



Orange Pi

用户手册



目录

一、 Orange Pi 基本特性.....	1
1. GPIO 规格.....	1
2. CSI 摄像头接口规格.....	2
二、 开发板使用介绍.....	4
1. 步骤 1: 准备需要的配件.....	4
2. 步骤 2: 准备板子启动需要的 TF 卡或 EMMC 镜像.....	5
3. 步骤 3: 启动你的香橙派开发板.....	11
4. 步骤 4: 正确关闭你的香橙派开发板.....	12
5. 其他设置.....	12
6. 通用软件配置.....	16
三、 Linux 内核源码编译.....	30
1. 下载 Linux 源码.....	30
2. 源码编译.....	31
3. 内核镜像文件和模块替换.....	33
四、 Android 源码编译.....	37
1. JDK 的安装.....	37
2. 安装平台支持软件.....	38
3. 下载 Android 源码.....	38
4. 编译工具链的安装.....	38
5. Lichee 源码编译.....	39
6. Android 源码编译命令.....	39
五、 使用工程配置文件.....	42
1. sys_config.fex 简介.....	42
2. Uboot 和 boot 更新方法.....	42
六、 Orange Pi 驱动程序开发.....	46
1. 设备驱动和应用程序的编写.....	46
2. 设备驱动的编译方法.....	48
3. 交叉编译器编译应用程序.....	51
4. 驱动和程序的运行方式.....	53
七、 串口调试工具介绍.....	54
1. 基于 Windows 平台的使用.....	54
2. 基于 Linux 平台的使用.....	58

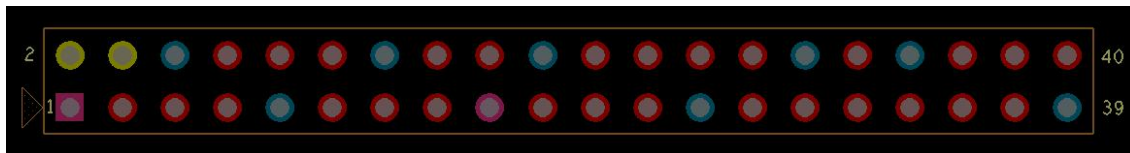


一、Orange Pi 基本特性

1. GPIO 规格

香橙派开发板有 40 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：



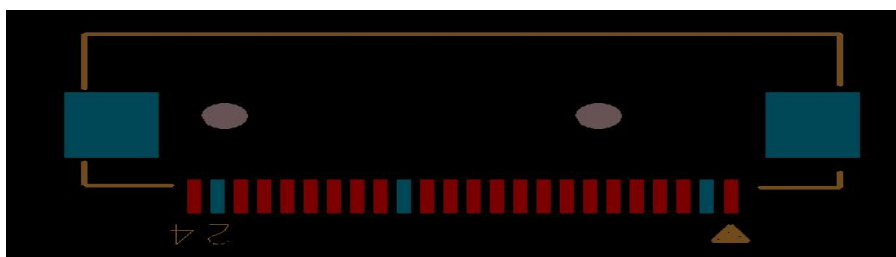
OrangePi(H3)		
CON3-P01	VCC-3V3	
CON3-P02	VCC-5V	
CON3-P03	TWI0-SDA	PA12
CON3-P04	VCC-5V	
CON3-P05	TWI0-SCK	PA11
CON3-P06	GND	
CON3-P07	PWM1	PA6
CON3-P08	UART3_TX	PA13
CON3-P09	GND	
CON3-P10	UART3_RX	PA14
CON3-P11	UART2_RX	PA1
CON3-P12	PD14	PD14
CON3-P13	UART2_TX	PA2
CON3-P14	GND	
CON3-P15	UART2_CTS	PA3
CON3-P16	PC4	PC4
CON3-P17	VCC-3V3	
CON3-P18	CAN_RX	PC7
CON3-P19	SPI0_MOSI	PC0
CON3-P20	GND	
CON3-P21	SPI0_MISO	PC1



CON3-P22	UART2_RTS	PA2
CON3-P23	SPI0_CLK	PC Plus
CON3-P24	SPI0_CS0	PC3
CON3-P25	GND	
CON3-P26	PA21	PA21
CON3-P27	TWI1-SDA	PA19
CON3-P28	TWI1-SCK	PA18
CON3-P29	PA7	PA7
CON3-P30	GND	
CON3-P31	PA8	PA8
CON3-P32	UART1_RTS	PG8
CON3-P33	PA9	PA9
CON3-P34	GND	
CON3-P35	PA10	PA10
CON3-P36	UART1_CTS	PG9
CON3-P37	PA20	PA20
CON3-P38	UART1_TX	PG6
CON3-P39	GND	
CON3-P40	UART1_RX	PG7

2. CSI 摄像头接口规格

CSI摄像头接口是一个可以通过适当的信号引脚映射来连接外部摄像头模块的24pin FPC 接口。CSI 接口的引脚定义如下所示。在香橙派开发板上标有“CON1”的就是摄像头接口。



OrangePi-CSI

CON1-P01	NC	
----------	----	--



CON1-P02	GND	
CON1-P03	TWI2-SDA	PE13
CON1-P04	VCC-CSI	
CON1-P05	TWI2-SCK	PE12
CON1-P06	CSI-RESET#	PE15
CON1-P07	CSI-VSYNC	PE3
CON1-P08	CSI-STBY-EN	PE15
CON1-P09	CSI-HSYNC	PE2
CON1-P10	VDD1V8-CSI	
CON1-P11	VCC-CSI	
CON1-P12	CSI-D7	PE11
CON1-P13	CSI-MCLK	PE1
CON1-P14	CSI-D6	PE10
CON1-P15	GND	
CON1-P16	CSI-D5	PE9
CON1-P17	CSI-PCLK	PE0
CON1-P18	CSI-D4	PE8
CON1-P19	CSI-D0	PE4
CON1-P20	CSI-D3	PE7
CON1-P21	CSI-D1	PE5
CON1-P22	CSI-D2	PE6
CON1-P23	GND	
CON1-P24	AFVCC-CSI	



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，你至少需要准备如下的一些配件：

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
2	标准 HDMI 转 HDMI 线 标准 HDMI 转 DVI 线	标准 HDMI 转 HDMI 线用于连接 HD TV 或者 HD 显示器 标准 HDMI 转 DVI 线用于接 DVI 显示器
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）	网络属于可选项，它能够更加方便的更新和安装你的香橙派开发板上的软件。
6	电源适配器	至少 5V/2A 高品质电源适配器，大部分板子 OTG 不能用作电源输入（除 Zero Plus2 H3 外）。
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。



HDMI 转 HDMI



HDMI 转 DVI



AV 视频线



TF 卡



电源适配器



2. 步骤 2: 准备板子启动需要的TF卡或EMMC镜像

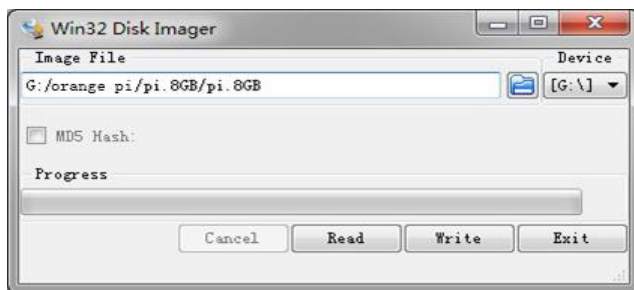
为了能够正常使用香橙派开发板，必须先在TF卡上安装对应板子的操作系统。

1) 基于 Windows 平台将 Linux 操作系统写入 TF 卡

- a. 把TF卡插入电脑，TF卡的容量必须比操作系统镜像大，通常需要 8GB或更大容量
- 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

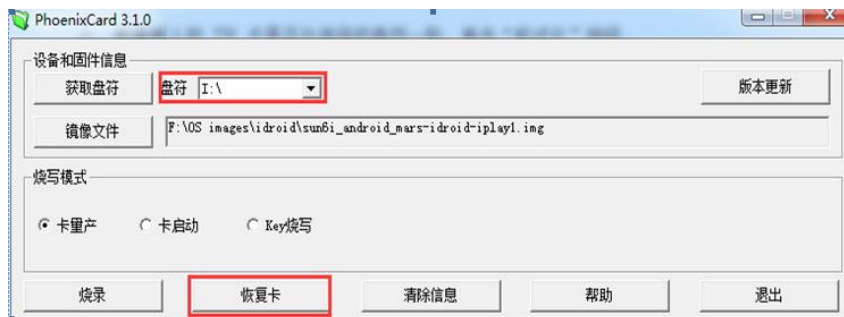
2) 基于 Windows 平台将 linux 操作系统写入 TF 卡

- 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 8GB或更大容量
- 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。

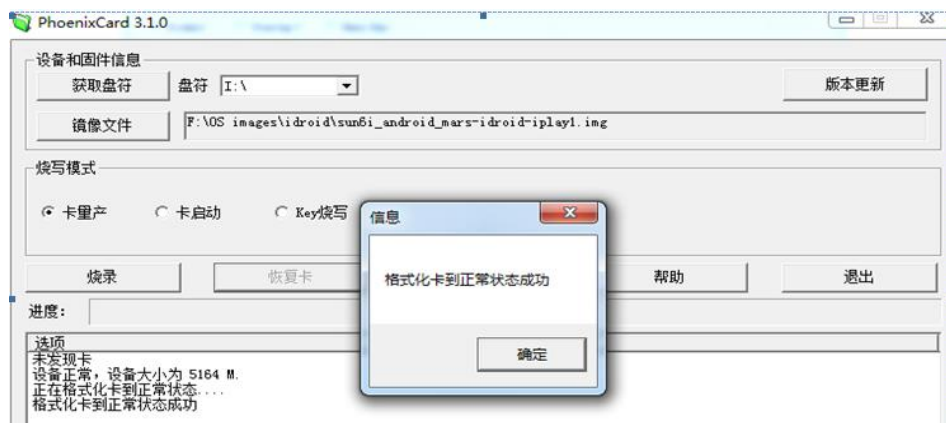
3) 使用工具 PhoenixCard 工具将 Android 系统镜像写入 TF 卡

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

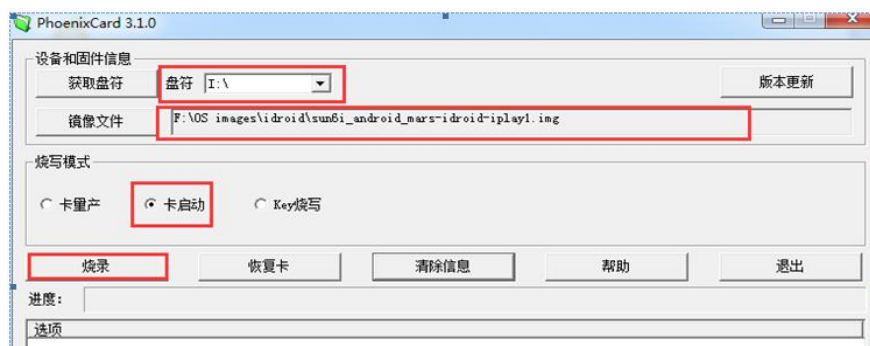


- c. 检查插入的 TF 卡是否与选择的盘符一致，单击“恢复卡”按钮，开始格式 TF

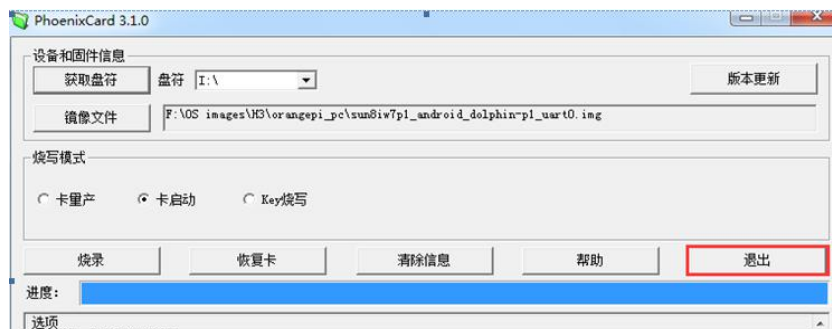


格式化 TF 卡成功后，单击“确定”按钮。

- 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。



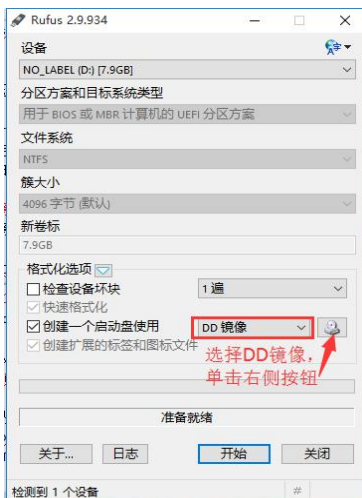
4) Armbian 镜像写入 TF 卡

- a. 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 8GB 或更大容量。下载操作系统镜像文件，镜像下载页面地址如下：

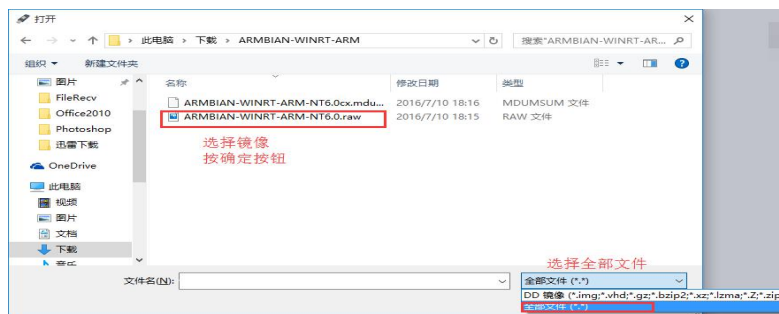
<http://www.armbian.com/download/>

- b. 镜像烧录至 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。

5) Android 镜像文件烧写至 EMMC

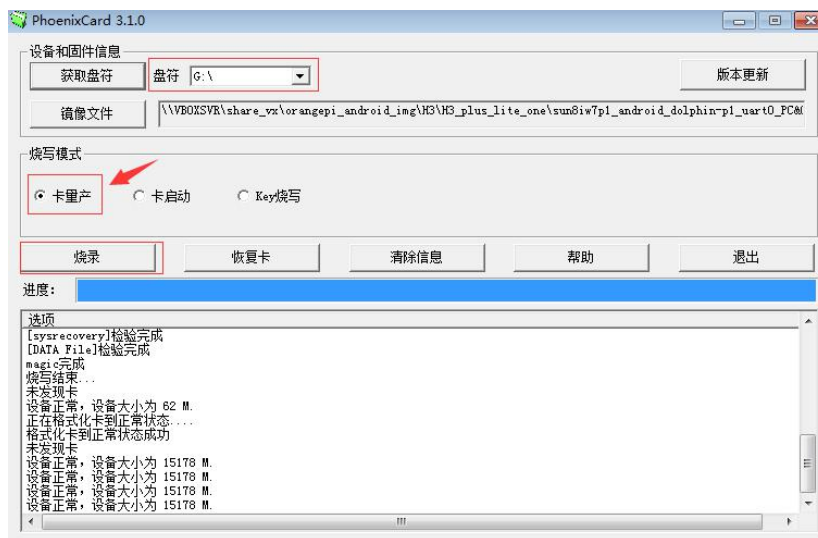
烧写镜像至 EMMC 的步骤基本和烧写至 TF 卡的步骤一样。

- a. 下载对应 Android 系统和 PhoenixCard 烧写工具

- b. 格式化 TF 卡



- c. 确认 TF 卡是否与盘符一致，单击“恢复卡”，开始格式化卡
- d. 将 Android 系统写入 TF 卡中，注意下方红色标记的地方，与写入 TF 卡不一样。



用“卡量产”的方式烧完之后，退出即可

- e. 上电，可以发现板子上的红灯一闪一闪的，表示正在烧录系统，当红灯熄灭，表示已经烧录完成，拔掉 TF 卡，上电系统即可跑起来。

6) Linux 镜像文件烧写至 EMMC

H3 系列的烧录至 EMMC 的方式都一样，本手册以 PC PLUS 为例，使用 `install_to_emmc` 即可。下面分别以官网系统和 Armbian 的系统演示说明如何操作。

- a. 官网系统

```
$ sudo install_to_emmc
```

输入 Y，确认清除 EMMC 数据。

```
root@OrangePI:~# install_to_emmc

Thu Feb 11 16:29:30 UTC 2016
=====
Installing Linux system to emmc
=====

WARNING: EMMC WILL BE ERASED !, Continue (y/N)? y
Erasing EMMC ...
```

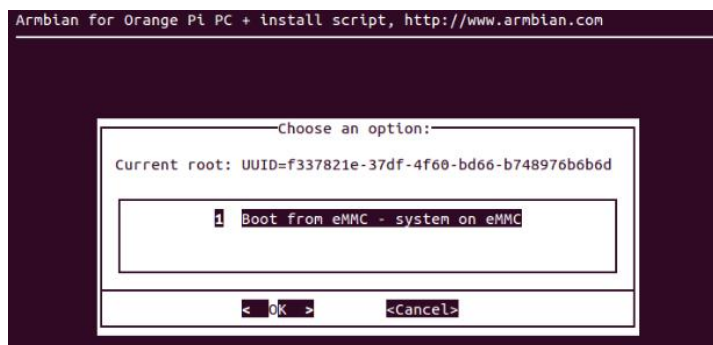
烧录至 EMMC 完成，拔掉 SD 卡，即可从 EMMC 启动



```
Installing u-boot to EMMC ...  
Mounting EMMC partitions...  
FAT partitions mounted to /tmp/_fatdir  
linux partition mounted to /tmp/_extdir  
Copying file system to EMMC ...  
Creating "fstab"  
*****  
Linux system installed to EMMC.  
*****
```

b. Armbian 系统

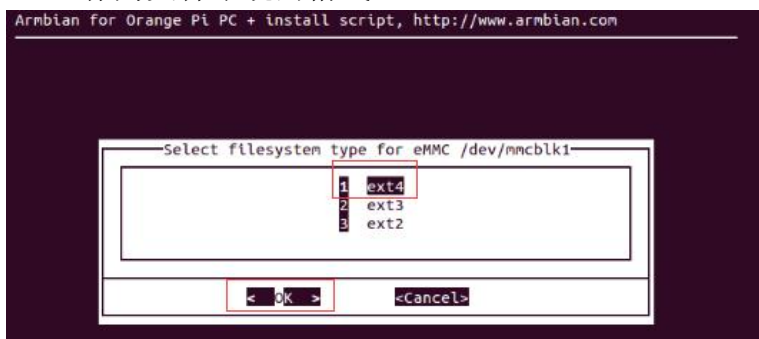
```
$ sudo nand-sata-install
```



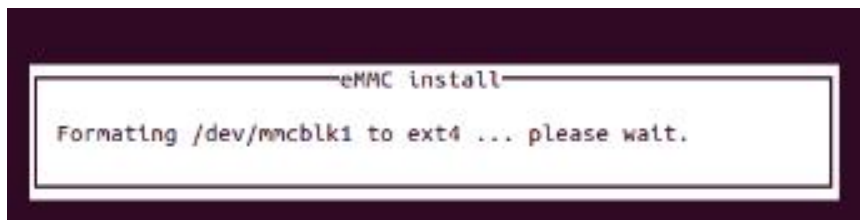
选择 OK



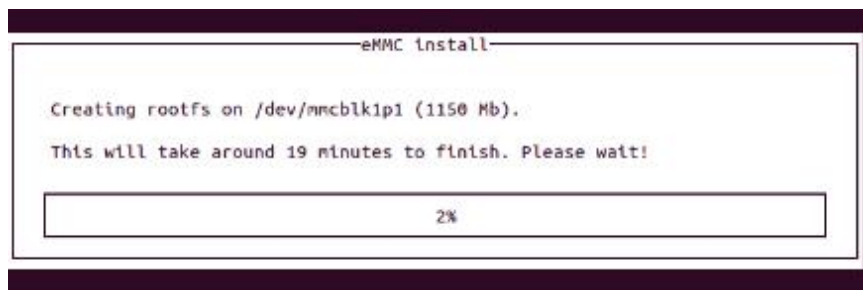
选择 EXT4 作为文件系统的格式



格式化分区



等待系统至 EMMC 完成即可。



3. 步骤 3: 启动你的香橙派开发板

1) 详细启动步骤

- a. 将写好镜像的TF卡插入香橙派开发板
- b. 用HDMI线连接香橙派开发板和HDMI TV或者显示器，或者可以连接AV接口来输出视频和音频到模拟TV或显示器，如果板子既没有HDMI接口也没有AV接口，可以利用转接板上的AV接口实现输出。
- c. 将 USB 键盘和鼠标插入板子上的USB接口
- d. 插入网线将香橙派开发板接入有线网络
- e. 插入 5V/3A 的电源适配器，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

注意：部分板子不能用Micro-usb（OTG）作为电源输入，一定要连接DC电源才能正常给板子供电。

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较长的时间，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钮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钮关闭系统可能会损坏TF卡或者是文件系统。系统关闭后可以长按 5 秒以上的电源按钮进行断电。

5. 其他设置

1) 扩展TF卡

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。

a. 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。

b. 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   56M   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
```

2) 连接有线网络

Orange Pi 在加电开机前如果已连接网线, 则系统启动后自动获取 IP 地址, 如果没有连接网线, 或是其它网络问题, 则会导致获取 IP 地址失败, 同时系统启动会因此等待一小段时间, 不影响系统的正常运行。

连接上网络之后应该是绿灯常亮, 黄灯闪烁。要确定烧录的镜像是该板子的, 因为有些网卡是百兆网口 (pc, one), 有些网口是千兆 (PC Plus), 不能混用。

百兆网卡用的是芯片内部的数据线 (internal phy), 配置如下:

2 表示 internal phy

```
[gmac0]
gmac_used          = 2
;gmac_rxd3         = port:PD00<2><default><3><default>
;gmac_rxd2         = port:PD01<2><default><3><default>
;gmac_rxd1         = port:PD02<2><default><3><default>
```

千兆网卡用外部的千兆网卡芯片, 用的是 external phy

1 表示 external phy



```
[gmac0]
gmac_used      = 1
gmac_rxd3      = port:PD00<2><default><3><default>
gmac_rxd2      = port:PD01<2><default><3><default>
gmac_rxd1      = port:PD02<2><default><3><default>
gmac_rxd0      = port:PD03<2><default><3><default>
gmac_rxcclk    = port:PD04<2><default><3><default>
gmac_rxdv      = port:PD05<2><default><3><default>
```

以上配置默认已经配置好，可作为了解使用。

3) 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/authoriz

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

# To enable empty passwords, change t
PermitEmptyPasswords no

# Change to yes to enable challenge-r
# 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] [~]
-> #
```

4) HDMI或者 3.5mm输出声音

- 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
```

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。

- b. 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。

- c. 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
```

5) 测试GPU

启动系统，在 hami 界面下登陆，打开终端，并运行命令：

```
glmark2-es2
```

可以看到 mali400 的测试结果和跑分情况



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:orangepi,,,:/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=="sd*", RUN+="/usr/bin/pmount --sync --umask 000 %k"
ACTION=="remove", KERNEL=="sd*", 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-zenhe
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.

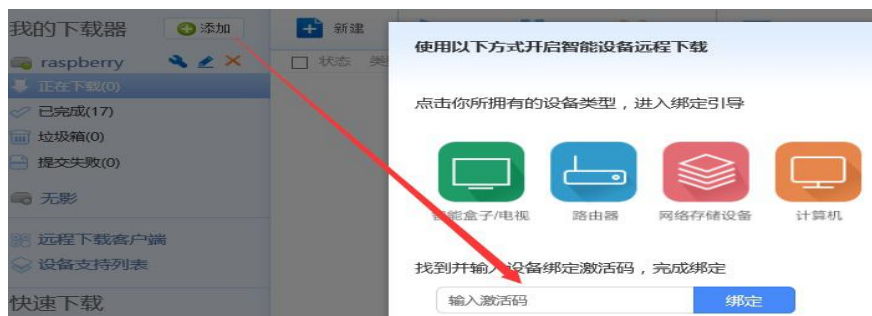
getting xunlei service info...
Connecting to 127.0.0.1:9000 (127.0.0.1:9000)

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）。运行时会有一两个错误，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

a. 扩大文件系统

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 kill 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 kill 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(保证文件系统无错误, 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。

上面的方法是使用 `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。

官网的 `Lubuntu` 和 `raspbian` 暂不可用

9) Linux 下如何使用 gc2035 摄像头

a. 用 `find` 命令找到下面的模块在文件中的位置, 按照指定的顺序加载模块

```
insmod videobuf-core.ko
insmod videobuf-dma-contig.ko
insmod uvcvideo.ko
insmod cci.ko
insmod vfe_os.ko
insmod vfe_subdev.ko
insmod gc2035.ko
insmod vfe_v4l2.ko
```

加载之后, 应该会在 `/dev/` 下面生成 `video0`
底层驱动安装好, 安卓可以直接使用。

b. Linux 下使用摄像头

i 加载驱动

```
$ sudo modprobe gc2035
$ sudo modprobe vfe_v4l2
```

ii 安装 motion

```
$ sudo apt-get install motion
```



- iii 修改配置

```
$ sudo nano /etc/motion/motion.conf
$ stream_localhost off
```
- iv 创建文件夹，用于保存图片

```
$ mkdir ~/motion
```
- v 修改权限

```
$ chmod 777 motion
```
- vi 继续修改配置

```
$ sudo nano /etc/default/motion
$ start_motion_daemon=yes
```
- vii 启动服务器

```
$ sudo /etc/init.d/motion start
```

最后一步，在浏览器中输入

localhost:8081

即可查看摄像头输出的图像

详细步骤参照以下链接：

<http://www.cnx-software.com/2015/09/26/how-to-use-orange-pi-camera-in-linux-with-motion/>

10) 设置 eth0 和 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]
sWantedBy=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.

11) Orange Pi 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

12) WiringPi 安装和使用

a. 安装 WiringPi

i 安装源码需要的编译工具

```
$ sudo apt-get install gcc g++ make
```

ii 编译 GPIO 的 H3 的驱动

```
git clone https://github.com/kazukioishi/WiringOP.git -b h3
```

```
cd WiringOP
```

```
chmod +x ./build
```

```
sudo ./build
```

iii GPIO 打印信息

```
# gpio -v
```

```
gpio version: 2.20
```

```
Copyright (c) 2012-2014 Gordon Henderson
```

```
This is free software with ABSOLUTELY NO WARRANTY.
```

```
For details type: gpio -warranty
```

```
Banana Pro Details:
```

```
Type: Banana Pro, Revision: 1.2, Memory: 1024MB, Maker: LeMaker
```

iv 显示



gpio readall

Orange Pi											
BCM	wPi	Name	Mode	V	Physical	V	Mode	Name	wPi	BCM	
		3.3v			1	2		5v			
12	8	SDA.0	ALT5	0	3	4		5V			
11	9	SCL.0	ALT5	0	5	6		0v			
6	7	GPIO.7	ALT3	0	7	8	0	ALT5	TxD3	15	13
		0v			9	10	0	ALT5	RxD3	16	14
1	0	RxD2	ALT5	0	11	12	0	ALT3	GPIO.1	1	110
0	2	TxD2	ALT5	0	13	14		0v			
3	3	CTS2	ALT5	0	15	16	0	ALT3	GPIO.4	4	68
		3.3v			17	18	0	ALT3	GPIO.5	5	71
64	12	MOSI	ALT4	0	19	20		0v			
65	13	MISO	ALT0	0	21	22	0	ALT5	RTS2	6	2
66	14	SCLK	ALT4	0	23	24	0	ALT4	CE0	10	67
		0v			25	26	0	ALT3	GPIO.11	11	21
19	30	SDA.1	ALT4	0	27	28	0	ALT4	SCL.1	31	18
7	21	GPIO.21	ALT3	0	29	30		0v			
8	22	GPIO.22	ALT3	0	31	32	0	ALT5	RTS1	26	200
9	23	GPIO.23	ALT3	0	33	34		0v			
10	24	GPIO.24	ALT3	0	35	36	0	ALT5	CTS1	27	201
20	25	GPIO.25	OUT	1	37	38	0	ALT5	TxD1	28	198
		0v			39	40	0	ALT5	RxD1	29	199
BCM	wPi	Name	Mode	V	Physical	V	Mode	Name	wPi	BCM	

13) 配置网络的具体步骤

方法 1:

a. 命令行输入

```
$ ifconfig
```

看是否有网卡(wlan*)

b. 如果没有，根据网卡的型号加载相应模块

```
$ insmod 8189*.ko
```

例如：RTL8189ftv 对应的 8189fs.ko，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 &
dhcpd wlan0 &
```

- c. 测试 `wifi` 连接情况

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

14) OrangePi 使用微雪 5 寸屏

微雪屏在 Orange Pi 上使用要修改 `script.bin` 的配置参数

修改 `[hdmi_para]` 参数, 加入如下定义

```
hdcp_enable = 0
hdmi_cts_compatibility = 1
```

修改输出的分辨率为 `800*480`

```
screen0_output_mode = 31
```

再用 `sunxi-tools` 将 `script.fex` 转换成 `script.bin` 替换即可



15) 使用官方 USB wifi

- a. 将 USBwifi 插上(确保 USB 打开)。命令行输入 lsusb 查看 USB 设备的详细信息

```
$ dmesg
```

```
$ lsusb
```

(Bus 008 Device 002: ID 0bda:8176 Realtek Semiconductor Corp. RTL8188CUS 802.11n WLAN Adapter) id 为 8176, 在网上查找后需要的驱动为 rtl8188cu.

可通过下面的链接找到

<https://sites.google.com/site/easylinuxtipsproject/reserve-7#TOC-Realtek-RTL8188CUS-and-RTL8192CU-chipsets-0bda:8176-and-0bda:8178->

1 Determine the chipset

2 Realtek RTL8188CUS and RTL8192CU chipsets (0bda:8176 and 0bda:8178)

源码编译, lib/modules/*/drivers/net/wireless/realtek/下面生成 8192CU.ko

- b. 光盘里面有提供的源码

(rtl8188C_8192C_usb_linux_v4.0.2_9000.20130911.tar.gz)

将编译出来的 uImage 和 lib 的目录更新

- c. 重新上电, 会自动加载如下模块

rtlwifi.ko,rtl8192c_common.ko,mac80211.ko,rtl8192cu.ko.

- d. 将 rtl8192cu.ko 这个模块卸载,安装 8192cu.ko。修改/etc/modules, 行末加上

8192cu, 使其能上电自启

- e. 修改/etc/network/interfaces 添加 ssid 和 psk.

- f. 重启可用 USBwifi



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：Orange Pi 开发板，读卡器和一张 TF 卡

1. 下载 Linux 源码

源代码可以在 Orange Pi 官网上进行下载：

<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：



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. 源码编译

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
```

- 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
```

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 android



```
Welcome to mkscrip setup progress
All available chips:
  0. sun6i
  1. sun8iw6p1
  2. sun8iw7p1
  3. sun9iw1p1
Choice: 2
All available platforms:
  0. android
  1. dragonboard
  2. linux
Choice: 1
not set business, to use default!
LICHEE_BUSINESS=
using kernel 'linux-3.4':
All available boards:
  0. dolphin-cmcc-wasu-p1
  1. dolphin-p1
  2. dolphin-perf
  3. fpga
Choice: 1
```

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 sun8iw7p1 dragonboard lichee OK
-----
```

成功显示

- uboot.bin和boot0.bin源码编译
 1. 编译uboot



```
$ cd */lichee/brandy/u-boot-2011.09
$ make clean
$ make sun8iw7p1_config
$ make -j16
```

在*/lichee/tools/pack/chips/sun8iw7p1/bin/ 生成u-boot-sun8iw7p1.bin

2. 编译boot0

```
$ cd */lichee/brandy/u-boot-2011.09
$ make boot0
```

在*/lichee/tools/pack/chips/sun8iw7p1/bin/ 生成
boot0_sdcard_sun8iw7p1.bin

3. 烧录boot0 和uboot.bin至sd卡

```
dd if=boot0_sdcard_sun8iw7p1.bin of=/dev/sdb bs=1024 seek=8
dd if=u-boot-sun8iw7p1.bin of=/dev/sdb bs=1024 seek=16400
```

3. 内核镜像文件和模块替换

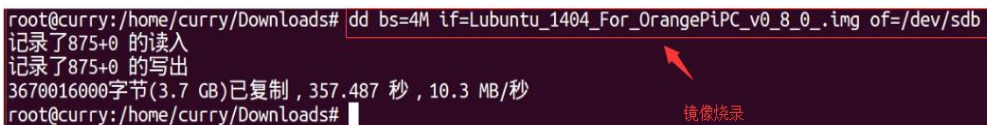
- 编译完成之后，将会目录下生成如下文件：

libs: lichee/out/sun8iw7p1/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
```



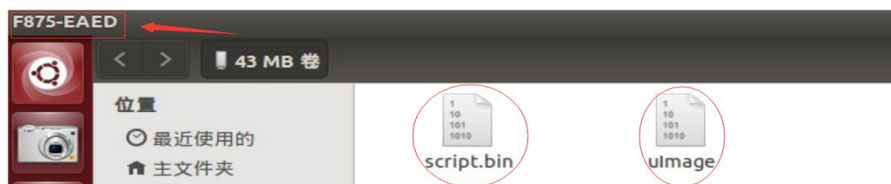
```
root@curry:/home/curry/Downloads# dd bs=4M if=Lubuntu_1404_For_OrangePiPC_v0_8_0_.img of=/dev/sdb
记录了875+0 的读入
记录了875+0 的写出
3670016000字节(3.7 GB)已复制, 357.487 秒, 10.3 MB/秒
root@curry:/home/curry/Downloads#
```

镜像烧录

- 拔掉读卡器，再插一次。

此时，将 SD 卡插入到 PC 上，查看 SD 卡的挂载点(如果不知道如何获得 SD 卡的挂载点，请参考下图)。

第一个分区是 boot 分区



第二个分区是 rootfs 分区



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

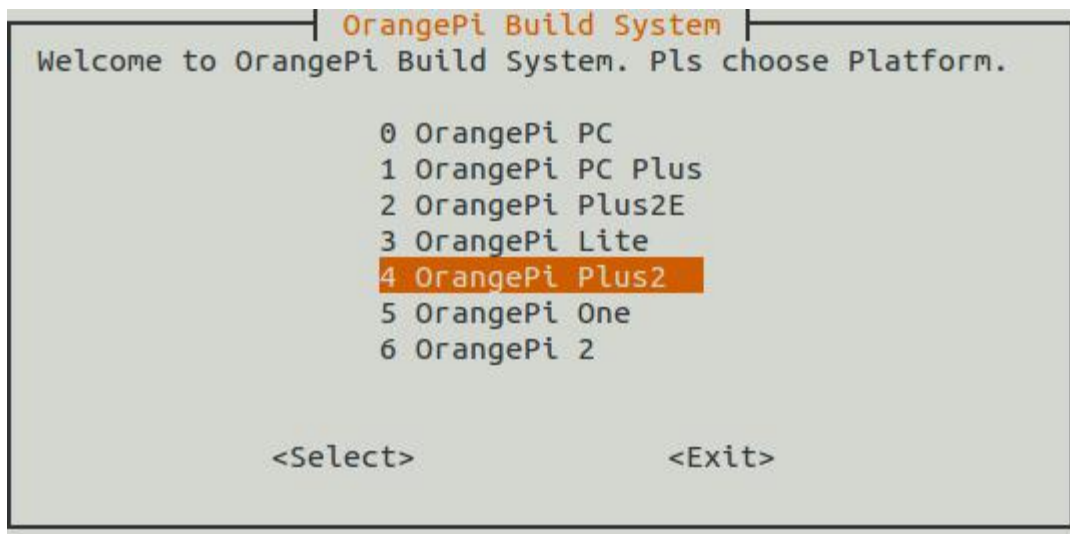
output 生成的文件

script 编译的脚本

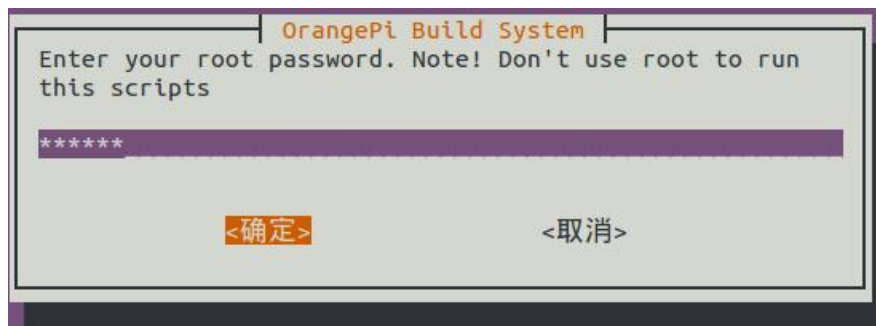
toolchain 存放交叉编译器

uboot uboot源码

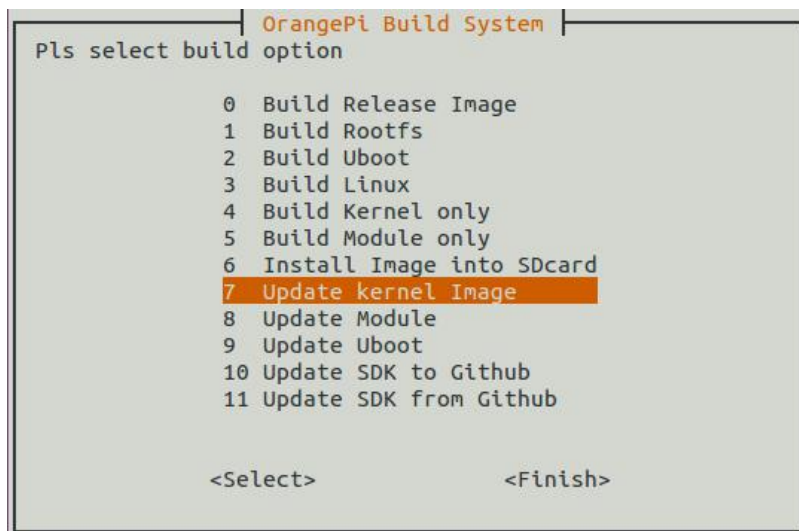
执行./build.sh, 进入图形化界面, 并选择PC Plus



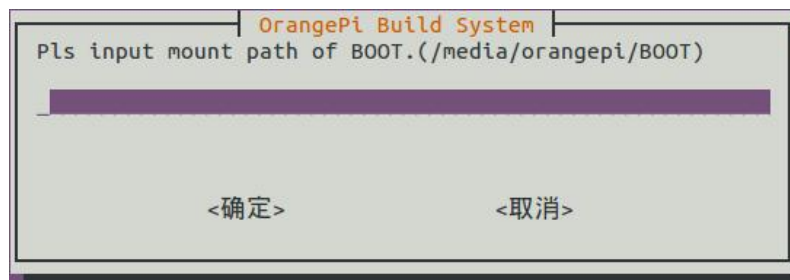
输入root密码



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、Android 源码编译

需要硬件：

Orange Pi 开发板，读卡器和一张 TF 卡

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；

Java 1.6 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 libnc6-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.org/downloadresources/>



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
```

```
root@curry:/home/curry/OrangePi/android/lichee# ls
brandy buildroot build.sh linux-3.4 README tools
root@curry:/home/curry/OrangePi/android/lichee# ./build.sh lunch
All available lichee lunch:
0. sun8iw6p1-android-eagle
1. sun8iw6p1-android-secure
2. sun8iw7p1-android-dolphin
3. sun8iw7p1-android-secure
4. sun8iw7p1-android-karaok
5. sun8iw8p1-android
6. sun9iw1p1-android-jaws
7. sun9iw1p1-android-secure
8. sun9iw1p1-android-optimus
Choice: 2
```

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# ls
abi      build    development  frameworks  libnativehelper  pdk      tools
art      cts      device      frameworks.tar.gz  Makefile         prebuilts  vendor
bionic   dalvik   docs        hardware     ndk              sdk
bootable developers external  libcore       packages        system
root@curry:/home/curry/OrangePi/android/android# source ./build/envsetup.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

```

root@curry:/home/curry/OrangePi/android/android# extract-bsp
/home/curry/OrangePi/android/android/device/*/dolphin-fvd-p1/bImage copied!
/home/curry/OrangePi/android/android/device/*/dolphin-fvd-p1/modules copied!

```

\$ make #后面的数值为同时编译的进程，依赖于主机的配置，不推荐“+”“-”

```

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

```

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

```

\$ cd */lichee/tools/pack/

```

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/>



General Tools

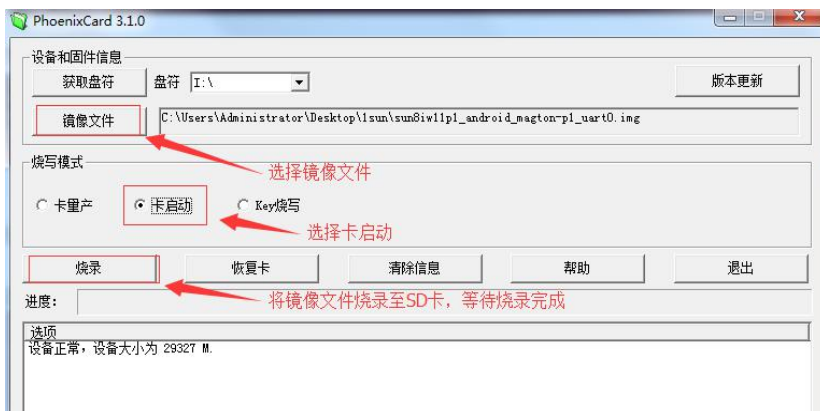
USB update tools for Linux 64bits OS (LiveSuitV306_For_Linux64.zip)		
burn TF/SD card BATCH-TOOL (PhoenixCard_V310_20130618.rar)		
USB update tools for linux 32bits (LiveSuitV306_For_Linux32.zip)		

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

ParserManager.dll	2013/6/19 13:40	应用程序扩展
PhoenixCard	2013/6/19 14:58	应用程序
PhoenixCard	2013/6/18 11:25	配置设置

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。

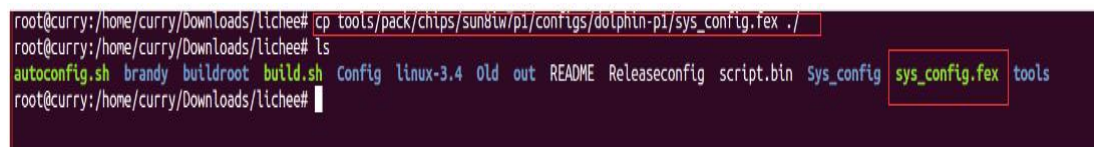
对于 OrangePi，其工程化配置文档的位置是：

lichee/tools/pack/chips/sun8iw7p1/configs/dolphin-p1/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-p1/sys_config.fex ./
```



```
root@curry:/home/curry/Downloads/lichee# cp tools/pack/chips/sun8iw7p1/configs/dolphin-p1/sys_config.fex ./
root@curry:/home/curry/Downloads/lichee# ls
autoconfig.sh  brandy  buildroot  build.sh  Config  linux-3.4  Old  out  README  Releaseconfig  script.bin  sys_config  sys_config.fex  tools
root@curry:/home/curry/Downloads/lichee#
```

用户可根据 sysconfig1.fex_manul_linux_BSP_v0.4.pdf 文件对 sys_config.fex 进行个人化配置。

sysconfig1.fex_manul_linux_BSP_v0.4.pdf 的目录为 /lichee/buildroot/docs

2. Uboot 和 boot 更新方法

修改完 sys_config.fex 项目之后，将新的内容更新到 SD 卡上。(注意，此时 SD 卡应已烧录好镜像，如果 SD 卡未烧录镜像，请参考：)

可以在终端输入以下命令：(可写成脚本，一键执行)

```
$ cd /lichee
```

```
$ busybox unix2dos sys_config.fex
```

```
$ cp ../tools/pack/chips/sun8iw7p1/bin/boot0_sdcard_sun8iw7p1.bin ./
```

```
$ cp ../tools/pack/chips/sun8iw7p1/bin/u-boot-sun8iw7p1.bin ./
```

```
$ tools/pack/pctools/linux/mod_update/script sys_config.fex >/dev/null
```

```
$ tools/pack/pctools/linux/mod_update/update_boot0boot0_sdcard_sun8iw7p1.bin
```

```
sys_config.bin SDMMC_CARD >/dev/null
```



```
$ tools/pack/pctools/linux/mod_update/update_ubootu-boot-sun8iw7p1.bin
sys_config.bin >/dev/null
$ mv sys_config.bin script.bin
```

如果已经烧录好系统的挂载点是 “/dev/sdb”，那么在终端使用命令进行 u-boot 和 boot 以及 script.bin 的更新，命令如下：

```
$ dd if=boot0_sdcard_sun8iw7p1.bin of=/dev/sdb bs=1024 seek=8
$ dd if=u-boot-sun8iw7p1.bin of=/dev/sdb bs=1024 seek=16400
```

操作完以上步骤，u-boot 和 boot 将更新完毕，但工程化配置的二进制文件还未更新。

将生成的 script.bin 拷贝到 boot 分区：

```
$ cd /lichee/
$ cp script.bin /media/*/boot/ -rf
```

至此，工程配置化文件更新完成，OrangePi 运行的系统将使用新的配置项进行初始化。

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

3) 修改分辨率

找到 disp_init 的选项, 修改子项 screen0_output_mode, 可显示不同的分辨率。

例如: 屏 0 输出模式 (used for tv/hdmi output, 0:480i 1:576i 2:480p 3:576p 4:720p50 5:720p60 6:1080i50 7:1080i60 8:1080p24 9:1080p50 10:1080p60 11:pal 14:ntsc)

以上是一些基本的主流分辨率, 如果以上分辨率不适合, 到创客群里找到补丁 z-0003-add-additional-video-modes.patch, 这个补丁支持一些其他的分辨率。

打补丁的方法:

```
patch -p1 < z-0003-add-additional-video-modes.patch
```

打完补丁重新源码编译, 替换 uImage 和 script.bin 即可

例如:

```

1024*768   是 32
1280*1024  是 33
1360*768   是 34
1440*900   是 35
1680*1050  是 36

```

注: 默认输出是 HDMI, 修改成 vga 输出, 方法相同。



六、Orange Pi 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：Orange Pi 开发板，读卡器和一张 TF 卡

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;
}

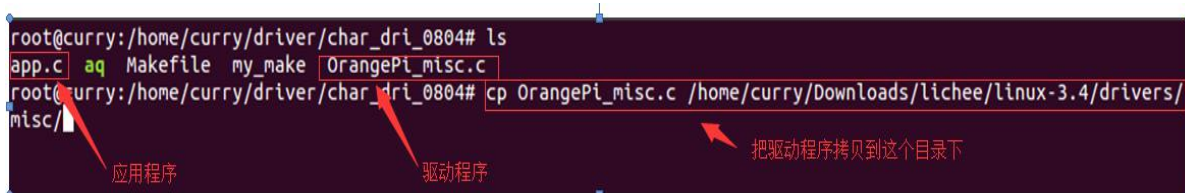
static void __exit OrangePi_misc_exit(void)
{
    int ret;
    printk("func : %s, line : %d\r\n", __func__, __LINE__);
    ret = misc_deregister(&OrangePi_Misc);
    if(ret < 0){

        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-3.4.39/driver/misc



进入*/lichee/linux-3.4.39/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)      += akm8975.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 orange_pi_misc.ko 文件:



```

root@curry:/home/curry/Downloads/lichee# ./build.sh
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 ...
external toolchain has been installed
INFO: build buildroot OK.
INFO: build kernel ...
INFO: prepare toolchain ...
use last time build config
Building kernel
/home/curry/Downloads/lichee/linux-3.4/output/lib/modules/3.4.39
CHK include/linux/version.h
CHK include/generated/utsrelease.h
make[1]: "include/generated/mach-types.h"是最新的。
CALL scripts/checksyscalls.sh
CHK include/generated/compile.h
AS arch/arm/mach-sunxi/power/brom/resumes.o
CC arch/arm/mach-sunxi/power/brom/resume_head.o

```

```

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
-----

```

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。

第二个分区是 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交叉编译器。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
ugurl=file:///usr/share/doc/gcc-4.8/README.Bugs --enable-languages=c,c++,java,go,d,fort
+ --prefix=/usr --program-suffix=-4.8 --enable-shared --enable-linker-build-id --libexe
-de/c++/4.8.4 --libdir=/usr/lib --enable-nls --with-sysroot=/ --enable-clocale=gnu --ena
ebug --enable-libstdc++-time=yes --enable-gnu-unique-object --disable-libmudflap --disa
sable-libquadmath --enable-plugin --with-system-zlib --disable-browser-plugin --enable-
enable-gtk-cairo --with-java-home=/usr/lib/jvm/java-1.5.0-gcj-4.8-armhf-cross/jre --ena
-with-jvm-root-dir=/usr/lib/jvm/java-1.5.0-gcj-4.8-armhf-cross --with-jvm-jar-dir=/usr/
/java-1.5.0-gcj-4.8-armhf-cross --with-arch-directory=arm --with-ecj-jar=/usr/share/jav
ar --disable-libgcj --enable-objc-gc --enable-multiarch --enable-multilib --disable-sjl
with-arch=armv7-a --with-fpu=vfpv3-d16 --with-float=hard --with-mode=thumb --disable-we
hecking=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)
root@curry:/home/curry/lichee#
```

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

解压并进入解压之后的目录

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

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-addr2line  arm-linux-gnueabihf-dwp  arm-linux-gnueabihf-gcc-ranlib  arm-linux-gnueabihf-ld  arm-linux-gnueabihf-ld.gold  arm-linux-gnueabihf-ranlib
arm-linux-gnueabihf-ar  arm-linux-gnueabihf-elfedit  arm-linux-gnueabihf-gcov  arm-linux-gnueabihf-gold  arm-linux-gnueabihf-readelf
arm-linux-gnueabihf-as  arm-linux-gnueabihf-g++  arm-linux-gnueabihf-gdb  arm-linux-gnueabihf-gn  arm-linux-gnueabihf-size  arm-linux-gnueabihf-strings
arm-linux-gnueabihf-c++  arm-linux-gnueabihf-gcc  arm-linux-gnueabihf-gfortran  arm-linux-gnueabihf-objcopy  arm-linux-gnueabihf-strip
arm-linux-gnueabihf-cfilt  arm-linux-gnueabihf-gcc-4.9.1  arm-linux-gnueabihf-gprof  arm-linux-gnueabihf-objdump  arm-linux-gnueabihf-strip
arm-linux-gnueabihf-cpp  arm-linux-gnueabihf-gcc-a  arm-linux-gnueabihf-ld  arm-linux-gnueabihf-pkg-config
arm-linux-gnueabihf-ct-ng.config  arm-linux-gnueabihf-gcc-nm  arm-linux-gnueabihf-ld.bfd  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 755 /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
修改权限之后
修改权限

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/toolschain/4.5.1/bin:/home/curry/tools/1_arm-linux-gnueabihf-gcc/gcc-linaro-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：Orange Pi 开发板，读卡器，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.zip，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	914 KB ← 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 ← 点击安装
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

a. 安装好之后打开程序如下图所示

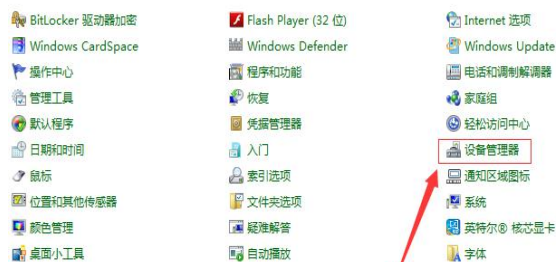


3) 调试的连接方式

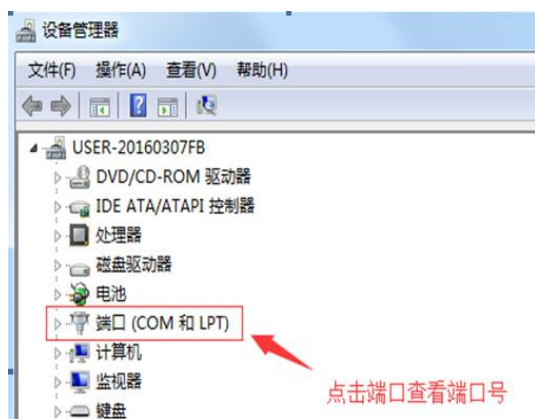
使用 TTL 转串口线，一端连接 OrangePi，另一端连接 PC

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



```

[mmc]: *****SD/MMC 0 init OK!!*****
[mmc]: erase_grp_size      : 0x1WtBlk*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
```

```

$~$sudo apt-get install ckermit

```

- 配置 kermit

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

```

$~$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
c
```

```
root@orange-All-Series: /home/orange
; This is /etc/kernit/kernrc
; It is executed on startup if ~/.kernrc is not found.
; See "man kermit" and http://www.kermit-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 \S(CKERMODO) assign _myinit \S(CKERMODO)
if not def _myinit assign _myinit \v(home).mykernrc

xif exist \m(_myinit) {                ; If it exists,
    echo Executing \m(_myinit)...      ; print message,
    take \m(_myinit)                   ; and TAKE the file.
}

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
c
```

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          l2c-4          psaux          sda7           tty21          tty47          ttyS13          uhid
block           l2c-5          ptmx           sda8           tty22          tty48          ttyS14          uinput
bsg             input          pts            sda9           tty23          tty49          ttyS15          urandom
bttrfs-control  kmsg          ram0           serial         tty24          tty5           ttyS16          val
bus             log            ram1           sg0            tty25          tty50          ttyS17          vboxusb
cdrom           loop0          ram10          sg1            tty26          tty51          ttyS18          vcs
char            loop1          ram11          shn            tty27          tty52          ttyS19          vcs1
console         loop2          ram12          snapshot       tty28          tty53          ttyS20          vcs2
core            loop3          ram13          snd            tty29          tty54          ttyS21          vcs3
cpu             loop4          ram14          sr0            tty3           tty55          ttyS22          vcs4
cpu_dma_latency loop5          ram15          stderr         tty30          tty56          ttyS23          vcs5
cuse            loop6          ram2           stdin          tty31          tty57          ttyS24          vcs6
disk            loop7          ram3           stdout         tty32          tty58          ttyS25          vcs7
dri             loop-control  ram4           tty            tty33          tty59          ttyS26          vcs8
ecryptfs        lp0            ram5           tty0           tty34          tty60          ttyS27          vcs9
fb0             mapper        ram6           tty1           tty35          tty61          ttyS28          vcsa1
fd              mcelog        ram7           tty10          tty36          tty62          ttyS29          vcsa2
full            me10          ram8           tty11          tty37          tty63          ttyS30          vcsa3
fuse            mem           ram9           tty12          tty38          tty64          ttyS31          vcsa4
hidraw0         memory_bandwidth random          tty13          tty39          tty65          ttyS32          vcsa5
hidraw1         ndctl0        rkill          tty14          tty40          tty66          ttyS33          vcsa6
hidraw2         net           rtc            tty15          tty41          tty67          ttyS34          vfto
hpet            network_latency network_throughput sda            tty16          tty42          tty68          vga_arbiter
hwrng           null          sda            tty17          tty43          tty69          ttyS35          vhost-net
l2c-0           parport0      sda1           tty18          tty44          tty70          ttyS36          video0
l2c-1           port          sda2           tty19          tty45          tty71          ttyS37          zero
l2c-2           ppp           sda3           tty20          tty46          tty72          ttyS38          zero
l2c-3
root@orange-All-Series:/home/orange#

```

串口号

- 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

```

kermit (/etc/kermit) - VIM
: CKERMOD or ~/.mykermitrc

if def $(CKERMOD) assign _myinit $(CKERMOD)
if not def _myinit assign _myinit $(home).mykermitrc

if 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

```

设置串口号

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