面地址如下：

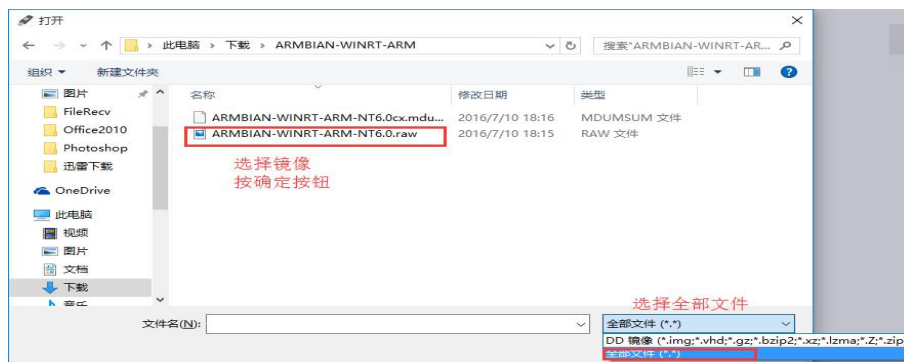
<http://www.armbian.com/download/>

- b. 烧录(镜像)至 TF 卡

i 下载镜像写入工具，例如 Rufus，下载页面：<https://rufus.akeo.ie/>



ii 选择已经解压的镜像文件路径。



iii 点击开始按钮，耐心等待镜像写入

iv 镜像写入完成后，点击“关闭”按钮。

3. 步骤 3: 启动你的香橙派开发板



1) 硬件连接示意图



Orange Pi R1运行Android系统



Orange Pi R1运行Debian系统



Orange Pi R1运行Ubuntu系统



Orange Pi R1电视盒子，高清流畅播放视频



2) 详细启动步骤

- 将写好镜像的TF卡插入香橙派开发板左边缘底部的TF卡插槽
- 你可以使用扩展板上的音频接口来输出视频和音频到模拟TV或显示器。
- 开发板右侧 13pin接口可以连接扩展板，上有 2 个USB和mic, 红外等可供使用
- HDMI旁边的是WIFI模块，你可以将香橙派开发板接入无线网络
- 下边缘的左边是电源输入接口，Micro-usb (OTG) 用作电源输入连接一个 5V和至少 2A的电源适配器。比 2A大也可以。避免使用较小功率的GSM手机充电器，即使上面标明了“5V/2A”，它也不一定能够输出 2A

如果上面的步骤都很顺利的话，开发板将会在几分钟内启动。显示屏上将显示系统的图形界面。首次启动时可能会需要很长的时间，请耐心等待，往后使用时板子将会很快就能启动。

4. 步骤 4：正确关闭你的香橙派开发板

- 你可以使用界面中的关机按键来安全关闭香橙派开发板。
- 你也可以在 shell 里面输入命令来关闭系统：

```
sudo halt
```

or

```
sudo shutdown -h
```

如果以上步骤进行顺利，你将能安全地关闭香橙派开发板，如果直接使用电源按键关闭系统可能会损坏TF卡或者是文件系统。

5. 其他设置

1) 连接无线网络

方法 1:

- 命令行输入

```
$ ifconfig
```

看是否有网卡(wlan*)

- 如果没有，根据网卡的型号加载相应模块

```
$ insmod 8189es.ko
```



例如: RTL8189ETV 对应的 8189es.ko

- c. 输入命令 `ifconfig`, 应该可以看到 `wlan0` (假设是 `wlan0`)
- d. 配置无线, 首先要知道 `ssid` 和 `psk` (账号, 密码), 输入对应的 `wlan*`, `ssid`, `psk`

`$ sudo nano /etc/network/interfaces` (添加如下内容)

```
auto wlan0
```

```
iface wlan0 inet dhcp
```

```
wpa-ssid xxxx
```

```
wpa-psk xxxx
```

- e. 之后重启电脑, 无线即可连接上

`$ sudo reboot`

方法 2:

- a. 在 `/etc/network/` 目录下创建 `wifi` 热点配置文件 `wpa_supplication.conf`, 添加内容如下:

```
network={
    ssid="wifi 热点名字"
    psk="wifi 热点密码"
    priority=1
}
```

- b. 连接 `wifi`, 命令如下:

```
ifconfig wlan0 up
```

```
sudo wpa_supplicant -i wlan0 -c /etc/network/wpa_supplication.conf &
```

```
dhcpcd wlan0 &
```

- c. 测试 `wifi` 连接情况

使用 `iwconfig` 命令, 可以看到 `wlan0` 相关信息。使用 `ping` 命令进行测试

2) vnc和ssh登陆系统

如果没有连接 `hdmi` 的条件, 可以通过 `vnc` 或者是 `ssh` 远程登陆。

- 先用串口登陆, 安装 `ssh`,

```
apt-get install ssh
```

- 修改 `ssh` 配置文件 `/etc/ssh/sshd_config`,



```
# Logging
SyslogFacility AUTH
LogLevel INFO

# Authentication:
LoginGraceTime 120
PermitRootLogin yes
StrictModes yes

RSAAuthentication yes
PubkeyAuthentication yes
#AuthorizedKeysFile %h/.ssh/authorized_keys

# Don't read the user's ~/.rhosts and ~/.shosts files
IgnoreRhosts yes
# For this to work you will also need to configure
# the sshd_config file
RhostsRSAAuthentication no
# similar for protocol version 2
HostbasedAuthentication no
# Uncomment if you don't trust ~/.ssh/known_hosts
#IgnoreUserKnownHosts yes

# To enable empty passwords, change to yes (not recommended)
PermitEmptyPasswords no

# Change to yes to enable challenge-response authentication
# (e.g., RSA, DSA, and some PAM modules and threads)
ChallengeResponseAuthentication no
```

- ifconfig 查看 ip, root 用户用 ssh 登陆

```
curry@curry:~$ ssh root@192.168.1.178
root@192.168.1.178's password:
Welcome to Ubuntu 15.10 (GNU/Linux 3.4.39-02-lobo armv7l)

 * Documentation:  https://help.ubuntu.com/
Last login: Tue Apr 11 15:20:33 2017 from 192.168.1.111
root@OrangePI [10:03:27 AM] [~]
-> #
```

3) 3.5mm输出声音(3.5mm输出声音需要扩展板)

- a. 镜像默认从 hdmi 输出声音, 用 alsamixer 可以查看并切换

ls /etc/asound.conf

里面 card 代表卡号, device 代表设备号

aplay -l 可以查看系统加载的声卡个数详细信息

cat /proc/asound/cards 也可以查看

用 alsamixer 切换声卡之后, 可以使用

alsactl store -f /var/lib/alsa/asound.state 来保存修改的参数

- b. 3.5mm 接口输出需要修改文件系统上的配置文件/etc/asound.conf.

将 card1 修改成 card0 即可, 或者用 amixer 修改, 默认的已经配置好。或者直接使用图形界面的播放器选择声道的切换。

- c. 图形界面切换方式:

打开 smplayer, 选择 options 中的 preferences, 选择 alsa (audiocodec), HDMI



和 audiocodec 同时只能打开一个。

d. 如何使用 mic 录音

```
arecord -d 5 -f cd -t wav 123.wav
```

录音之后, 用

```
aplay 123.wav
```

 放音

6. 通用软件配置

1) 更改默认账号

香橙派默认的登陆账号 orangepi, 为了安全, 建议修改这个默认的 orangepi 账号成为你自己的账号, 例如 zhangsan, 步骤如下:

a. root 账号登陆(不要以 orangepi 用户登录)

b. `$ usermod -l zhangsan orangepi`

修改 orangepi 的账号为 zhangsan

```
@orangepi:~$ usermod -l zhangsan orangepi
```

c. `$ groupmod -n zhangsan orangepi`

修改组

```
@orangepi:~$ groupmod -n zhangsan orangepi
```

d. `$ mv /home/orangepi /home/zhangsan`

把原来 orangepi 目录改掉

```
@orangepi:~$ mv /home/orangepi /home/zhangsan
```

e. `$ usermod -d /home/orangepi orangepi`

把这目录设置成 orangepi 用户的 home 目录

```
@orangepi:~$ usermod -d /home/zhangsan zhangsan
```

f. `$ cat /etc/passwd`



```
pulse:x:112:121:PulseAudio daemon,,,:/var/run/pulse:/bin/false
zhangsan:x:1001:1001:orange pi,,,:/home/zhangsan:/bin/bash
```

以上修改完后就可以使用新账号 zhangsan 登陆了。

2) 设置 U 盘自动挂载

- a. `sudo apt-get install usbmount`
- b. `sudo vim /etc/udev/rules.d/automount.rules`

```
ACTION=="add",KERNEL=="sdb*", RUN+="/usr/bin/pmount --sync --umask
000 %k"
ACTION=="remove", KERNEL=="sdb*", RUN+="/usr/bin/pumount %k"
ACTION=="add",KERNEL=="sdc*", RUN+="/usr/bin/pmount --sync --umask
000 %k"
ACTION=="remove", KERNEL=="sdc*", RUN+="/usr/bin/pumount %k"
```
- c. `udevadm control -reload-rules`

可以参照链接：

<http://unix.stackexchange.com/questions/134797/how-to-automatically-mount-an-usb-device-on-plugin-time-on-an-already-running-sy>

3) 配置系统源

配置系统源为国内源可以使更新，安装软件时速度更快，下面以 Ubuntu 为例

- a. 打开源文件

```
$ sudo vi /etc/apt/sources.list
```

```
root@curry:/home/curry# vim /etc/apt/sources.list
root@curry:/home/curry#
```

- b. 编辑源文件

把源文件替换成自己喜欢的源，例如 Ubuntu 16.04 的中科大源为：

```
deb http://mirrors.ustc.edu.cn/ubuntu-ports/ xenial main multiverse restricted
universe
deb http://mirrors.ustc.edu.cn/ubuntu-ports/ xenial-backports main multiverse
restricted universe
deb http://mirrors.ustc.edu.cn/ubuntu-ports/ xenial-proposed main multiverse
restricted universe
deb http://mirrors.ustc.edu.cn/ubuntu-ports/ xenial-security main multiverse
restricted universe
deb http://mirrors.ustc.edu.cn/ubuntu-ports/ xenial-updates main multiverse
```




```
restricted universe
deb-src http://mirrors.ustc.edu.cn/ubuntu-ports/ xenial main multiverse restricted
universe
deb-src http://mirrors.ustc.edu.cn/ubuntu-ports/ xenial-backports main multiverse
restricted universe
deb-src http://mirrors.ustc.edu.cn/ubuntu-ports/ xenial-proposed main multiverse
restricted universe
deb-src http://mirrors.ustc.edu.cn/ubuntu-ports/ xenial-security main multiverse
restricted universe
deb-src http://mirrors.ustc.edu.cn/ubuntu-ports/ xenial-updates main multiverse
restricted universe
```

(注：在此源中xenial字样是版本代号，若是Ubuntu其他版本替换成相应版本代号即可，版本代号可在网上查到)

4) 远程桌面安装

软件有很多，VNC、XRDP、X2GO 等，X2GO 功能多些，桌面色彩还原很好不需要多少配置，其次 XRDP、VNC，xrdp 比 vnc 更安全些。

a. \$ sudo apt-get install tightvncserver

安装 vnc

```
apt-get install tightvncserver
```

b. vncpasswd

设置密码；不运行此命令，直接运行 vncserver 也会提示你输入密码，一共两次，当提示是否需要只读密码时选 N 即可。

```
root@curry:/home/curry/tools/minidlna/minidlna-1.1.0# vncpasswd
Using password file /root/.vnc/passwd
VNC directory /root/.vnc does not exist, creating.
Password:
Verify:
```

c. 通过 vncserver 或者 vncserver:1(vncserver:2)……等开启一个或多个桌面，也可以通过完整的命令传送更多参数，如

```
vncserver :1 -geometry 1024x768 -depth 16 -pixelformat rgb565
```

(注意，如果安装时提示未找到文件或其他错误，请运行 sudo apt-get update 更新下软件源再尝试安装)

5) 设置系统中文化

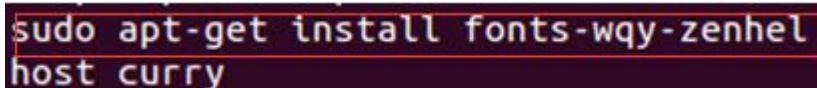
a. 打开终端，输入如下命令后回车：



```
$ sudo apt-get install --reinstall locales
```

- b. 输入如下命令回车执行:

```
$ sudo vi /etc/default/locale
```



```
sudo apt-get install fonts-wqy-zenhei
host curry
```

- c. 按 i 键进入修改模式修改其内容为:

```
LANG="zh_CN.UTF-8"
LC_ALL="en_US.UTF-8"
LANGUAGE="zh_CN:en_US:en"
LC_NUMERIC="en_US.UTF-8"
LC_TIME="en_US.UTF-8"
LC_MONETARY="en_US.UTF-8"
LC_PAPER="en_US.UTF-8"
LC_IDENTIFICATION="en_US.UTF-8"
LC_NAME="en_US.UTF-8"
LC_ADDRESS="en_US.UTF-8"
LC_TELEPHONE="en_US.UTF-8"
LC_MEASUREMENT="en_US.UTF-8"
```

- d. 按 ESC 键退出编辑模式, 输入 wq 保存退出

- e. 安装中文字体:

```
$ sudo apt-get install fonts-wqy-zenhei
```

- f. 如需 HOME 文件夹也为中文, 可以删除/home/orangepi/.config 下面文件夹中文文件 user-dirs.dirs 和 user-dirs.locale 然后重启。

- g. 如果连接上网络, 系统的时间与当地的时间不对, 选择该目录下对应的洲的城市, 替换 etc 目录下的 localtime.

例: 替换使用上海的时间

```
$ cp /usr/share/zoneinfo/Asia/Shanghai /etc/localtime
```

6) NAS 和 DLNA 配置

- a. NAS:

网上有很多教程可以借鉴, <http://www.geekfan.net/5003/>, 里面将的很详细, 可以一步一步跟着操作即可.

- b. DLNA



主要通过 minidlna 软件实现媒体资源的局域网内共享，比如视频、音乐等。安装步骤如下：

i \$ sudo apt-get install minidlna

ii 执行如下命令修改配置文件：

\$ sudo nano /etc/minidlna.conf

注：也可以用其他文本编辑器进行修改。

iii 增加以下内容：

media_dir=A, /nas 路径/DLNA/Music

media_dir=V, /nas 路径/DLNA/Video

media_dir=P, /nas 路径/DLNA/Picture

db_dir=/nas 路径/DLNA/log

db_dir=/nas 路径/DLNA/db

iv ctrl +o 回车, ctrl +x 保存退出。

v 分别建立以上的文件夹，注意，路径一致，给读写权限

\$ sudo chmod 755 /nas 路径/DLNA/Music

vi 重启 minidlna 让配置生效：/etc/init.d/minidlna restart, 完毕。

vii 电脑上把相应的文件通过 samba 传到对应的文件夹。

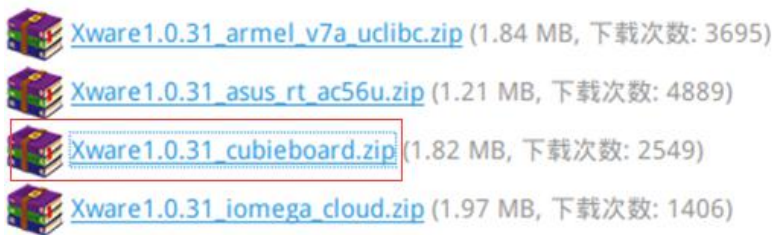
注：在移动设备上建议下载 MoliPlayer，安卓和 IOS 都有，效果不错，蓝光无压力。

7) 迅雷远程下载

a. 首先去到迅雷路由论坛下载所需要的安装包，稳定版地址：

<http://luyou.xunlei.com/thread-12545-1-1.html>

下载 Xware1.0.31_cubieboard 压缩包



可以下载最新的测试版：

<http://luyou.xunlei.com/thread-15167-1-1.htm>

b. 相应的版本解压缩上传到香橙派上面后，进入所在目录。这里建议解压缩好后把文件夹改名为 xunlei

c. **1.0.31 版本安装方法：**

i \$ cd /xxx/xunlei xxx 为你拷如 xunlei 安装文件的目录

ii \$ chmod 755 portal



iii \$./portal

```
root@curry:/home/curry/Downloads/xunlei# ls
EmbedThunderManager ETMDaemon portal vod_httpserver
root@curry:/home/curry/Downloads/xunlei# chmod 755 portal
root@curry:/home/curry/Downloads/xunlei#
```

iv 运行后会出现如下界面，得到一个激活码

```
YOUR CONTROL PORT IS: 9000

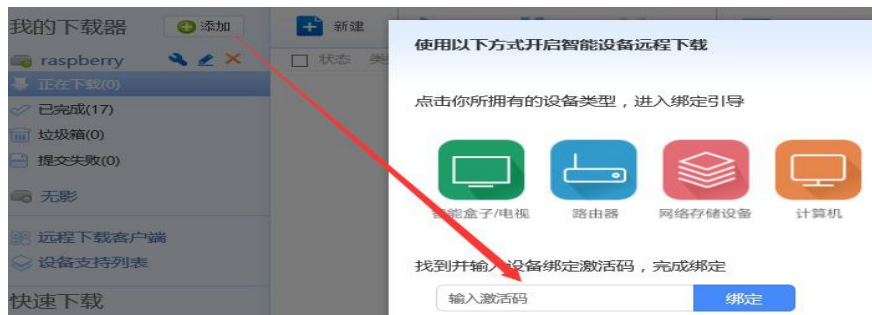
starting xunlei service...
etm path: /home/echo/xunlei
execv: /home/echo/xunlei/lib/ETMDaemon.

getting xunlei service info...
Connecting to 127.0.0.1:9000 (127.0.0.1:9000)

THE ACTIVE CODE IS: ██████████ ← 这里你会得到一个激活码

go to http://yuancheng.xunlei.com, bind your device with the active code.
finished.
```

v 复制此号码，到 <http://yuancheng.xunlei.com>，（需要迅雷账号登陆），然后点右上角的添加，按下图填入激活码完成绑定



vi 设置开机运行

```
$ sudo nano /etc/rc.local 在 exit 0 行之上添加如下两行
    cd /xx/xunlei
    ./portal &
ctrl +o、回车、ctrl +x 保存退出。
```

d. 3.0.32.253 版本的安装:

- i \$ cd /xxx/xunlei xxx 为你拷如 xunlei 安装文件的目录
- ii \$ sudo nano thunder_mounts.cfg 修改下载路径



```
#仅接受以下列路径开头的挂载路径
available_mounts
{
    /media/SATA 把这路径改成你NAS路径
}

#下列目录被认为是分区，并在程序运行期间不变
virtual_mounts
{
```

iii \$ chmod +x etm_monitor

iv \$./etm_monitor 运行，会出现 1.0.32 版本一样的激活码页面，然后到迅雷远程页面绑定（上面步骤 4、5）。运行时会有两三个错误，忽略它（shell 类型选择还有 ini 文件的生成）。

v 设置开机运行

\$ sudo nano /etc/rc.local 在 exit 0 行之上添加如下两行

```
cd /xx/xunlei
./etm_monitor &
```

ctrl +o 、回车、ctrl +x 保存退出

之后就可以在电脑上或是手机、开发板上登陆 yuancheng.xunlei.com 进行远程下载了。

8) 如何修改 ext4 文件系统的大小

做好系统运行卡之后立即进行文件系统 rootfs 分区的扩展，这将能大大提升系统的性能，避免空间不足带来的各种繁琐问题。

● 方法 1:

在 PC 机上扩展 TF 卡 rootfs 文件系统分区：

选择指定盘符，右键选择相应盘符，选择“更改大小”调整成自己想要的大小，单击“调整大小”，关闭对话框单击“应用全部操作”，选择应用，完成扩容操作。

● 方法 2:

进入系统，通过 shell 扩容

未分区之前

```
root@OrangePi:~# df -lh
Filesystem      Size  Used Avail Use% Mounted on
/dev/mmcblk0p2  2.0G  565M  1.4G  30% /
devtmpfs         482M   0    482M   0% /dev
tmpfs            490M   0    490M   0% /dev/shm
tmpfs            490M  13M   478M   3% /run
tmpfs            5.0M  4.0K   5.0M   1% /run/lock
tmpfs            490M   0    490M   0% /sys/fs/cgroup
/dev/mmcblk0p1   50M   13M   38M  26% /boot
```

进入系统后使用 fs_resize 扩容：



```
+ DEVICE=/dev/mmcblk0
+ PART=2
+ resize
+ fdisk -l /dev/mmcblk0
+ grep /dev/mmcblk0p2
+ awk {print $2}
+ start=143360
+ echo 143360
143360
+ set +e
+ fdisk /dev/mmcblk0

Welcome to fdisk (util-linux 2.27.1).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.
```

命令行输入: `fs_resize`, 系统自动分区扩容, 重启系统

`df -lh` 检查分区扩容是否成功

```
+ set -e
+ partx -u /dev/mmcblk0
+ resize2fs /dev/mmcblk0p2
resize2fs 1.42.13 (17-May-2015)
Filesystem at /dev/mmcblk0p2 is mounted on /; on-line resizing required
old_desc_blocks = 1, new_desc_blocks = 1
The filesystem on /dev/mmcblk0p2 is now 3871616 (4k) blocks long.

+ echo Done!
Done!
root@OrangePi:/usr/local/sbin# df -lh
Filesystem      Size  Used Avail Use% Mounted on
/dev/mmcblk0p2  15G  566M   14G   4% /
devtmpfs        482M    0  482M   0% /dev
tmpfs           490M    0  490M   0% /dev/shm
tmpfs           490M   13M  478M   3% /run
tmpfs           5.0M  4.0K  5.0M   1% /run/lock
tmpfs           490M    0  490M   0% /sys/fs/cgroup
/dev/mmcblk0p1  50M   13M   38M  26% /boot
```

a. 扩大文件系统

i 启动到 Linux, `umount /dev/sdb1` 和 `/dev/sdb2`, 若提示磁盘忙的话使用 `fuser` 将正在使用磁盘的程序破坏掉。(推荐使用另外的 Linux 启动盘来引导系统)

ii 使用 `fdisk /dev/sdb` 调整分区大小, 进去之后, 输入 `p`, 记下要扩大分区起始位置的大小。

iii 输入 `d` 删除需要变化的分区(我的文件系统是 `/dev/sdb2`, 也就是第二个分区)

iv 输入 `n` 建立新分区, 注意分区起始位置和之前删除分区保持一致, 结束输入你期望的大小。

v 输入 `w` 保存分区表

vi 使用下面命令检查文件系统(保证文件系统没有错误, 为修改文件系统做准备)

`e2fsck -f /dev/sdb2`

vii 调整分区大小

`resize2fs /dev/sdb2`

viii 可以挂载一个磁盘分区, 看大小是否改变

b. 缩小文件系统

i 启动到 Linux, `umount` 掉 `/dev/sdb1` 和 `/dev/sdb2`, 若提示磁盘忙的话使用 `fuser` 将正在使用磁盘的程序破坏掉。(推荐使用另外的 Linux 启动盘来引导系统)。

ii 使用 `fsck` 检查文件系统(保证文件系统没有错误, 为修改文件系统做准备)

`e2fsck -f /dev/sdb2`



iii 修改文件系统的大小(使用 `resize2fs`)

```
resize2fs /dev/sdb2 900M
```

数字后面的“s”表示通过扇区数(按每扇区 512 字节算)来指定文件系统的大小。还可以指定 K(KB), M(MB), G(GB)等。

iv 使用 `fdisk /dev/sdb` 调整分区大小, 进去之后, 输入 `p`, 记下要扩大分区起始位置的大小。因为 `fdisk` 无法动态的修改分区大小, 所以只能先删除分区, 然后再重建一个小一点的分区 (size 要计算好, 必须要能容纳下我们在上一步调整后的文件系统)。

v 输入 `d` 删除需要变化的分区(我的文件系统是 `/dev/sdb2`, 也就是第二个分区)

vi 输入 `n` 建立新分区, 注意分区起始位置和之前删除分区保持一致, 结束输入你期望的大小。另外, 如果你修改的是可引导分区, 注意它的可引导标志要保留, 否则可能导致系统无法 boot。

上述方法是使用 `fdisk` 和 `resize2fs` 来修改分区和文件系统, 也可以使用 `gparted`。`gparted` 提供了图形界面, 而且它在 `resize` 分区的同时会帮你 `resize` 文件系统, 用起来更方便, 不容易出错。目前官网的 `Lubuntu` 和 `raspbian` 暂不可用。

9) 设置 `eth0` 或者 `wlan0` 静态 mac 地址

a. 系统未使用 `systemd`, 可直接修改 `rc.local`, 并添加如下内容

```
$ vim /etc/rc.local
MAC=00:e0:4c:a1:2b:d4
ifconfig wlan0 down
ifconfig wlan0 hw ether $MAC
ifconfig wlan0 up
dhclient &
```

重启电脑即可, `ifconfig` 查看 mac 地址是否改变

b. 系统使用 `systemd`, 除了做上述的步骤之外, 还需要添加服务

```
$ cd /etc/systemd/system/
$ vim change_mac_address.service (服务名字可以自己定, 格式如下)
[unit]
Description=Change OrangePi Wifi mac address
[Service]
ExecStart=/etc/rc.local
RemainAfterExit=yes
[Install]
WantedBy=multi-user.target
$ systemctl enable change_mac_address.service
```

修改 `eth0` 的 mac 地址和 `wlan0` 的一样, 替换 `wlan0` 为 `eth0` 即可。



10) Orangepi 安卓 root

开发板上默认安装的 android 系统已经有 root 权限，但是缺少授权管理软件，下面的步骤为添加授权软件

需要的软件 UsbModeSwitch.apk 和 UPDATE-SuperSU-v2.46.zip，电脑安装 kingroot，并且确保开发板的 otg 可以连接到电脑。

a. 打开 adb 调试模式

将 UsbModeSwitch.apk 用 U 盘或读卡器安装进开发板系统中，打开 UsbModeSwitch.apk 应用，勾选“enable usb device mode”，然后用调试线(要数据线，有些线不识别)将开发板的 otg 接口与计算机相连，一般计算机将会自动搜索安装 adb 驱动软件，若驱动安装失败，可安装 PC 版豌豆荚，可成功安装对应的驱动软件(或者去创客群共享文件里面下载 otg.zip)。

b. 成功将开发板与计算机相连后，打开计算机的命令行模式，输入 adb 相关命令(需要安装 adb 调试命令，豌豆荚里自带有 adb 命令)，命令如下

```
adb remount
```

```
adb shell
```

windows(win+r)命令行进入命令行模式，进入 kingroot 的目录. 执行下列步骤

```
adb shell
```

```
root@rabbit-p1:/ # mkdir /tmp
```

```
root@rabbit-p1:/ # cd /system/bin
```

```
root@rabbit-p1:/ # mount -o remount, rw /system
```

```
root@rabbit-p1:/system/bin # ln -s busybox-smp unzip
```

退出 adb shell 模式

```
root@rabbit-p1:/exit (或 Ctrl + C)
```

解压 UPDATE-SuperSU-v2.46.zip,

将解压之后得到的 META-INF/com/google/android/update-binary 文件放到指定的目录中。

```
adb push /path/UPDATE-SuperSU-v2.46.zip /data/local/tmp    path 是文件路径
```

```
adb push /path/ update-binary /data/local/tmp
```

```
adb shell
```

```
root@rabbit-p1:/ #cd /data/local/tmp
```

```
root@rabbit-p1:/ #sh update-binary 0 1
```

```
/data/local/tmp/UPDATE-SuperSU-v2.46.zip
```

```
.....
```

```
.....
```

等待执行完脚本后，输入重启命令 reboot 后，可正常使用设备中的授权软件。

reboot 之后，不一定会出现超级管理员的图标，删除桌面配置文件，重启即可



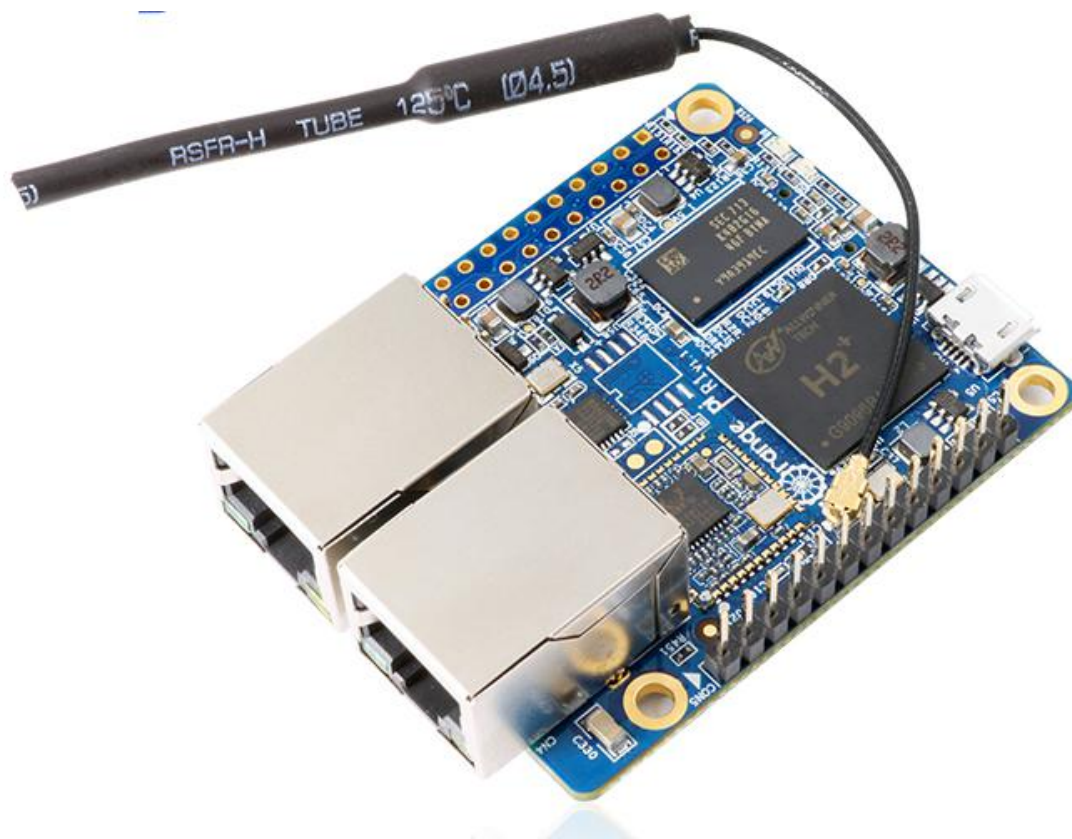
三、Linux 内核源码编译

为了支持快速工程开发，OrangePi 将工程配置选项写入到二进制文件中。系统运行时，通过读取该二进制文件来获得系统运行时的信息，这能大大简化工程开发的时间。

本手册主要描述如何使用该机制来加速客户的工程开发。

***表示通配符，实际值根据路径填写**

需要硬件： 下图Orange Pi开发板一块，读卡器一个，TF卡一张和电源适配器一个





1. 下载 Linux 源码

源代码可以在 OrangePi 官网上进行下载，可以在官方工具找到 lichee 和 android 的具体编译文档：

<http://www.orangepi.org/downloadresources/>

下载完毕后先进行分卷压缩再进行解压，解压之后将可获得以下目录：

```
root@curry:/home/curry/lichee# ls
brandy  buildroot  build.sh  linux-3.4  README  Releaseconfig  tools
root@curry:/home/curry/lichee#
```

uboot源码 交叉编译工具 工程编译脚本 编译脚本 内核源码 工程编译工具

buildroot: 工程编译脚本

brandy: boot, uboot 源码以及开源交叉编译工具 gcc-linaro

linux-3.4: 内核源码

tools: 工程编译工具

build.sh: 编译脚本

2. 项目源码的编译

第一次使用源码，需要全编整个工程。在 lichee 下使用下面命令来全编工程。编译之前的注意事项：

- 进到 lichee 目录下，用命令

```
$ ls -la
```

查看 build.sh 是否有可执行权限，如果没有请修改权限

```
$ chmod 755 build.sh
```

- ls -la 之后如果有发现 !buildconfig 删除

```
$ rm -rf .buildconfig
```

```
root@curry:/home/curry/lichee# ll -a
总用量 128
drwxr-xr-x 7 curry curry 4096 8月 5 10:23 ./
drwxr-xr-x 24 curry curry 4096 8月 5 10:21 ../
drwxr-xr-x 5 curry curry 4096 1月 27 2015 brandy/
-rw-r--r-- 1 root root 152 8月 3 14:39 .buildconfig
drwxr-xr-x 14 curry curry 4096 1月 27 2015 buildroot/
-rwxr-xr-x 1 curry curry 55 1月 27 2015 build.sh*
lrwxrwxrwx 1 curry curry 33 7月 12 15:18 .git -> ../.allgitrepositories/lichee.git
-rw-r--r-- 1 curry curry 351 1月 27 2015 .gitignore
drwxr-xr-x 25 curry curry 4096 8月 4 10:06 linux-3.4/
drwxr-xr-x 3 root root 4096 8月 3 14:39 out/
-rw-r--r-- 1 curry curry 232 1月 27 2015 README
-rw----- 1 curry curry 83529 7月 9 09:02 Releaseconfig
drwxr-xr-x 7 curry curry 4096 1月 27 2015 tools/
root@curry:/home/curry/lichee# chmod 777 build.sh
```

查看当前目录所有文件 解压后发现有的就删除 修改权限



- 使用下面命令全编工程

```
$ ./build.sh config
```

```
root@curry:/home/curry/lichee# ls
brandy buildroot build.sh linux-3.4 README Releaseconfig tools
root@curry:/home/curry/lichee# ./build.sh config
```

使用此命令全编工程

此时系统会提示芯片的选择，如下图，对于 OrangePi ，选择 sun8iw7p1

此时系统会提示平台的选择，如下图，对于 OrangePi ，选择 dragonboard

选择 dolphin，选择 dolphin-p2

```
curry@curry:$ sudo ./build.sh config

Welcome to mkscript setup progress
All available chips:
 0. sun8iw6p1
 1. sun8iw7p1
 2. sun8iw8p1
 3. sun9iw1p1
Choice: 1
All available platforms:
 0. android
 1. dragonboard
 2. linux
Choice: 1
All available business:
 0. dolphin
 1. secure
 2. karaok
Choice: 0
LICHEE_BUSINESS=dolphin
using kernel 'linux-3.4':
All available boards:
 0. dolphin-cmcc-wasu-p1
 1. dolphin-karaok
 2. dolphin-p1
 3. dolphin-p1-secure
 4. dolphin-p2
 5. dolphin-perf
Choice: 4
```

出现此画面等待编译

```
=====
INFO: -----
INFO: build lichee ...
INFO: chip: sun8iw7p1
INFO: platform: dragonboard
INFO: business:
INFO: kernel: linux-3.4
INFO: board: dolphin-p1
INFO: output: out/sun8iw7p1/dragonboard/dolphin-p1
INFO: -----
INFO: build buildroot ...
installing external toolchain
please wait for a few minutes ...
```




等待十五分钟左右，编译完成。

```
make[1]:正在离开目录`/home/curry/Downloads/lichee/buildroot/target/`
generating rootfs...
blocks: 85M -> 112M
Creating filesystem with parameters:
  Size: 117440512
  Block size: 4096
  Blocks per group: 32768
  Inodes per group: 7168
  Inode size: 256
  Journal blocks: 1024
  Label:
  Blocks: 28672
  Block groups: 1
  Reserved block group size: 7
Created filesystem with 3653/7168 inodes and 23020/28672 blocks
e2fsck 1.42.9 (4-Feb-2014)
success in generating rootfs
Build at: 2016年 08月 03日 星期三 14:55:30 CST
INFO: build rootfs OK.
-----
build sun8iw7pi dragonboard lichee OK
-----
```

成功显示

3. 内核镜像文件和库的替换

- 编译完成之后，将会目录下生成如下文件：

libs: lichee/out/sun8iw7pi/android/common/lib/modules/3.4.39

到官方网站上下载镜像文件：<http://www.orangepi.org/downloadresources/>

- 镜像烧录

```
$ sudo dd bs=4M if=*.img of=/dev/sdb
```

- 拔掉读卡器，再插一次。

将编译完产生的内核镜像文件拷贝到第一个分区 (boot分区)

将编译完产生的lib库拷贝到第二个分区 (rootfs分区)

推荐使用官网github上的编译系统

```
curry@curry:$ ls
build.sh  external  kernel  output  scripts  toolchain  uboot
```

build.sh 执行脚本进入到编译的图形化界面

external 里面放的补丁和一些配置文件

kernel 内核目录

output 生成的文件

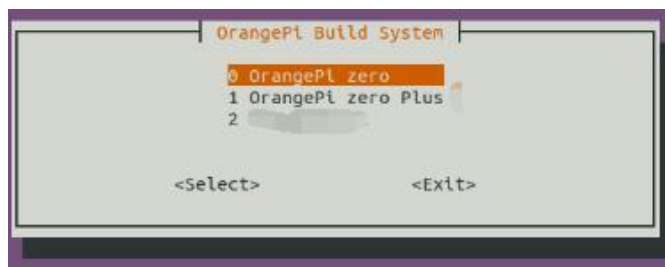
script 编译的脚本

toolchain 存放交叉编译器

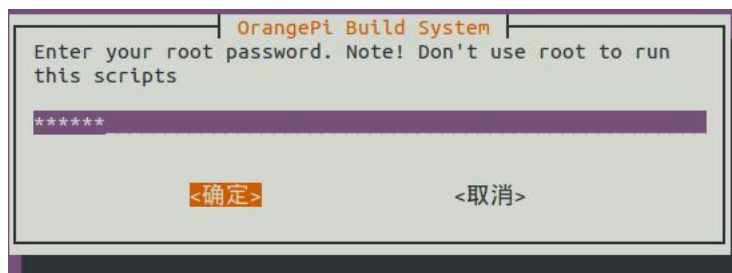


uboot uboot源码

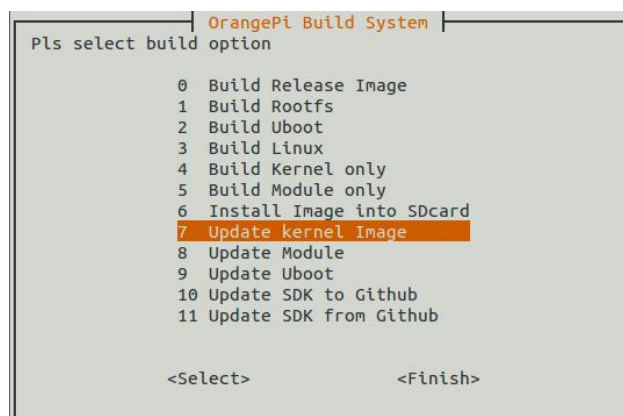
执行./build.sh, 进入图形化界面, 并选择 zero , zero 和 R1 是一样的



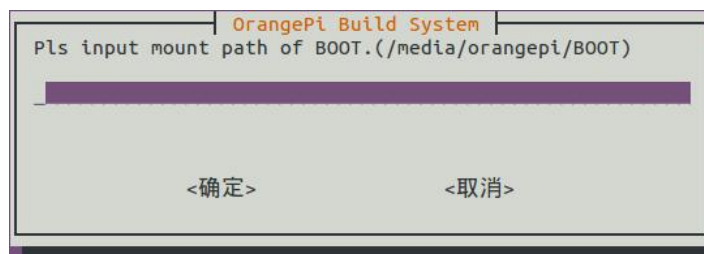
输入root密码



更新内核目录和模块



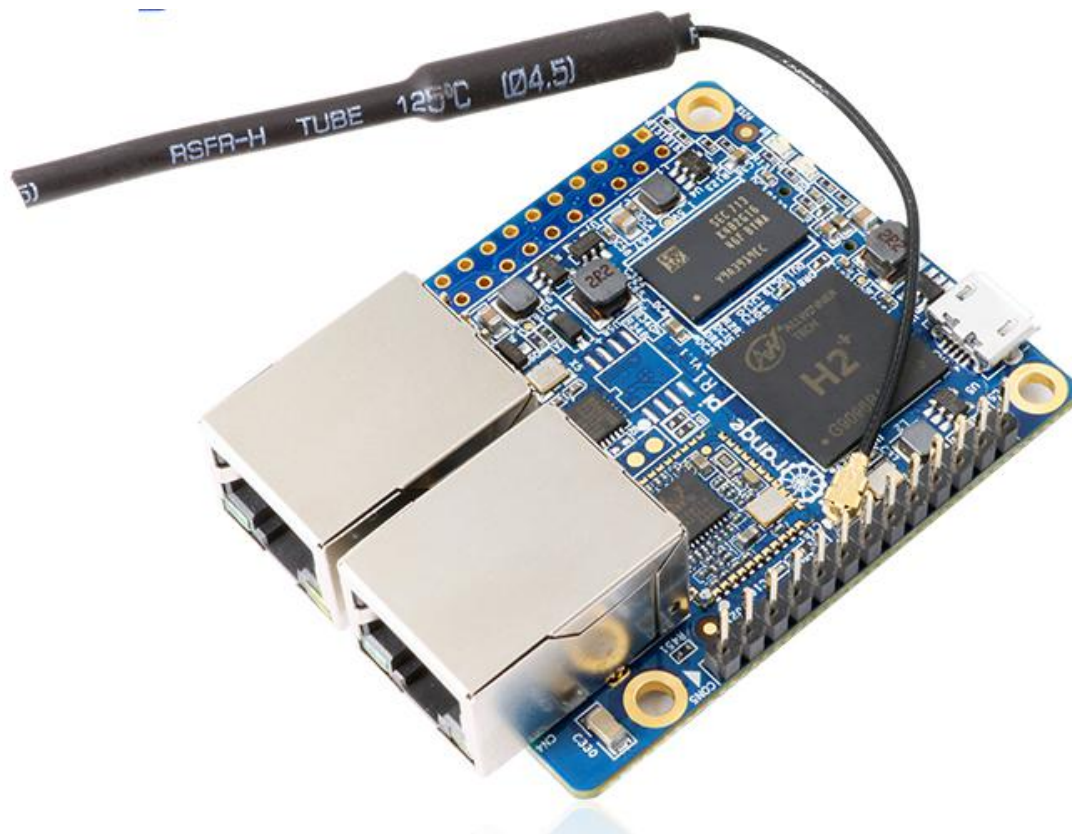
选择自己对应的文件目录更新uImage和modules





四、Linux 内核源码编译

需要硬件： 下图 Orange Pi 开发板一块，读卡器一个，TF 卡一张和电源适配器一个



软件：

Linux主机 硬盘空间至少 50G(可满足一次完全编译)

Linux主机中需要：

Python 的 2.7.3 版本；

GNU Make 的 3.81-3.82 版本；

git 的 1.7 或更高版本；



1. JDK 的安装

下面给出的是 jdk1.6 的安装方法。

- 网上下载并安装 JDK，下载得到 jdk-6u31-linux-x64.bin
- 修改 jdk-6u31-linux-x64.bin 的权限，之前的没有可执行权限
- `$./jdk-6u31-linux-x64.bin`

之后生成一个文件夹

```
root@curry:/home/curry/tools# ls
1_arm-linux-gnueabi-hf-gcc      java1.6_environment.sh  jdk-6u31-linux-x64.bin
arm-linux-gcc-4.5.1-v6-vfp-20120301.tgz  jdk1.6.0_31            opt
```

- 终端输入

注意 JAVA_HOME 是当前目录名字，根据自己存放目录填写

```
root@curry:/home/curry/tools# ls
1_arm-linux-gnueabi-hf-gcc      java1.6_environment.sh  jdk-6u31-linux-x64.bin
arm-linux-gcc-4.5.1-v6-vfp-20120301.tgz  jdk1.6.0_31            opt
```

```
$ export JAVA_HOME=*/jdk1.6.0_31
$ export PATH=$PATH:/$JAVA_HOME/bin
$ export CLASSPATH=.:$JAVA_HOME/lib
$ export JRE_HOME=$JAVA_HOME/jre
```

```
root@curry:/home/curry/tools# export JAVA_HOME=/home/curry/tools/jdk1.6.0_31
root@curry:/home/curry/tools# export PATH=$PATH:/$JAVA_HOME/bin
root@curry:/home/curry/tools# export CLASSPATH=.:$JAVA_HOME/lib
root@curry:/home/curry/tools# export JRE_HOME=$JAVA_HOME/jre
```

- 命令行输入 java 按下 tab 看是否能自动补全 (java)，能说明成功安装确认 java 的版本是不是 1.6

2. 安装平台支持软件

```
$ sudo apt-get install git gnupg flex bison gperf build-essential \
zip curl libc6-dev libncurses5-dev:i386 x11proto-core-dev \
libx11-dev:i386 libreadline6-dev:i386 libgl1-mesa-glx:i386 \
libgl1-mesa-dev g++-multilib mingw32 tofrodos \
python-markdown libxml2-utils xsltproc zlib1g-dev:i386
```



```
$ sudo ln -s /usr/lib/i386-linux-gnu/mesa/libGL.so.1  
/usr/lib/i386-linux-gnu/libGL.so
```

3. 下载 Android 源码

下载链接地址:

<http://www.orangepi.cn/downloadresourcescn/>

解压之后得到这两个目录

```
curry@curry:$ ls  
android  lichee
```

4. 编译工具链的安装

编译工具链已经集成在 Android SDK 中

工具链位于 Android SDK 中的 lichee/brandy/gcc-linaro/(已存在)

```
brandy buildroot build.sh linux-3.4 README tools  
root@curry:/home/curry/OrangePi/android/lichee# cd brandy/gcc-linaro/bin/  
root@curry:/home/curry/OrangePi/android/lichee/brandy/gcc-linaro/bin# ls  
arm-linux-gnueabi-addr2line  arm-linux-gnueabi-gprof  
arm-linux-gnueabi-ar         arm-linux-gnueabi-ld  
arm-linux-gnueabi-as         arm-linux-gnueabi-ld.bfd  
arm-linux-gnueabi-c++        arm-linux-gnueabi-ldd  
arm-linux-gnueabi-c++filt    arm-linux-gnueabi-ld.gold  
arm-linux-gnueabi-cpp        arm-linux-gnueabi-nm
```

5. Lichee 源码的编译

解压之后的压缩包有 android 和 lichee, 进入 lichee 目录下

```
$ cd lichee
```

```
$ ./build.sh lunch
```

选择 sun8iw7p1

编译成功的打印信息



```

sun8iw7p1 compile Kernel successful

INFO: build kernel OK.
INFO: build rootfs ...
INFO: skip make rootfs for android
INFO: build rootfs OK.
-----
build sun8iw7p1 android dolphin lichee OK

```

6. Android 源码编译命令

命令行输入

```
$ cd android
```

```
$ source ./build/envsetup.sh
```

```

root@curry:/home/curry/OrangePi/android/android# source ./build/envsetup.sh
including device/generic/armv7-a-neon/vendorsetup.sh
including device/generic/x86/vendorsetup.sh
including device/generic/mips/vendorsetup.sh
including device/asus/tilapia/vendorsetup.sh
including device/asus/grouper/vendorsetup.sh
including device/asus/deb/vendorsetup.sh
including device/asus/flo/vendorsetup.sh

```

```
$ lunch dolphin_fvd_p1-eng      #选择方案号
```

```

root@curry:/home/curry/OrangePi/android/android# source ./build/envsetup.sh
including device/generic/armv7-a-neon/vendorsetup.sh
including device/generic/x86/vendorsetup.sh
including device/generic/mips/vendorsetup.sh
including device/asus/tilapia/vendorsetup.sh
including device/asus/grouper/vendorsetup.sh
including device/asus/deb/vendorsetup.sh
including device/asus/flo/vendorsetup.sh

```

```
$ extract-bsp      #拷贝内核及驱动模块
```

```

curry@curry:$ extract-bsp
/expansion/xspace/OrangePi/Android/H5/zeroplus/android/device/*/cheetah-p1/binImage copied!
/expansion/xspace/OrangePi/Android/H5/zeroplus/android/device/*/cheetah-p1/modules copied!
curry@curry:$ make
=====
PLATFORM_VERSION_CODENAME=REL
PLATFORM_VERSION=5.1
TARGET_PRODUCT=cheetah_fvd_p1
TARGET_BUILD_VARIANT=eng
TARGET_BUILD_TYPE=release

```

```
$ make      #后面的数值为同时编译的进程，依赖于主机的配置
```




```

Creating filesystem with parameters:
  Size: 805306368
  Block size: 4096
  Blocks per group: 32768
  Inodes per group: 8192
  Inode size: 256
  Journal blocks: 3072
  Label:
  Blocks: 196608
  Block groups: 6
  Reserved block group size: 47
Created filesystem with 1393/49152 inodes and 79017/196608 blocks
+ '[' 0 -ne 0 ']'
Install system fs image: out/target/product/dolphin-fvd-p1/system.img
out/target/product/dolphin-fvd-p1/system.img+out/target/product/dolphin-fvd-p1/obj/PACKAGING
/recovery_patch_intermediates/recovery_from_boot.p maxsize=822163584 blocksize=4224 total=31
3479604 reserve=8308608

```

\$ pack #打包生成固件

\$ cd */lichee/tools/pack/

```

Dragon execute image.cfg SUCCESS !
-----image is at-----
/home/curry/OrangePi/android/lichee/tools/pack/sun8iw7p1_android_dolphin-p1_uart0.img
pack finish

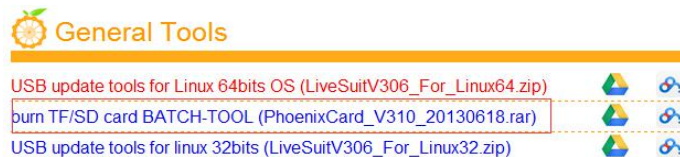
root@curry:/home/curry# cd /home/curry/OrangePi/android/lichee/tools/pack/
root@curry:/home/curry/OrangePi/android/lichee/tools/pack# ls
chips common createkeys out pack parser.sh ptools sun8iw7p1_android_dolphin-p1_uart0.img
root@curry:/home/curry/OrangePi/android/lichee/tools/pack#

```

烧录镜像:

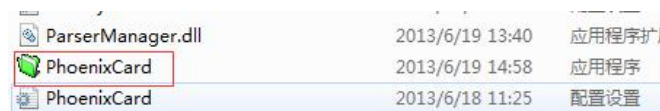
将生成的镜像文件拷贝到 SD 卡上, 切换到 windows 操作系统, 下载烧录软件

下载地址: <http://www.orangepi.org/downloadresources/>

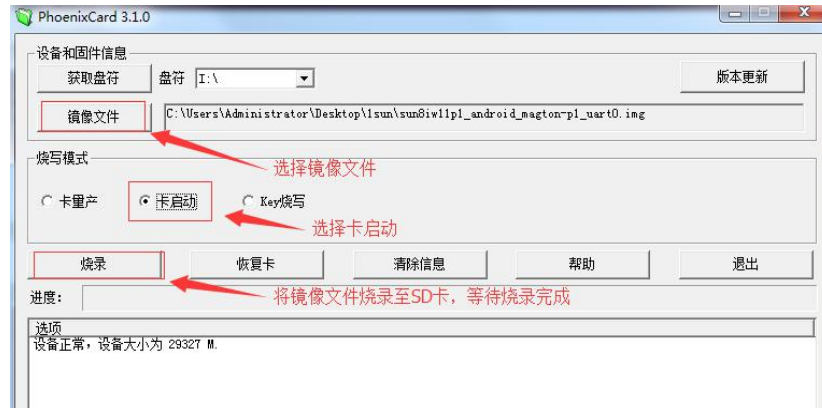


下载得到压缩包, 解压得到文件夹

进入文件夹, 以管理员身份打开运行该程序



在 windows 下用此工具烧录 android 镜像文件



将烧录好的 TF 卡插入 orangepi，开机即可进入安卓系统



五、使用工程配置化文件

1. sys_config.fex 简介

配置硬件: sys_config.fex

sys_config.fex 是被全志 SOC 内核驱动或 LiveSuit 使用的针对特定目标板的二进制配置文件, 包含如何设置基于目标版的各种外设, 端口, I/O 针脚信息。

对于 OrangePi , 其工程化配置文档的位置是:

lichee/tools/pack/chips/sun8iw7p1/configs/dolphin-p2/sys_config.fex

将文档拷贝到 /lichee 目录下, 使用命令

```
$ cd ./lichee
```

```
$ cp ./tools/pack/chips/sun8iw7p1/configs/dolphin-p2/sys_config.fex ./
```

2. 例程

1) 修改输出的模式为 tv

- tv-out 输出, 屏 0 的输出类型无效, 要设置屏 1 的输出类型. 输出模式 pal. 启动显示输出的默认配置改成 tv

```
[tv0]
```

```
used = 1
```

```
tv_dac_used = 1
```

```
dac_src0 = 0
```

```
dac_type0= 0
```

```
interface= 1
```

```
[tvout_para]
```

```
tvout_used= 1
```

```
tvout_channel_num= 1
```

```
[disp]
```

```
disp_init_enable= 1
```

```
disp_mode= 1
```

```
screen0_output_type= 2
```



```

screen0_output_mode= 11
screen1_output_type= 2
screen1_output_mode= 11
dev0_output_type = 4
dev0_output_mode = 4
dev0_screen_id = 0
dev0_do_hpd = 1
dev1_output_type = 2
dev1_output_mode = 11

```

修改 sys_config, 重新生成 script.bin 并替换, 使用官网 github 上的编译系统来进行更新比较快速, 方法参照 Linux 编译这一章节

2) 开机自动加载 tv.ko 模块

进入/lib/目录, 输入命令

```
depmod -a
```

在/etc/modules 加上一行

```
tv
```

开机即可 tv 输出

● 电容屏(capacitor tp)

| 配置项 | 配置项含义 |
|----------------------|--|
| ctp_used=xx | 该选项为是否开启电容触摸, 支持的话置 1, 反之置 0 |
| ctp_name =xx | 用于指明方案采用的触控方案, 目前可选: "ft5x_ts" 或 "Goodix-TS" |
| ctp_twi_id=xx | 用于选择 i2c adapter, 可选 0, 2 |
| ctp_twi_addr =xx | 指明 i2c 设备地址, 与具体硬件相关 |
| ctp_screen_max_x=xx | 触摸板的 x 轴最大坐标 |
| ctp_screen_max_y=xx | 触摸板的 y 轴最大坐标 |
| ctp_revert_x_flag=xx | 是否需要翻转 x 坐标, 需要则置 1, 反之置 0 |
| ctp_revert_y_flag=xx | 是否需要翻转 y 坐标, 需要则置 1, 反之置 0 |
| ctp_int_port=xx | 电容屏中断信号的 GPIO 配置 |
| ctp_wakeup=xx | 电容屏唤醒信号的 GPIO 配置 |
| ctp_io_port=xx | 电容屏 io 信号, 目前与中断信号公用管脚 |

配置举例:

```
ctp_used = 1
```



```
ctp_name          = "ft5x_ts"
ctp_twi_id        = 2
ctp_twi_addr      = 0x70
ctp_screen_max_x  = 800
ctp_screen_max_y  = 480
ctp_revert_x_flag = 0
ctp_revert_y_flag = 0
ctp_int_port      = port:PH21<6><default>
ctp_wakeup        = port:PB13<1><default><default><1>
ctp_io_port       = port:PH21<0><default>
```

注意事项:

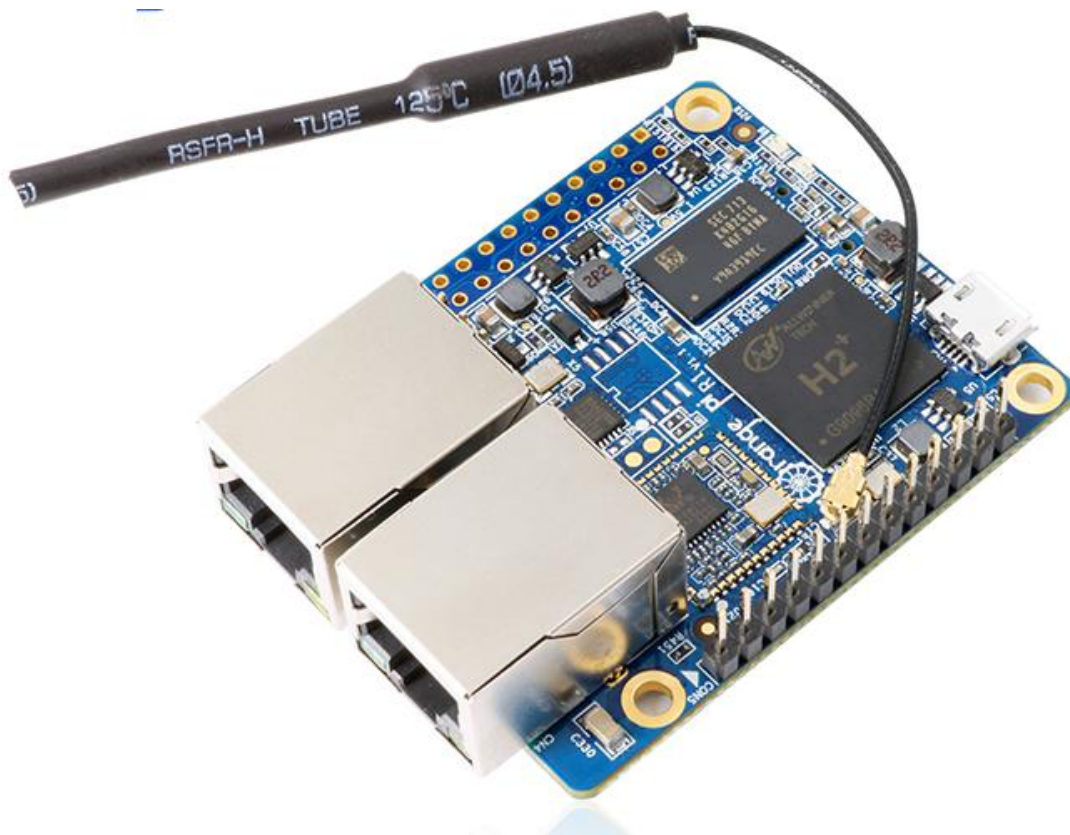
若要支持新的电容触控 ic, 在原有电容触控 ic 的代码基础上, 须结合 A10 bsp 层的配置情况, 作相应修改。具体说来, 1) 在 sys_config 中: ctp_twi_id 应与硬件连接一致; 2) 在驱动部分代码中: 使用的 twi 从设备名字+地址, 应与 sys_config 中的 ctp_name, ctp_twi_addr 配置一致。同时, sysconfig 中的其他子键也要正确配置, 在程序中, 要对这些配置进行相应的处理



六、OrangePi 驱动程序开发

为帮助开发者更加熟悉 OrangePi，本手册主要描述如何在开发板上使用简单设备驱动模块和应用程序。

需要硬件： 下图 Orange Pi 开发板一块，读卡器一个，TF 卡一张和电源适配器一个





1. 设备驱动和应用程序的编写

1) 应用程序(app.c):

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <string.h>

int main(int argc, char *argv[])
{
    int cnt, fd;
    char buf[32] = {0};
    if(argc != 2)
    {
        printf("Usage : %s </dev/xxx>\r\n", argv[0]);
        return -1;
    }

    fd = open(argv[1], O_RDWR);
    if(fd < 0)
    {
        printf("APP Error : open device is Failed!\r\n");
        return -1;
    }
    read(fd, buf, sizeof(buf));
    printf("buf = %s\r\n", buf);
    close(fd);
    return 0;
}
```

2) 驱动程序(OrangePi_misc.c):



```
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/fs.h>
#include <linux/miscdevice.h>
#include <linux/init.h>
#include <asm-generic/uaccess.h>

static int orangepi_open(struct inode *inodp, struct file *filp)
{
    return 0;
}

static ssize_t orangepi_read(struct file *filp, char __user *buf, size_t
count, loff_t *offset)
{
    char str[] = "Hello World";
    copy_to_user(buf, str, count);
    return 0;
}

static struct file_operations tOrangePiFops = {
    .owner = THIS_MODULE,
    .open = orangepi_open,
    .read = orangepi_read,
};

static struct miscdevice OrangePi_Misc = {
    .minor = 255,
    .name = "orangepimisc",
    .fops = &tOrangePiFops,
};
```



```
static int __init OrangePi_misc_init(void)
{
    int ret;
    printk("func : %s, line : %d\r\n", __func__, __LINE__);

    ret = misc_register(&OrangePi_Misc);
    if(ret < 0){
        printk("Driver Error : misc_register is Failed!\r\n");
        return -1;
    }
    return 0;
}

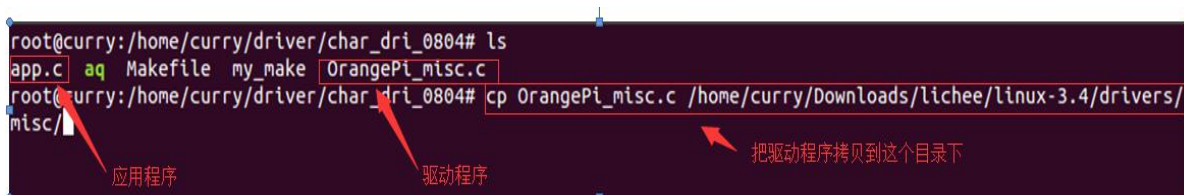
static void __exit OrangePi_misc_exit(void)
{
    int ret;
    printk("func : %s, line : %d\r\n", __func__, __LINE__);
    ret = misc_deregister(&OrangePi_Misc);
    if(ret < 0){

        printk("Driver Error : misc_register is Failed\r\n");
    }
}

module_init(OrangePi_misc_init);
module_exit(OrangePi_misc_exit);
```

2. 设备驱动的编译方法

OrangePi_misc.c 拷贝到源码目录下*/lichee/linux-3.4/driver/misc



进入*/lichee/linux-3.4/drivers/misc/, 并修改makefile
修改当前文件的Makefile(如下图所示)



```

43 obj-$(CONFIG_SPEAR13XX_PCIE_GADGET) += spear13xx_pcie_gadget.o
44 obj-$(CONFIG_VMWARE_BALLOON) += vmw_balloon.o
45 obj-$(CONFIG_ARM_CHARLCD) += arm-charlcd.o
46 obj-$(CONFIG_PCH_PHUB) += pch_phub.o
47 obj-y += ti-st/
48 obj-$(CONFIG_AB8500_PWM) += ab8500-pwm.o
49 obj-y += lis3lv02d/
50 obj-y += carma/
51 obj-$(CONFIG_USB_SWITCH_FSA9480) += fsa9480.o
52 obj-$(CONFIG_ALTERA_STAPL) += altera-stapl/
53 obj-$(CONFIG_MAX8997_MUIC) += max8997-muic.o
54 obj-$(CONFIG_WL127X_RFKILL) += wl127x-rfkill.o
55 obj-$(CONFIG_SENSORS_AK8975) += ak8975.o
56 obj-$(CONFIG_SUNXI_VIBRATOR) += sunxi-vibrator.o
57 obj-$(CONFIG_SUNXI_BROM_READ) += sunxi_brom_read.o
58 obj-$(CONFIG_NET) += rf_pm/
59 obj-$(CONFIG_ORANGEPI_MISC) += OrangePi_misc.o

```

重新修改Makefile

和Makefile同级的文件夹下有Kconfig, 每个Kconfig分别描述了所属目录源文件相关的内核配置菜单, 在内核配置make menuconfig时候, 从Kconfig中读取配置菜单, 用户配置后保存到.config中。在内核编译时, 主Makefile调用这个.config, 就知道用户对内核的配置情况。

所以Kconfig就是对应着内核的配置菜单。加入要添加新的驱动到内核源码中, 可以通过修改Kconfig来增加对我们驱动的配置菜单, 这样就可以在menuconfig里面选择我们驱动是否被编译。

```

config SUNXI_BROM_READ
    tristate "Read the BROM infonation"
    depends on ARCH_SUN8I
    default n
    ---help---
    This option can allow program access brom space by the file node.

config ORANGEPI_MISC
    tristate
    default n

```

修改Kconfig

回到源码目录下

```
root@curry:/home/curry/Downloads/lichee# cd /home/curry/Downloads/lichee/
```

回到源码目录下

\$./build.sh

编译内核, 之后会在 lichee/linux-3.4/output/lib/modules/3.4.39

下面产生一个 orangepi_misc.ko 文件:

```

Reserved block group size: 7
Created filesystem with 3654/7168 inodes and 23042/28672 blocks
e2fsck 1.42.9 (4-Feb-2014)
success in generating rootfs
Build at: 2016年 08月 04日 星期四 15:53:34 CST
INFO: build rootfs OK.
-----
build sun8iw7p1 dragonboard lichee OK
-----

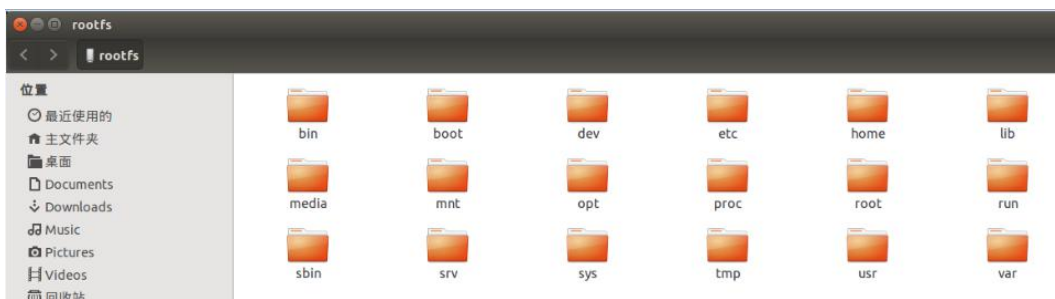
```

编译结束



能发现在*/lichee/linux-3.4/output/lib/modules/3.4.39/产生一个.ko文件，这就是刚刚的OrangePi_misc.c经过编译之后产生的模块。

插上 U 盘（注意此时 SD 卡已经烧好镜像）如果此时 SD 卡挂载到系统的 /dev/sdb 目录下，SD 将会有两个子挂载点，分别为 /dev/sdb1 和 /dev/sdb2。SD 卡的两个分区会自动挂载到 PC 上的 /media/ 目录下，第一个分区是 boot 分区，第二个分区为 rootfs 分区。



把OrangePi_misc.ko文件复制到/media/*/lib/modules/3.4.39 里

```
$ cp OrangePi_misc.ko /media/*/lib/modules/3.4.39
```

3. 交叉编译器编译应用程序

以arm-linux-gnueabi-hf-gcc为例子讲述如何安装变差编译器。首先查询是否有下面这个交叉编译器，没有下载安装

```
$ arm-linux-gnueabi-hf-gcc -v
```

```
root@curry:/home/curry/lichee# arm-linux-gnueabi-hf-gcc -v
Using built-in specs.
COLLECT_GCC=arm-linux-gnueabi-hf-gcc
COLLECT_LTO_WRAPPER=/usr/lib/gcc-cross/arm-linux-gnueabi-hf/4.8/lto-wrapper
Target: arm-linux-gnueabi-hf
Configured with: ../src/configure -v --with-pkgversion='Ubuntu/Linaro 4.8.4-2ubuntu1-14.04.1'
--with-arch=armv7-a --with-fpu=vfpv3-d16 --with-float=hard --with-mode=thumb --disable-w
checking=release --build=x86_64-linux-gnu --host=x86_64-linux-gnu --target=arm-linux-gnu
m-prefix=arm-linux-gnueabi-hf --includedir=/usr/arm-linux-gnueabi-hf/include
Thread model: posix
gcc version 4.8.4 (Ubuntu/Linaro 4.8.4-2ubuntu1-14.04.1)
```

编译应用程序，发现所需要的交叉编译器是arm-linux-gnueabi-hf-gcc，网上下载并安装



```
curry@curry:~/tools/1_arm-linux-gnueabihf-gcc$ ls
gcc-linaro-arm-linux-gnueabihf-4.9-2014.07_linux.tar.xz
curry@curry:~/tools/1_arm-linux-gnueabihf-gcc$
```

下载好的压缩包

解压并进入解压之后的目录

```
curry@curry:~/tools/1_arm-linux-gnueabihf-gcc$ tar -xvf gcc-linaro-arm-linux-gnueabihf-4.9-2014.07_linux.tar.xz
gcc-linaro-arm-linux-gnueabihf-4.9-2014.07_linux$ ls
curry@curry:~/tools/1_arm-linux-gnueabihf-gcc$ cd gcc-linaro-arm-linux-gnueabihf-4.9-2014.07_linux/
curry@curry:~/tools/1_arm-linux-gnueabihf-gcc/gcc-linaro-arm-linux-gnueabihf-4.9-2014.07_linux$ ls
arm-linux-gnueabihf bin lib libexec share
curry@curry:~/tools/1_arm-linux-gnueabihf-gcc/gcc-linaro-arm-linux-gnueabihf-4.9-2014.07_linux$
```

解压压缩包

进入当前目录下查看文件

进入 bin 目录下，查看内容

```
curry@curry:~/tools/1_arm-linux-gnueabihf-gcc/gcc-linaro-arm-linux-gnueabihf-4.9-2014.07_linux$ ls
arm-linux-gnueabihf bin lib libexec share
curry@curry:~/tools/1_arm-linux-gnueabihf-gcc/gcc-linaro-arm-linux-gnueabihf-4.9-2014.07_linux$ cd bin/
curry@curry:~/tools/1_arm-linux-gnueabihf-gcc/gcc-linaro-arm-linux-gnueabihf-4.9-2014.07_linux/bin$ ls
arm-linux-gnueabihf-addrtline arm-linux-gnueabihf-dwp arm-linux-gnueabihf-gcc-ranlib arm-linux-gnueabihf-ldd arm-linux-gnueabihf-ranlib
arm-linux-gnueabihf-ar arm-linux-gnueabihf-elfedit arm-linux-gnueabihf-gcov arm-linux-gnueabihf-gdb arm-linux-gnueabihf-readelf
arm-linux-gnueabihf-as arm-linux-gnueabihf-g++ arm-linux-gnueabihf-gfortran arm-linux-gnueabihf-gprof arm-linux-gnueabihf-objcopy arm-linux-gnueabihf-size
arm-linux-gnueabihf-c++ arm-linux-gnueabihf-gcc-4.9.1 arm-linux-gnueabihf-gprof arm-linux-gnueabihf-objdump arm-linux-gnueabihf-strings
arm-linux-gnueabihf-c++ arm-linux-gnueabihf-gcc-4.9.1 arm-linux-gnueabihf-gprof arm-linux-gnueabihf-objdump arm-linux-gnueabihf-strip
arm-linux-gnueabihf-ct-ng.config arm-linux-gnueabihf-gcc-nm arm-linux-gnueabihf-ld.bfd arm-linux-gnueabihf-pkg-config arm-linux-gnueabihf-pkg-config-real
curry@curry:~/tools/1_arm-linux-gnueabihf-gcc/gcc-linaro-arm-linux-gnueabihf-4.9-2014.07_linux/bin$
```

发现所需要的编译工具

pwd显示该路径，并将这个路径倒到全局

```
curry@curry:~/tools/1_arm-linux-gnueabihf-gcc/gcc-linaro-arm-linux-gnueabihf-4.9-2014.07_linux/bin$ pwd
/home/curry/tools/1_arm-linux-gnueabihf-gcc/gcc-linaro-arm-linux-gnueabihf-4.9-2014.07_linux/bin
curry@curry:~/tools/1_arm-linux-gnueabihf-gcc/gcc-linaro-arm-linux-gnueabihf-4.9-2014.07_linux/bin$ vim /etc/environment
```

显示路径

环境变量

\$ ll /etc/environment 发现该文件只能读，需要

\$ chmod 777 /etc/environment

修改权限

```
root@curry:/home/curry/tools/1_arm-linux-gnueabihf-gcc/gcc-linaro-arm-linux-gnueabihf-4.9-2014.07_linux/bin# ll /etc/environment
-rw-r--r-- 1 root root 151 8月 4 15:24 /etc/environment
root@curry:/home/curry/tools/1_arm-linux-gnueabihf-gcc/gcc-linaro-arm-linux-gnueabihf-4.9-2014.07_linux/bin# chmod 777 /etc/environment
root@curry:/home/curry/tools/1_arm-linux-gnueabihf-gcc/gcc-linaro-arm-linux-gnueabihf-4.9-2014.07_linux/bin# ll /etc/environment
-rwxrwxrwx 1 root root 151 8月 4 15:24 /etc/environment*
root@curry:/home/curry/tools/1_arm-linux-gnueabihf-gcc/gcc-linaro-arm-linux-gnueabihf-4.9-2014.07_linux/bin#
```

只能读 要修改权限

修改权限

修改权限之后

把路径加入全局环境变量中



```
PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/home/curry/tools/opt/FriendlyARM/toolchain/4.5.1/bin:/home/curry/tools/1_arm-linux-gnueabihf-gcc/occ-tn
aro-arm-linux-gnueabihf-4.9-2014.07/linux/bin"
修改之后把路径加进去
```

有了交叉编译器，编译应用程序

```
$ arm-linux-gnueabihf-gcc app.c -o aq
```

之后在目录下产生一个aq的应用程序，将应用程序aq复制到开发板文件系统 (rootfs的/home/orangepi/下)

```
$ cp aq /media/*/home/orangepi/
```

4. 驱动和程序的运行方式

将卡取下，插入开发板，上电开机。

首先要切换到root用户，开发板下加载模块驱动模块

```
$ insmod /lib/modules/Orangepi_misc.ko
```

```
$ lsmod      看下是否加载上
```

```
root@orangepi:/# lsmod
Module              Size  Used by
8189fs              935152  0
Orangepi_misc      1315  0
查看加载的模块
刚刚加载上去的字符设备驱动
```

\$ ll /dev/orangepimisc (杂项设备自动生成设备文件，具体看驱动代码)

```
root@orangepi:/home/orangepi# ll /dev/orangepimisc
crw----- 1 root root 10, 41 Jan  1 1970 /dev/orangepimisc
查看字符设备详细信息
```

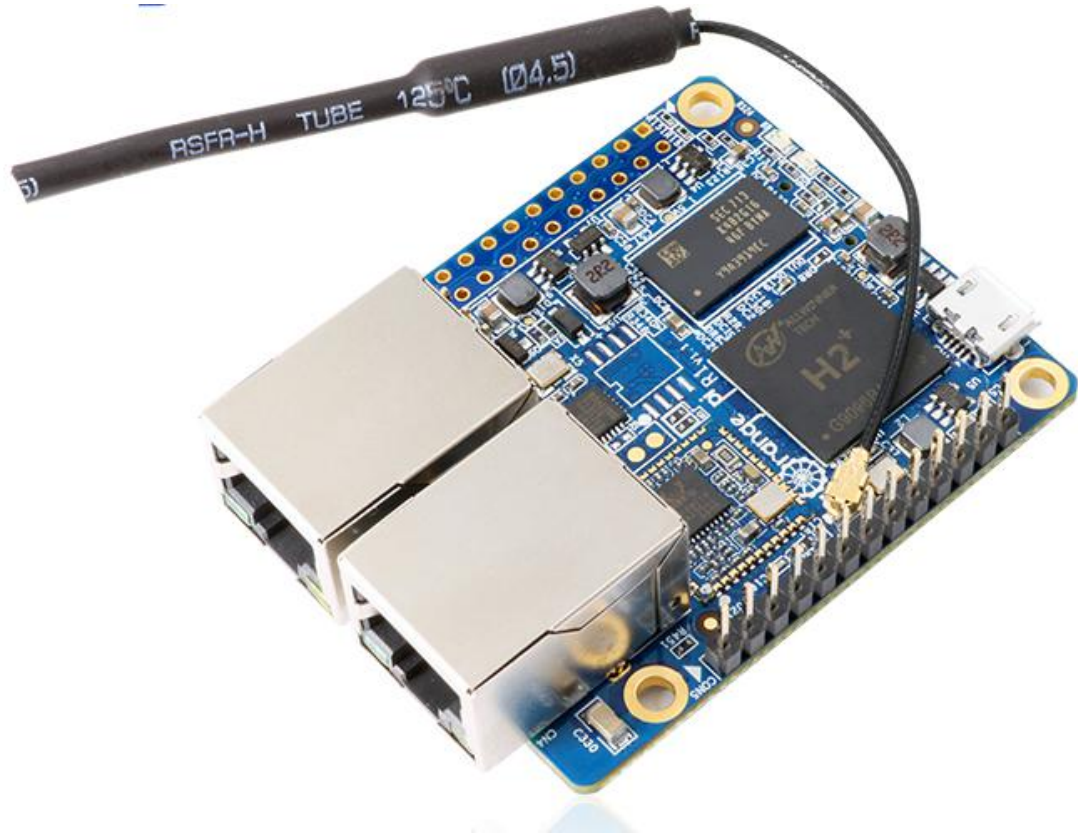
执行应用程序 (注意应用程序的用法，具体看代码)

```
$ ./aq /dev/orangepimisc
```




七、串口调试工具介绍

需要硬件：OrangePi R1 开发板，读卡器和一张 TF 卡



TTL 转 USB 线





1. 基于 Windows 平台的使用

在使用 OrangePi 做项目开发过程中, 为了获得更多的调试信息, OrangePi 默认支持串口信息调试。对于开发者而言, 只需准备上面提到的材料, 即可简单的获得串口调试信息。不同的上位机使用的串口调试工具大同小异, 基本可以参考下文的方法进行部署。使用 Windows 平台进行串口调试的工具很多, 通常使用的工具是 putty。本节以 putty 作为例子进行部署讲解。

1) Windows 下 USB 驱动安装

- 目前最新版的驱动 PL2303_Prolific_DriverInstaller_v130, 下载解压。

| | | | | |
|--------------------------------------|-----------------|------------------|----------|-------------|
| PL2303_Prolific_DriverInstaller_v130 | 2010/7/15 10:41 | 应用程序 | 3,099 KB | ← 解压之后的应用程序 |
| PL2303_Prolific_DriverInstaller_v130 | 2016/8/3 9:20 | WinRAR ZIP 压缩... | 2,316 KB | ← 下载的压缩包 |
| releasenote | 2010/7/22 10:14 | 文本文档 | 2 KB | |

- 以管理员身份选择应用程序安装



- 等待安装完成



2) Windows 下 Putty 安装

- 下载 putty 安装包

| | | | |
|-----------------|----------------|-------------|------------|
| putty | 2016/1/21 9:56 | 文件夹 | ← 解压之后的文件夹 |
| puTTY.xp510.com | 2016/8/3 9:29 | WinRAR 压缩文件 | ← putty压缩包 |

- 解压安装



| | | | |
|--------------|-----------------|-------------------|--------|
| 636网址导航 | 2015/5/4 14:21 | Internet 快捷方式 | 1 KB |
| putty中文版1.0v | 2016/1/20 17:13 | 应用程序 | 604 KB |
| XP510下载须知 | 2015/5/4 14:21 | 文本文档 | 2 KB |
| 软件使用说明 | 2015/5/13 9:23 | 360 se HTML Do... | 1 KB |

点击安装

- 安装好之后打开程序如下图所示



3) 调试的连接方式

使用 TTL 转串口线，一端连接 OrangePi，另一端连接 PC

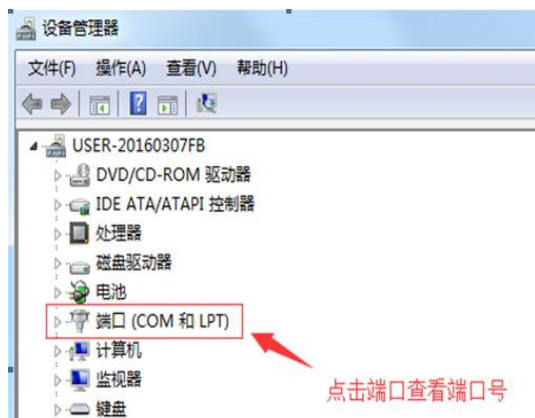
4) 设备信息的获取

- 开始菜单选择控制面板



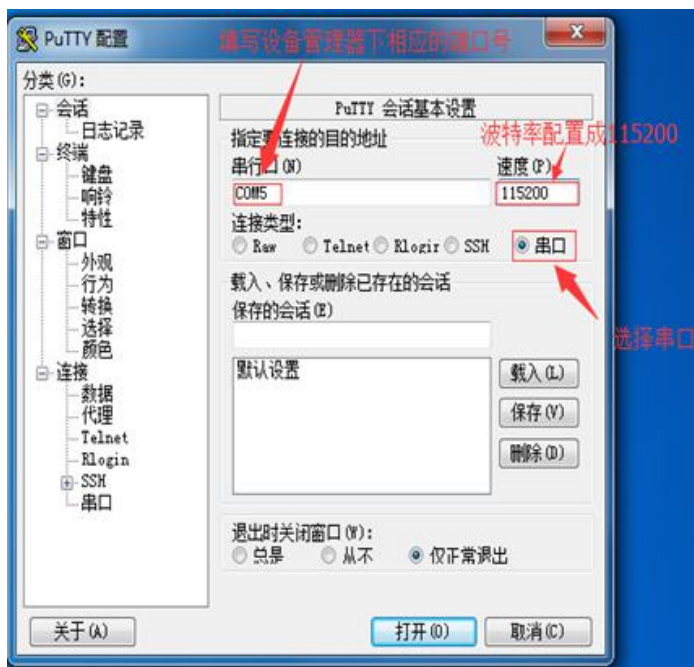


- 点击设备管理器，查看端口号





5) Putty 配置



串行口设置成相应的端口号 (COM5)，速度设置成 115200

6) 调试串口

OrangePi 上电开机，串口自动打印串口 log

```
COM5 - PuTTY
[mmc]: *****SD/MMC 0 init OK!!*****
[mmc]: erase_grp_size      : 0x1WrBlk*0x200=0x200 Byte
[mmc]: secure_feature      : 0x0
[mmc]: secure_removal_type : 0x0
[  1.606]sunxi flash init ok
[  1.627]start
drv_disp_init
init_clocks: finish init_clocks.
enable power vcc-hdmi-33, ret=0
drv_disp_init finish
reading disp_rsl.fex
FAT: Misaligned buffer address (76e93030)
8 bytes read in 7 ms (1000 Bytes/s)
display resolution 4, type 4
display output attr: type 4, used 1, channel 0, mode 4
reading disp_rsl.fex
FAT: Misaligned buffer address (76e93030)
8 bytes read in 6 ms (1000 Bytes/s)
could not get output resolution for 'cvbs_channel'
display output attr: type 2, used 1, channel 1, mode 11
boot_disp.auto_hpd=1
boot_disp.hdmi_mode_check=1
boot_disp.output_type=3
```

2. 基于 Linux 平台的使用

使用Linux平台进行串口调试工具有minicom和kermit。本文以kermit作为例子

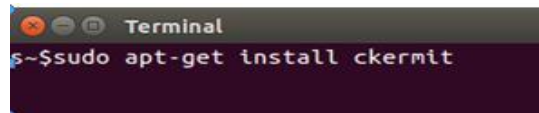


进行讲解。

1) Kermit 安装

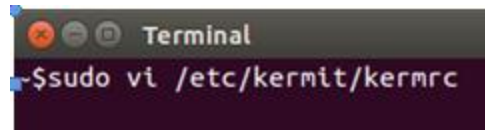
- 使用命令进行安装:

```
$ sudo apt-get install ckermit
```



- 配置 kermit

```
$ sudo vi /etc/kermit/kermitrc
```



- 添加行:

```
set line          /dev/ttyUSB1
set speed         115200
set carrier-watch off
set handshake none
set flow-control  none
robust
set file type     bin
set file name     lit
set rec pack      1000
set send pack     1000
set window        5
```



```

root@orange-All-Series:/home/orange
; This is /etc/kernit/kernrc
; It is executed on startup if ~/.kernrc is not found.
; See "nan kernit" and http://www.kernit-project.org/ for details on
; configuring this file, and /etc/kernit/kernrc.full
; for an example of a complex configuration file

; If you want to run additional user-specific customisations in
; addition to this file, place them in ~/.mykernrc

; Execute user's personal customization file (named in environment var
; CKERMODO or ~/.mykernrc)
;

if def ${CKERMODO} assign _myinit ${CKERMODO}
if not def _myinit assign _myinit ${v(home).mykernrc}

xif exist ${_myinit} {                                : If it exists,
  echo Executing "${_myinit}"...                       : print message,
  take ${_myinit}                                       : and TAKE the file.
}

set line /dev/ttyUSB1
set speed 115200
set carrier-watch off
set handshake none
set flow-control none
set robust
set file type bin
set file name lit
set rec pack 1000
set send pack 1000
set window 5
c

```

添加这些行

2) 调试的连接方式

使用 TTL 转串口线，一端连接 OrangePi，另一端连接 PC

3) 设备信息的获取

\$ ls /dev/ (在 PC 终端输入命令，查询 TTL 转串口 线的设备号)

```
root@orange-All-Series:/home/orange# ls /dev
autofs          t2c-4          psaux          sda7           tty21           tty47           ttys513        uhid
block           t2c-5          psbmx          sda8           tty22           tty48           ttys514        uinput
bsg             input          pts            sda9           tty23           tty49           ttys515        urandom
btfs-control    kmsg          ram0           serial         tty24           tty5            ttys516        v4l
bus             log            ram1           sg0            tty25           tty58           ttys517        vboxusb
cdrom           loop0          ram10          sg1            tty26           tty51           ttys518        vcs
char            loop1          ram11          shn            tty27           tty52           ttys519        vcs1
console         loop2          ram12          snapshot       tty28           tty53           ttys52         vcs2
core            loop3          ram13          sr0            tty29           tty54           ttys520        vcs3
cpu             loop4          ram14          snd            tty3            tty55           ttys521        vcs4
cpu_dma_latency ram15          ram15          snderr         tty30           tty56           ttys522        vcs5
fuse            loop6          ram2           stdin          tty31           tty57           ttys523        vcs6
disk            loop7          ram3           stdio          tty32           tty58           ttys524        vcsa
dri             loop-control   ram4           tty            tty33           tty59           ttys525        vcsa1
ecryptfs        lp0            ram5           tty0           tty34           tty6            ttys526        vcsa2
fb0             mapper        ram6           tty1           tty35           tty60           ttys527        vcsa3
fd              mcelog        ram7           tty10          tty36           tty61           ttys528        vcsa4
full            net0           ram8           tty11          tty37           tty62           ttys529        vcsa5
fuse            nen            ram9           tty12          tty38           tty63           ttys53         vcsa6
hidraw0         network_bandwidth random          tty13          tty7           tty30           tty530        vrflo
hidraw1         ndctl0        ram10          rtc            tty14           tty8            tty531        vga_arbiter
hidraw2         net            rtk            tty15          tty40           tty9            tty54         vhci
hpet            network_latency rtc0            tty16          tty41           ttyprntk        tty55         vhost-net
hwrng           sda            tty17          tty42           tty58           tty56           tty58         video0
t2c-0           null          sda1           tty18          tty43           tty51           tty57         zero
t2c-1           parport0      sda2           tty19          tty44           tty510          tty58
t2c-2           port          sda5           tty2           tty45           tty511          ttys9
t2c-3           ppp           sda6           tty20          tty46           tty512          ttys80
```

- 从图中可以看出，“TTL 转串口”线被识别为“ttyUSB0”，配置 `/etc/kermit/kermitc` 文件，更新串口信息。
`$ sudo vi /etc/kermit/kermitc`
- 将 `setline` 的值设置为 `/dev/ttyUSB0`



```
kernrc (/etc/kernrc) - VIM
: CKERMOD or ~/.mykernrc

if def \$(CKERMOD) assign _myinit \$(CKERMOD)
if not def _myinit assign _myinit \v(home).mykernrc

xif exist \n(_myinit) {
    echo Executing \n(_myinit)...
    take \n(_myinit)
}

set line /dev/ttyUSB0
set speed 115200
set carrier-watch off
set handshake none
set flow-control none
robust
set file type bin
set file name ltt
set rec pack 1000
set send pack 1000
set window 5
c
```

4) 开始调试串口

- 在上位机终端输入命令，进入 kermit 模式：

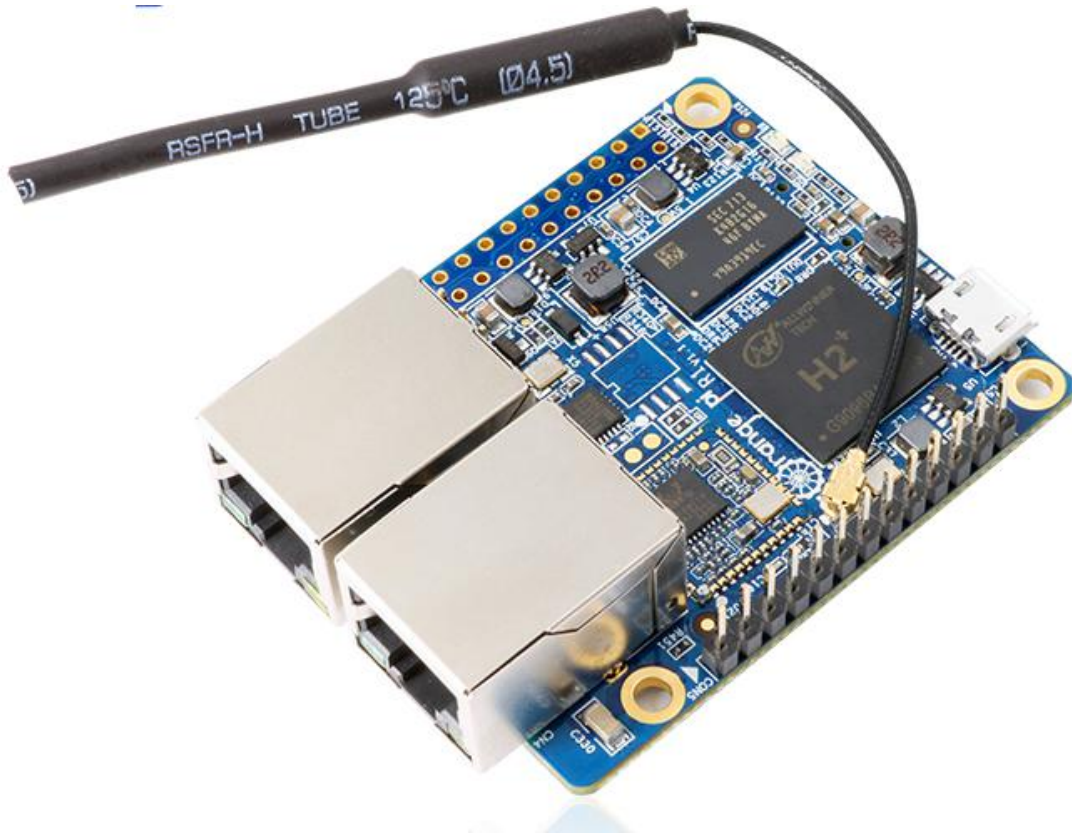
```
$ sudo kermit -c
```

```
root@orange-All-Series: /home/orange
root@orange-All-Series: /home/orange# kermit -c
Connecting to /dev/ttyUSB0, speed 115200
Escape character: Ctrl-\ (ASCII 28, FS): enabled
Type the escape character followed by C to get back,
or followed by ? to see other options.
.....
```

- OrangePi 上电开机，串口自动打印串口 log



八、OrangePi R1 路由系统适配



1. Quagga 的配置、编译和安装

1) 配置 Quagga 时可能遇到的问题:

问题:

编译过程中, 遇到以下错误该如何修复?

"libtoolize: No such file or directory" error?



```
Can't exec "libtoolize": No such file or directory at
/usr/share/autoconf/Autom4te/FileUtils.pm line 345, line 5.
autoreconf: failed to run libtoolize: No such file or directory
autoreconf: libtoolize is needed because this package uses Libtool
```

答案:

这个错误表示你还没有在系统安装 libtool. 修复这种错误, 你需要先根据如下方法安装 libtool:

libtool 是一个库的工具, 旨在简化建设复杂的静态/软件过程共享库的依赖性通过便携式接口。

在 Debian, Ubuntu 或 Linux Mint:

```
$ sudo apt-get install libtool
```

2) OrangePiR1 的 Ubuntu15.04 系统首先需要安装如下软件:

a. 先将 sources.list_ubuntu15.04_ports_vivid(见最后) 复制到 /etc/apt/ 下重命名为 sources.list

b. 然后安装如下软件

```
$ sudo apt-get update
$ sudo apt-get install -y automake
$ sudo apt-get install -y libtool
$ sudo apt-get install -y gawk
$ sudo apt-get install -y texinfo
$ sudo apt-get install -y telnet
```

3) Quagga 的配置、编译、安装过程如下 (在源码的根目录下执行):

```
$ autoreconf -vif
$ ./configure --enable-user=root --enable-group=root --enable-vty-group=root
$ make
$ make install
```

2. 搭建测试环境 一个小型的局域网

1) 该局域网由三台机器组成, 其中机器 B 至少有两个网卡, 所有机器对应的 IP 地址如下所示:

机器 A[eth0: 192.168.1.10] <—> [eth0: 192.168.1.12] 机器 B[eth1 : 192.168.2.12] <—> [eth0: 192.168.2.10] 机器 C

2) 为每台机器配置对应的静态 IP 地址



Ubuntu 系统对应的配置文件为: `/etc/network/interfaces`

机器 A

```
auto eth0
iface eth0 inet static
address 192.168.1.10
gateway 192.168.1.12 #确认网关是否正确
netmask 255.255.255.0
```

机器 B

```
auto eth0

iface eth0 inet static
address 192.168.1.12
gateway 192.168.1.10 #确认网关是否正确
netmask 255.255.255.0
```

```
auto eth0
iface eth0 inet static
address 192.168.2.12
gateway 192.168.2.10 #确认网关是否正确
netmsk 255.255.255.0
```

机器 C

```
auto eth0
iface eth0 inet static
address 192.168.2.10
gateway 192.168.2.12 #确认网关是否正确
netmask 255.255.255.0
```

配置完后, 重启机器, 查看对应接口的 IP 地址是否自动配置且正确, 然后通过 ping 命令测试所有的链接是否是连通的

3) 在所有参与路由转发的机器上开启 IP 转发功能



i linux 发行版默认情况下是不开启 ip 转发功能的。这是一个好的做法，因为大多数人是用不到 ip 转发的，但是如果我们架设一个 linux 路由或者 vpn 服务我们就需要开启该服务了

ii 通过访问内核的 `ipv4.ip_forward` 来判断转发是否开启

- 使用 `sysctl`:

```
$ sysctl net.ipv4.ip_forward
net.ipv4.ip_forward = 0
```

- 或者检查 `/proc` 下的文件:

```
$ cat /proc/sys/net/ipv4/ip_forward
0
```

如果值为 0，则表示 ipv4 转发没有开启

iii 开启 IP 转发

- 通过 `sysctl` 我们可以开启 ipv4 的转发功能（无需重启）:

```
$ sysctl -w net.ipv4.ip_forward=1
```

- 或者 `echo 1 > /proc/sys/net/ipv4/ip_forward`

这两种设置只是暂时的；它的效果会随着计算机的重启而失效

- 如果你想使 ip 转发永久生效，就请修改 `/etc/sysctl.conf`

将其中的 `#net.ipv4.ip_forward = 1` 中的注释去掉，改为

`net.ipv4.ip_forward = 1`

如果你的 ipv4 转发项已被设为 0 那么你只需要将它改为 1，

然后运行 `/etc/init.d/procps.sh restart` 使其生效，或者重启机器使其生效

4) 测试网络的连通性

在机器 A 中输入如下的命令

```
$ ping 192.168.2.10
```

```
root@OrangePI:~# ping 192.168.2.10
PING 192.168.2.10 (192.168.2.10) 56(84) bytes of data.
64 bytes from 192.168.2.10: icmp_seq=1 ttl=63 time=1.81 ms
64 bytes from 192.168.2.10: icmp_seq=2 ttl=63 time=1.22 ms
64 bytes from 192.168.2.10: icmp_seq=3 ttl=63 time=1.25 ms
64 bytes from 192.168.2.10: icmp_seq=4 ttl=63 time=1.27 ms
64 bytes from 192.168.2.10: icmp_seq=5 ttl=63 time=1.28 ms
^C
--- 192.168.2.10 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4006ms
rtt min/avg/max/mdev = 1.222/1.369/1.817/0.227 ms
root@OrangePI:~#
```

如果输出如上图所示，则表示连通性测试成功，从机器 A 发出的送往机器 C 的数据包能通过机器 B 进行转发



3. 配置 OSPF 路由协议

- 1) 在 2 中的局域网中并没有启用路由协议也可以进行转发，因为使用的是默认路由，当网络再复杂一点时，配置静态路由的工作量会成倍的增加，这时我们一般都会使用相应的路由协议来进行路由。
- 2) 在机器 B 中配置 OSPF 路由协议的过程如下：
 - a. 在第一节已经安装好了 Quagga 路由协议套件，接下来我们进行相应的配置：

i 首先查看 /usr/local/etc 目录下是否有如下配置文件

```
root@OrangePizero:/usr/local/etc# ls
babeld.conf.sample  isisd.conf.sample  ripd.conf.sample
bgpd.conf.sample    ospf6d.conf.sample  ripngd.conf.sample
bgpd.conf.sample2   ospfd.conf.sample  zebra.conf.sample
root@OrangePizero:/usr/local/etc#
```

ii 将要启用的协议的配置文件重命名（无论启用哪个协议 zebra 都是必须要开启的）

```
$ cp ripd.conf.sample ripd.conf
$ cp zebra.conf.sample zebra.conf
$ cp bgpd.conf.sample bgpd.conf
```

b. 开启对应的路由协议的进程

```
$ zebra -d
$ ospfd -d
```

如果出现如下的错误

```
root@OrangePizero:~/quagga-0.99.24# zebra -d
zebra: error while loading shared libraries: libzebra.so.0: cannot open shared object file: No such file or directory
root@OrangePizero:~/quagga-0.99.24#
```

则将 /usr/local/lib 下对应的库文件拷贝到/lib 下即可

```
root@OrangePizero:/usr/local/lib# ls
libospf.a          libospfapiclient.so.0.0.0  libzebra.a        python2.7
libospfapiclient.a  libospf.la                 libzebra.la       python3.4
libospfapiclient.la  libospf.so                 libzebra.so
libospfapiclient.so  libospf.so.0              libzebra.so.0
libospfapiclient.so.0  libospf.so.0.0.0         libzebra.so.0.0
root@OrangePizero:/usr/local/lib# cp lib* /lib/
root@OrangePizero:/usr/local/lib#
```

c. 通过 telnet 连接对应协议的 daemon（Password 为：zebra）



```
root@OrangePizero:/usr/local/etc# telnet localhost ospfd
Trying ::1...
Connected to localhost.
Escape character is '^]'.

Hello, this is Quagga (version 0.99.24).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

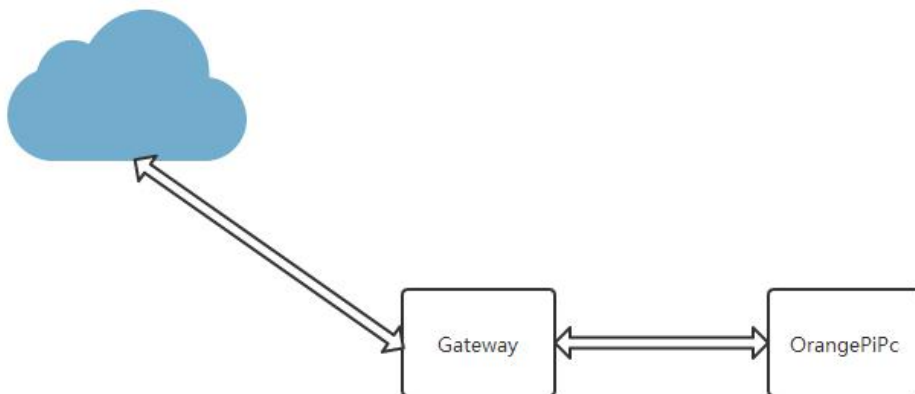
User Access Verification

Password:
ospfd> en
ospfd#
```

d. 配置 OSPF 协议的过程（以图一的拓扑为例）

```
ospfd# configure t
ospfd# configure terminal
ospfd(config)# ro
ospfd(config)# router
ospfd(config)# router o
ospfd(config)# router ospf
ospfd(config-router)# netwo
ospfd(config-router)# network 192.168.1.0/24
ospfd(config-router)# network 192.168.1.0/24 area 0
ospfd(config-router)# network 192.168.2.0/24 area 0
ospfd(config-router)#
```

4. 配置 NAT



- 1) 在图二中, Gateway 为 OrangePiR1 开发板, 它其中的一个网卡(eth1)和 Internet 相连, 另外一个网卡(eth0)和内部局域网相连
 - 2) Gateway 的配置步骤如下:
 - i 安装 iptables
- ```
$ sudo apt-get install -y iptables
```



ii /etc/network/interfaces 的配置如下, 其中 eth1 使用系统自动分配的 IP 地址

```
auto eth0
iface eth0 inet static
address 192.168.3.12
#gateway 192.168.3.10
netmask 255.255.255.0
```

iii iptables 的配置如下:

```
iptables -F
```

```
iptables -P INPUT ACCEPT
```

```
iptables -P FORWARD ACCEPT
```

```
iptables -t nat -A POSTROUTING -o eth1 -j MASQUERADE
```

3) OrangePiPc 的配置如下:

/etc/network/interfaces 的配置如下

```
auto eth0
iface eth0 inet static
address 192.168.3.10
gateway 192.168.3.12
netmask 255.255.255.0
```

在/etc/resolv.conf 中配置 DNS 服务器的地址

```
Generated by NetworkManager
#nameserver 127.0.1.1
nameserver 114.114.114.114
```

4) 测试

在 OrangePiPc 上如果能 ping 通 www.baidu.com 说明 NAT 设置成功

## 5. 配置 DHCP

1) DHCP 服务器端的配置过程 (OrangePiR1)

a. OrangePiR1 的 Ubuntu15.04 系统上安装 dhcp 服务器的命令

```
$ sudo apt-get install -y isc-dhcp-server (注意包名)
```

b. 在/etc/default/isc-dhcp-server 文件中修改网口, 如下:

```
On what interfaces should the DHCP server (dhcpd) serve DHCP requests?
Separate multiple interfaces with spaces, e.g. "eth0 eth1".
INTERFACES="eth0"
```



- c. 另外，还要修改/etc/dhcp/dhcpd.conf 文件，该文件主要是配置 DHCP 分配的 IP 范围，下面是设置为 192.168.3.0 网段的示例：

```
subnet 192.168.3.0 netmask 255.255.255.0 {
 range 192.168.3.100 192.168.3.105;
 option routers 192.168.3.; #配置客户端默认网关，必须要添加
 option broadcast-address 192.168.3.255;
 default-lease-time 600;
 max-lease-time 7200;
}
```

- d. 在 OrangePiR1 上启动 DHCP 服务

```
$ service isc-dhcp-server start
```

## 2) DHCP 客户端的配置过程 (OrangePiPc)

- a. /etc/network/interface 的配置如下

```
#auto eth0
#iface eth0 inet static
#address 192.168.3.10
#gateway 192.168.3.12
#netmask 255.255.255.0

auto eth0
iface eth0 inet dhcp
```

- b. OrangePiPc 客户端的使用如下命令自动获取 IP

```
$ dhclient eth0
```