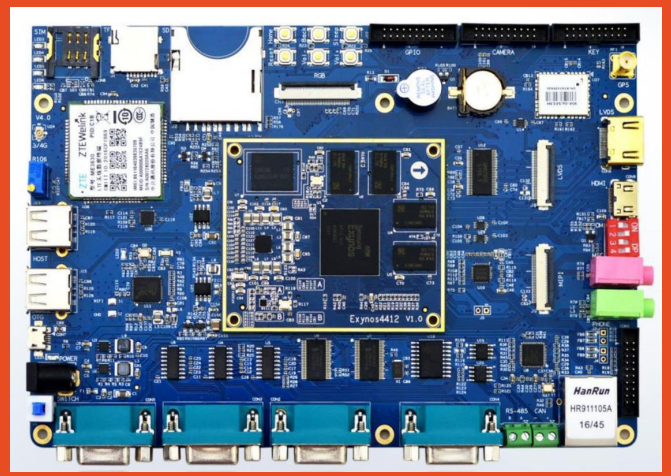
系统移植



彭老师

想入门和进阶ARM

请加关注一口君公众号：一口Linux

视频配套资料后台回复：arm

ARM

01

课程说明

嵌入式系列课程学习顺序

- 1.《从0学ARM-第一期》
- 2.《从0开始学ARM第二期-裸机开发》 资料获取，回复arm
- 3.《系统移植》 
- 4.《从0学Linux驱动第一期》 资料获取，回复ubuntu

配套图书

京东、当当、淘宝均有销售，

谁便宜买谁的！

但是请支持正版！

关注公众号：一口Linux 回复：arm 获取视频中所有资料

课程目标

- 掌握一款嵌入式产品软件移植步骤
- 掌握uboot工具的使用
- 掌握uboot、内核、文件系统的制作
- 为学习驱动做好准备工作

本课程需要的基础知识

- 1. C语言
- 2. linux基本操作
  - 命令、Makfile、shell
- 3. arm基础

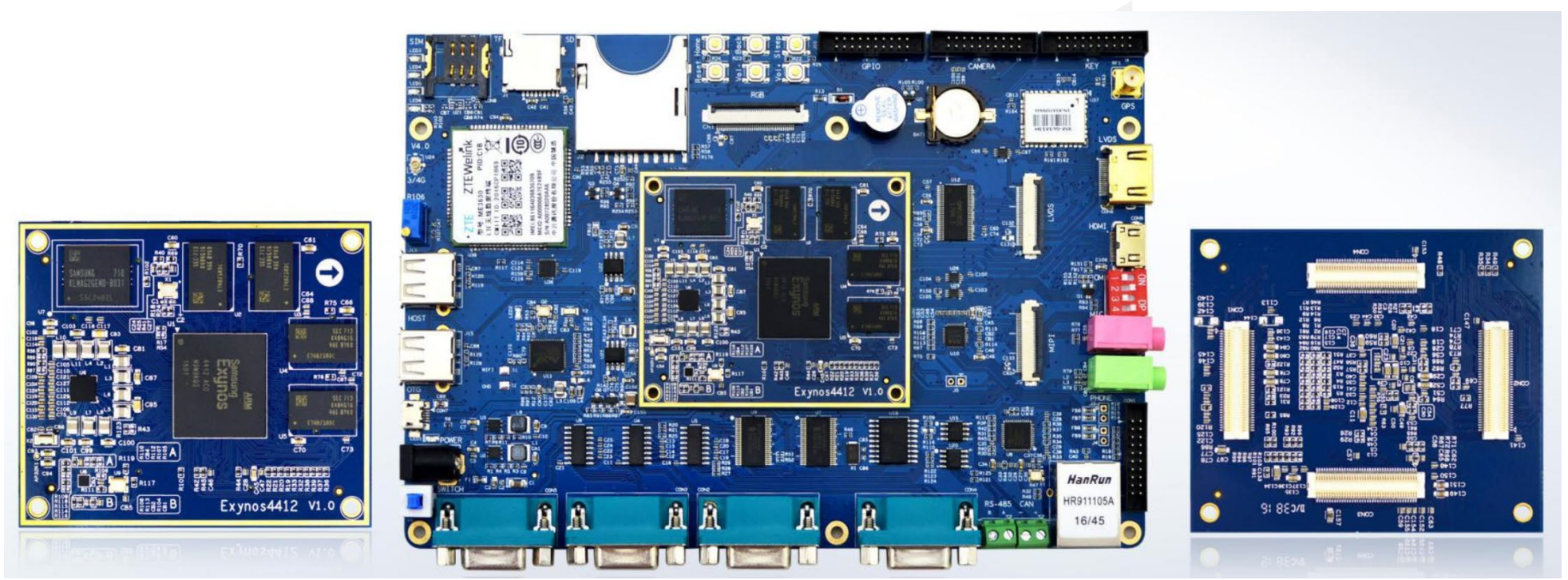
课程所需要的硬件

- 1. ARM开发板

- 2. tf卡或/SD卡 + 读卡器
  - 实测BanQ的32G、8G的没有问题

课程演示开发板

- 讯为4412全能板



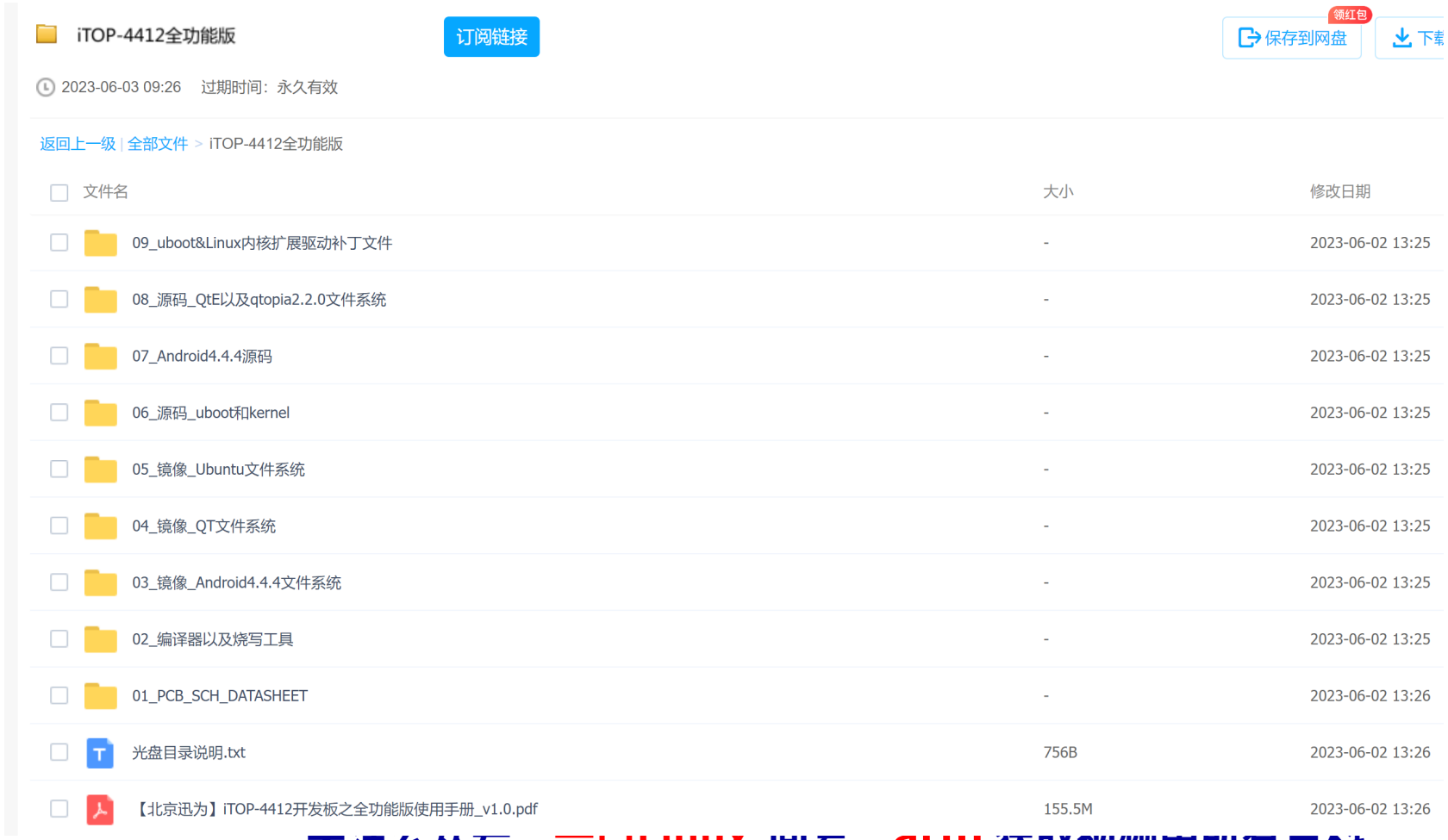
购买地址&&优惠券领取

- 购买链接
  - https://item.taobao.com/item.htm?spm=a21dvs.23580594.0.0.3f063d0drttxFQ&ft=t&id=729075103408
- 优惠券
  - 添加彭老师好友：yikoupeng
  - 群公告中领取

关注公众号：一口Linux 回复：arm 获取视频中所有资料

全能板资料下载地址

- https://pan.baidu.com/s/1BzNnNEXZKmXNK8bb4P4hXA 提取码：kpf6



关注公众号：一口Linux 回复：arm 获取视频中所有资料

其他相关资料下载

- https://pan.baidu.com/s/1yQT8g9nLYVxfEugyXXLlTw 提取码：fpz9

关注公众号：一口Linux 回复：arm 获取视频中所有资料

整理过的资料——建议下这个资料

脑 › 视频 (F:) › 录制视频 › 从0学arm第三期-系统移植-讯为4412 › xunwei

名称

- code
- uboot+kernel
- 补充pdf
- 工具软件
- 芯片手册
- 原理图
- 【北京迅为】iTOP-4412开发板之全功能版使用手册_v1.0.pdf
- SEC_Exynos 4412 SCP_Users Manual_Ver.0.10.00_Preliminary.pdf
- 讯为uboot环境变量.txt

扫描右侧二维码
回复arm即可以获取

关注公众号：一口Linux 回复：arm 获取视频中所有资料

ARM

02

系统移植
环境搭建

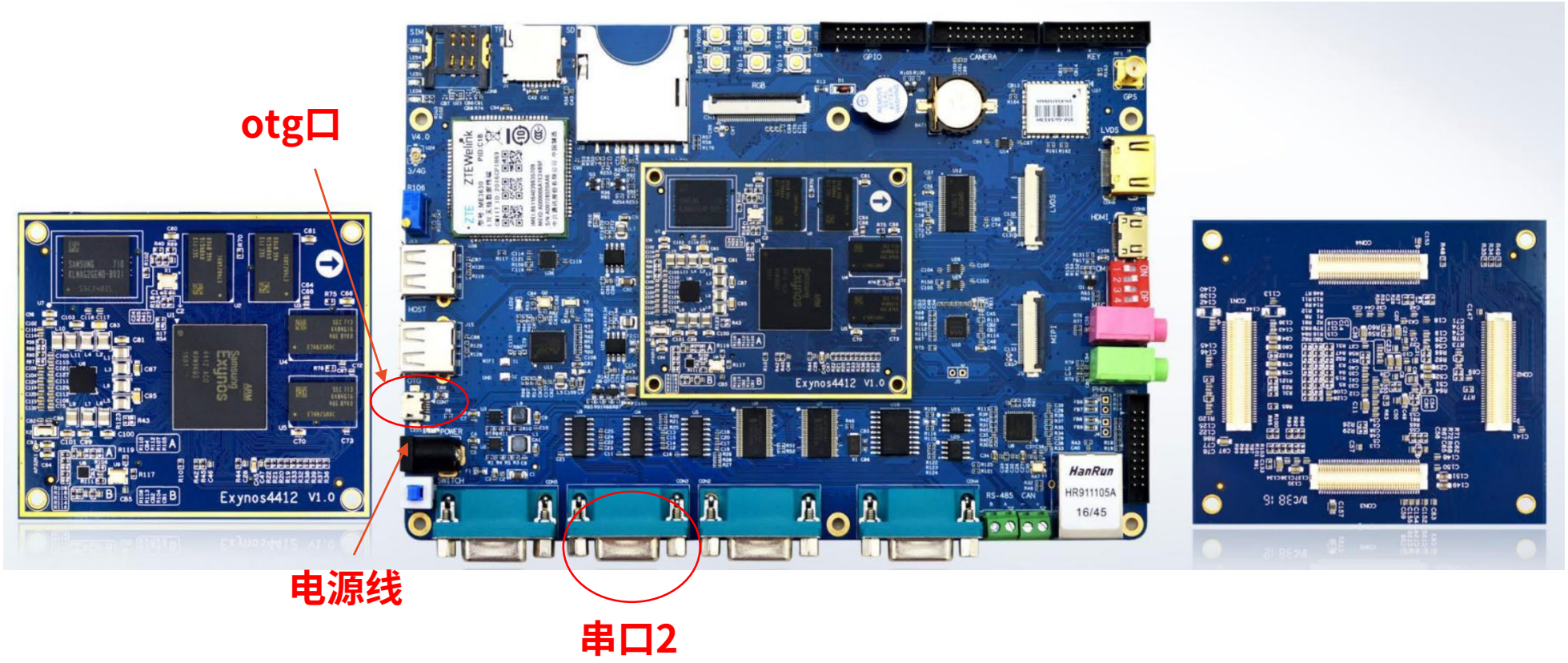
本节内容

- 1. 讯为开发板简介
- 2. 串口驱动安装
- 3. 安卓驱动安装
- 4. fastboot工具
- 5. 交叉工具链
- 6.制作文件系统工具

1.讯为4412全能板简介

讯为4412全能板

- 开发只需要连接这3根线

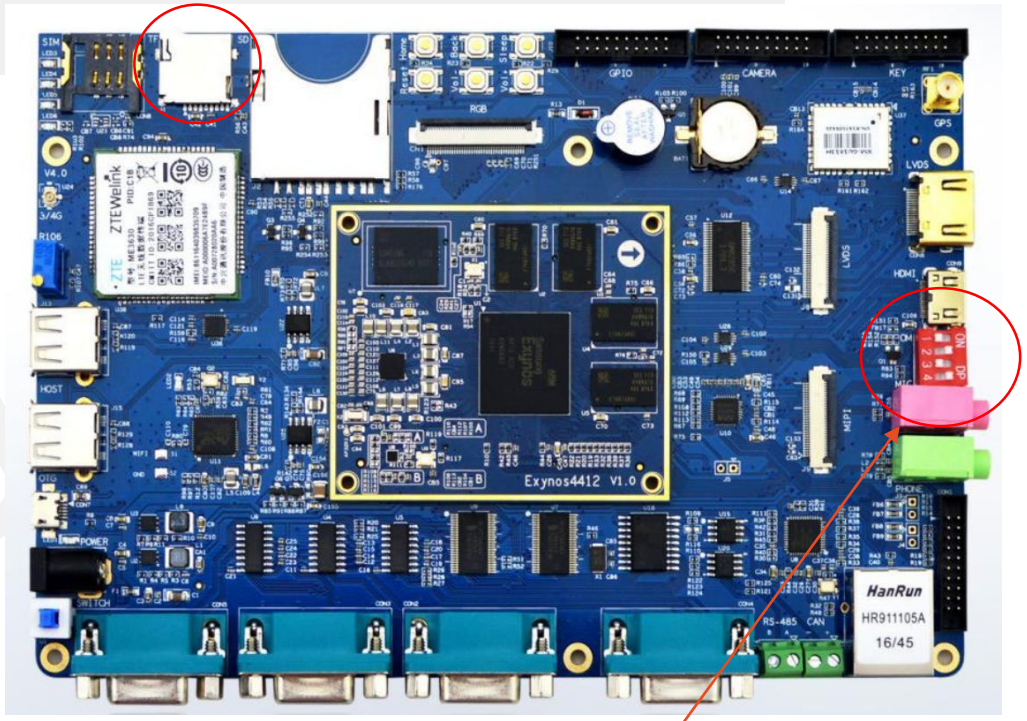


关注公众号：一口Linux 回复：arm 获取视频中所有资料

开发板启动-拨码开关

拨码开关编号	1	2
EMMC启动	0	1
TF卡启动	1	0

拨码开关编号	3	4	分辨率
9.7 寸屏幕	0	0	1024*768
塑胶壳 7 寸屏幕	0	1	1280*800
4.3 寸屏幕	1	0	480*272
金属框 7 寸屏幕	1	1	1024*600
10.1 寸屏幕	1	1	1024*600
HDMI 屏幕	1	1	1080P

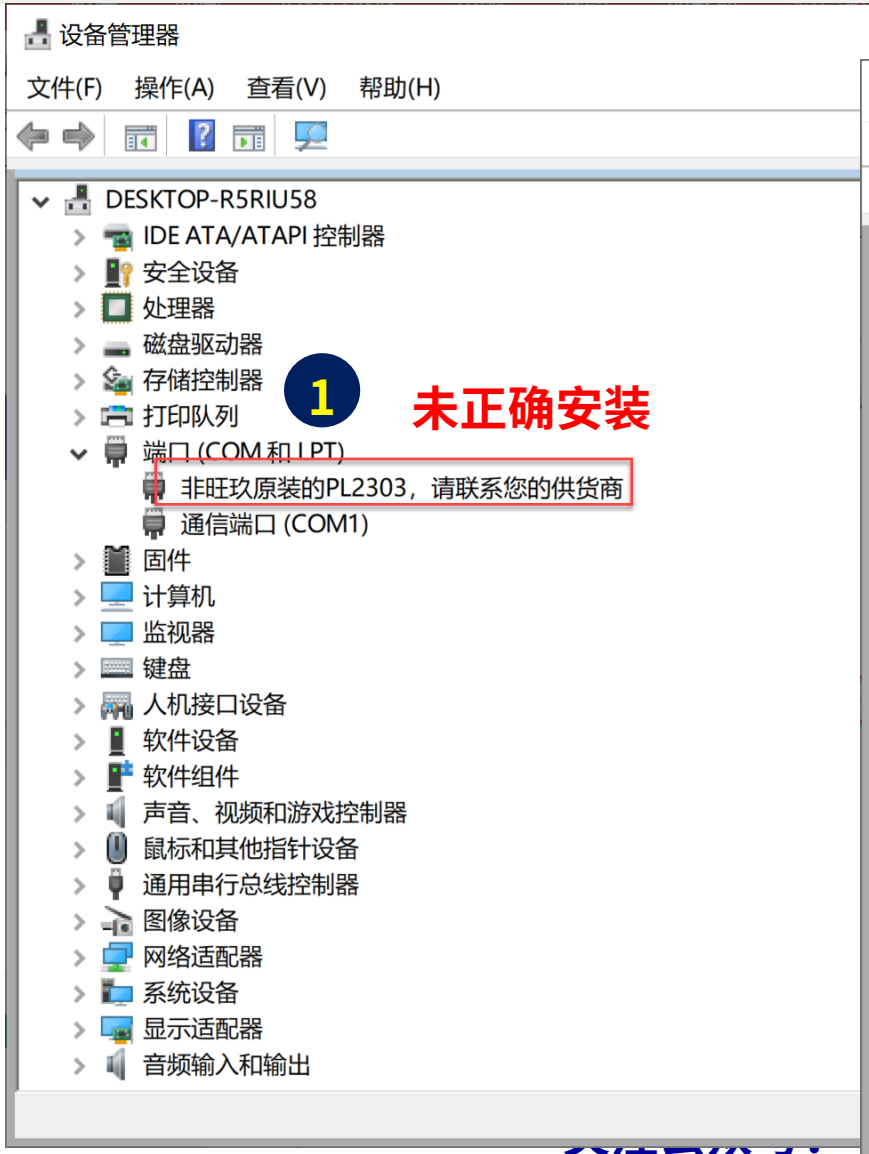


HDMI 默认输出是 1024*600， 如果需要支持 HDMI 输出 1080P 分辨率，需要修改内核

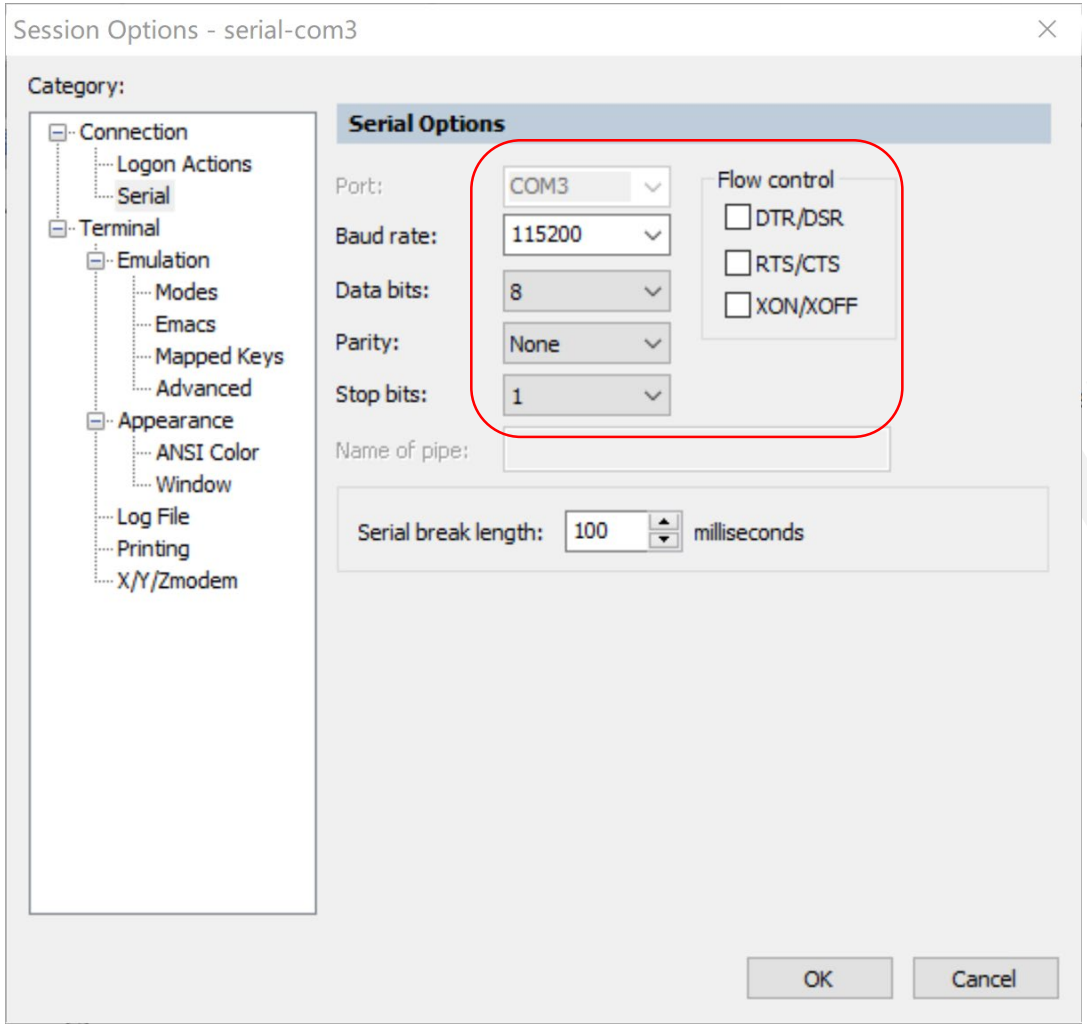
2.串口驱动

串口驱动安装

USB驱动\win10串口驱动



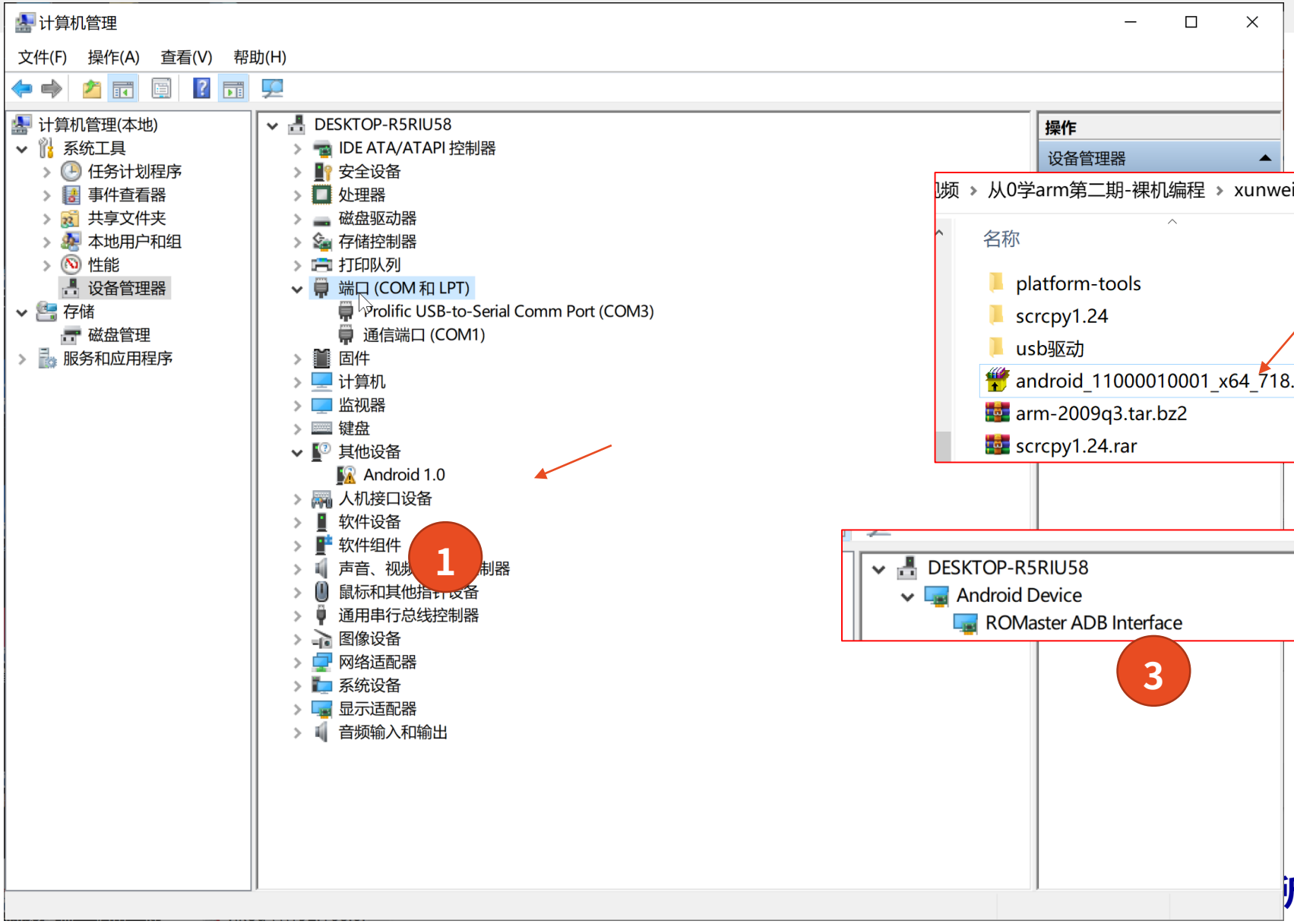
串口配置



3. 安卓驱动

安卓驱动安装

xunwei\工具软件\android_11000010001_x64_718.exe



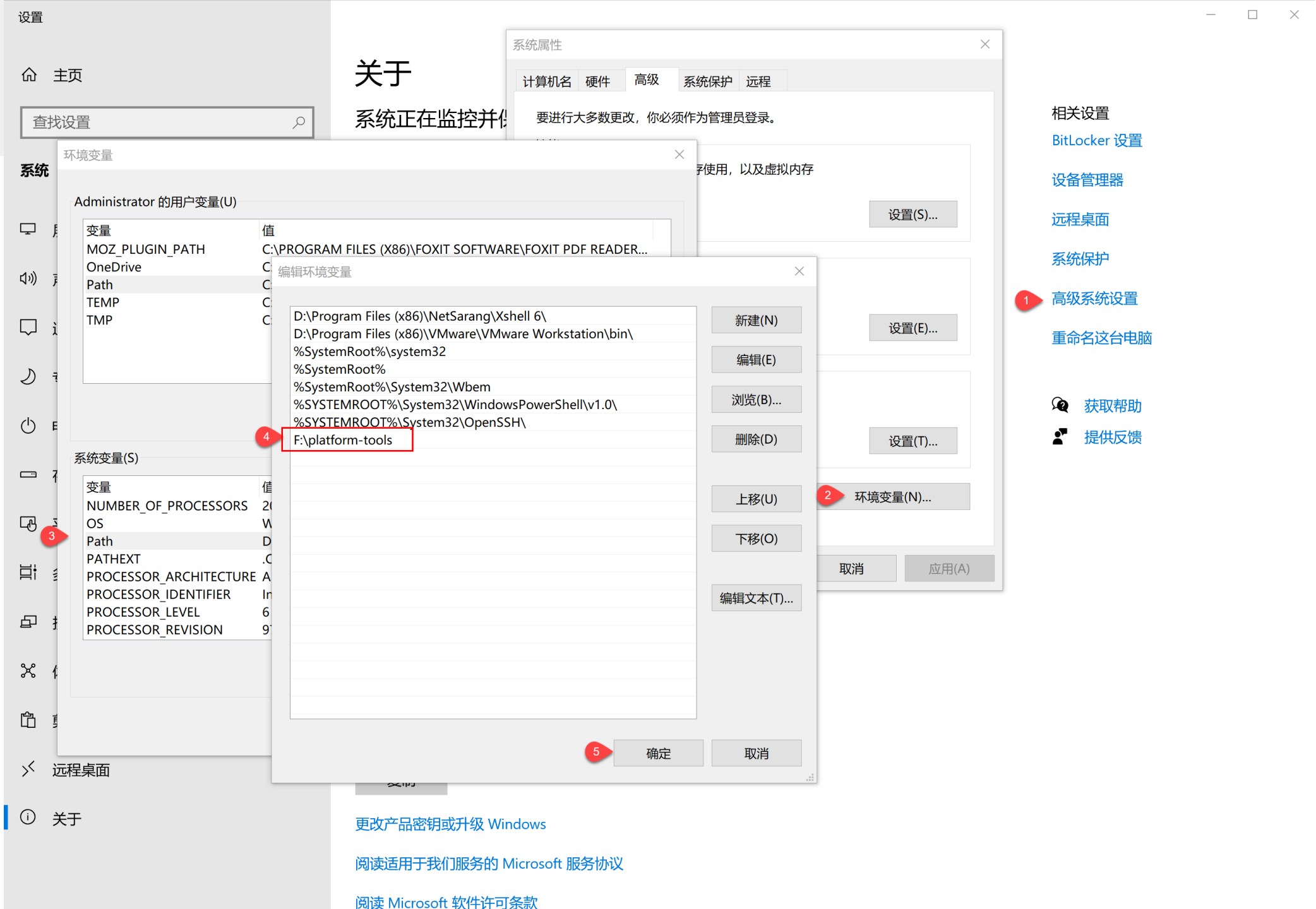
所有资料

也可以用其他串口线

安装Android手机驱动

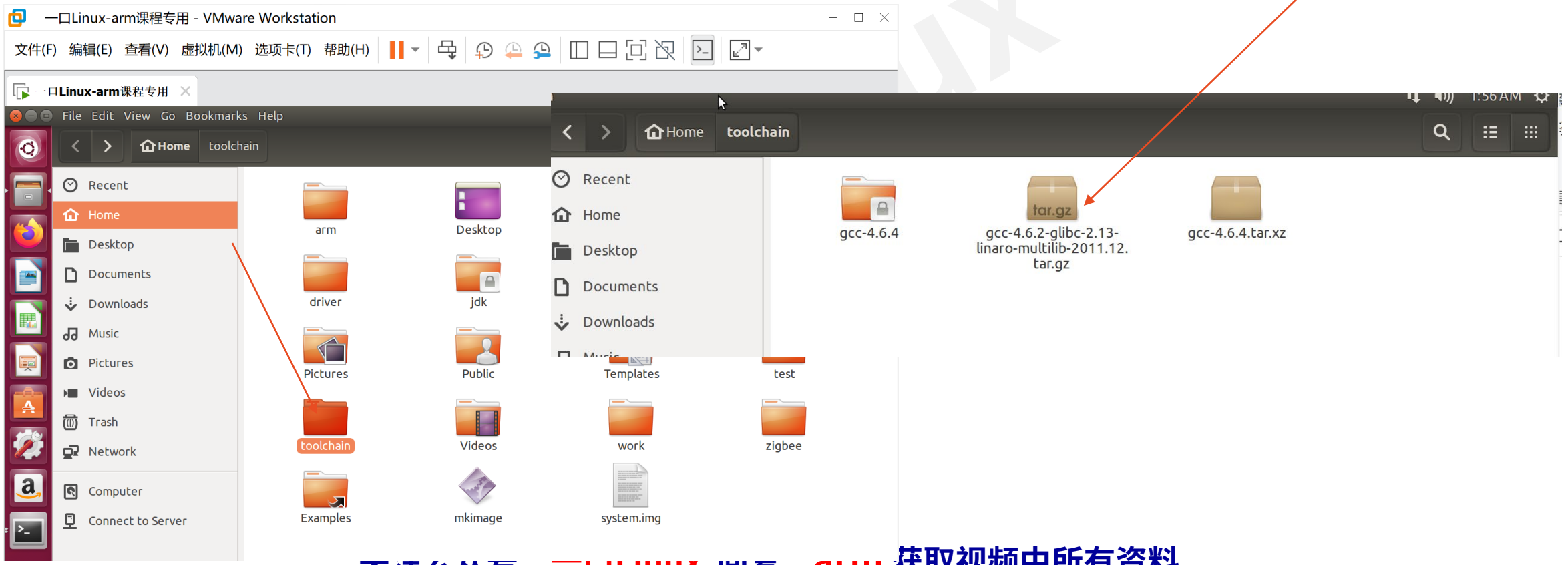
4. fastboot工具链

- xunwei\工具软件\USB_fastboot_tool\platform-tools


设置
主页
查找设置
系统
远程桌面
关于
关于
系统正在监控并保
更改产品密钥或升级 Windows
阅读适用于我们服务的 Microsoft 服务协议
阅读 Microsoft 软件许可条款
相关设置
BitLocker 设置
设备管理器
远程桌面
系统保护
1
高级系统设置
重命名这台电脑
获取帮助
提供反馈
系统属性
计算机名
硬件
高级
系统保护
远程
要进行大多数更改，你必须作为管理员登录。
字使用，以及虚拟内存
设置(S)...
设置(E)...
设置(T)...
2
环境变量(N)...
取消
应用(A)
环境变量
Administrator 的用户变量(U)
变量
值
MOZ_PLUGIN_PATH
C:\PROGRAM FILES (X86)\FOXIT SOFTWARE\FOXIT PDF READER...
OneDrive
Path
TEMP
TMP
系统变量(S)
变量
NUMBER_OF_PROCESSORS
OS
3
Path
PATHEXT
PROCESSOR_ARCHITECTURE
PROCESSOR_IDENTIFIER
PROCESSOR_LEVEL
PROCESSOR_REVISION
编辑环境变量
D:\Program Files (x86)\NetSarang\Xshell 6\
D:\Program Files (x86)\VMware\VMware Workstation\bin\
%SystemRoot%\system32
%SystemRoot%
%SystemRoot%\System32\Wbem
%SYSTEMROOT%\System32\WindowsPowerShell\v1.0\
%SYSTEMROOT%\System32\OpenSSH\
4
F:\platform-tools
新建(N)
编辑(E)
浏览(B)...
删除(D)
上移(U)
下移(O)
编辑文本(T)...
5
确定
取消

5. 交叉编译工具链

- **工具链**
  - xunwei\工具软件\ gcc-4.6.2-glibc-2.13-linaro-multilib-2011.12.tar.gz



解压缩

```
peng@ubuntu:~/toolchain$ tar -zxvf gcc-4.6.2-glibc-2.13-linaro-multilib-2011.12.tar.gz
```

修改配置文件

```
peng@ubuntu:~/toolchain$ sudo vim /etc/bash.bashrc
```

```
65              return 127
66          fi
67      }
68 fi
69 #export PATH=$PATH:/home/peng/toolchain/gcc-4.6.4/bin
70 export PATH=$PATH:/home/peng/toolchain/gcc-4.6.2-glibc-2.13-linaro-multilib-2011.12/fsl-linaro-toolchain/bin
/etc/bash.bashrc                                                    70,1          Bot
```

export PATH=$PATH:/home/peng/toolchain/gcc-4.6.2-glibc-2.13-linaro-multilib-2011.12/fsl-linaro-toolchain/bin

测试

```
peng@ubuntu:~/toolchain$ source /etc/bash.bashrc
peng@ubuntu:~/toolchain$ arm-none-linux-gnueabi-gcc -v
Using built-in specs.
COLLECT_GCC=arm-none-linux-gnueabi-gcc
COLLECT_LTO_WRAPPER=/home/peng/toolchain/gcc-4.6.2-glibc-2.13-linaro-multilib-2011.12/fsl-linaro-toolchain/bin/../li
bexec/gcc/arm-fsl-linux-gnueabi/4.6.2/lto-wrapper
Target: arm-fsl-linux-gnueabi
Configured with: /work/build/.build/src/gcc-linaro-4.6-2011.06-0/configure --build=i686-build_pc-linux-gnu --host=i6
86-build_pc-linux-gnu --target=arm-fsl-linux-gnueabi --prefix=/work/fsl-linaro-toolchain-2.13 --with-sysroot=/work/f
sl-linaro-toolchain-2.13/arm-fsl-linux-gnueabi/multi-libs --enable-languages=c,c++ --with-pkgversion='Freescale MAD
-- Linaro 2011.07 -- Built at 2011/08/10 09:20' --enable-__cxa_atexit --disable-libmudflap --disable-libgomp --disab
le-libssp --with-gmp=/work/build/.build/arm-fsl-linux-gnueabi/build/static --with-mpfr=/work/build/.build/arm-fsl-li
nux-gnueabi/build/static --with-mpc=/work/build/.build/arm-fsl-linux-gnueabi/build/static --with-ppl=/work/build/.bu
ild/arm-fsl-linux-gnueabi/build/static --with-cloog=/work/build/.build/arm-fsl-linux-gnueabi/build/static --with-lib
elf=/work/build/.build/arm-fsl-linux-gnueabi/build/static --with-host-libstdcxx='-static-libgcc -Wl,-Bstatic,-lstdc+
+,-Bdynamic -lm -L/work/build/.build/arm-fsl-linux-gnueabi/build/static/lib -lpwl' --enable-threads=posix --enable-t
arget-optspace --enable-plugin --enable-multilib --with-local-prefix=/work/fsl-linaro-toolchain-2.13/arm-fsl-linux-g
nueabi/multi-libs --disable-nls --enable-c99 --enable-long-long --with-system-zlib
Thread model: posix
gcc version 4.6.2 20110630 (prerelease) (Freescale MAD -- Linaro 2011.07 -- Built at 2011/08/10 09:20)
```

6. 制作文件系统工具

- xunwei\工具软件\linux_tools.tgz
  - mkimage
  - make_ext4fs

ARM

03

开发板如何烧录镜像

uboot基础

uboot基础

1. Bootloader

- Bootloader是在操作系统运行之前执行的一段小程序。
- 通过这段小程序，我们可以初始化硬件设备、建立内存空间的映射表，从而建立适当的系统软硬件环境，为最终调用操作系统内核做好准备。

Bootloader种类

Bootloader	描述	x86	ARM	PowerPC
LILO	Linux 磁盘引导程	是	否	否
GRUB	GNU 的 LILO 替代程	是	否	否
Loadlin	从 DOS 引导 Linux	是	否	否
ROLO	从 ROM 引导 Linux 而不需要 BIOS	是	否	否
Etherboot	通过以太网卡启动 Linux 系统的固件	是	否	否
LinuxBIOS	完全替代 BUIS 的 Linux 引导程序	是	否	否
BLOB	LART 等硬件平台的引导程序	否	是	否
U-boot	通用引导程序	是	是	是
RedBoot	基于 eCos 的引导程序	是	是	是

2. uboot

- U-Boot，全称 Universal Boot Loader，是遵循GPL条款的开放源码项目，是一套在GNU通用公共许可证之下发布的自由软件。

3. Uboot对硬件平台的支持

- U-Boot是一个主要用于嵌入式系统的引导加载程序
- 可以支持多种不同的计算机系统结构，包括
  - PPC、ARM、AVR32、MIPS、x86、68k、Nios与MicroBlaze等诸多常用系列的处理器。

4. Uboot对os的支持

- 在操作系统方面，U-Boot不仅支持嵌入式Linux系统的引导。
  - 目前支持的目标操作系统有OpenBSD、NetBSD、FreeBSD、4.4BSD、Linux、SVR4、Esix、Solaris、Irix、SCO、Dell、NCR、VxWorks、LynxOS、pSOS、QNX、RTEMS、ARTOS、Android。

5. 嵌入式系统启动流程

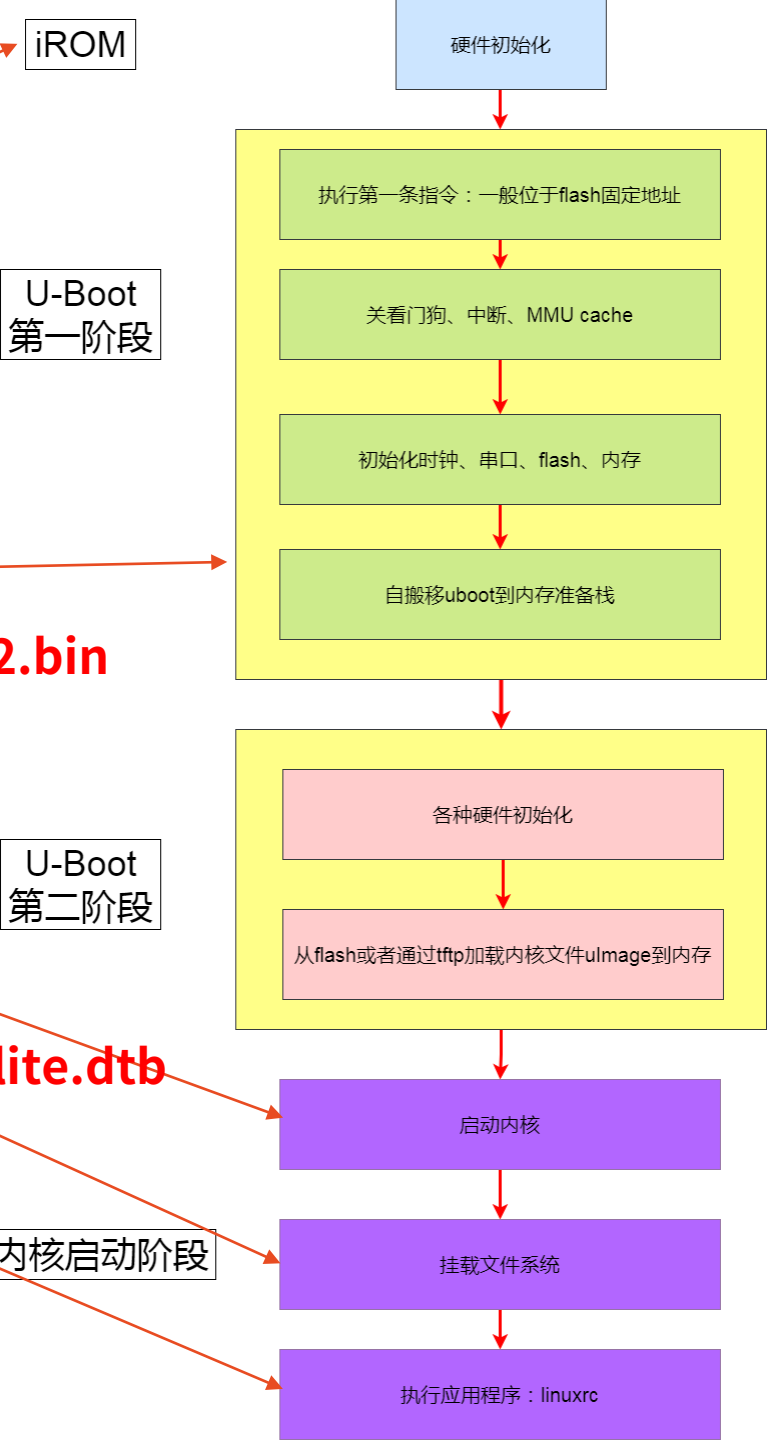
1)设备上电之后，先执行iROM中的出厂代码(启动设备选择)；

2)执行U-Boot

3)启动Linux内核；

4)挂载文件系统，如：rootfs；

5)然后执行应用程序linuxrc。



6. 存储镜像有哪些外设?

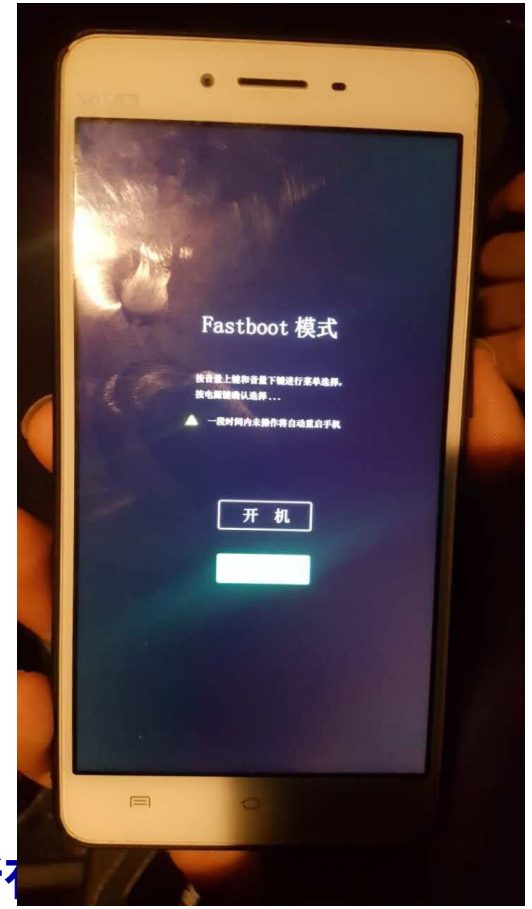
- 1. nand flash
  - nand
- 2. emmc/SD
  - mmc
- 3. qspi flash
  - sf
- 4. 没有外设存储
  - 网络tftp、fastboot

7. 如何拷贝镜像到开发板?

- 1. 网络
  - tftp
- 2. 串口
  - loadb + kermit
- 3. usb
  - Fastboot
- 4. JLink/Jtag
- 5. tf卡
  - fatload

8.什么是fastboot?

- Fastboot协议是一种通过USB连接与bootloader通讯的机制。
- 适用于Linux、Windows或者macOS等多种平台。



```
λ fastboot.exe
usage: fastboot [ <option> ] <command>

commands:
  update <filename>                        reflash device from update.zip
  flashall                                 flash boot + recovery + system
  flash <partition> [ <filename> ]         write a file to a flash partition
  erase <partition>                        erase a flash partition
  getvar <variable>                        display a bootloader variable
  boot <kernel> [ <ramdisk> ]              download and boot kernel
  flash:raw boot <kernel> [ <ramdisk> ]    create bootimage and flash it
  devices                                  list all connected devices
  continue                                 continue with autoboot
  reboot                                   reboot device normally
  reboot-bootloader                        reboot device into bootloader
  help                                     show this help message

options:
  -w                                       erase userdata and cache
  -s <serial number>                       specify device serial number
  -p <product>                             specify product name
  -c <cmdline>                             override kernel commandline
  -i <vendor id>                           specify a custom USB vendor id
  -b <base_addr>                           specify a custom kernel base address
  -n <page size>                           specify the nand page size. default: 2048
```

```
u-boot # fastboot
fastboot - use USB Fastboot protocol

Usage:
fastboot <USB_controller>
    - run as a fastboot usb device
```

fatboot烧写命令

- 开发板uboot命令行
  - fastboot 0
- Windows命令行
  - fastboot.exe flash bootloader u-boot-iTOP-4412.bin
  - fastboot.exe flash kernel uImage
  - fastboot.exe flash dtb exynos4412-itop-elite.dtb
  - fastboot.exe flash system system.img
  - fastboot.exe flash ramdisk ramdisk-uboot.img

Uboot 烧写操作

1.讯为Uboot版本说明

- Uboot 2010　　讯为出厂默认是uboot2010
  - 不支持设备树
  - 烧录文件包括：
    - Uboot、zImage、ramdisk-uboot.img 、system.img
- Uboot 2017
  - 支持设备树
  - 烧录文件包括：
    - Uboot、uImage、dtb、system.img

2. Uboot2010 – 烧写成uboot 2017

serial-com5 - SecureCRT

File Edit View Options Transfer Script Tools Window Help

Enter host <Alt+R>

Session Manager

Filter by session name <A

Sessions
- 192.168.0.113
- 192.168.0.113 (1)
- 192.168.0.113 (2)
- serial-com3
- serial-com4
- serial-com5
- serial-com6
- serial-com7

serial-com5

```
EMMC clock output: 40000 KHz
max_emmc_clock:40 MHZ
Input CLK [ 50 MHz] is higher than limit [40 MHZ]
Set CLK to 40000 KHz
EMMC clock output: 40000 KHz
MMC0:    14910 MB
SD sclk_mmc is 400K HZ
raise: Signal # 8 caught
raise: Signal # 8 caught
MMC1:    0 MB
0 MB
*** Warning - using default environment

In:    serial
Out:   serial
Err:   serial
eMMC OPEN Success.!!
                        !!!Notice!!!
!You must close eMMC boot Partition after all image writing!
!eMMC boot partition has continuity at image writing time.!
!So, Do not close boot partition, Before, all images is written.!

MMC read: dev # 0, block # 48, count 16 ...16 blocks read: OK
eMMC CLOSE Success.!!


Checking Boot Mode ... EMMC4.41
SYSTEM ENTER NORMAL BOOT MODE
Hit any key to stop autoboot:  0
iTOP-4412 # fastboot
[Partition table on MoviNAND]
ptn 0 name='bootloader' start=0x0 len=N/A (use hard-coded info. (c
ptn 1 name='kernel' start=N/A len=N/A (use hard-coded info. (cmd:
ptn 2 name='ramdisk' start=N/A len=0x300000(~3072KB) (use hard-cod
 movi))
ptn 3 name='Recovery' start=N/A len=0x600000(~6144KB) (use hard-co
: movi))
ptn 4 name='system' start=0x1000000 len=0x40000000(~1048576KB)
ptn 5 name='userdata' start=0x41000000 len=0x40000000(~1048576KB)
ptn 6 name='cache' start=0x81000000 len=0x12C00000(~307200KB)
ptn 7 name='fat' start=0x93C00000 len=0xF800000(~253952KB)
Received 17 bytes: download:00076e56
Starting download of 486998 bytes

downloading of 486998 bytes finished
Received 16 bytes: flash:bootloader
flashing 'bootloader'
writing bootloader.. 0, 1038
eMMC OPEN Success.!!
                        !!!Notice!!!
!You must close eMMC boot Partition after all image writing!
!eMMC boot partition has continuity at image writing time.!
!So, Do not close boot partition, Before, all images is written.!

MMC write: dev # 0, block # 0, count 1038. 1038 blocks write finish
1038 blocks verify1: OK
eMMC CLOSE Success.!!
completed
partition 'bootloader' flashed
```

cmd

```
C:\Users\Administrator
λ fastboot.exe flash bootloader F:\录制视频\linux驱动第二期-讯为4412\xunwei\uboot+kernel\uboot-2017-tree\u-boot
-iTOP-4412.bin
 sending 'bootloader' (475 KB)... OKAY [  0.081s]
          writing 'bootloader'... OKAY [  0.193s]
finished. total time: 0.276s

C:\Users\Administrator
λ
```

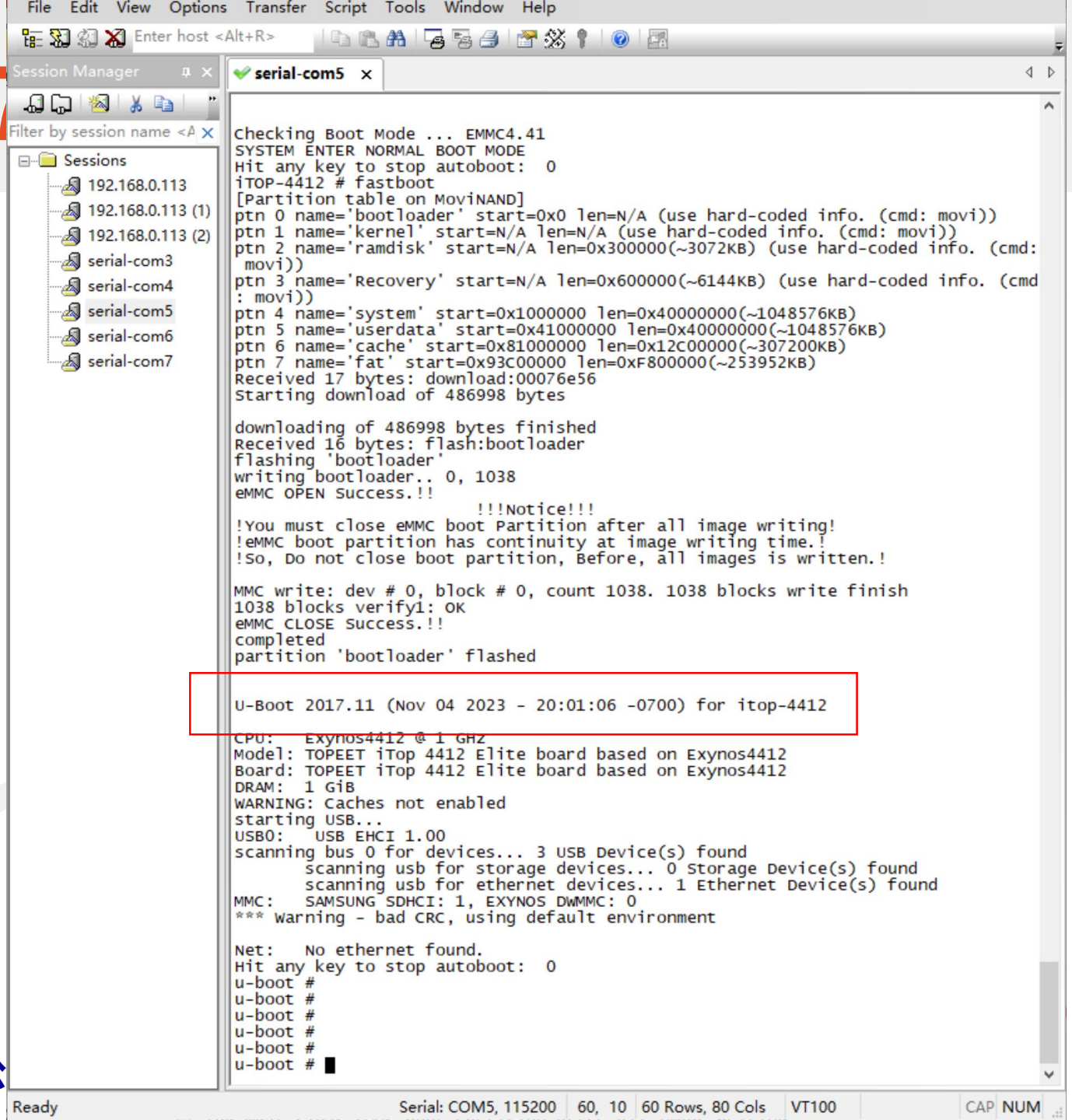
拨码：01xx

cmd.exe

Search

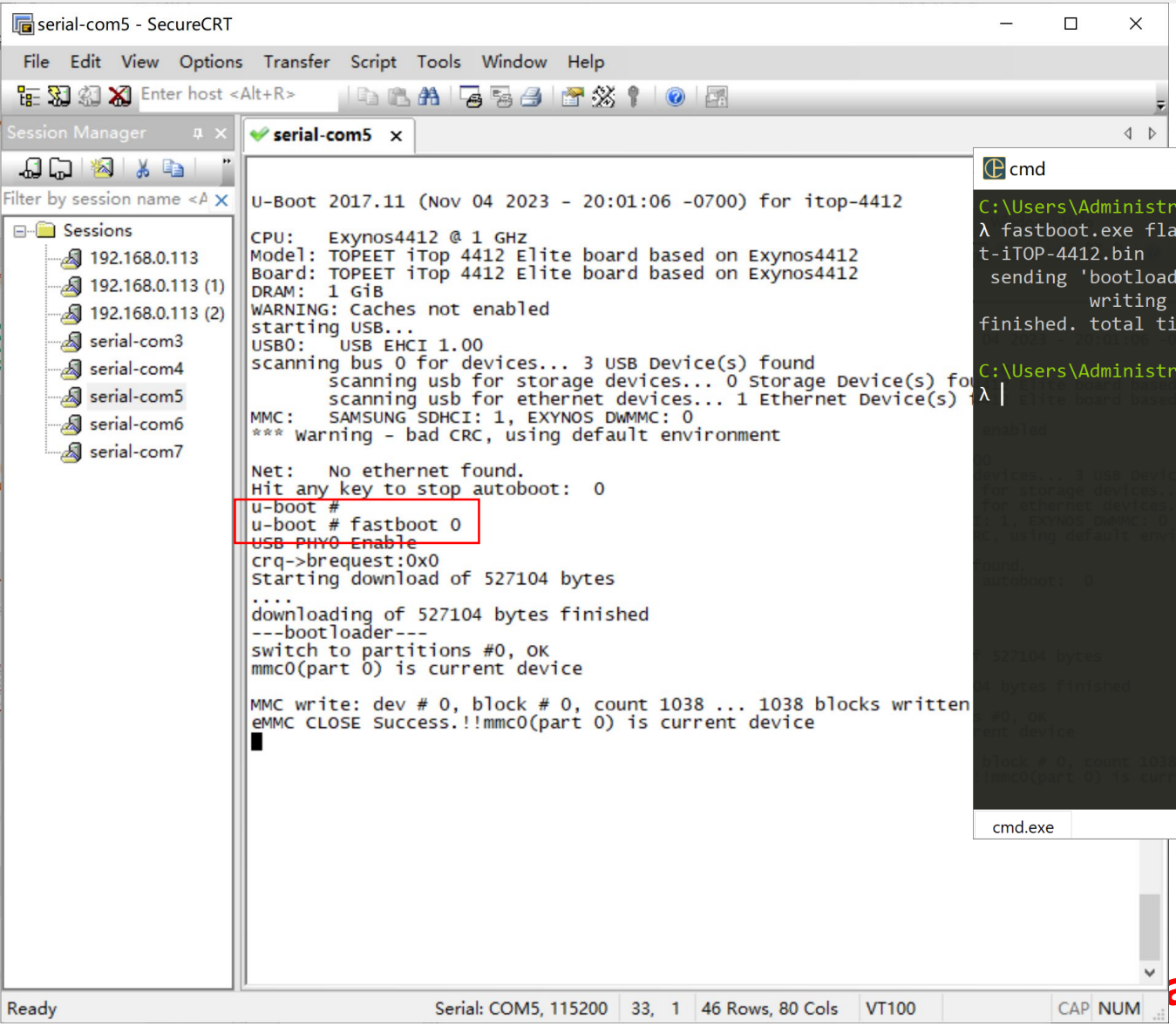
ux 回复：arm 获取视频中所有资料

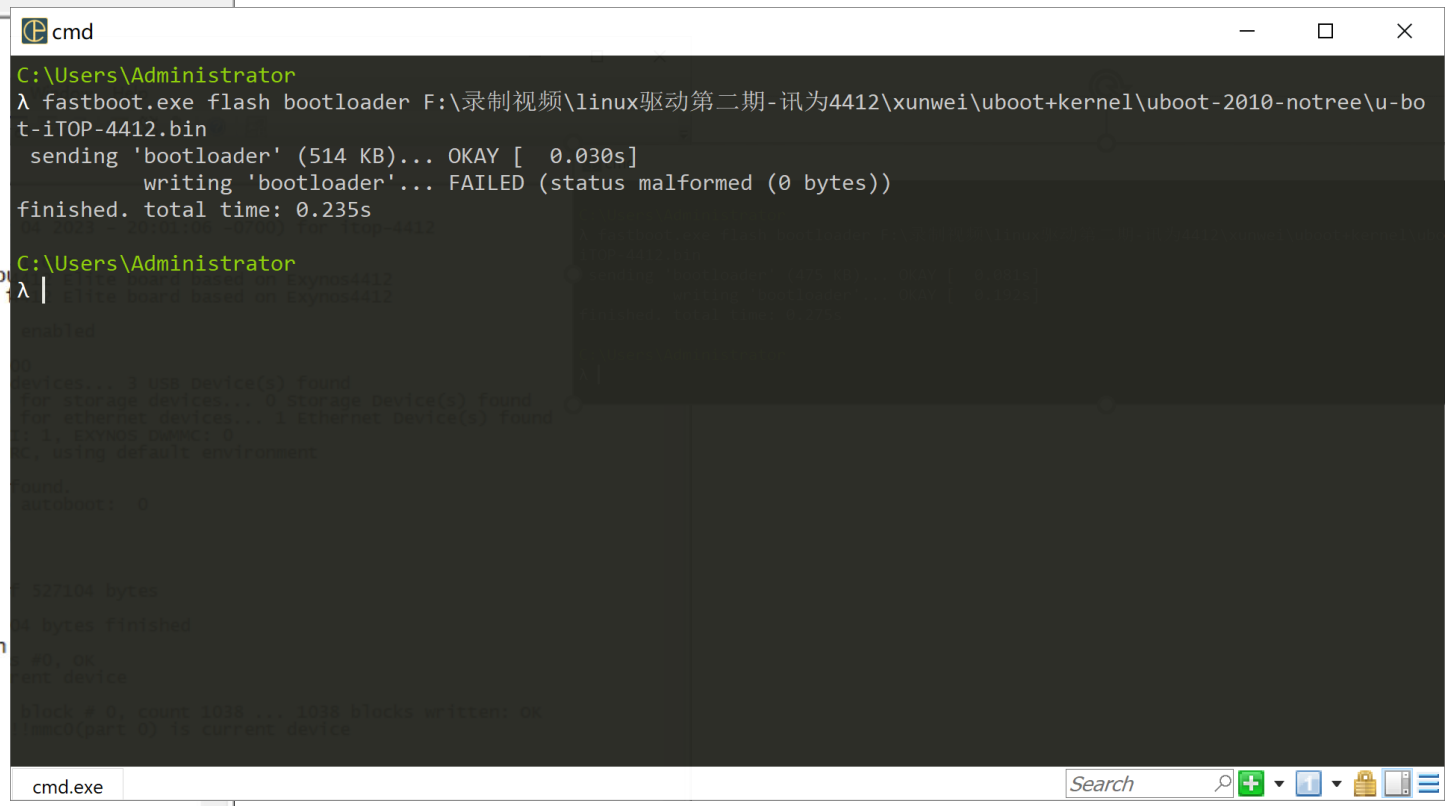
Uboot 2017



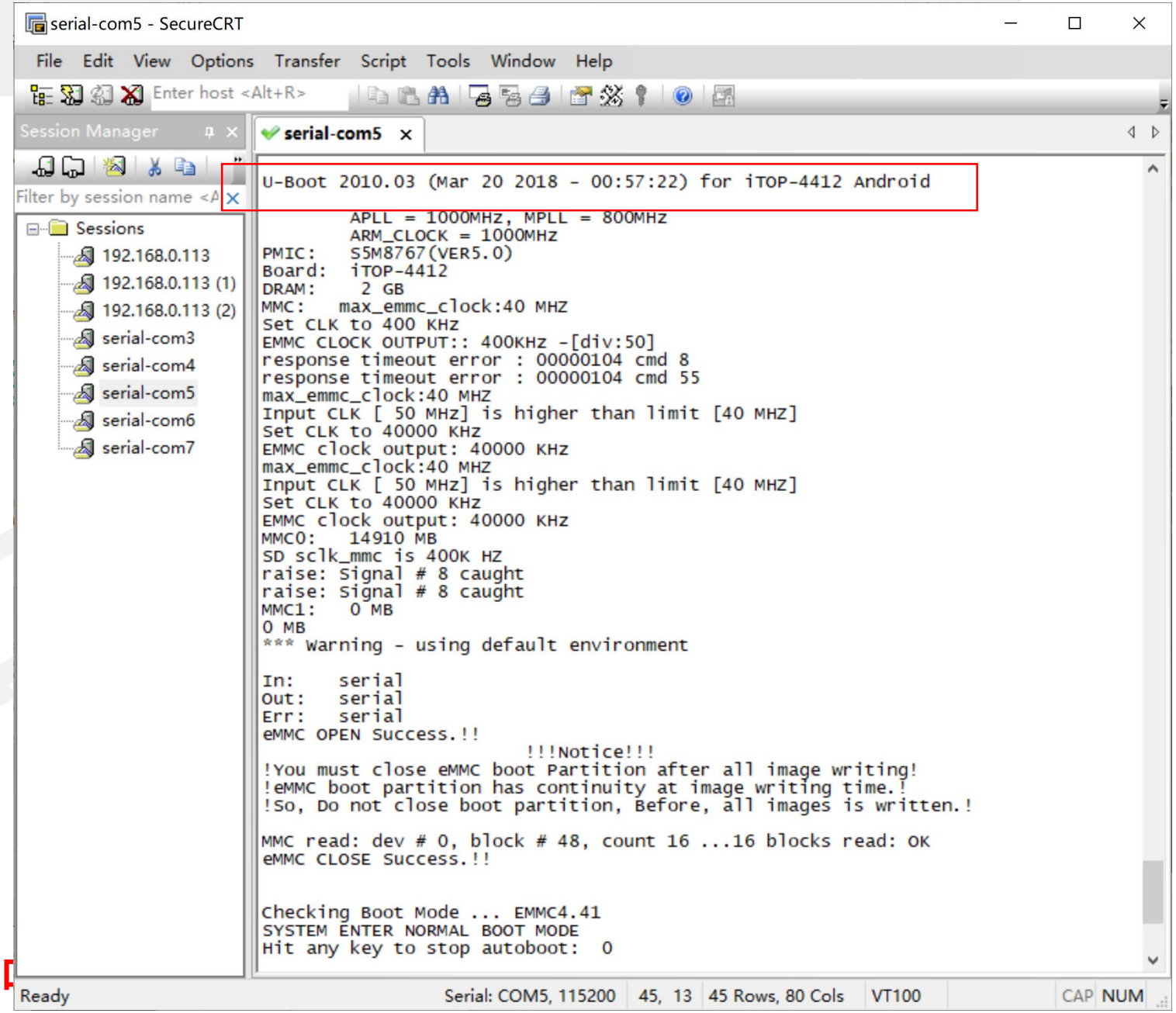
关注公

3. Uboot2017 – 烧写成uboot 2010





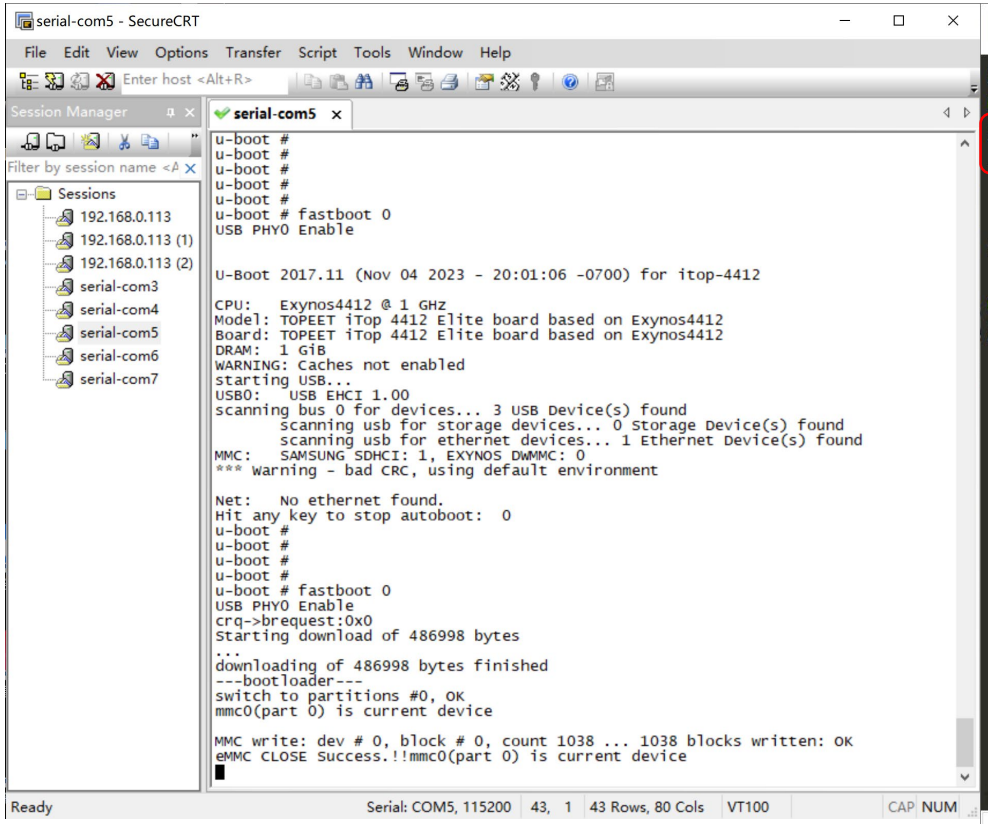
Uboot 2010

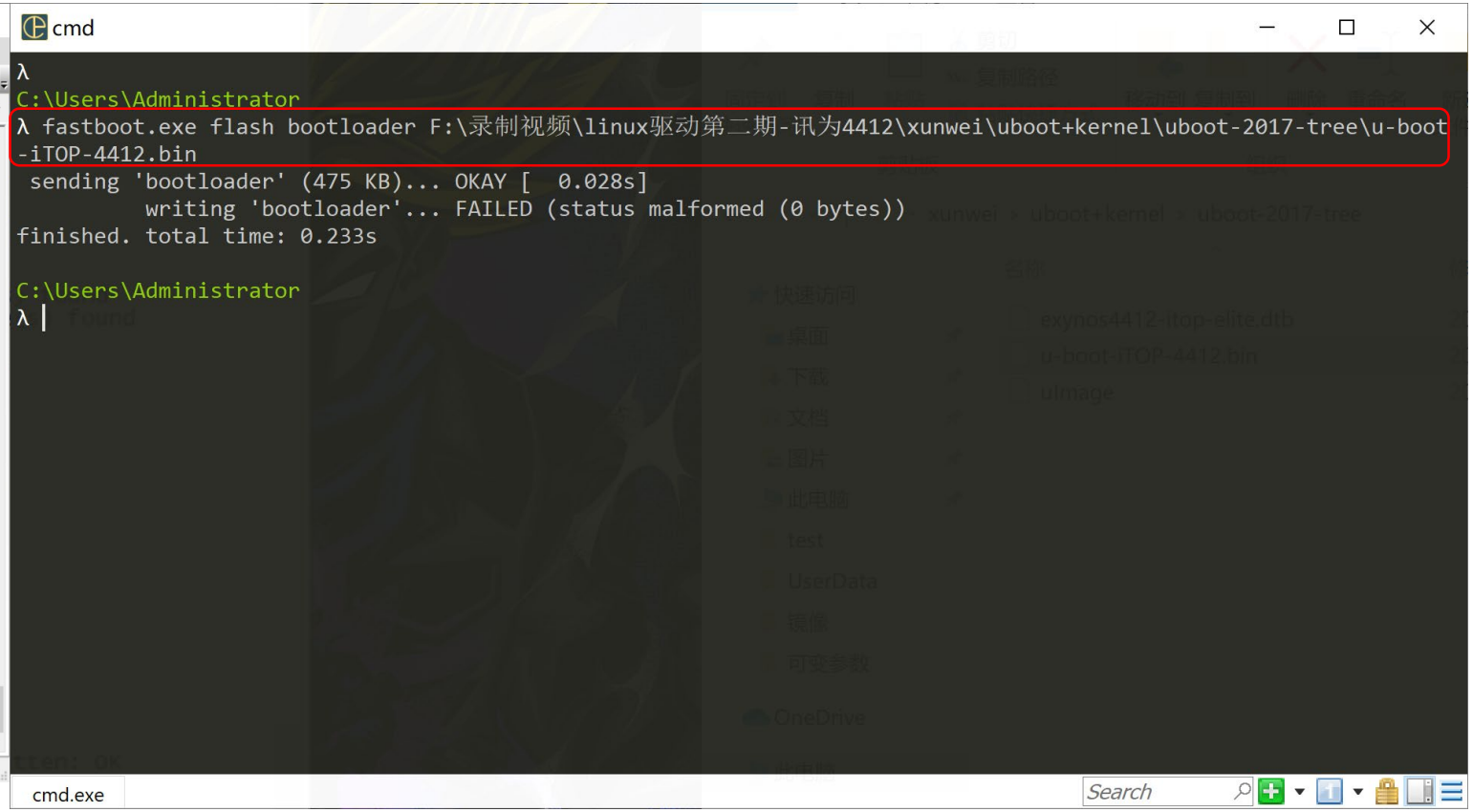


uboot 2017
通过fastboot烧写镜像

1.烧录uboot

通常不用重复烧录





2.烧录kernel

cmd

```
 sending 'bootloader' (475 KB)... OKAY [  0.028s]
        writing 'bootloader'... FAILED (status malformed (0 bytes))
finished. total time: 0.233s

C:\Users\Administrator
λ fastboot.exe flash kernel F:\录制视频\linux驱动第二期-讯为4412\xunwei\uboot+kernel\uboot-2017-tree\uImage
   sending 'kernel' (5538 KB)... OKAY [  0.299s]
          writing 'kernel'... FAILED (status malformed (0 bytes))
finished. total time: 1.811s

C:\Users\Administrator
λ
```

cmd.exe

3.烧录设备树dtb

cmd

```
 sending 'bootloader' (475 KB)... OKAY [  0.028s]
        writing 'bootloader'... FAILED (status malformed (0 bytes))
finished. total time: 0.233s

C:\Users\Administrator
λ fastboot.exe flash kernel F:\录制视频\linux驱动第二期-讯为4412\xunwei\uboot+kernel\uboot-2017-tree\uImage
    sending 'kernel' (5538 KB)... OKAY [  0.299s]
           writing 'kernel'... FAILED (status malformed (0 bytes))
finished. total time: 1.811s

C:\Users\Administrator
λ fastboot.exe flash dtb F:\录制视频\linux驱动第二期-讯为4412\xunwei\uboot+kernel\uboot-2017-tree\exynos4412-it
p-elite.dtb
         sending 'dtb' (55 KB)... OKAY [  0.006s]
                writing 'dtb'... FAILED (status malformed (0 bytes))
finished. total time: 0.034s

C:\Users\Administrator
λ
```

cmd.exe

4.烧录system.img

Usb线必须接usb3.0

serial-com5

```
MMC write: dev # 0, block # 130672, count 40 ... 40 blocks written: OK
MMC write: dev # 0, block # 130712, count 40 ... 40 blocks written: OK
MMC write: dev # 0, block # 130752, count 40 ... 40 blocks written: OK
MMC write: dev # 0, block # 130792, count 40 ... 40 blocks written: OK
MMC write: dev # 0, block # 130832, count 40 ... 40 blocks written: OK
MMC write: dev # 0, block # 130872, count 40 ... 40 blocks written: OK
MMC write: dev # 0, block # 130912, count 40 ... 40 blocks written: OK
MMC write: dev # 0, block # 130952, count 40 ... 40 blocks written: OK
MMC write: dev # 0, block # 130992, count 24 ... 24 blocks written: OK
MMC write: dev # 0, block # 131016, count 24 ... 24 blocks written: OK
MMC write: dev # 0, block # 131040, count 24 ... 24 blocks written: OK
MMC write: dev # 0, block # 131064, count 24 ... 24 blocks written: OK
MMC write: dev # 0, block # 131088, count 64 ... 64 blocks written: OK
MMC write: dev # 0, block # 131152, count 64 ... 64 blocks written: OK
MMC write: dev # 0, block # 131216, count 64 ... 64 blocks written: OK
MMC write: dev # 0, block # 131280, count 64 ... 64 blocks written: OK
MMC write: dev # 0, block # 131344, count 40 ... 40 blocks written: OK
MMC write: dev # 0, block # 131384, count 40 ... 40 blocks written: OK
MMC write: dev # 0, block # 131424, count 40 ... 40 blocks written: OK
MMC write: dev # 0, block # 131464, count 40 ... 40 blocks written: OK
MMC write: dev # 0, block # 131504, count 112 ... 112 blocks written: OK
MMC write: dev # 0, block # 131616, count 112 ... 112 blocks written: OK
MMC write: dev # 0, block # 131728, count 112 ... 112 blocks written: OK
MMC write: dev # 0, block # 131840, count 112 ... 112 blocks written: OK
MMC write: dev # 0, block # 131952, count 2376 ... 2376 blocks written: OK
MMC write: dev # 0, block # 134328, count 2616 ... 2616 blocks written: OK
MMC write: dev # 0, block # 136944, count 2376 ... 2376 blocks written: OK
MMC write: dev # 0, block # 139320, count 2616 ... 2616 blocks written: OK
MMC write: dev # 0, block # 141936, count 480 ...
```

cmd

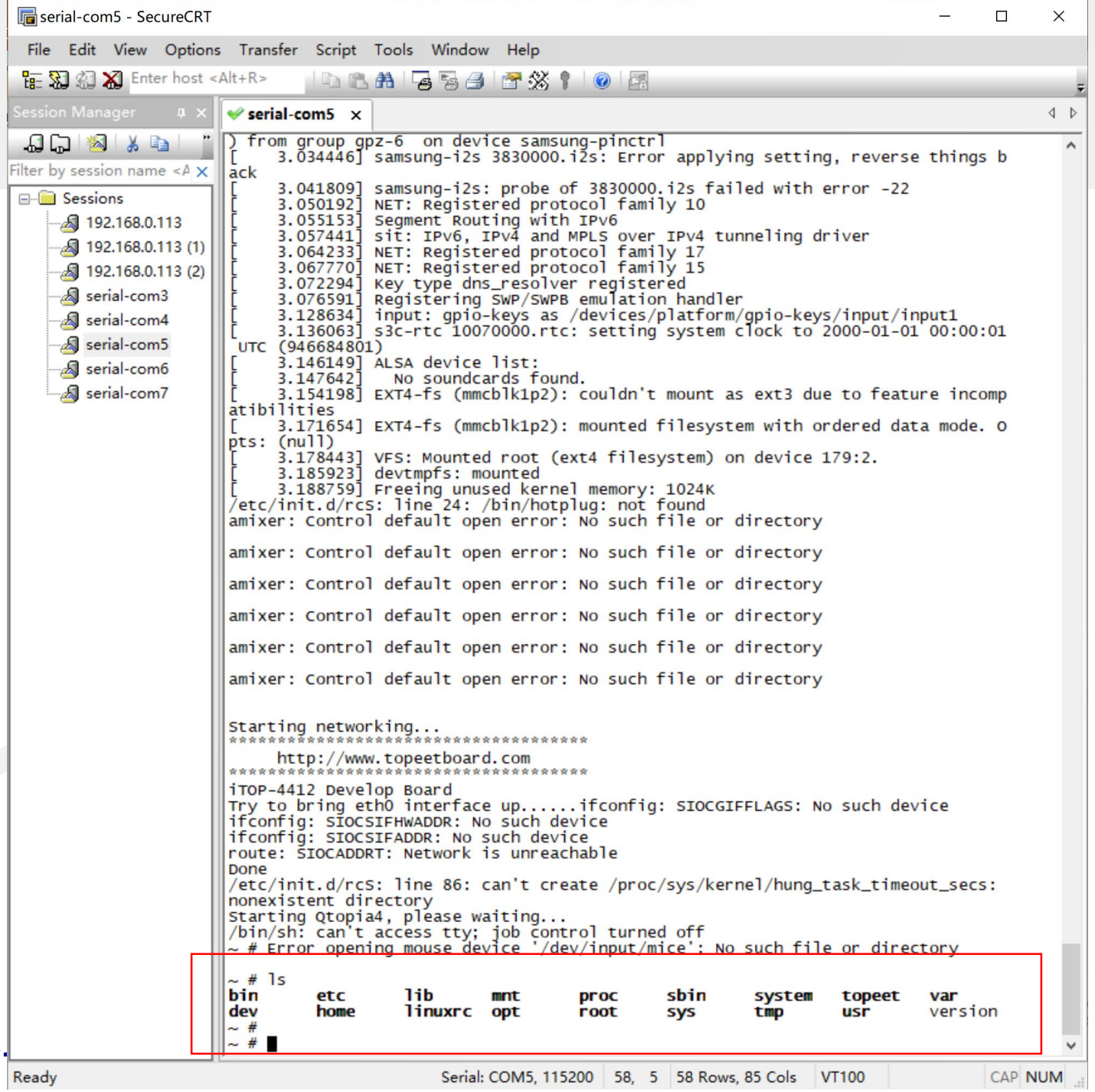
```
C:\Users\Administrator
λ
C:\Users\Administrator
λ fastboot.exe flash system F:\录制视频\linux驱动第二期-讯为4412\xunwei\uboot+kernel\system.img
  sending 'system' (204831 KB)... OKAY [ 11.020s]
          writing 'system'... FAILED (status malformed (0 bytes))
finished. total time: 119.797s

C:\Users\Administrator
λ
```

cmd.exe

Search

关注公众号：Linux 回复：arm 获取视频中所有资料

5.启动linux



关注公众

其他

其他镜像烧录

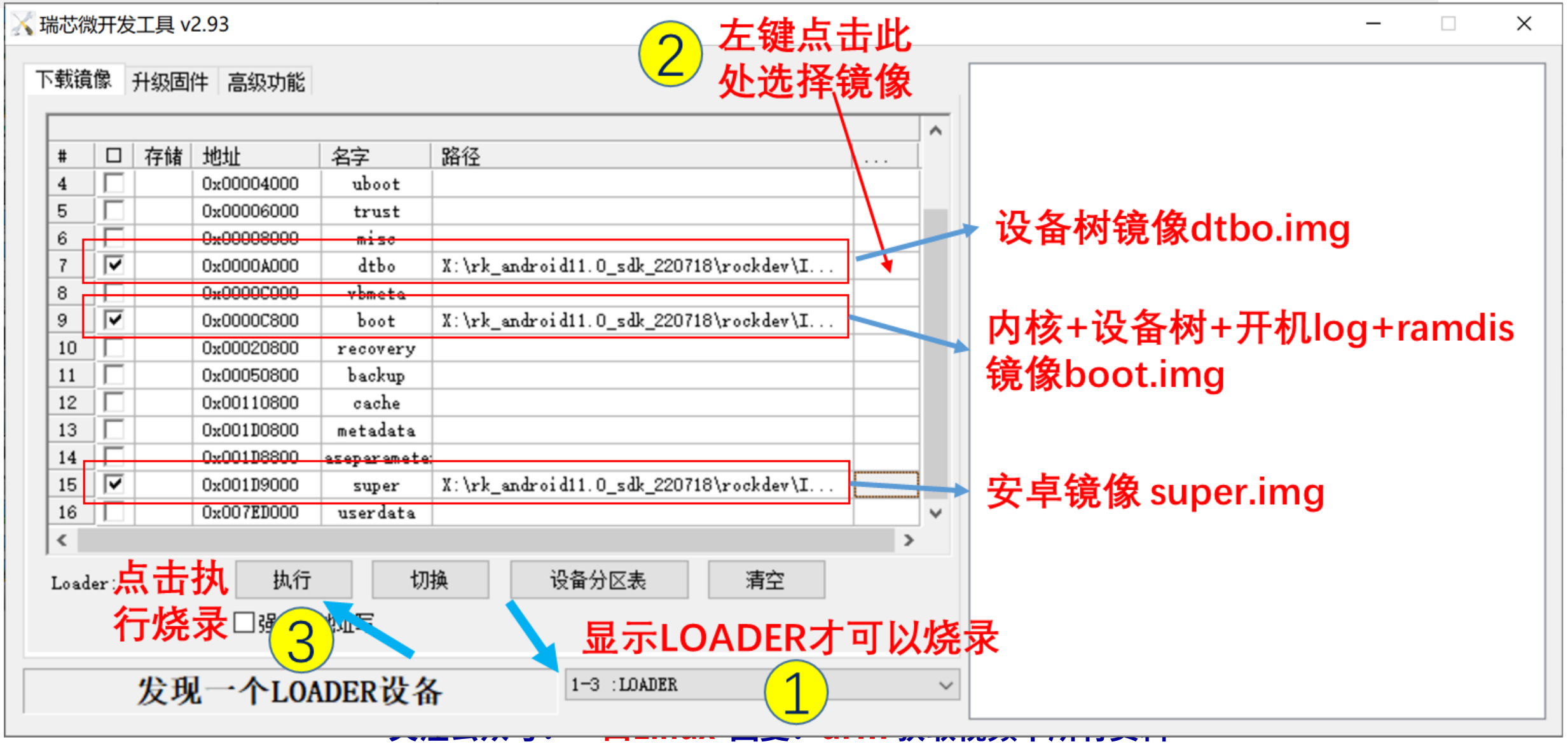
- 03_镜像_Android4.4.4文件系统
- 04_镜像_QT文件系统
- 05_镜像_Ubuntu文件系统

fastboot.exe flash ramdisk ramdisk-uboot.img

均不支持设备树

没有设备树的内核，必须用uboot2010启动

瑞芯微集成烧录工具

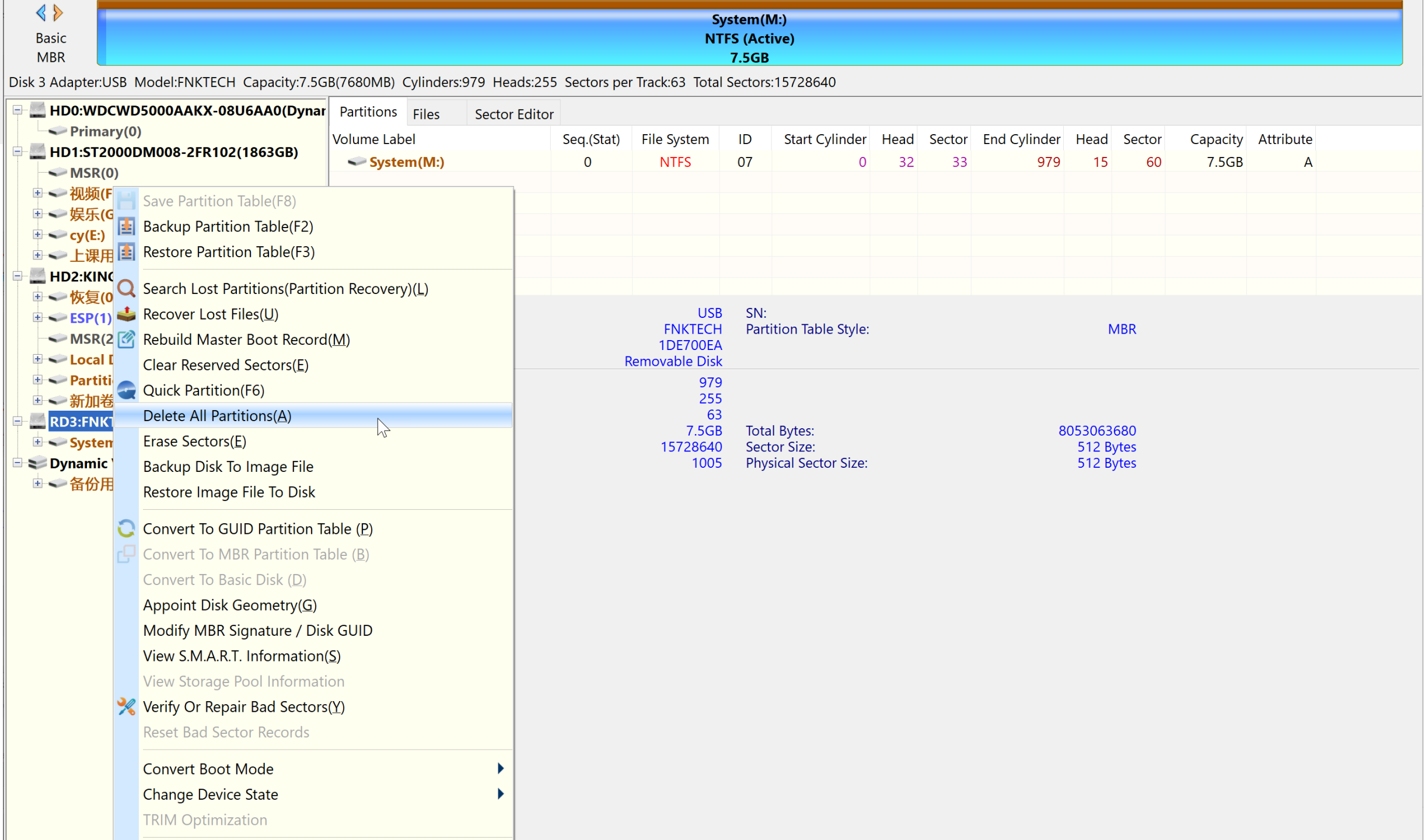

瑞芯微开发工具 v2.93
下载镜像
升级固件
高级功能
2
左键点击此处选择镜像
存储 地址 名字 路径
4 0x00004000 uboot
5 0x00006000 trust
6 0x00008000 misc
7 0x0000A000 dtbo X:\rk_android11.0_sdk_220718\rockdev\I...
8 0x0000C000 vbmeta
9 0x0000C800 boot X:\rk_android11.0_sdk_220718\rockdev\I...
10 0x00020800 recovery
11 0x00050800 backup
12 0x00110800 cache
13 0x001D0800 metadata
14 0x001D8800 aseperamete
15 0x001D9000 super X:\rk_android11.0_sdk_220718\rockdev\I...
16 0x007ED000 userdata
设备树镜像dtbo.img
内核+设备树+开机log+ramdis
镜像boot.img
安卓镜像 super.img
Loader:点击执行烧录
执行
切换
设备分区表
清空
3
显示LOADER才可以烧录
发现一个LOADER设备
1-3 :LOADER
1

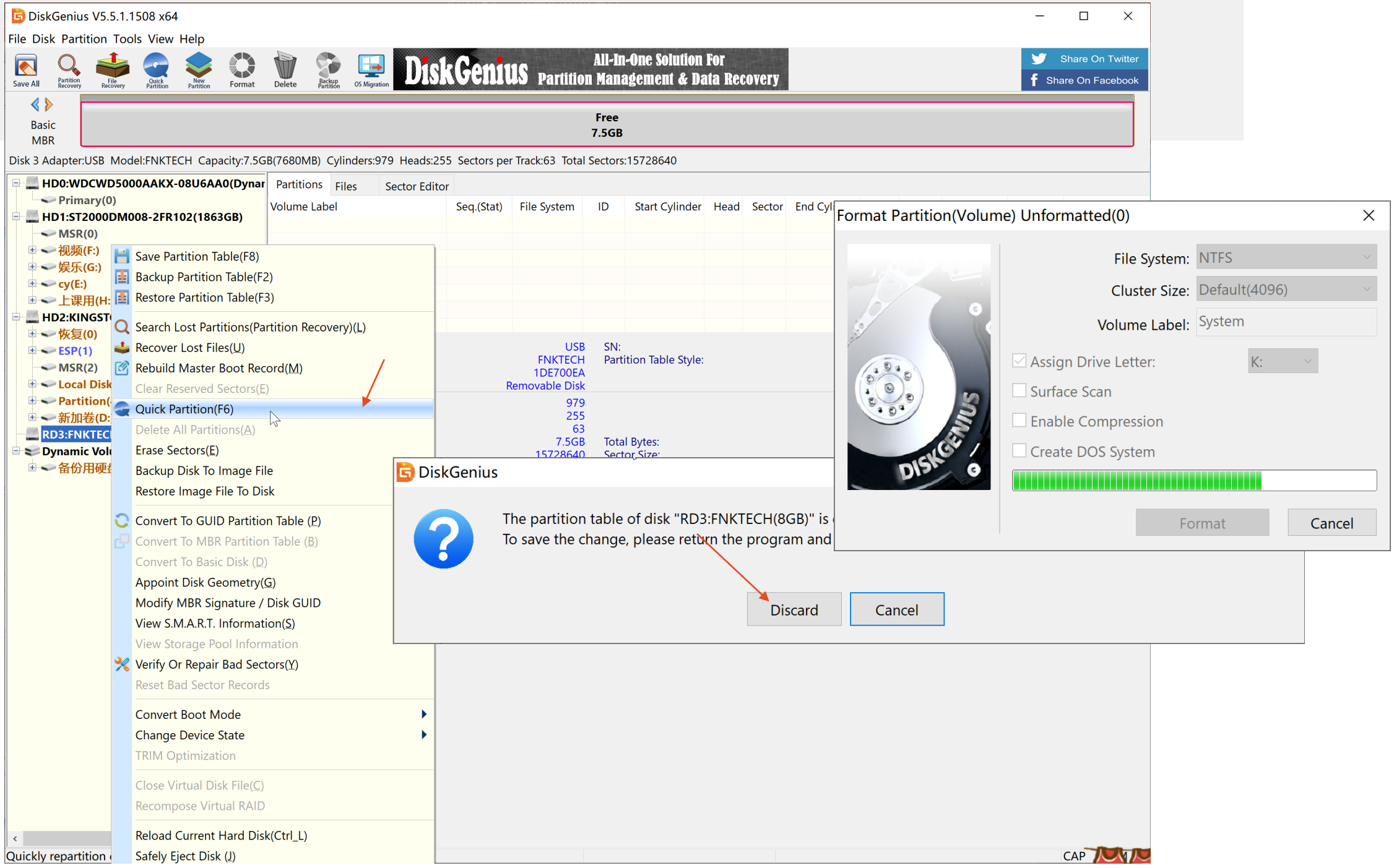
ARM®

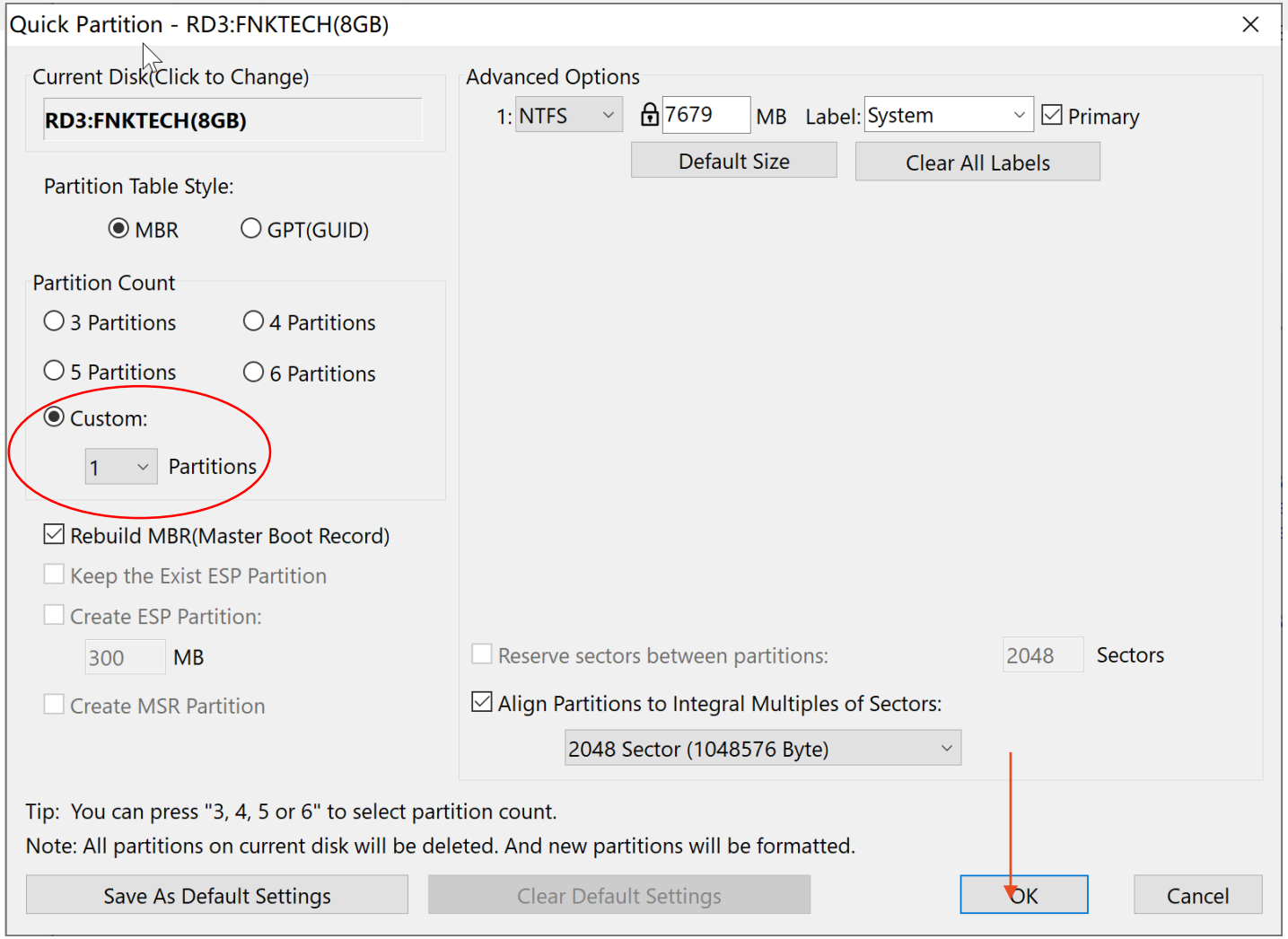
03

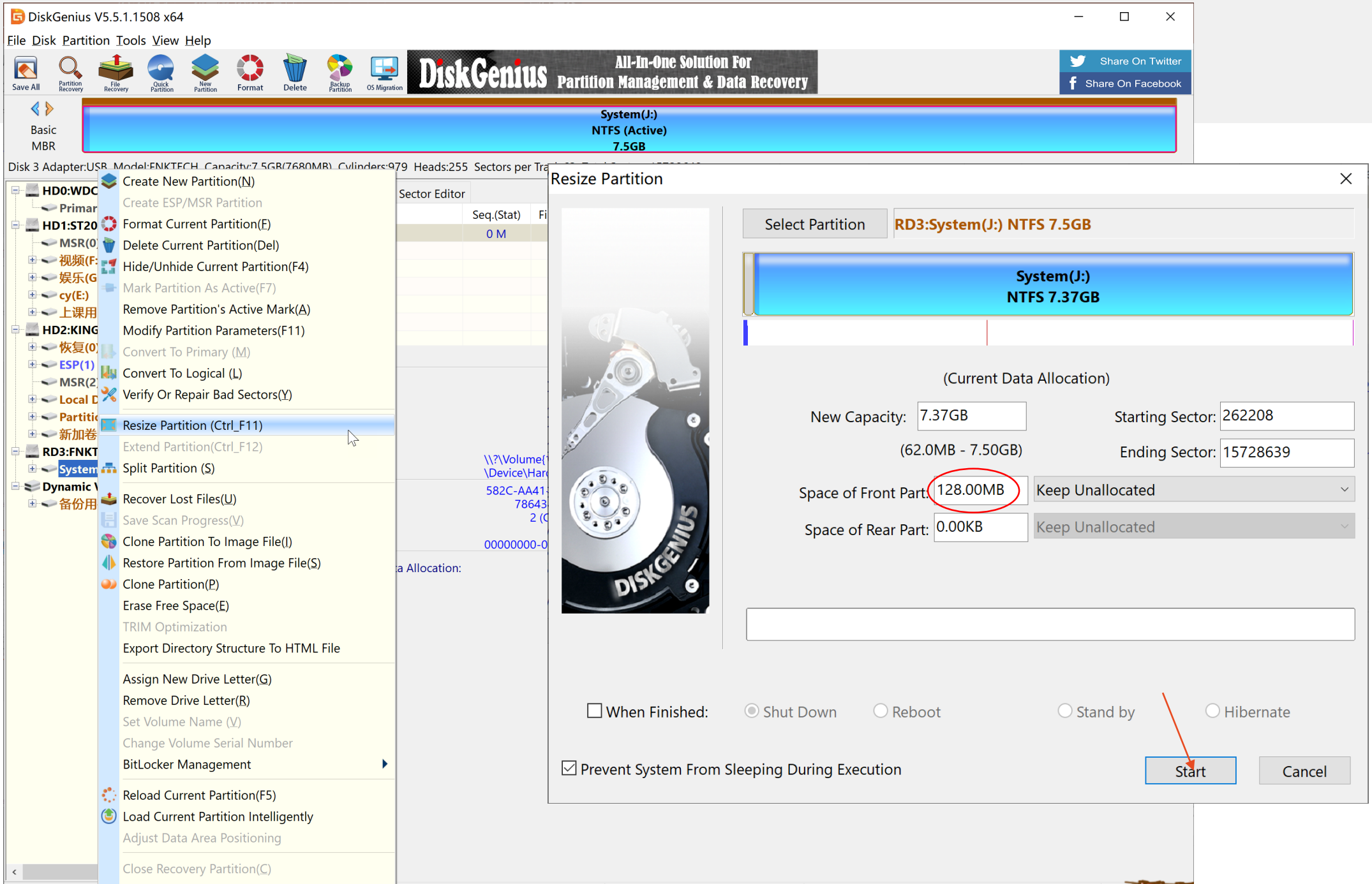
TF启动卡制作

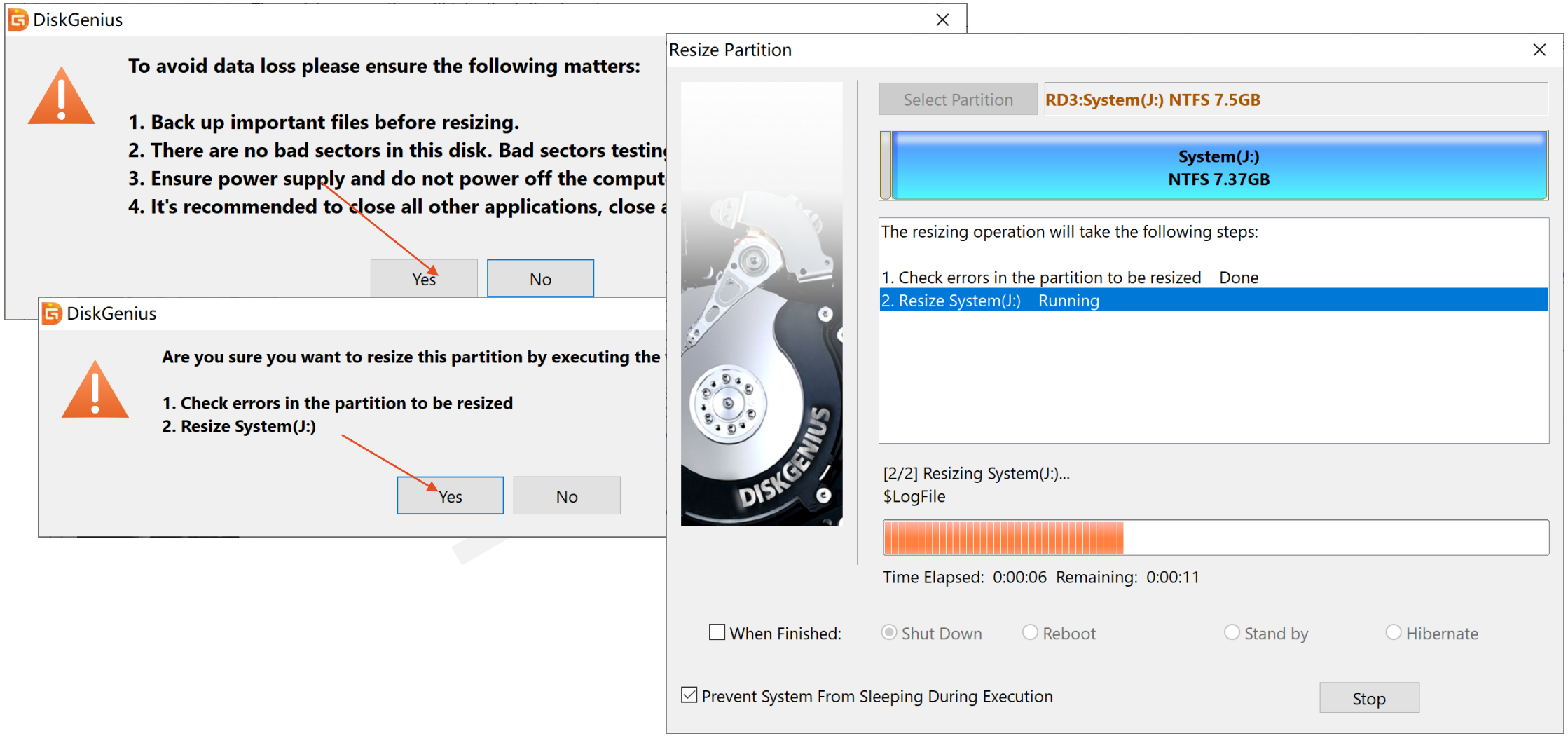
tf启动卡制作
-支持uboot启动

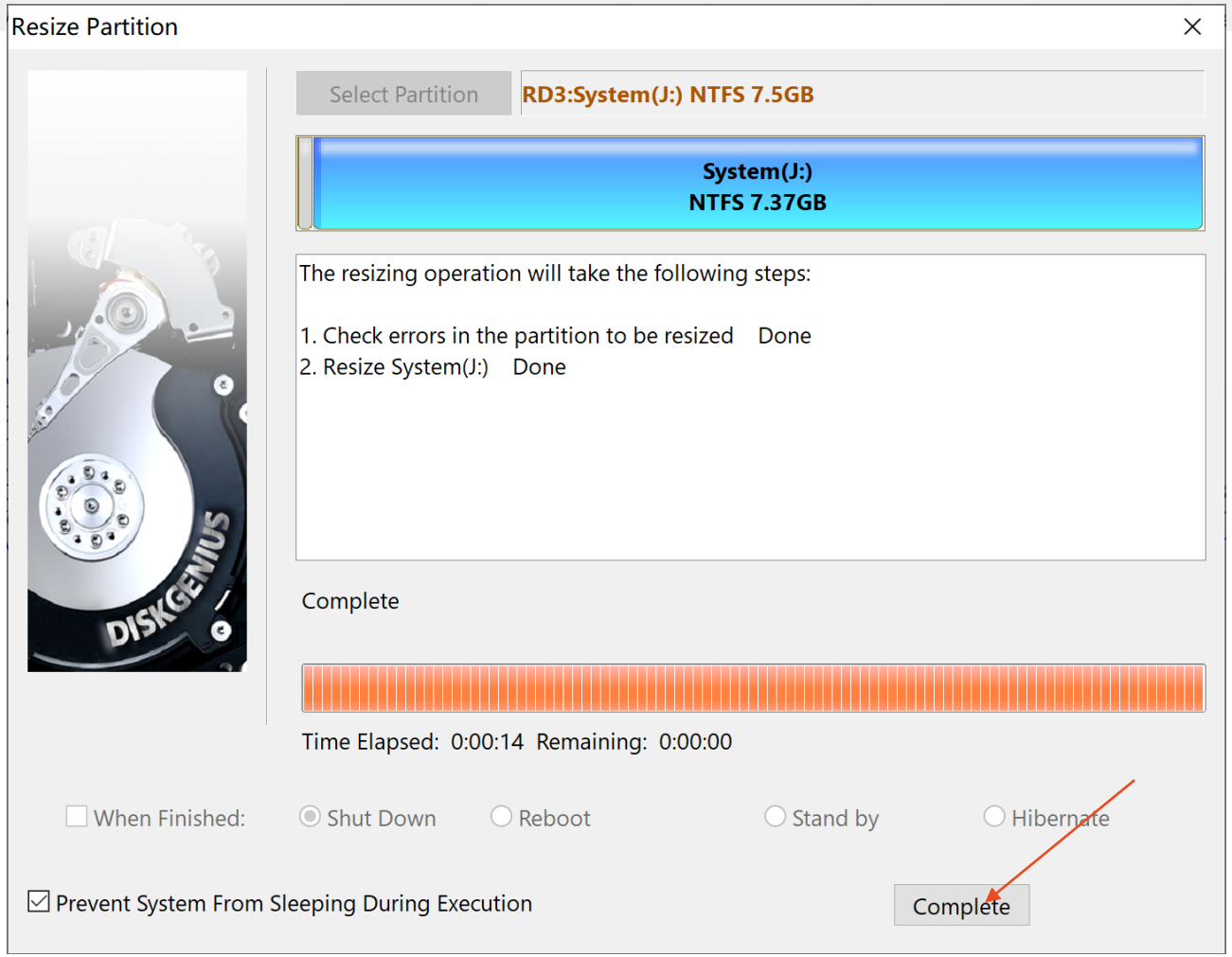

Basic
MBR
System(M:)
NTFS (Active)
7.5GB
Disk 3 Adapter:USB Model:FNKTECH Capacity:7.5GB(7680MB) Cylinders:979 Heads:255 Sectors per Track:63 Total Sectors:15728640
HD0:WDCWD5000AAKX-08U6AA0(Dynar
Primary(0)
HD1:ST2000DM008-2FR102(1863GB)
MSR(0)
视频(F
娱乐(G
cy(E:)
上课用
HD2:KING
恢复(0
ESP(1)
MSR(2
Local D
Partiti
新加卷
RD3:FNKT
System
Dynamic
备份用
Partitions
Files
Sector Editor
Volume Label
Seq.(Stat)
File System
ID
Start Cylinder
Head
Sector
End Cylinder
Head
Sector
Capacity
Attribute
System(M:)
0
NTFS
07
0
32
33
979
15
60
7.5GB
A
USB
FNKTECH
1DE700EA
Removable Disk
SN:
Partition Table Style:
MBR
979
255
63
7.5GB
15728640
1005
Total Bytes:
8053063680
Sector Size:
512 Bytes
Physical Sector Size:
512 Bytes
Save Partition Table(F8)
Backup Partition Table(F2)
Restore Partition Table(F3)
Search Lost Partitions(Partition Recovery)(L)
Recover Lost Files(U)
Rebuild Master Boot Record(M)
Clear Reserved Sectors(E)
Quick Partition(F6)
Delete All Partitions(A)
Erase Sectors(E)
Backup Disk To Image File
Restore Image File To Disk
Convert To GUID Partition Table (P)
Convert To MBR Partition Table (B)
Convert To Basic Disk (D)
Appoint Disk Geometry(G)
Modify MBR Signature / Disk GUID
View S.M.A.R.T. Information(S)
View Storage Pool Information
Verify Or Repair Bad Sectors(Y)
Reset Bad Sector Records
Convert Boot Mode
Change Device State
TRIM Optimization


DiskGenius V5.5.1.1508 x64
File Disk Partition Tools View Help
Save All
Partition Recovery
File Recovery
Quick Partition
New Partition
Format
Delete
Backup Partition
OS Migration
DiskGenius All-In-One Solution For Partition Management & Data Recovery
Share On Twitter
Share On Facebook
Basic MBR
Free 7.5GB
Disk 3 Adapter:USB Model:FNKTECH Capacity:7.5GB(7680MB) Cylinders:979 Heads:255 Sectors per Track:63 Total Sectors:15728640
HD0:WDCWD5000AAKX-08U6AA0(Dynar
Primary(0)
HD1:ST2000DM008-2FR102(1863GB)
MSR(0)
视频(F:)
娱乐(G:)
cy(E:)
上课用(H:
HD2:KINGST
恢复(0)
ESP(1)
MSR(2)
Local Disk
Partition(
新加卷(D:
RD3:FNKTEC
Dynamic Vol
备份用硬盘
Partitions
Files
Sector Editor
Volume Label
Seq.(Stat)
File System
ID
Start Cylinder
Head
Sector
End Cyl
USB
FNKTECH
1DE700EA
Removable Disk
SN:
Partition Table Style:
979
255
63
7.5GB
15728640
Total Bytes:
Sector Size:
Save Partition Table(F8)
Backup Partition Table(F2)
Restore Partition Table(F3)
Search Lost Partitions(Partition Recovery)(L)
Recover Lost Files(U)
Rebuild Master Boot Record(M)
Clear Reserved Sectors(E)
Quick Partition(F6)
Delete All Partitions(A)
Erase Sectors(E)
Backup Disk To Image File
Restore Image File To Disk
Convert To GUID Partition Table (P)
Convert To MBR Partition Table (B)
Convert To Basic Disk (D)
Appoint Disk Geometry(G)
Modify MBR Signature / Disk GUID
View S.M.A.R.T. Information(S)
View Storage Pool Information
Verify Or Repair Bad Sectors(Y)
Reset Bad Sector Records
Convert Boot Mode
Change Device State
TRIM Optimization
Close Virtual Disk File(C)
Recompose Virtual RAID
Reload Current Hard Disk(Ctrl_L)
Safely Eject Disk (J)
Quickly repartition
DiskGenius
The partition table of disk "RD3:FNKTECH(8GB)" is
To save the change, please return the program and
Discard
Cancel
Format Partition(Volume) Unformatted(0)
File System: NTFS
Cluster Size: Default(4096)
Volume Label: System
Assign Drive Letter: K:
Surface Scan
Enable Compression
Create DOS System
Format
Cancel
CAP



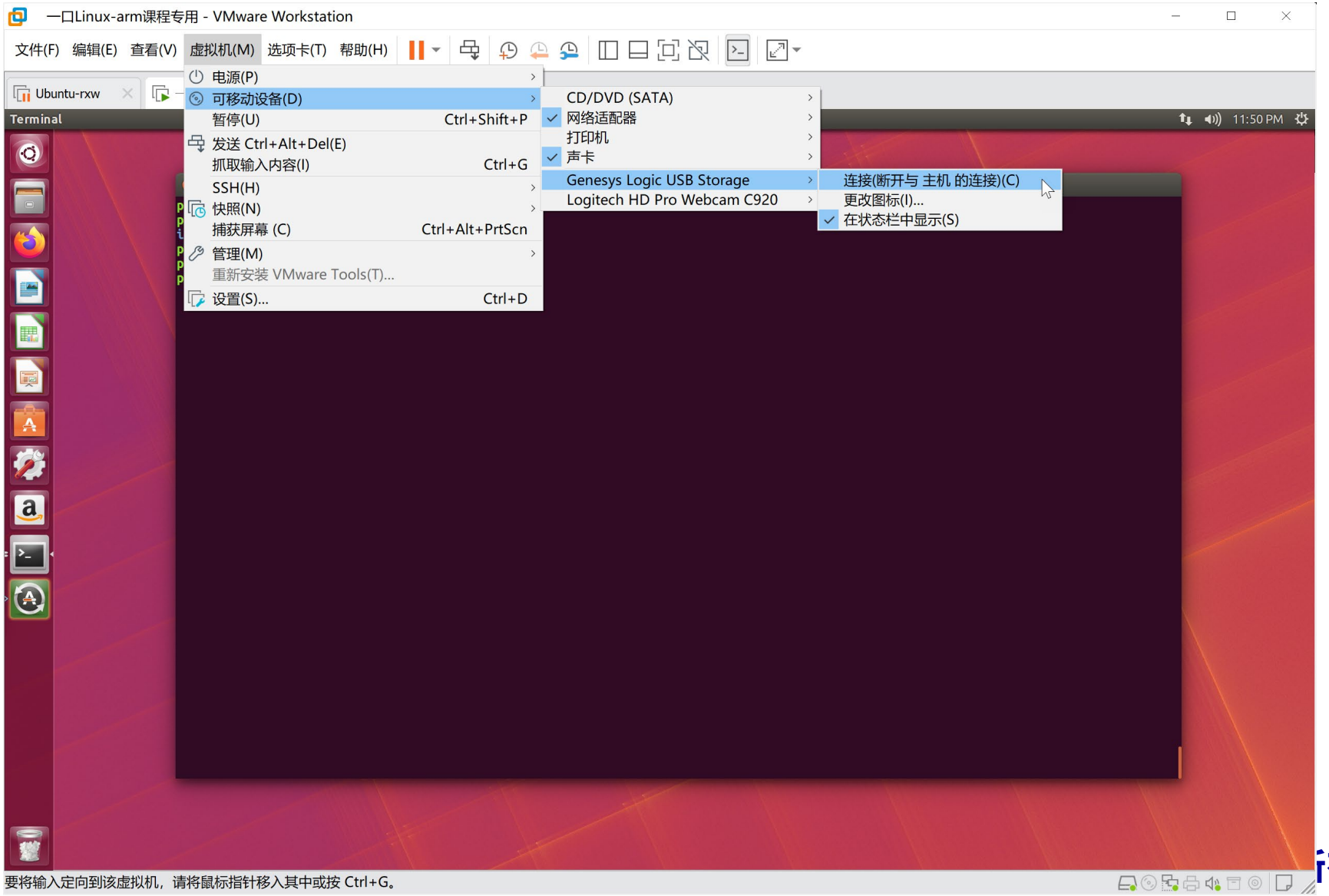

DiskGenius V5.5.1.1508 x64
File Disk Partition Tools View Help
Save All
Partition Recovery
File Recovery
Quick Partition
New Partition
Format
Delete
Backup Partition
OS Migration
DiskGenius
All-In-One Solution For
Partition Management & Data Recovery
Share On Twitter
Share On Facebook
Basic
MBR
System(J:)
NTFS (Active)
7.5GB
Disk 3 Adapter:USB Model:FNKTECH Capacity:7.5GB(7680MB) Cylinders:979 Heads:255 Sectors per Tra
HD0:WDC
Primar
HD1:ST20
MSR(0)
视频(F:
娱乐(G
cy(E:)
上课用
HD2:KING
恢复(0)
ESP(1)
MSR(2)
Local D
Partitio
新加卷
RD3:FNKT
System
Dynamic V
备份用
Sector Editor
Seq.(Stat)
0 M
Create New Partition(N)
Create ESP/MSR Partition
Format Current Partition(F)
Delete Current Partition(Del)
Hide/Unhide Current Partition(F4)
Mark Partition As Active(F7)
Remove Partition's Active Mark(A)
Modify Partition Parameters(F11)
Convert To Primary (M)
Convert To Logical (L)
Verify Or Repair Bad Sectors(Y)
Resize Partition (Ctrl_F11)
Extend Partition(Ctrl_F12)
Split Partition (S)
Recover Lost Files(U)
Save Scan Progress(V)
Clone Partition To Image File(I)
Restore Partition From Image File(S)
Clone Partition(P)
Erase Free Space(E)
TRIM Optimization
Export Directory Structure To HTML File
Assign New Drive Letter(G)
Remove Drive Letter(R)
Set Volume Name (V)
Change Volume Serial Number
BitLocker Management
Reload Current Partition(F5)
Load Current Partition Intelligently
Adjust Data Area Positioning
Close Recovery Partition(C)
Resize Partition
Select Partition
RD3:System(J:) NTFS 7.5GB
System(J:)
NTFS 7.37GB
(Current Data Allocation)
New Capacity: 7.37GB
(62.0MB - 7.50GB)
Starting Sector: 262208
Ending Sector: 15728639
Space of Front Part: 128.00MB
Keep Unallocated
Space of Rear Part: 0.00KB
Keep Unallocated
When Finished:
Shut Down
Reboot
Stand by
Hibernate
Prevent System From Sleeping During Execution
Start
Cancel



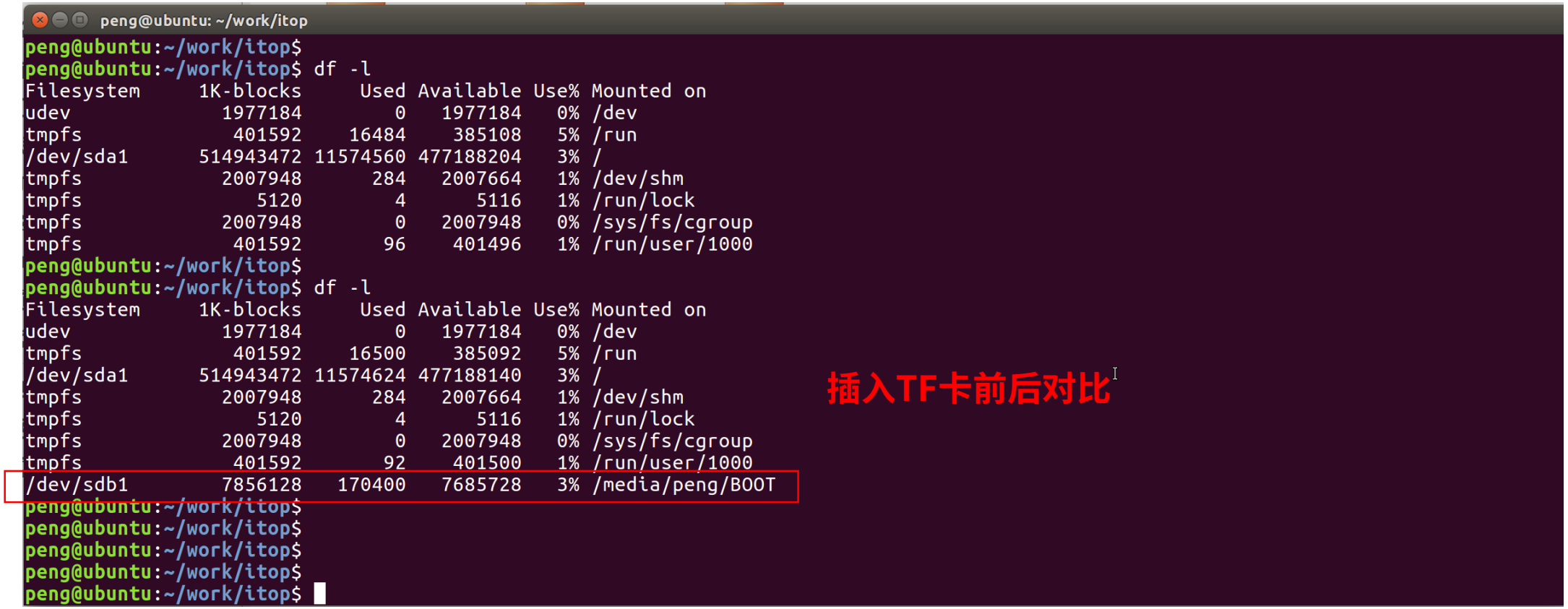


抓取TF卡

使用读卡器将 TF 卡连接到 PC
让vmware抓取tf卡



检查tf卡名称 df -l



插入TF卡前后对比

烧录uboot到tf卡

```
peng@ubuntu:~/work/itop/sdfuse_q$ sudo ./mkuboot.sh /dev/sdb
Fuse FS4412 trustzone uboot file into SD card
/dev/sdb reader is identified.
u-boot-iTOP-4412.bin fusing...
1029+1 records in
1029+1 records out
527104 bytes (527 kB, 515 KiB) copied, 3.58666 s, 147 kB/s
u-boot-iTOP-4412.bin image has been fused successfully.
Eject SD card
```

通过Tf卡升级uboot

serial-com5 - SecureCRT

File Edit View Options Transfer Script Tools Window Help

Enter host <Alt+R>

Session Manager

Filter by session name <A

Sessions
- 192.168.0.113
- 192.168.0.113 (1)
- 192.168.0.113 (2)
- serial-com3
- serial-com4
- serial-com5
- serial-com6
- serial-com7

serial-com5

```
Checking Boot Mode ... SDMMC
SYSTEM ENTER NORMAL BOOT MODE
Hit any key to stop autoboot:  2 OK

U-Boot 2010.03 (Mar 20 2018 - 00:57:22) for iTOP-4412 Android

        APLL = 1000MHz, MPLL = 800MHz
        ARM_CLOCK = 1000MHz
PMIC:   S5M8767(VER5.0)
Board:  iTOP-4412
DRAM:    2 GB
MMC:   max_emmc_clock:40 MHZ
Set CLK to 400 KHz
EMMC CLOCK OUTPUT:: 400KHz -[div:50]
response timeout error : 00000104 cmd 8
response timeout error : 00000104 cmd 55
max_emmc_clock:40 MHZ
Input CLK [ 50 MHz] is higher than limit [40 MHZ]
Set CLK to 40000 KHz
EMMC clock output: 40000 KHz
max_emmc_clock:40 MHZ
Input CLK [ 50 MHz] is higher than limit [40 MHZ]
Set CLK to 40000 KHz
EMMC clock output: 40000 KHz
MMC0:   14910 MB
SD sclk_mmc is 400K HZ
SD sclk_mmc is 50000K HZ
SD sclk_mmc is 50000K HZ
 >>>get part_type fail : 0
MMC1:   30007 MB
*** Warning - using default environment

In:    serial
Out:   serial
Err:   serial


Checking Boot Mode ... SDMMC
SYSTEM ENTER NORMAL BOOT MODE
Hit any key to stop autoboot:  0
reading kernel.. 1120, 12288
MMC read: dev # 1, block # 1120, count 12288 ...12288 block
completed
reading RFS.. 13408, 2048
MMC read: dev # 1, block # 13408, count 2048 ...2048 blocks
completed
Wrong Image Format for bootm command
ERROR: can't get kernel image!
iTOP-4412 # fastboot
[Partition table on MoviNAND]
ptn 0 name='bootloader' start=0x0 len=N/A (use hard-coded
ptn 1 name='kernel' start=N/A len=N/A (use hard-coded info. (cmd: movi))
ptn 2 name='ramdisk' start=N/A len=0x300000(~3072KB) (use hard-coded info. (cmd: movi))
ptn 3 name='Recovery' start=N/A len=0x600000(~6144KB) (use hard-coded info. (cmd: movi))
ptn 4 name='system' start=0x1000000 len=0x40000000(~1048576KB)
ptn 5 name='userdata' start=0x41000000 len=0x40000000(~1048576KB)
ptn 6 name='cache' start=0x81000000 len=0x12C00000(~307200KB)
ptn 7 name='fat' start=0x93C00000 len=0xF800000(~253952KB)
```

Ready | Serial: COM5, 115200 | 60, 1 | 60 Rows, 132 Cols | VT100 | CAP NUM

拨码：10xx

cmd

```
C:\Users\Administrator
λ fastboot.exe flash bootloader F:\录制视频\linux驱动第二期-讯为4412\xunwei\uboot+kernel\uboot-2017-tree\u-boot
iTOP-4412.bin
 sending 'bootloader' (475 KB)... OKAY [  0.081s]
          writing 'bootloader'... OKAY [  0.192s]
finished. total time: 0.275s

C:\Users\Administrator
λ
```

```
Received 17 bytes: download:00076e56
Starting download of 486998 bytes

downloading of 486998 bytes finished
Received 16 bytes: flash:bootloader
flashing 'bootloader'
writing bootloader.. 0, 1038
eMMC OPEN Success.!!
                        !!!Notice!!!
!You must close eMMC boot Partition after all image writing!
!eMMC boot partition has continuity at image writing time.!
!So, Do not close boot partition, Before, all images is written.!

MMC write: dev # 0, block # 0, count 1038. 1038 blocks write finish
1038 blocks verify1: OK
eMMC CLOSE Success.!!
completed
partition 'bootloader' flashed
```

频中所有资料

tf启动卡制作脚本分析

脚本文件

- mkuboot.sh
- sd_fusing_exynos4x12.sh

- param1=`echo "$1" | awk '{print substr($1,1,7)}'`
  - $1 /dev/sdb
  - awk调用命令

- dd iflag=dsync oflag=dsync if=$2 of=$1 seek=1 && \

- [-b FILE] 如果 FILE 存在且是一个块特殊文件则为真。
- [-z STRING] “STRING” 的长度为零则为真。

umount

- 卸载分区
  - umount /dev/sdb1

dd

- dd作用是将一个指定文件拷贝到磁盘的指定块。
- 可以用于磁盘备份、程序烧写等应用。
- 基本语法：
  - dd iflag=dsync oflag=dsync if=<输入文件> of=<输出文件> seek=<跳过扇区数量>

- dd iflag=dsync oflag=dsync if=$2 of=$1 seek=1
  - iflag=dsync：
    - 表示输入文件读取时不经过缓冲区，一块一块地读取（块的大小为缓冲区大小），直到读取完成。
  - oflag=dsync：
    - 表示输出文件不经过缓冲区，来一块，写入一块。
  - if=$2：
    - 表示输入文件为$2,即 u-boot-iTOP-4412.bin
  - of=$1：
    - 表示输出文件为$1 ，即/dev/sdb
  - seek=1：
    - 表示跳过第0块，从第1块开始写入

ARM®

04

Uboot环境变量及命令

本节内容

- 0. uboot环境变量以及操作
- 1. uboot常用命令
- 2. mmc/SD基础知识
- 3. Uboot mmc操作命令
- 4. uboot启动参数分析
- 5. Uboot手动引导Linux启动的几种方式

Uboot环境变量以及操作

1. 环境变量显示、设置与保存

printenv	打印U-Boot环境变量
setenv	设置U-Boot环境变量。 例:setenv envname value设置环境变量 envname的值，如果没有value，则表示删除envname环境变量。
saveenv	将修改的环境变量保存到固态存储器中

help printenv
help setenv

举例

- 删除环境变量
  - Setenv yikou
- 保存环境变量
  - saveenv

2. U-Boot常见环境变量

命令	含义
bootdelay	执行自动启动（bootcmd中的命令）的等候秒数
baudrate	串口控制台的波特率
netmask	以太网的网络掩码
ethaddr	以太网的MAC地址
bootfile	默认的下载文件名
ipaddr	本地的IP地址
gateway	以太网的网关
serverip	TFTP服务器端的IP地址
stdin	标准输入设备，一般是串口
stdout	标准输出，一般是串口，也可是LCD（VGA）
stderr	标准出错，一般是串口，也可是LCD（VGA）

U-boot最重要的2个环境变量

命令	含义
bootcmd	自动启动时执行命令。 U-Boot开机后会自动倒计时， 在倒计时结束前如果没有外部按键打断自动计时， U-Boot将自动执行bootcmd变量保存的命令。
bootargs	传递给Linux内核的启动参数。

bootcmd与bootargs可以说是uboot最重要的两个环境参数

bootm	引导启动存储在内存中的程序映像。 这些内存包括RAM和可以永久保存的Flash。

环境变量补充

- (1)默认环境变量
  - 在uboot/common/env_common.c 中 default_environment，
  - 本质是一个字符数组，大小为CFG_ENV_SIZE（16kb），
  - 每个环境变量最末端以’\0’结束。
- (2)SD卡中环境变量分区，在uboot的raw分区中。
  - 当saveenv时其实整个环境变量都被保存了一遍，而不是只保存更改了的。
- (3)DDR中环境变量，在default_environment中，实质是字符数组。
  - 在uboot中其实是一个全局变量，链接时在数据段，重定位时default_environment就被重定位到DDR中一个内存地址处了。
-

Uboot常用命令

1. help

- help查看所有支持的命令

2. bdinfo

- 查看开发板信息

```
u-boot # bdinfo
arch_number = 0x000013FB
boot_params = 0x40000100
DRAM bank   = 0x00000000
-> start    = 0x40000000
-> size     = 0x10000000
DRAM bank   = 0x00000001
-> start    = 0x50000000
-> size     = 0x10000000
DRAM bank   = 0x00000002
-> start    = 0x60000000
-> size     = 0x10000000
DRAM bank   = 0x00000003
-> start    = 0x70000000
-> size     = 0x10000000
baudrate    = 115200 bps
TLB addr    = 0x7FEF0000
relocaddr   = 0x7FE5C000
reloc off   = 0x3C05C000
irq_sp      = 0x7AE56A40
sp start    = 0x7AE56A30
Early malloc usage: f4 / 400
fdt_blob = 7ae56a58
```

DRAM起始地址

波特率

Sp堆栈地址

3. version

```
u-boot # version
U-Boot 2017.11 (Jan 02 2020 - 15:22:16 +0800) for itop-4412

arm-none-linux-gnueabi-gcc (Freescale MAD -- Linaro 2011.07 -- Built at 2011/08/10 09:20) 4.6.2
 20110630 (prerelease)
GNU ld (Freescale MAD -- Linaro 2011.07 -- Built at 2011/08/10 09:20) 2.21.52.20110702
```

4. reset

- 重启 uboot
  - u-boot # reset
  - resetting ...
  - OK

5.run 命令

- 用于运行环境变量中定义的命令

- 复用其他环境变量
  - setenv yikou "echo ${bootdelay}“
- 多条命令
  - setenv yikou "echo qqqqq && echo 222222“
- If then
  - setenv yikou "if mmc rescan; then echo mmc ok; fi;“

6. go 命令

- go 命令用于跳到指定的地址处执行应用
  - go addr [arg ...]
    - addr 是应用在 DRAM 中的首地址，通过 go 命令我们就可以在 uboot 中运行裸机实验。

7.uboot网络命令

命令	含义
ping	测试网络能否使用
dhcp	动态获取IP地址
nfs	nfs(Network File System)网络文件系统，通过nfs命令可以下载镜像文件到开发板的内存中
tftp	通过网络下载镜像到内存中， tftp命令使用的TFTP 协议

8. Nand Flash操作命令

命令	含义
nand info	显示可使用的Nand Flash
nand device [dev]	用于切换 NAND Flash，如果你的板子支持多片NAND 的话就可以使用此命令来设置当前所使用的NAND
nand read addr off size	读取命令，从Nand的off偏移地址处读取size字节的数据到SDRAM的addr地址
nand write addr off size	烧写命令，将SDRAM的addr地址处的size字节的数据烧写到Nand的off偏移地址

9. USB操作命令

命令	含义
usb reset	初始化USB控制器
usb stop [f]	关闭USB控制器
usb tree	已连接的USB设备树
usb info [dev]	显示USB设备[dev]的信息
usb storage	显示已连接的USB存储设备
usb dev [dev]	显示和设置当前USB存储设备
usb part [dev]	显示USB存储设备[dev]的分区信息
usb read addr blk# cnt	读取USB存储设备数据

使用USB操作命令前必须确保USB设备连接好，usb reset，以初始化USB控制器，获取设备信息。

10.mmc

- Sd/MMC

11.内存、寄存器操作命令

命令	含义
nm [.b, .w, .l] address	修改指定地址address的内存值
mm [.b, .w, .l] address	修改指定地址address内存值(地址自动加一)
md [.b, .w, .l] address [# of objects]	显示地址address开始的内存值， [# of objects]表示要查看的数据长度
mw [.b, .w, .l] address value [count]	用指定的数据value填充内存，起始地址为address，填充count个数据块
cp [.b, .w, .l] source target count	数据拷贝命令，用于将 DRAM 中的数据从一段内存拷贝到另一段内存中，或者把 Nor Flash 中的数据拷贝到 DRAM 中。

其中:[.b .w .l]对应 byte、 word 和 long，分别以 1 个字节、 2 个字节、 4 个字节来显示内存值。

md

https://www.bilibili.com/video/BV1K14y1S7yJ/?vd_source=07570058a62e0e8a6cf489efac35cfec

- ”md“命令用于显示内存值，
- 格式如下：
  - md[.b，.w，.l] address [#of objects]
    - [.b,.w,.l]对应byte，word，long
    - address就是要查看的内存起始地址，
    - [#of object]表示要查看的内存数据长度

```
iTOP-4412 # md.l   11000060
11000060: 00000000 00000078 00001555 00000000    ....x...U.......
11000070: 00000000 00000000 00001555 00000000    ........U.......
11000080: 02222222 0000007f 00003ff0 00003fff    """.....?...?..
11000090: 00000000 00000000 00003ff0 00003fff    ........?...?..
110000a0: 00000000 00000000 00001555 00000000    ........U.......
110000b0: 00000000 00000000 00001555 00000000    ........U.......
110000c0: 00000000 00000000 00001555 00000000    ........U.......
110000d0: 00000000 00000000 00001555 00000000    ........U.......
110000e0: 00000000 00000000 00000005 00000002    ................
110000f0: 00000000 00000000 00000005 00000002    ................
11000100: 00000001 00000081 00005555 00000000    ........UU......
11000110: 00000000 00000000 0000557f 00000003    ..........U.....
11000120: 00000000 0000003f 00000fff 00000aaa    ....?...........
11000130: 00000000 00000000 00000fff 00000aab    ................
11000140: 00000000 0000000f 000000ff 000000aa    ................
11000150: 00000000 00000000 000000ff 000000ab    ................
```

nm

- nm用于修改指定内存地址的值，
- 命令格式如下：
  - nm[.b，.w，.l] address

```
iTOP-4412 # nm.l 11000060
11000060: 00000000 ? 00000010
11000060: 00000010 ? q
iTOP-4412 # md.l 11000060
11000060: 00000010 00000078 00001555 00000000    ....x...U.......
11000070: 00000000 00000000 00001555 00000000    ........U.......
11000080: 02222222 0000007e 00003ff0 00003fff    """.~....?...?..
11000090: 00000000 00000000 00003ff0 00003fff    ........?...?..
110000a0: 00000000 00000000 00001555 00000000    ........U.......
110000b0: 00000000 00000000 00001555 00000000    ........U.......
110000c0: 00000000 00000000 00001555 00000000    ........U.......
110000d0: 00000000 00000000 00001555 00000000    ........U.......
110000e0: 00000000 00000000 00000005 00000002    ................
110000f0: 00000000 00000000 00000005 00000002    ................
11000100: 00000001 00000081 00005555 00000000    ........UU......
11000110: 00000000 00000000 0000557f 00000003    ........U.......
11000120: 00000000 0000003f 00000fff 00000aaa    ....?...........
11000130: 00000000 00000000 00000fff 00000aab    ................
11000140: 00000000 0000000f 000000ff 000000aa    ................
11000150: 00000000 00000000 000000ff 000000ab    ................
```

emmc/SD基础知识

SD/MMC基础概念

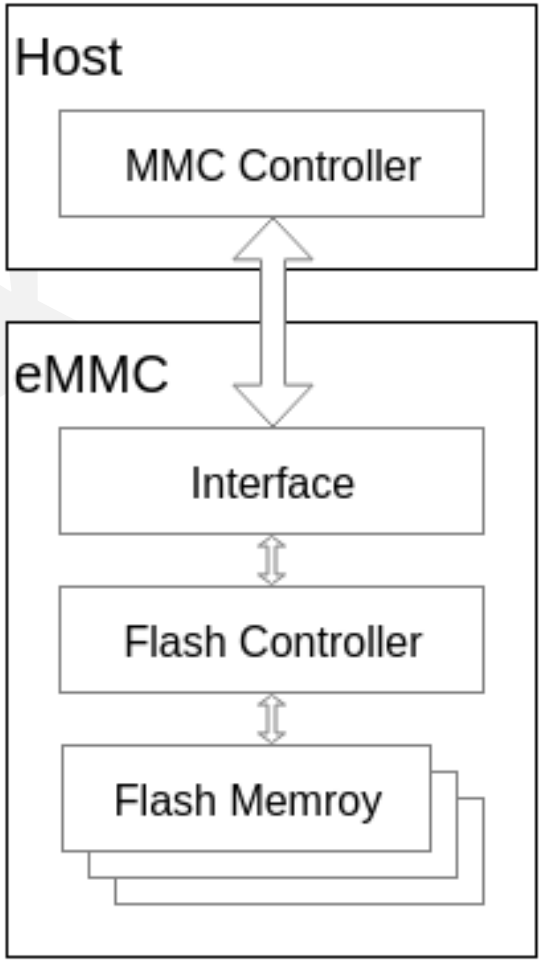
- eMMC 是 embedded MultiMediaCard 的简称。
- 是一种闪存卡（Flash Memory Card）标准，它定义了 MMC 的架构以及访问 Flash Memory 的接口和协议

- 一般认为EMMC和 SD卡是同一个东西,所以没有特殊说明统一使用MMC来代指EMMC和SD卡。

- uboot支持EMMC和SD卡,因此也要提供EMMC和SD卡的操作命令。

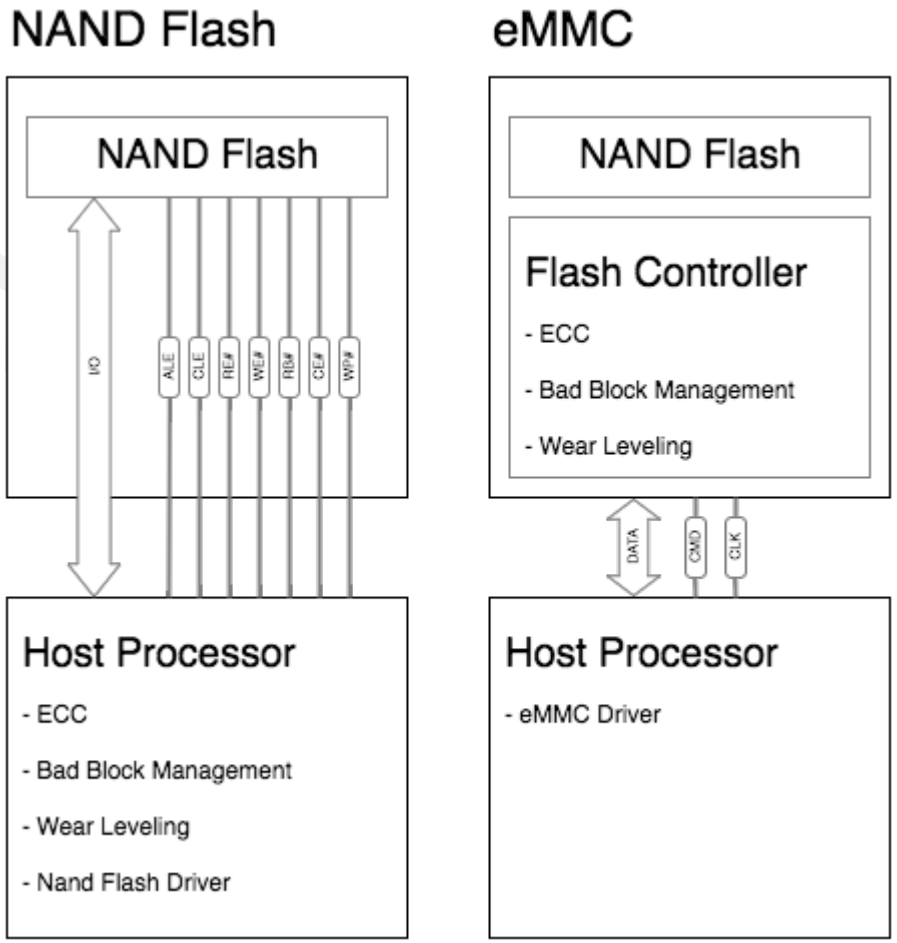
eMMC 整体架构

- eMMC 内部主要可以分为：
  - Flash Memory
  - Flash Controller
  - Host Interface

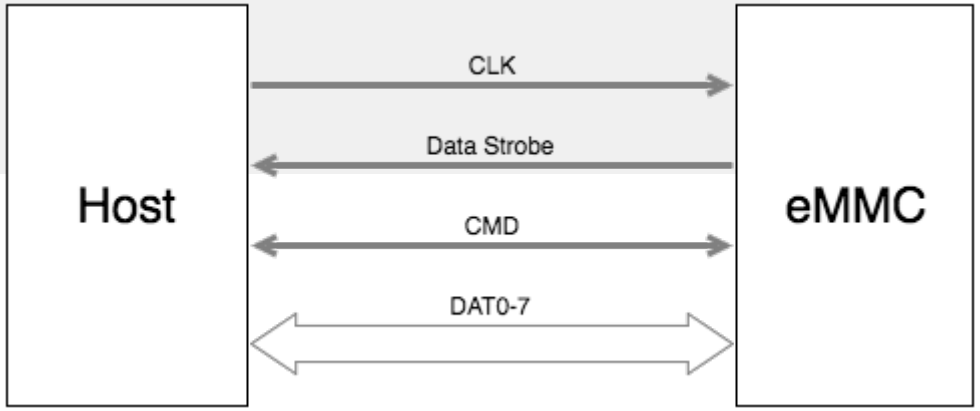


1） Flash Controller

- NAND Flash 直接接入 Host 时，Host 端通常需要有 NAND Flash Translation Layer，即 NFTL 或者 NAND Flash 文件系统来做坏块管理、ECC等的功能。
- eMMC 则在其内部集成了 Flash Controller，用于完成擦写均衡、坏块管理、ECC校验等功能。
- 相比于直接将 NAND Flash 接入到 Host 端，eMMC 屏蔽了 NAND Flash 的物理特性，可以减少 Host 端软件的复杂度，让 Host 端专注于上层业务，省去对 NAND Flash 进行特殊的处理。



2）Host Interface



- CLK
  - 用于同步的时钟信号
- Data Strobe
  - 此信号是从 Device 端输出的时钟信号，频率和 CLK 信号相同，用于同步从 Device 端输出的数据。该信号在 eMMC 5.0 中引入。
- CMD
  - 此信号用于发送 Host 的 command 和 Device 的 response。
- DAT0-7
  - 用于传输数据的 8 bit 总线。
    Host 与 eMMC 之间的通信都是 Host 以一个 Command 开始发起的。
    针对不同的 Command，Device 会做出不同的响应。

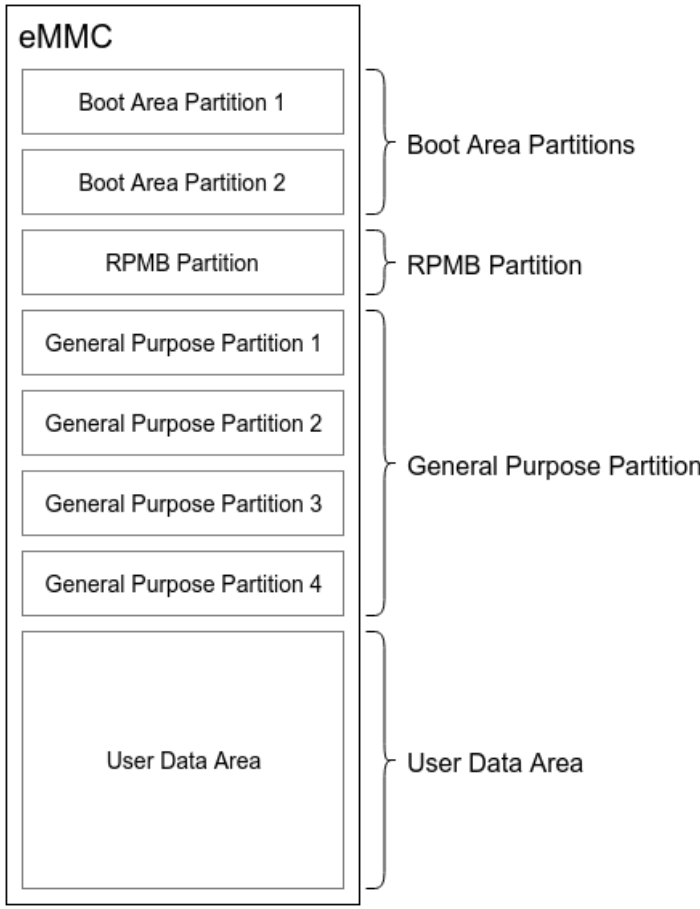
3） Flash Memory

- Flash Memory 是一种非易失性的存储器，通常在嵌入式系统中用于存放系统、应用和数据等，类似于 PC 系统中的硬盘

- 绝大部分手机和平板等移动设备中所使用的 eMMC 内部的 Flash Memory 都属于 NAND Flash

eMMC 在内部对 Flash Memory 划分

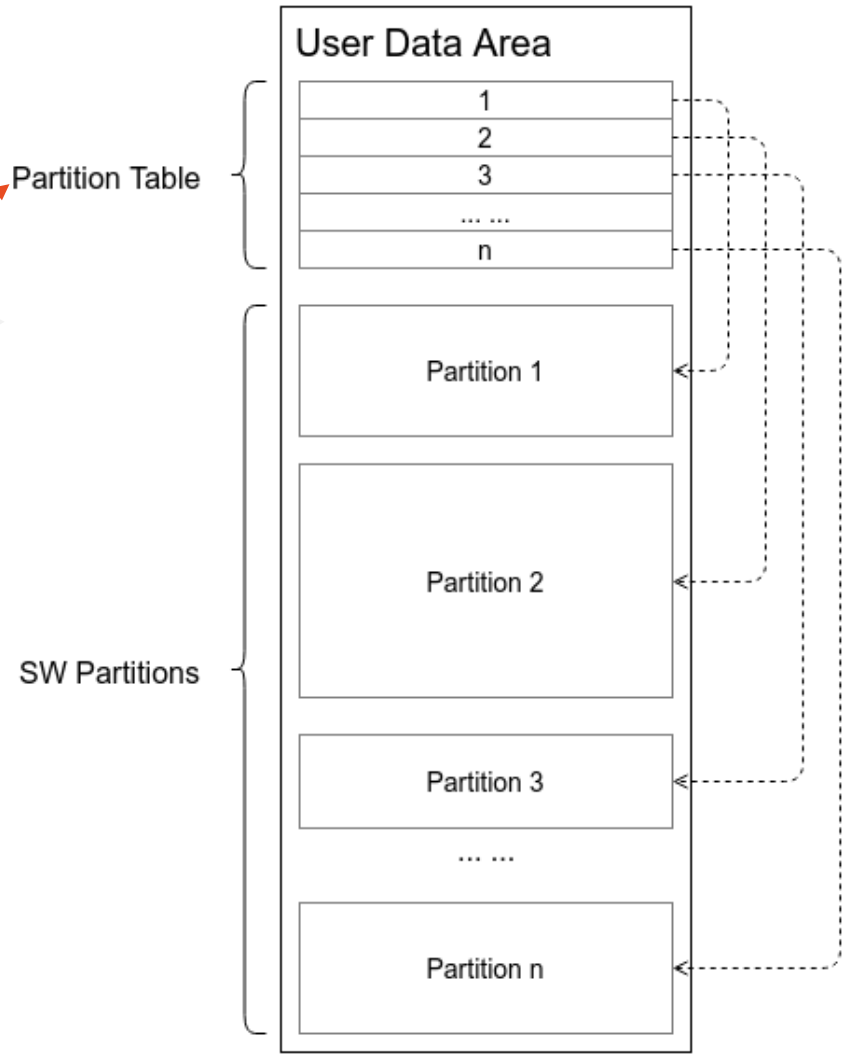
- **BOOT Area Partition 1 & 2**
  - 主要用于存储 Bootloader，支持 SOC 从 eMMC 启动系统。
    该分区的数据，在 eMMC 上电后，可以通过很简单的协议就可以读取出来。
  - 大部分的 SOC 都可以通过 GPIO 或者 FUSE 的配置，让 ROM 代码在上电后，将 eMMC BOOT 分区的内容加载到 SOC 内部的 SRAM 中执行。
- **RPMB Partition（Replay Protected Memory Block）**
  - eMMC 在写入数据到 RPMB 时，会*校验数据的合法性*，只有指定的 Host 才能够写入。
  - 在实际应用中，RPMB 分区通常用来保存安全相关的数据，例如指纹数据、安全支付相关的密钥等。
- **General Purpose Partition 1～4**
  - 此区域则主要用于存储系统或者用户数据。
  - General Purpose Partition 在芯片出厂时，通常是不存在的，需要主动进行配置后，才会存在。
- **User Data Area**
  - 此区域则主要用于存储系统和用户数据。
    User Data Area 通常会进行再分区，例如 Android 系统中，通常在此区域分出 boot、system、userdata 等分区。



eMMC 内部分区

UDA软件分区

- User Data Area (UDA) 通常是 eMMC 中最大的一个分区，是实际产品中，最主要的存储区域。
- 目前主流的软件分区技术有 MBR（Master Boot Record）和 GPT（GUID Partition Table）
- 软件分区技术一般是将存储介质划分为多个区域，然后通过一个 Partition Table 来维护这些 SW Partitions。



当前mmc分区

```
~ # ls /dev/mmcblk* -l
brw-rw----    1 root     root      179,  64 Jan  1 00:23 /dev/mmcblk0
brw-------    1 root     root      179,  65 Jan  1 00:23 /dev/mmcblk0p1
brw-rw----    1 root     root      179,   0 Jan  1  1970 /dev/mmcblk1
brw-rw----    1 root     root      179,  16 Jan  1  1970 /dev/mmcblk1boot0
brw-rw----    1 root     root      179,  32 Jan  1  1970 /dev/mmcblk1boot1
brw-rw----    1 root     root      179,   1 Jan  1  1970 /dev/mmcblk1p1
brw-rw----    1 root     root      179,   2 Jan  1  1970 /dev/mmcblk1p2
brw-rw----    1 root     root      179,   3 Jan  1  1970 /dev/mmcblk1p3
brw-rw----    1 root     root      179,   4 Jan  1  1970 /dev/mmcblk1p4
brw-rw----    1 root     root      179,  48 Jan  1  1970 /dev/mmcblk1rpmb
```

前面我们制作的tf启动卡

emmc

emmc分区与linux中文件名对应关系

- mmcblk1 ←----------→ eMMC 的块设备;
- mmcblk1boot0 mmcblk1boot1 ←----------→ 对应两个 Boot Area Partitions;
- mmcblk1rpmb ←----------→则为 RPMB Partition;
- mmcblk1px ←----------→ UDA 划分出来的 SW Partitions;
- GPP ←----------→ mmcblk1gp1、mmcblk1gp2、mmcblk1gp3、mmcblk1gp4;

Linux中分区设备名

```
~ # ls /dev/mmcblk1* -l
brw-rw----    1 root     root      179,   0 Jan  1  1970 /dev/mmcblk1
brw-rw----    1 root     root      179,  16 Jan  1  1970 /dev/mmcblk1boot0
brw-rw----    1 root     root      179,  32 Jan  1  1970 /dev/mmcblk1boot1
brw-rw----    1 root     root      179,   1 Jan  1  1970 /dev/mmcblk1p1
brw-rw----    1 root     root      179,   2 Jan  1  1970 /dev/mmcblk1p2
brw-rw----    1 root     root      179,   3 Jan  1  1970 /dev/mmcblk1p3
brw-rw----    1 root     root      179,   4 Jan  1  1970 /dev/mmcblk1p4
brw-rw----    1 root     root      179,  48 Jan  1  1970 /dev/mmcblk1rpmb
```

挂载的文件系统system.img

Uboot中分区号

```
u-boot # mmc part

Partition Map for MMC device 0  --   Partition Type: DOS

Part    Start Sector    Num Sectors     UUID            Type
  1     4841472         25673728        00000000-01     0c
  2     32768           2097152         00000000-02     83
  3     2129920         2097152         00000000-03     83
  4     4227072         614400          00000000-04     83
```

Uboot mmc操作命令

SD/MMC操作命令列表

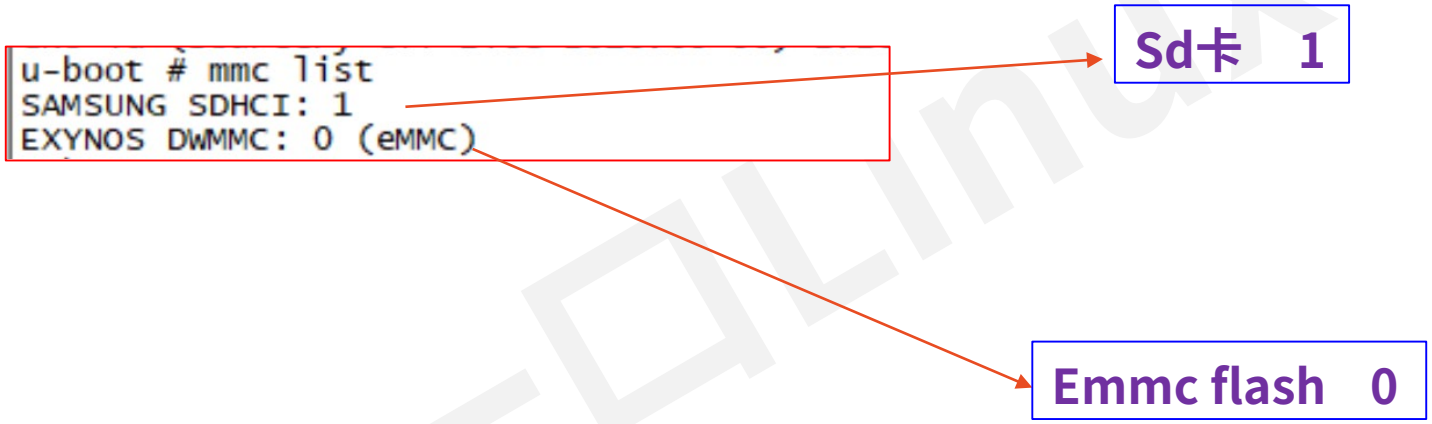
命令	含义
mmc init [dev]	初始化MMC子系统
mmc info	输出MMC设备信息
mmc read	读出MMC中数据
mmc write	向MMC设备写入数据
mmc rescan	扫描MMC设备
mmc part	列出MMC设备的分区
mmc dev	切换MMC设备
mmc list	列出当前有效的所有MMC设备
mmc hwpartition	设置MMC设备的分区
mmc bootbus……	设置MMC设备的BOOT_BUS_WIDTH域的值
mmc bootpart……	设置MMC设备的boot和RPMB分区的大小
mmc bootconf……	设置MMC设备的PARTITION_CONFG域的值
mmc rst	复位MMC设备
mmc setdsr	设置DSR寄存器的值

mmc rescan命令

- mmc rescan命令用于扫描当前开发板上所有的MMC设备,包括EMMC和SD卡

mmc list

- mmc list命令用于来查看当前开发板一共有几个MMC设备



mmc dev 命令

- mmc dev命令用于切换当前MMC设备
  - mmc dev [dev] [part]
    - [dev]用来设置要切换的MMC设备号
      如果不写分区号的话默认为分区0
    - [part]是分区号
- 切换到SD卡
  - mmc dev 1//切换到EMMC卡,
    - 0为SD卡,1为eMMC

```
u-boot # mmc info
Device: EXYNOS DWMMC
Manufacturer ID: 15
OEM: 100
Name: AJTD4
Tran Speed: 52000000
Rd Block Len: 512
MMC version 5.1
High Capacity: Yes
Capacity: 14.6 GiB
Bus Width: 4-bit DDR
Erase Group Size: 512 KiB
HC WP Group Size: 8 MiB
User Capacity: 14.6 GiB WRREL
Boot Capacity: 4 MiB ENH
RPMB Capacity: 4 MiB ENH
u-boot #
u-boot #
u-boot #
u-boot # mmc dev 1
mmc1 is current device
u-boot #
u-boot #
u-boot # mmc info
Device: SAMSUNG SDHCI
Manufacturer ID: 6f
OEM: 303
Name: SDABC
Tran Speed: 50000000
Rd Block Len: 512
SD version 3.0
High Capacity: Yes
Capacity: 29.3 GiB
Bus Width: 4-bit
Erase Group Size: 512 Bytes
```

mmc part命令

- 查看EMMC的分区

```
u-boot # mmc part

Partition Map for MMC device 0  --   Partition Type: DOS

Part    Start Sector    Num Sectors     UUID            Type
  1     4841472         25673728        00000000-01     0c
  2     32768           2097152         00000000-02     83
  3     2129920         2097152         00000000-03     83
  4     4227072         614400          00000000-04     83
```

0c fat
83 linux分区

fstype

- fstype用于查看MMC设备某个分区的文件系统格式
  - fstype <interface> <dev>:<part>

```
u-boot # fstype mmc 0:1
fat
u-boot # fstype mmc 0:2
ext4
u-boot # fstype mmc 0:3
ext4
u-boot # fstype mmc 0:4
ext4
```

存放根文件系统。

EXT 格式文件系统操作命令

- uboot有ext2和ext4这两种格式的文件系统的操作命令，
- 常用命令分别为:ext2load、ext2ls、ext4load、ext4ls和 ext4write。

```
u-boot # fatls mmc 0:1

0 file(s), 0 dir(s)
```

```
u-boot # ext4ls mmc 0:2
<DIR>       4096 .
<DIR>       4096 ..
             121 .ash_history
<DIR>       4096 bin
<DIR>       4096 dev
<DIR>       4096 etc
<DIR>       4096 home
<DIR>       4096 lib
<SYM>         11 linuxrc
<DIR>       4096 mnt
<DIR>       4096 opt
<DIR>       4096 proc
<DIR>       4096 root
<DIR>       4096 sbin
<DIR>       4096 sys
<DIR>       4096 system
<DIR>       4096 tmp
<DIR>       4096 topeet
<DIR>       4096 usr
<DIR>       4096 var
              49 version
```

```
u-boot # ext4ls mmc 0:3
<DIR>       4096 .
<DIR>       4096 ..
<DIR>      16384 lost+found
u-boot # ext4ls mmc 0:4
<DIR>       4096 .
<DIR>       4096 ..
<DIR>      16384 lost+found
```

```
u-boot # ext4ls mmc 0:2 /etc
<DIR>       4096 .
<DIR>       4096 ..
             337 fstab
<DIR>       4096 init.d
             335 inittab
             257 profile
```

```
u-boot # ext4load mmc 0:2  0x44000000  /etc/fstab
337 bytes read in 13 ms (24.4 KiB/s)
u-boot # md.b 0x44000000 0x80
44000000: 23 64 65 76 69 63 65 20 20 20 20 20 6d 6f 75 6e    #device     moun
44000010: 74 2d 70 6f 69 6e 74 20 20 20 20 20 74 79 70 65    t-point     type
44000020: 20 20 20 20 20 6f 70 74 69 6f 6e 73 20 20 20 20         options
44000030: 20 20 20 20 20 64 75 6d 70 20 20 20 20 20 66 73         dump     fs
44000040: 63 6b 20 6f 72 64 65 72 0a 70 72 6f 63 20 20 20    ck order.proc
44000050: 20 20 20 20 20 2f 70 72 6f 63 20 20 20 20 20 20         /proc
44000060: 20 20 20 09 70 72 6f 63 20 20 20 20 20 64 65 66       .proc     def
44000070: 61 75 6c 74 73 20 20 20 20 20 09 30 20 20 20 20    aults     .0
```

fatinfo

- fatinfo命令用于查询指定MMC设备分区的文件系统信息
  - fatinfo <interface> [<dev[:part]>]
    - interface表示接口,比如mmc,
    - dev是查询的设备号
    - part是要查询的分区

```
u-boot # fatinfo mmc 0:1
Interface:  MMC
  Device 0: Vendor: Man 000015 Snr fc06e8b8 Rev: 0.3 Prod: AJTD4R
            Type: Removable Hard Disk
            Capacity: 14910.0 MB = 14.5 GB (30535680 x 512)
Filesystem: FAT32 "NO NAME    "
```

mmc read

- mmc read命令用于读取mmc设备的数据
  - mmc read addr blk# cnt
    - addr是数据读取到DRAM中的地址
    - blk是要读取的块起始地址(十六进制)
      一个块是512字节,这里的块和扇区是一个意思,在MMC设备中我们通常说扇区
    - cnt 是要读取的块数量(十六进制)

举例

- 从EMMC的第 13408(0x3460)个块开始,读取160(0xa0)个块的数据到DRAM的0x41000000地址处,

  - mmc read 0x41000000 0x3460 0xa0
  - md.b 0x41000000 100

dtb

```
u-boot # mmc read 0x41000000 0x3460 0xa0

MMC read: dev # 0, block # 13408, count 160 ... 160 blocks read: OK
u-boot # md.b 0x41000000 100
41000000: d0 0d fe ed 00 00 df 20 00 00 00 38 00 00 d3 28    ....... ...8...(
41000010: 00 00 00 28 00 00 00 11 00 00 00 10 00 00 00 00    ...(............
41000020: 00 00 0b f8 00 00 d2 f0 00 00 00 00 00 00 00 00    ................
41000030: 00 00 00 00 00 00 00 00 00 00 00 01 00 00 00 00    ................
41000040: 00 00 00 03 00 00 00 04 00 00 00 00 00 00 00 01    ................
41000050: 00 00 00 03 00 00 00 04 00 00 00 11 00 00 00 01    ................
41000060: 00 00 00 03 00 00 00 04 00 00 00 20 00 00 00 01    ........... ....
41000070: 00 00 00 03 00 00 00 39 00 00 00 2c 74 6f 70 65    .......9...,tope
41000080: 65 74 2c 69 74 6f 70 34 34 31 32 2d 65 6c 69 74    et,itop4412-elit
41000090: 65 00 73 61 6d 73 75 6e 67 2c 65 78 79 6e 6f 73    e.samsung,exynos
410000a0: 34 34 31 32 00 73 61 6d 73 75 6e 67 2c 65 78 79    4412.samsung,exy
410000b0: 6e 6f 73 34 00 00 00 00 00 00 00 03 00 00 00 31    nos4...........1
410000c0: 00 00 00 37 54 4f 50 45 45 54 20 69 54 6f 70 20    ...7TOPEET iTop
410000d0: 34 34 31 32 20 45 6c 69 74 65 20 62 6f 61 72 64    4412 Elite board
410000e0: 20 62 61 73 65 64 20 6f 6e 20 45 78 79 6e 6f 73     based on Exynos
410000f0: 34 34 31 32 00 00 00 00 00 00 00 01 73 6f 63 00    4412........soc.
u-boot #
```

mmc write 命令

- 要将数据写到MMC设备里面
  - mmc write addr blk# cnt
    - addr是要写入MMC中的数据在DRAM中的起始地址
    - blk是要写入MMC的块起始地址(十六进制)
    - cnt是要写入的块大小,一个块为512字节

举例

- 通过fastboot将uboot.bin烧写到mmc中，其实就是通过mmc write

```
u-boot # fastboot 0
USB PHY0 Enable
crq->brequest:0x0
Starting download of 486998 bytes
...
downloading of 486998 bytes finished
---bootloader---
switch to partitions #0, OK
mmc0(part 0) is current device

MMC write: dev # 0, block # 0, count 1038 ... 1038 blocks written: OK
eMMC CLOSE Success.!!mmc0(part 0) is current device
```

下载文件

切换分区

写入到mmc

- u-boot.bin大小为486998字节
- 向emmc中写入1038个块

mmc erase

- 如果要擦除MMC设备的指定块
  - mmc erase blk# cnt
    - blk为要擦除的起始块
    - cnt是要擦除的数量

emmc分区与linux中文件名对应关系

- mmcblk1 ←----------→ eMMC 的块设备;
- mmcblk1boot0 ←----------→ mmcblk1boot1 对应两个 Boot Area Partitions;
- mmcblk1rpmb ←----------→则为 RPMB Partition;
- mmcblk1px ←----------→ UDA 划分出来的 SW Partitions;
- GPP ←----------→ mmcblk1gp1、mmcblk1gp2、mmcblk1gp3、mmcblk1gp4;

Linux中分区设备名

```
~ # ls /dev/mmcblk1* -l
brw-rw----    1 root     root      179,   0 Jan  1  1970 /dev/mmcblk1
brw-rw----    1 root     root      179,  16 Jan  1  1970 /dev/mmcblk1boot0
brw-rw----    1 root     root      179,  32 Jan  1  1970 /dev/mmcblk1boot1
brw-rw----    1 root     root      179,   1 Jan  1  1970 /dev/mmcblk1p1
brw-rw----    1 root     root      179,   2 Jan  1  1970 /dev/mmcblk1p2
brw-rw----    1 root     root      179,   3 Jan  1  1970 /dev/mmcblk1p3
brw-rw----    1 root     root      179,   4 Jan  1  1970 /dev/mmcblk1p4
brw-rw----    1 root     root      179,  48 Jan  1  1970 /dev/mmcblk1rpmb
```

挂载的文件系统system.img

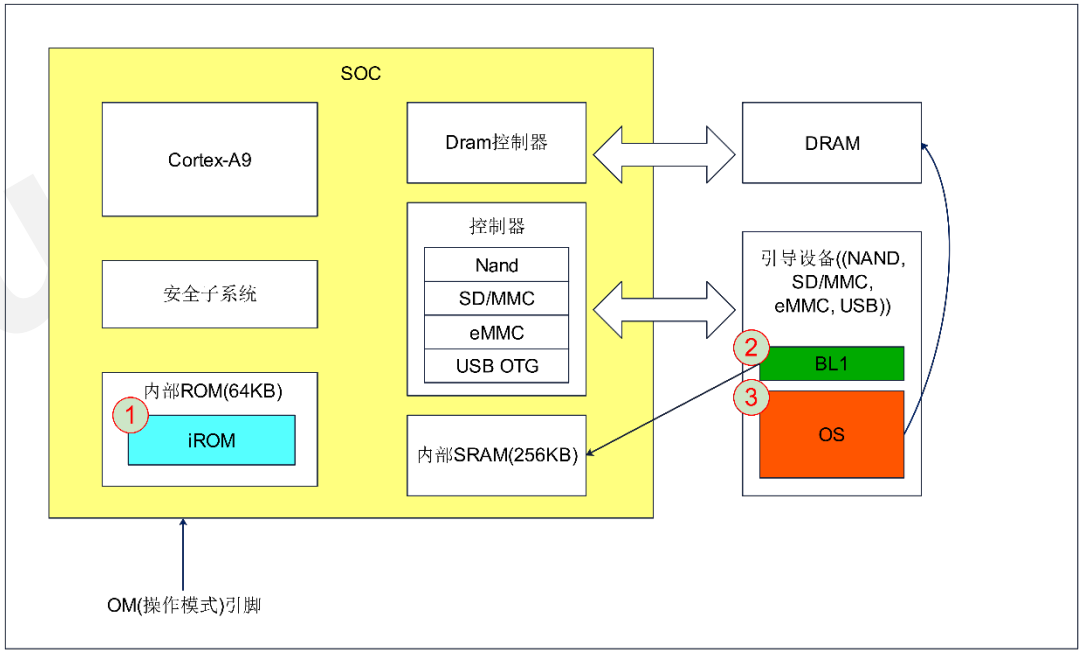
Uboot中分区号

```
u-boot # mmc part

Partition Map for MMC device 0  --   Partition Type: DOS

Part    Start Sector    Num Sectors     UUID            Type
  1     4841472         25673728        00000000-01     0c
  2     32768           2097152         00000000-02     83
  3     2129920         2097152         00000000-03     83
  4     4227072         614400          00000000-04     83
```

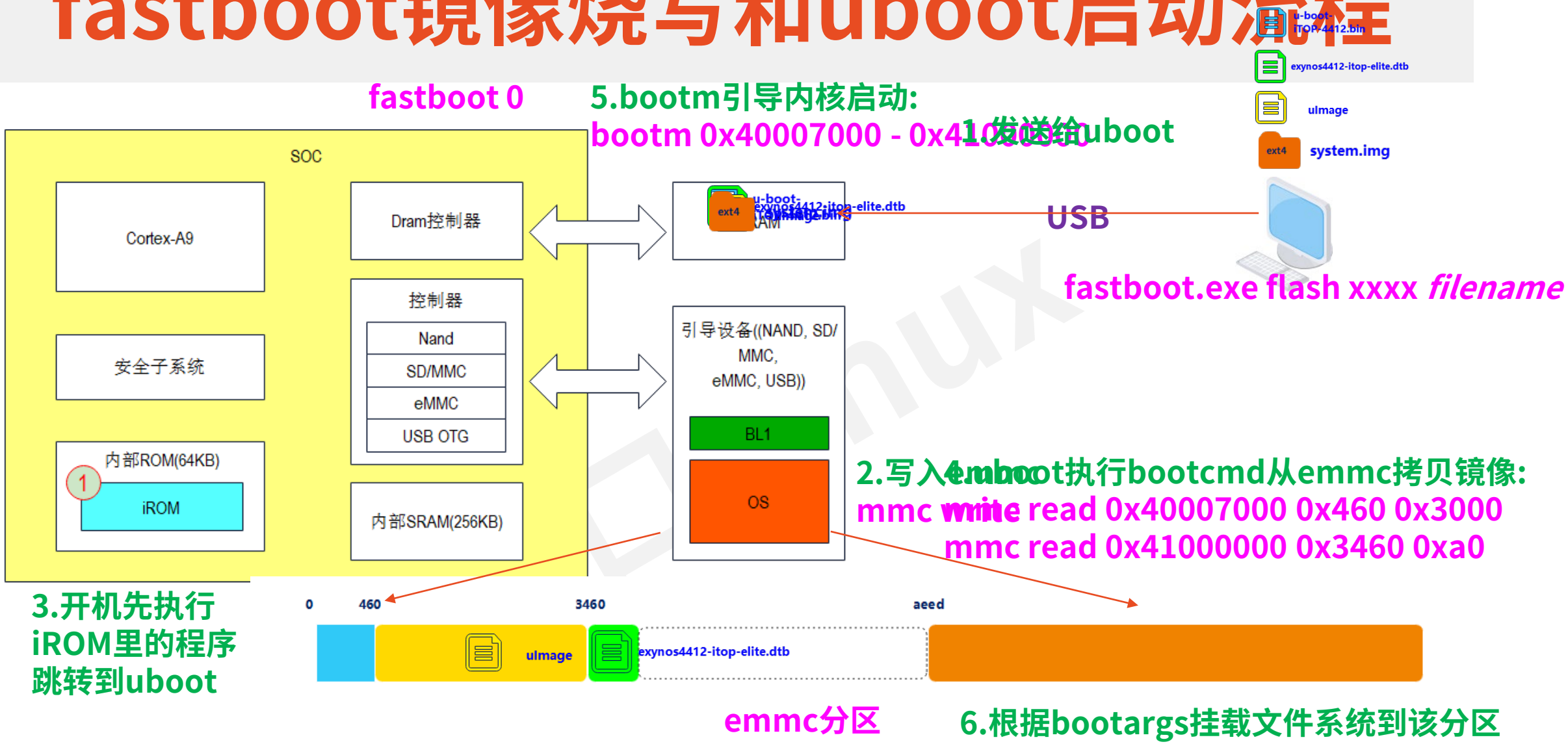
讯为4412全能版-uboot2017启动参数分析

Uboot启动一般步骤

- 1.执行内部只读存储器iROM中的一段代码（厂家固化在里面的）
  - iROM在SOC内部，是一个64KB的ROM，这段代码主要是初始化一些系统的基本配置，比如配置初步时钟、堆栈、启动模式；
  - iROM中的代码根据阶段一获取的启动模式（OM_STAT寄存器），从相应的存储介质中拷贝BL1镜像到SOC内部SRAM中。
  - 启动的外设（NAND、SD、eMMC或者USB）由操作按键来决定的，根据不同按键的值，iROM将会对BL1镜像文件做校验。
- 2.BL1主要是完善系统时钟的初始化工作、内存控制器一些时序的配置。
- 3.做完这些工作后把OS镜像拷贝到内存中
- 3) BL1 是三星公司提供的。BL1又把启动设备上另一个特定位置处的程序读入片内内存，并执行。这个程序被称为BL2（Bootloader2）



fastboot镜像烧写和uboot启动流程



环境变量

```
u-boot # printenv
arch=arm
baudrate=115200
board=itop4412
board_name=itop4412
bootargs=root=/dev/mmcblk1p2 rw console=ttySAC2,115200 init=/linuxrc earlyprintk
bootcmd=if mmc rescan; then echo SD/MMC found on device ${mmcdev};if run loadbootenv; then echo Loaded
environment from ${bootenv};run importbootenv;fi;if test -n $uenvcmd; then echo Running uenvcmd ...;run
 uenvcmd;fi;if run loadbootscript; then run bootscript; fi; fi;mmc read ${loadaddr} 0x460 0x3000; mmc r
ead ${dtb_addr} 0x3460 0xa0; bootm ${loadaddr} - ${dtb_addr}load mmc ${mmcdev} ${loadaddr} uImage; load
 mmc ${mmcdev} ${dtb_addr} ${dtb_name}; bootm ${loadaddr} - ${dtb_addr}
bootdelay=5
bootenv=uEnv.txt
bootscript=echo Running bootscript from mmc${mmcdev} ...; source ${loadaddr}
console=ttySAC2,115200n8
cpu=armv7
dtb_addr=0x41000000
dtb_name=exynos4412-itop-elite.dtb
fdtcontroladdr=7ae4b9f8
importbootenv=echo Importing environment from mmc ...; env import -t $loadaddr $filesize
kerneladdr=0x40007000
loadaddr=0x40007000
loadbootenv=load mmc ${mmcdev} ${loadaddr} ${bootenv}
loadbootscript=load mmc ${mmcdev} ${loadaddr} boot.scr
mmcdev=0
ramdiskaddr=0x48000000
rdaddr=0x48000000
soc=exynos
vendor=samsung
```

比较重要的参数1

- baudrate=115200
- bootenv=uEnv.txt
- console=ttySAC2,115200n8
- dtb_addr=0x41000000
- dtb_name=exynos4412-itop-elite.dtb
- kerneladdr=0x40007000
- loadaddr=0x40007000
- mmcdev=0

includ/configs/itop4412.h

比较重要的参数2

- bootargs=root=/dev/mmcblk1p2 rw console=ttySAC2,115200 init=/linuxrc earlyprintk
- bootcmd=if mmc rescan; then echo SD/MMC found on device ${mmcdev};if run loadbootenv; then echo Loaded environment from ${bootenv};run importbootenv;fi;if test -n $uenvcmd; then echo Running uenvcmd ...;run uenvcmd;fi;if run loadbootscript; then run bootscript; fi; fi;mmc read ${loadaddr} 0x460 0x3000; mmc read ${dtb_addr} 0x3460 0xa0; bootm ${loadaddr} - ${dtb_addr}load mmc ${mmcdev} ${loadaddr} uImage; load mmc ${mmcdev} ${dtb_addr} ${dtb_name}; bootm ${loadaddr} - ${dtb_addr}

bootcmd分析

```
8     bootcmd=
9         if mmc rescan;
10            then echo SD/MMC found on device ${mmcdev};
11            if run loadbootenv;
12                then echo Loaded environment from ${bootenv};
13                run importbootenv;
14            fi;
15            if test -n $uenvcmd;
16                then echo Running uenvcmd ...;
17                run uenvcmd;
18            fi;
19            if run loadbootscript;
20                then run bootscript;
21            fi;
22        fi;
23        mmc read ${loadaddr} 0x460 0x3000;
24        mmc read ${dtb_addr} 0x3460 0xa0;
25        bootm ${loadaddr} - ${dtb_addr}
26        load mmc ${mmcdev} ${loadaddr} uImage;
27        load mmc ${mmcdev} ${dtb_addr} ${dtb_name};
28        bootm ${loadaddr} - ${dtb_addr}
```

最终 核心命令

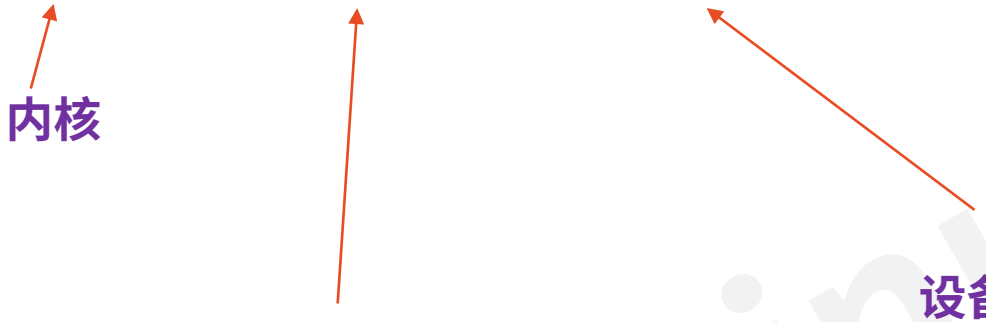
- uImage
  - mmc read 0x40007000 0x460 0x3000
- dtb
  - mmc read 0x41000000 0x3460 0xa0
- 启动linux内核
  - bootm 0x40007000 - 0x41000000

bootz、 bootm 和 boot

- uboot 引导启动Linux命令： bootz、 bootm 和 boot。
  - bootz
    - 命令用于自动 zImage 镜像文件
  - bootm
    - 用于启动 uImage 镜像文件
  - boot
    - boot 读取环境变量 bootcmd 来启动 Linux 系统

bootm

- bootm 0x40007000 - 0x41000000



ramdisk.img
没有指定的话，
内核就根据bootargs设
置挂载对应分区

还有一种image.ub格式的镜像，包
含内核、设备树和打包的rootfs

bootargs

```
lrwxrwxrwx 1 peng peng       11 Aug 10  2014 linuxrc -> bin/busybox
```

- bootargs=root=/dev/mmcblk1p2 rw console=ttySAC2,115200 init=/linuxrc earlyprintk

179:2

参数	含义
root=/dev/mmcblk1p2	根文件系统存放在 mmcblk1 设备的分区 2 ，即EMMC 的分区 2 。/dev/mmcblkx(x=0~n)表示 mmc 设备，而/dev/mmcblkxpy(x=0n,y=1n)表示 mmc 设备 x 的分区 y
rw	表示根文件系统是可以读写的，不加 rw 的话可能无法在根文件系统中进行写操作，只能进行读操作。
console=ttySAC2	设置 linux 终端
115200	设置串口的波特率
init=/linuxrc	init指定的是内核启起来后，进入系统中运行的第一个脚本
earlyprintk	内核console启动之前的调试输出接口
rootwait	表示等待 mmc 设备初始化完成以后再挂载，否则的话 mmc 设备还没初始化完成就挂载根文件系统会出错的。
root=/dev/nfs	根文件系统存放在 nfs 挂载目录

bootargs-root参数

存储介质	大小	标识	描述	举例
nor flash	16MByte~32MByte	mtdX		
nand Flash	128MByte+	mtdblockX(X：0,1,2,3…)		root=/dev/mtdblock1 rw root=/dev/mtdblock2 rw
emmc	~	mmcblkXpY(X=0~ n,Y=1~ n具体要看系统识别出来哪些)		root=/dev/mmcblk0p1 rw root=/dev/mmcblk0p2 rw
SD/TF卡	~	mmcblkXpY(X=0~ n,Y=1~ n具体要看系统识别出来哪些)	SD/TF卡本质上就是emmc	root=/dev/mmcblk0p1 rw root=/dev/mmcblk0p2 rw
内存	~	ram	内存文件系统	
网络		nfs	网络文件系统	root=/dev/nfs

uboot启动参数补充-重要

解压uboot源码

- Uboot2010源码
  - iTOP-4412全功能版\06_源码_uboot和kernel\iTop4412_uboot_20180320.tar.gz
- uboot2017
  - xunwei\uboot+kernel\itop4412_kernel_4_14_2_bsp_SCP-20200616.tar.gz

1. uboot中环境变量从哪来的?

- include/configs/itop4412.h

```
67 #define COPY_BL2_FNPTR_ADDR     0x02020030
68 #define CONFIG_SPL_TEXT_BASE    0x02023400 /* 0x02021410 */
69
70 #define CONFIG_EXTRA_ENV_SETTINGS \
71     "loadaddr=0x40007000\0" \
72     "rdaddr=0x48000000\0" \
73     "kerneladdr=0x40007000\0" \
74     "ramdiskaddr=0x48000000\0" \
75     "console=ttySAC2,115200n8\0" \
76     "mmcdev=0\0" \
77     "bootenv=uEnv.txt\0" \
78     "dtb_addr=0x41000000\0" \
79     "dtb_name=exynos4412-itop-elite.dtb\0" \
80     "loadbootenv=load mmc ${mmcdev} ${loadaddr} ${bootenv}\0" \
81     "bootargs=root=/dev/mmcblk1p2 rw console=ttySAC2,115200 init=/linuxrc earlyprintk\0" \
82     "importbootenv=echo Importing environment from mmc ...; " \
83     "env import -t $loadaddr $filesize\0" \
84     "loadbootscript=load mmc ${mmcdev} ${loadaddr} boot.scr\0" \
85     "bootscript=echo Running bootscript from mmc${mmcdev} ...; " \
86     "source ${loadaddr}\0"
87 #define CONFIG_BOOTCOMMAND \
88     "if mmc rescan; then " \
89         "echo SD/MMC found on device ${mmcdev};" \
90         "if run loadbootenv; then " \
91             "echo Loaded environment from ${bootenv};" \
92             "run importbootenv;" \
93         "fi;" \
94         "if test -n $uenvcmd; then " \
95             "echo Running uenvcmd ...;" \
96             "run uenvcmd;" \
97         "fi;" \
98         "if run loadbootscript; then " \
99             "run bootscript; " \
100        "fi; " \
101    "fi;" \
102    "mmc read ${loadaddr} 0x460 0x3000; mmc read ${dtb_addr} 0x3460 0xa0; bootm ${loadaddr} - ${dtb_addr}" \
103    "load mmc ${mmcdev} ${loadaddr} uImage; load mmc ${mmcdev} ${dtb_addr} ${dtb_name}; bootm ${loadaddr} - ${dtb_addr}"
104
105 #define CONFIG_CLK_1000_400_200
106
107 /* MIU (Memory Interleaving Unit) */
108 #define CONFIG_MIU_2BIT_21_7_INTERLEAVED
109
include/configs/itop4412.h [RO]
```

Uboot-2017源码

2. 下载地址，ram地址

Base Address	Limit Address	Size	Description
0x0000_0000	0x0001_0000	64 KB	iROM
0x0200_0000	0x0201_0000	64 KB	iROM (mirror of 0x0 to 0x10000)
0x0202_0000	0x0206_0000	256 KB	iRAM
0x0300_0000	0x0302_0000	128 KB	Data memory or general purpose of Samsung Reconfigurable Processor SRP.
0x0302_0000	0x0303_0000	64 KB	I-cache or general purpose of SRP.
0x0303_0000	0x0303_9000	36 KB	Configuration memory (write only) of SRP
0x0381_0000	0x0383_0000	–	AudioSS's SFR region
0x0400_0000	0x0500_0000	16 MB	Bank0 of Static Read Only Memory Controller (SMC) (16-bit only)
0x0500_0000	0x0600_0000	16 MB	Bank1 of SMC
0x0600_0000	0x0700_0000	16 MB	Bank2 of SMC
0x0700_0000	0x0800_0000	16 MB	Bank3 of SMC
0x0800_0000	0x0C00_0000	64 MB	Reserved
0x0C00_0000	0x0CD0_0000	–	Reserved
0x0CE0_0000	0x0D00_0000	–	SFR region of Nand Flash Controller (NFCON)
0x1000_0000	0x1400_0000	–	SFR region
0x4000_0000	0xA000_0000	1.5 GB	Memory of Dynamic Memory Controller (DMC)-0
0xA000_0000	0x0000_0000	1.5 GB	Memory of DMC-1

3. emmc中存储镜像首地址在哪定义的?

- drivers/usb/gadget/f_fastboot.c
  - uboot ： 0
  - kernel： 460
  - dtb ： 3460
  - system： aeed
- 支持命令
  - bootloader
  - kernel
  - dtb
  - system

```
669 static void cb_flash(struct usb_ep *ep, struct usb_request *req)
670 {
671     char *cmd = req->buf;
672     char response[FASTBOOT_RESPONSE_LEN];
673
674     char *argv[2]  = { "md","40000000"};
675         char *open_emmc[2]  = { "open","    "};
676         char *close_emmc[2]  = { "close","    "};
677         char *argv_uboot[5]  = { "mmc","write", "40000000","0","40e"};
678         char *argv_kernel[5]  = { "mmc","write", "40000000","460","3000"};
679         char *argv_dtb[5]  = { "mmc","write", "40000000","3460","a0"};
680         char *argv_system[5]  = { "mmc","write", "40000000","aeed","a000"};
681
```

Uboot-2017源码

4. mmc分区及格式化

- uboot2017没有fdisk，出厂是用uboot2010烧写的
  - fdisk -c 0 （0代表eMMC，1代表TF卡）
    - fdisk -c 1 300 300 300
  - fatformat mmc 0:1
  - ext3format mmc 0:2
  - ext3format mmc 0:3
  - ext3format mmc 0:4
  - fastboot

Uboot-2010源码

5.Emmc各个分区含义

- 三星平台一般把emmc（或者NAND）分为四个区
  - （1）、fat分区,作为sd卡用；
  - （2）、系统分区，相当为电脑c 盘，用来安装android系统；
  - （3）、userdata分区；
  - （4）、cache分区。

do_fdisk

- do_fdisk
- create_mmc_fdisk
- make_mmc_partition

```
17: #define     BLOCK_SIZE          512
18: #define     BLOCK_END           0xFFFFFFFF
19: #define     _10MB               (10*1024*1024)
20: #define     _100MB              (100*1024*1024)
21: #define     _300MB              (300*1024*1024)
22: #define     _8_4GB              (1023*254*63)
23: #define     _1GB                (1024*1024*1024)
24: #define     DISK_START          RAW_AREA_SIZE//mj (16*1024*1024) //same as raw area size
25:
26: #define     SYSTEM_PART_SIZE        _1GB //_300MB
27: #define     USER_DATA_PART_SIZE     _1GB //_300MB //_1GB
28: #define     CACHE_PART_SIZE         _300MB
```

```
247:     block_start = calc_unit(DISK_START, sdInfo);
248: /* modify by cym 20131206 */
249: #if 0
250:     block_offset    = calc_unit(SYSTEM_PART_SIZE, sdInfo);
251: #else
252:     if (flag)
253:         block_offset = calc_unit((unsigned long long)simple_strtoul(argv[3], NULL, 0)*1024*1024, sdInfo);
254:     else
255:         block_offset = calc_unit(SYSTEM_PART_SIZE, sdInfo);
256: #endif
257: /* end modify */
258:
259:     partInfo[0].bootable    = 0x00;
260:     partInfo[0].partitionId = 0x83;
261:
262:     make_partitionInfo(block_start, block_offset, sdInfo, &partInfo[0]);
```

uboot2010
cmd_mmc_fdisk.c

Uboot手动引导Linux内核举例

mmc 串口 sd卡 网络

测试1 从mmc自动加载镜像

- 目标：从mmc中读取内核和设备树，启动linux后挂载emmc分区中的文件系统

- 前置条件：

1. 我们已经通过fastboot或者其他方式烧录了内核和设备树到emmc分区中
2. 没有ramdisk,需要bootargs设置挂载文件位置 root=/dev/mmcblk1p2 ，并且system.img写到emmc分区中
    - uImage
        - mmc read 0x40007000 0x460 0x3000
    - dtb
        - mmc read 0x41000000 0x3460 0xa0
    - 启动linux内核
        - bootm 0x40007000 - 0x41000000
    - bootargs用讯为默认的
        - bootargs=root=/dev/mmcblk1p2 rw console=ttySAC2,115200 init=/linuxrc earlyprintk

合并命令到bootcmd

```
# setenv bootcmd mmc read 0x40007000 0x460 0x3000\; mmc read 0x41000000 0x3460 0xa0\;
bootm 40007000 – 41000000

setenv  bootargs  root=/dev/mmcblk1p2 rw console=ttySAC2,115200 init=/linuxrc earlyprintk
```

测试2 串口下载镜像

- 目标：通过loadb即串口下载内核设备树和ramdisk,并启动内核

- 前置条件：
  - 有串口+支持kermit协议

速率比较慢
Bootargs不需要

  - uImage
    - loadb 0x40007000
  - dtb
    - loadb 0x41000000
  - ramdisk
    - loadb 0x44000000
  - 启动linux内核
    - bootm 0x40007000 0x44000000 0x41000000

测试3 sd卡下载镜像

- 目标：通过sd卡手动下载内核、设备树和ramdisk

- 前置条件：
  - 支持sd卡槽
  - Sd卡文件格式uboot要支持

前面分区的sd卡是ntfs格式，用fat格式格式化

```
mmc list
mmc dev 1
fatls mmc 1:1
```

- uImage
  - fatload mmc 1:1 0x40007000 uImage
- dtb
  - fatload mmc 1:1 0x41000000 exynos4412-itop-elite-yikou.dtb
- ramdisk
  - fatload mmc 1:1 0x44000000 ramdisk.img
- 启动linux内核
  - bootm 0x40007000 0x44000000 0x41000000

- **setenv bootcmd** fatload mmc 1:1 0x40007000 uImage\; fatload mmc 1:1 0x41000000 exynos4412-itop-elite-yikou.dtb\; fatload mmc 1:1 0x44000000 ramdisk.img\; bootm 0x40007000 0x44000000 0x41000000

测试4 网络自动下载镜像

- 目标：通过tftp即串口下载内核、设备树和ramdisk，内核启动后解压ramdisk到ram中并挂载

下载完毕之后，可以通过mmc write将文件写入到emmc中

提供的sdk，uboot不支持网络功能，需要移植网络

- 前置条件：
  - Uboot支持网络
  - 内核必须支持

- uImage
  - tftp 0x40007000 uImage
- dtb
  - tftp 0x41000000 exynos4412-itop-elite.dtb
- ramdisk
  - tftp 0x44000000 ramdisk.img
- 启动linux内核
  - bootm 0x40007000 0x44000000 0x41000000
- 不需要设置bootargs

合并命令到bootcmd

```
# setenv bootcmd tftp 40007000 uImage\;tftp 41000000 exynos4412-itop-elite.dtb\;tftp
44000000  ramdisk.img\;bootm 40007000 44000000 41000000
```

内核支持ramdisk

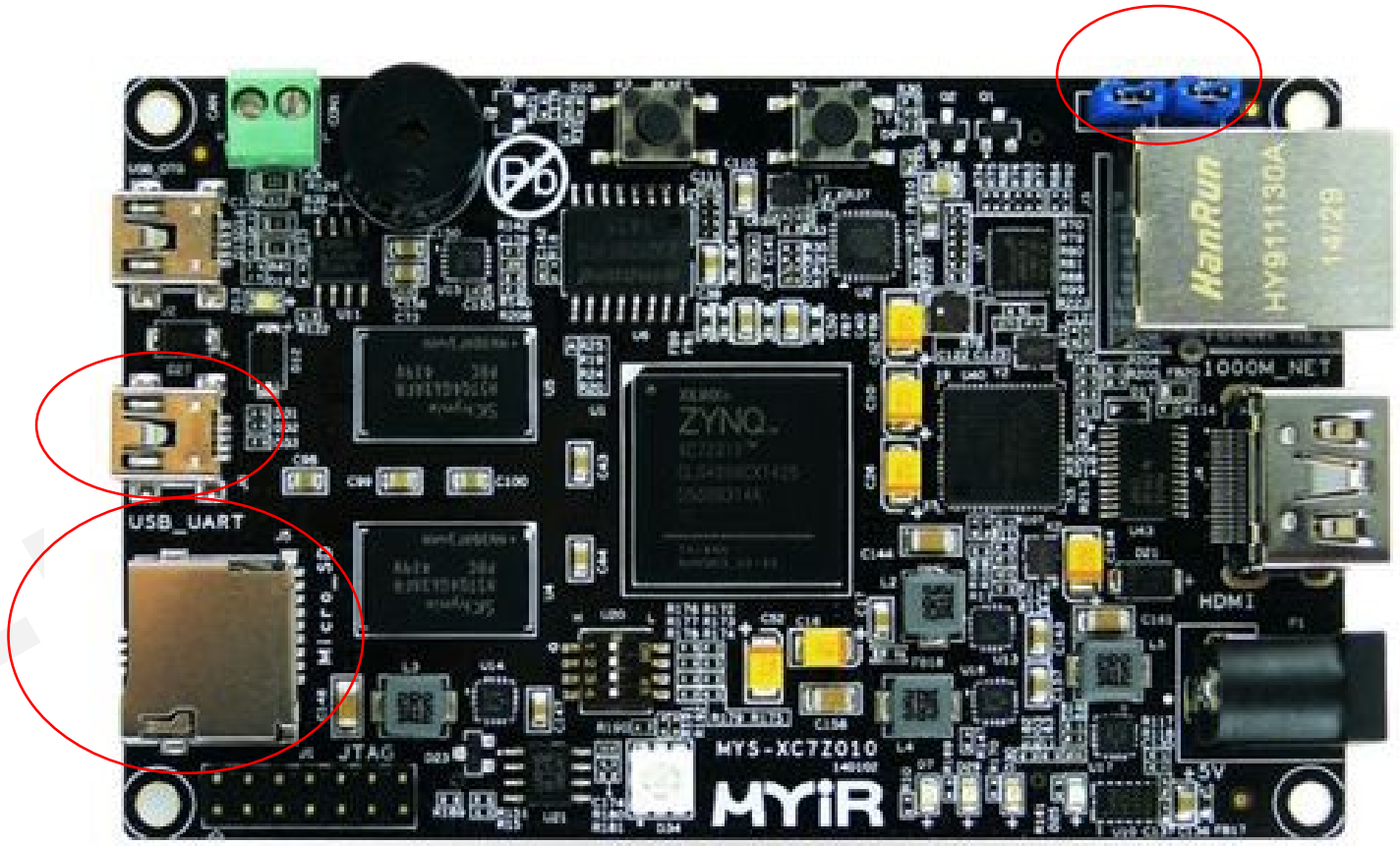
```
make menuconfig
        File systems  --->
                <*> Second extended fs support
        Device Drivers
                SCSI device support  --->
                        <*> SCSI disk support
                Block devices  --->
                        <*>RAM  block  device  support
                        (16)Default number of RAM disks
                        (8192) Default RAM disk size (kbytes)   (修改为8M)
        General setup  --->
                [*] Initial RAM filesystem and RAM disk (initramfs/initrd) support
```

- 还有很多其他基于不同设备(nand flash、usb)的启动命令
-

Zynq zturn
uboot启动参数分析

zynq

- Arm+fpga



镜像

- 7z020.bit
- uEnv.txt
- BOOT.bin
- devicetree.dtb
- uImage
- uramdisk.image.gz

Zynq zturn启动方式说明

- 只有2种
  - sd卡
  - qspi
- 并且只支持ramdisk，
- 如果文件系统部署在其他介质，需要自己修改脚本

1. Sd卡操作

- 查看sd卡分区
  - mmc list
  - mmc part
  - fatls mmc 0:1

  - run sdboot

Sd卡启动脚本

- sdboot
- 思路
  - 从sd卡下载bit、内核、设备树、ramdisk到指定ram地址
  - bootm引导启动linux内核

```
sdboot=
if mmcinfo;
then
    run uenvboot;
    get_bitstream_name
    && echo - load ${bitname} to PL...
    && fatload mmc 0 0x200000 ${bitname}
    && fpga loadb 0 0x200000 ${filesize}
    && echo Copying Linux from SD to RAM...
    && fatload mmc 0 ${kernel_load_address} ${kernel_image}
    && fatload mmc 0 ${devicetree_load_address} ${devicetree_image}
    && fatload mmc 0 ${ramdisk_load_address} ${ramdisk_image}
    && bootm ${kernel_load_address} ${ramdisk_load_address} ${devicetree_load_address};
fi
```

2. Qspi操作

- sf probe 0 0 0
- sf read 0x2000000 0x980000 0x010000

- run qspiboot

Qspi启动脚本

- qspiboot
- 思路
  - 从sd卡下载bit、内核、设备树、ramdisk到指定ram
  - Bootm引导启动linux内核

```
qspiboot=
echo Copying Linux from QSPI flash to RAM...
&& sf probe 0 0 0
&& qspi_get_bitsize 0x0A0000
&& sf read ${loadbit_addr} 0x0A0004 ${bitsize}
&& fpga loadb 0 ${loadbit_addr} ${bitsize}
&& sf read ${kernel_load_address} ${qkernel_addr} ${kernel_size}
&& sf read ${devicetree_load_address} ${qdevtree_addr} ${devicetree_size}
&& echo Copying ramdisk...
&& sf read ${ramdisk_load_address} ${qramdisk_addr} ${ramdisk_size}
&& bootm ${kernel_load_address} ${ramdisk_load_address} ${devicetree_load_address}
```

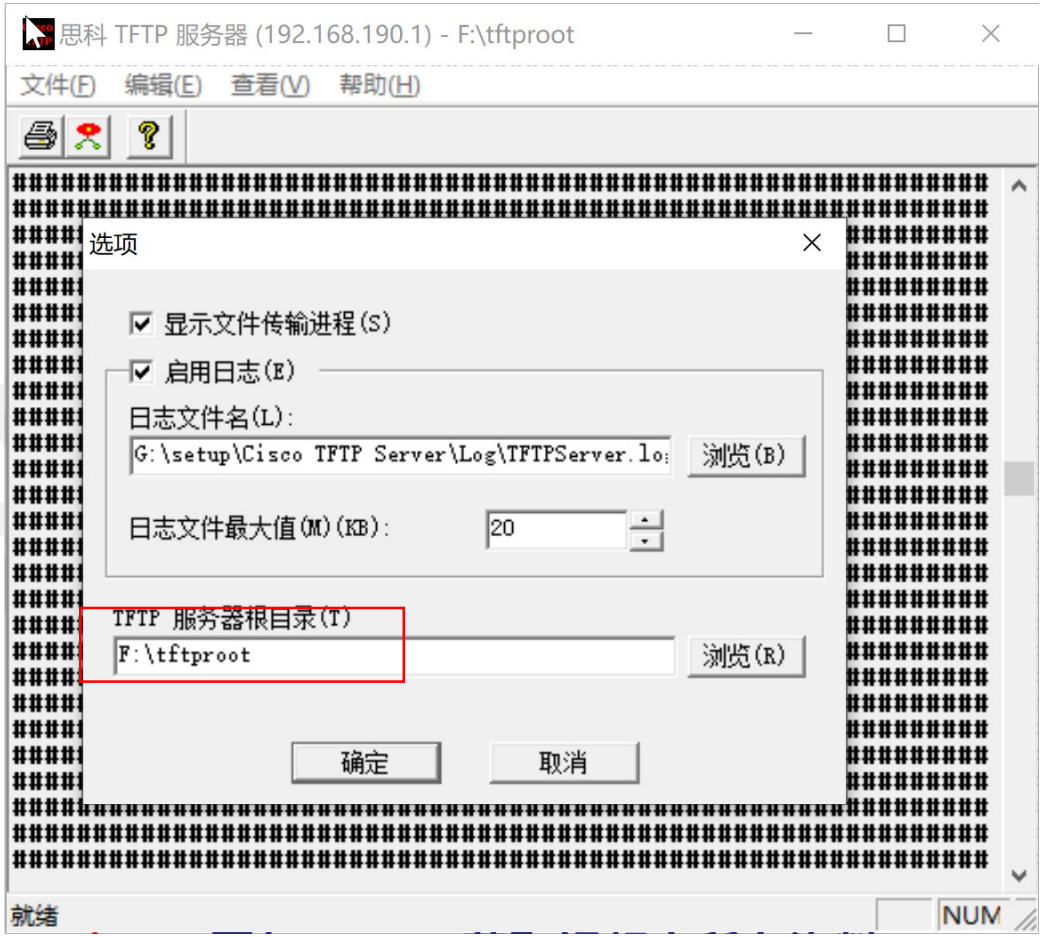
更新镜像到qspi flash

- 1. jtag直接烧录
  - sf write
- 2. qspiupdate
- 思路
  - 从sd卡先读取镜像到ram，
  - 然后再用sf命令写入到qspi flash

```
qspiupdate=
echo Update qspi images from sd card...
&& echo - Init mmc...
&& mmc rescan
&& echo - Init qspi flash...
&& sf probe 0 0 0
&& echo - Write boot.bin...
&& fatload mmc 0 0x200000 boot.bin
&& sf erase ${qboot_addr} ${boot_size}
&& sf erase ${qbootenv_addr} ${qbootenv_size}
&& sf write 0x200000 0 ${filesize}
&& get_bitstream_name
&& echo - Write ${bitstream_image}...
&& fatload mmc 0 0x200000 ${bitstream_image}
&& sf erase 0x0A0000 0x460000
&& mw.l 0x100000 ${filesize}
&& sf write 0x100000 0x0A0000 4
&& sf write 0x200000 0x0A0004 ${filesize}
&& echo - Write uImage...
&& fatload mmc 0 0x200000 uImage
&& sf erase ${qkernel_addr} ${kernel_size}
&& sf write 0x200000 ${qkernel_addr} ${filesize}
&& echo - Write device tree...
&& fatload mmc 0 0x200000 devicetree.dtb
&& sf erase ${qdevtree_addr} ${devicetree_size}
&& sf write 0x200000 ${qdevtree_addr} ${filesize}
&& echo - Write Ramdisk...
&& fatload mmc 0 0x200000 uramdisk.image.gz
&& sf erase ${qramdisk_addr} ${ramdisk_size}
&& sf write 0x200000 ${qramdisk_addr} ${filesize}
&& echo - Done.
```

3. 网络tftp下载镜像

- 搭建tftp服务器



tftp下载镜像

- 思路
- 通过tftp从tftp服务器下载镜像
  - setenv ipaddr 192.168.0.111
  - setenv serverip 192.168.0.102
  - run jtagboot

```
zynq-uboot> print ipaddr
ipaddr=192.168.0.111
zynq-uboot> print serverip
serverip=192.168.0.102
zynq-uboot> run jtagboot
TFTPing Linux to RAM...
Gem.e000b000 Waiting for PHY auto negotiation to complete..... done
Using Gem.e000b000 device
TFTP from server 192.168.0.102; our IP address is 192.168.0.111
Filename 'uImage'.
Load address: 0x2080000
Loading: T #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         ##############################################
         265.6 KiB/s
done
Bytes transferred = 3886152 (3b4c48 hex)
Gem.e000b000:3 is connected to Gem.e000b000.  Reconnecting to Gem.e000b000
Gem.e000b000 Waiting for PHY auto negotiation to complete....
```

```
jtagboot=
echo TFTPing Linux to RAM...
&& tftpboot ${kernel_load_address} ${kernel_image}
&& tftpboot ${devicetree_load_address} ${devicetree_image}
&& tftpboot ${ramdisk_load_address} ${ramdisk_image}
&& bootm ${kernel_load_address} ${ramdisk_load_address} ${devicetree_load_address}
```

其他启动方式

- nandboot
- norboot
- usbboot

- include/configs/zynq-zturn.h

ARM

05

linux挂载nfs文件系统

工作模式

- （1）启动加载模式
- （2）下载模式（开发者模式）

（1）启动加载模式

- 启动加载模式是Bootloader的正常工作模式
- 嵌入式产品发布时，Bootloader必须工作在这种模式下，Bootloader将嵌入式操作系统从FLASH中加载到SDRAM中运行，整个过程是自动的。
  - 通常镜像保存在emmc中，
  - 内核启动后将文件系统挂载在emmc某个分区
  - 文件没有存储在flash中的采用ramdisk文件系统

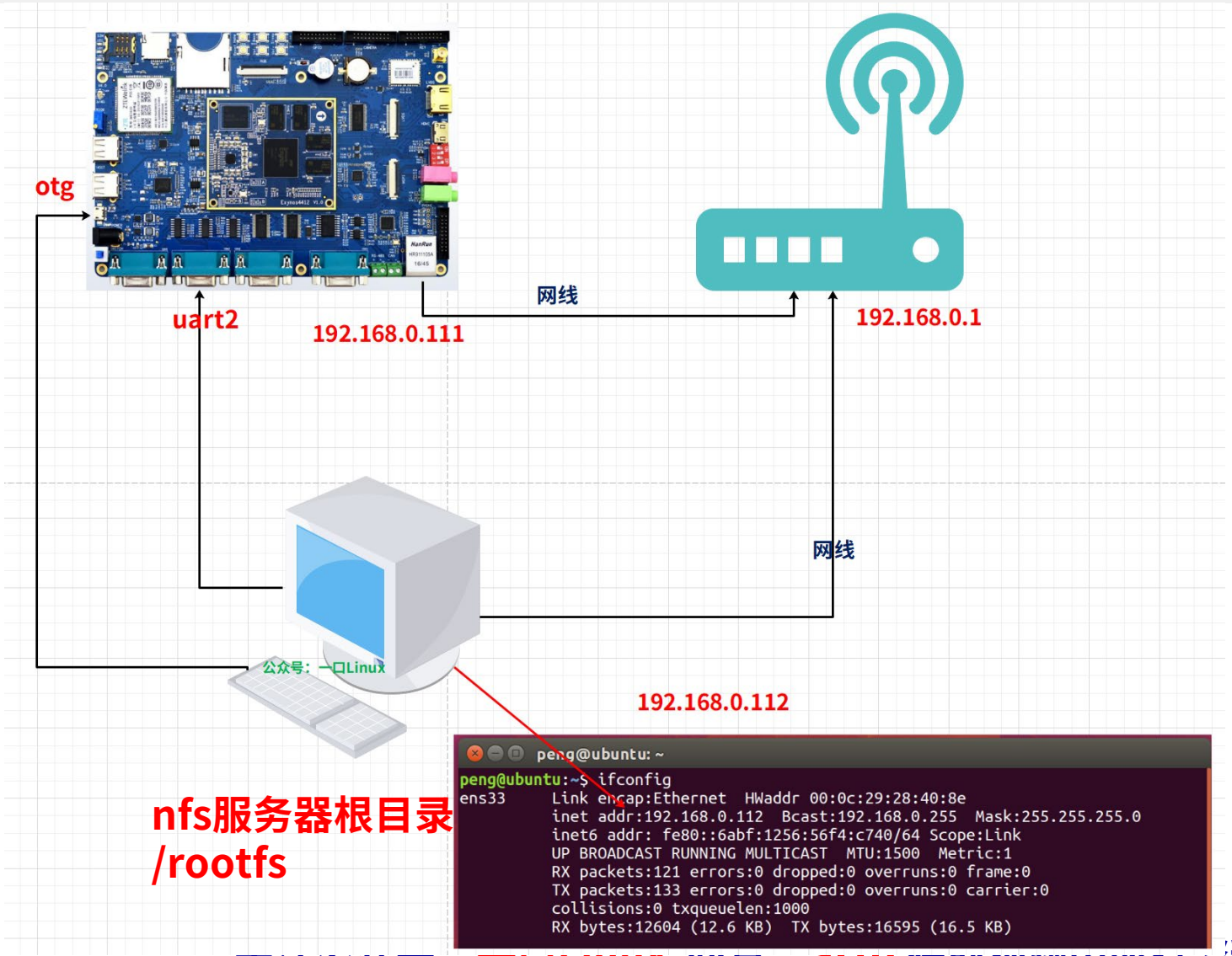
（2）下载模式（开发者模式）

- 下载模式就是Bootloader通过串口连接或网络连接某些通信手段将内核映像或根文件系统映像等从PC机中下载到目标板的FLASH中。
- 用户可以利用Bootloader提供的一些命令接口来完成自己想要的操作。

- 通常的操作是
  - 通过网络tftp下载内核设备树
  - 内核启动后通过nfs挂载文件系统到ubuntu的某个目录中

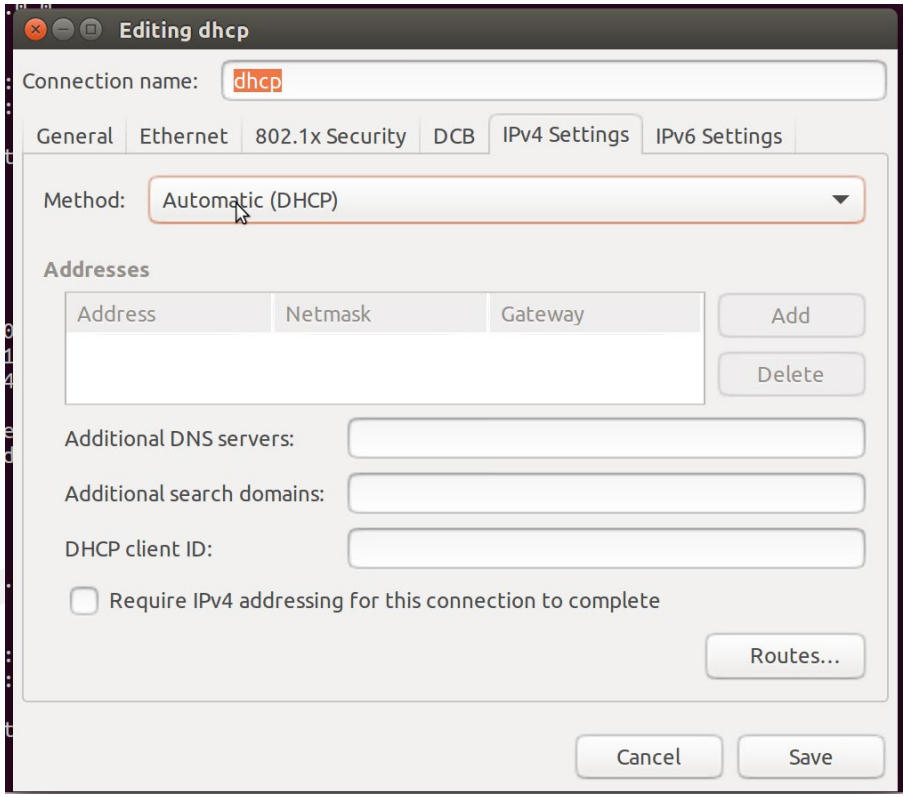
Nfs挂载配置举例

我的组网



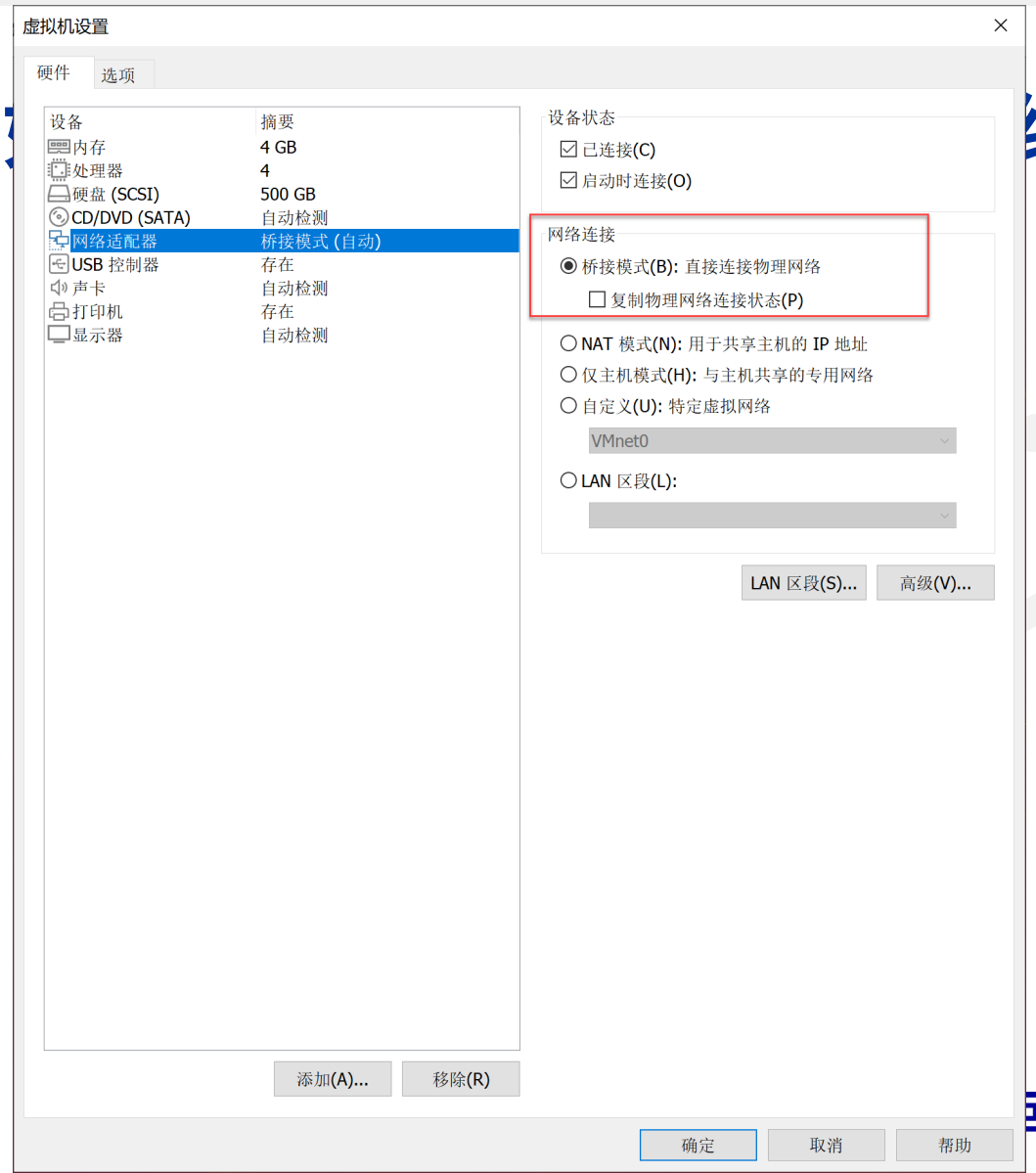
1. 配置Ubuntu ip地址

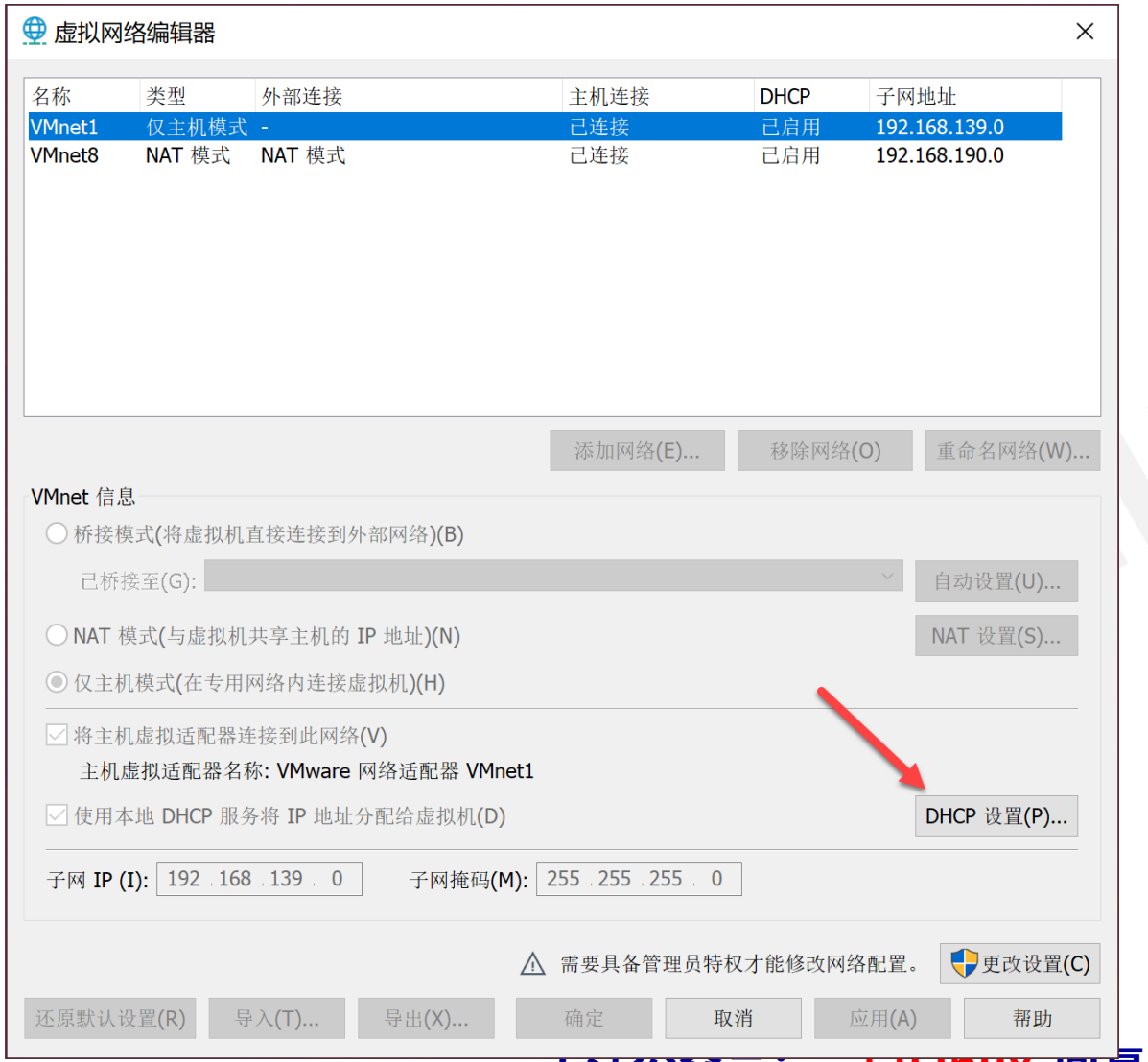
- ubuntu ip：
  - 192.168.0.112

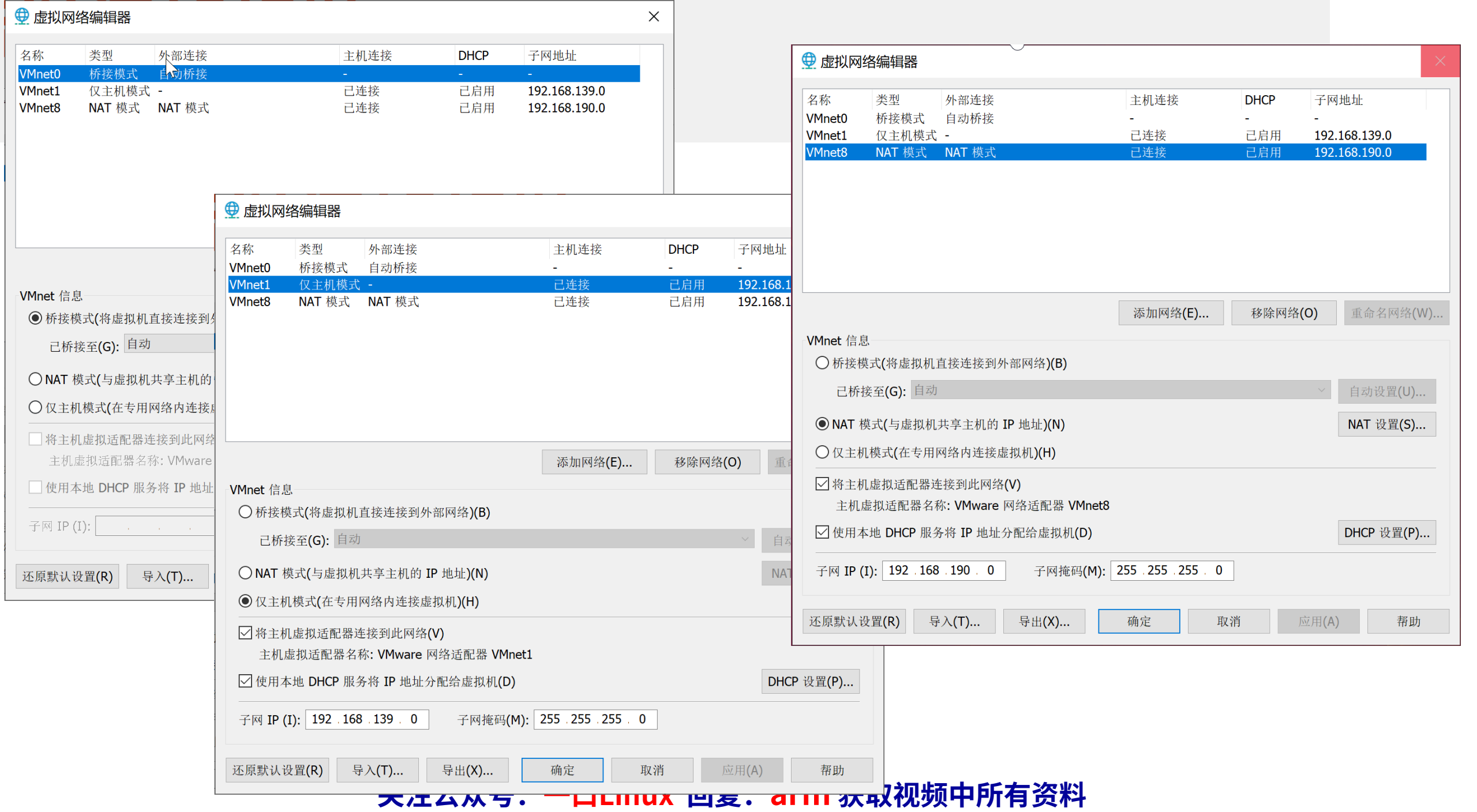


2. Vmware设置

- 线，用有线方式







关注公众号：一口Linux 回复：arm 获取视频中所有资料

3. nfs安装与测试

- 安装
  - sudo apt-get install nfs-kernel-server
- 修改 /etc/exports 文件
  - /rootfs *(rw,sync,no_root_squash,no_subtree_check)
    - /rootfs　　nfs服务器根目录
    - ro　　具有只读权限
    - rw　　具有读写权限
    - no_root_squash　　如果客户端是root的话，那么他对这个目录具有root的权限
    - root_squash　　如果客户端是root的话，那么他的权限被限制为匿名使用者
    - all_squash　　不论客户端是什么身份，他的权限都将被限制为匿名使用者
    - subtree_check　　如果共享/usr/bin之类的子目录时，强制NFS检查父目录的权限（默认）
    - sync　　文件同步写入到内存和硬盘
    - async　　文件先写入到内存，而不是直接写入到硬盘
- 启动 nfs 服务
  - sudo /etc/init.d/nfs-kernel-server restart
- 测试
  - mount -t nfs 127.0.0.1:/rootfs /mnt/nfs 或者 sudo mount -t nfs localhost:/rootfs /mnt/nfs
  - umount /mnt/nfs

Nfs服务器版本问题

- ubuntu17以上版本,默认只支持nfs3、4客户端
  - Vim /etc/default/nfs-kernel-server
  - 增加
    - RPCNFSDOPTS="--nfs-version 2,3,4 --debug --syslog"
  - 重启nfs
    - sudo /etc/init.d/nfs-kernel-server restart

4. 开发板网络设置

- 开发板ip：
  - 192.168.0.111
- U-Boot配置命令：
  - setenv ipaddr 192.168.0.111 ；板子的ip
  - setenv serverip 192.168.0.112 ；虚拟机的ip
  - setenv gatewayip 192.168.0.1 ；网关
  - saveenv ；保存配置

bootcmd

- setenv bootcmd tftp 0x40007000 uImage\；tftp 0x41000000 exynos4412-itop-elite.dtb\；bootm 0x40007000 - 0x41000000
  - bootcmd：
    - U-Boot启动之后，首先先执行找到这个参数，执行后面的命令；
  - tftp：
    - 从serverip 192.168.6.186的根目录下载文件uImage到地址41000000；
  - uImage：
    - 内核镜像；
  - exynos4412-itop-elite.dtb：
    - 设备树文件；
  - bootm 41000000 - 42000000
    - 引导内核，并传入地址。
- 该命令含义是从tftp服务器下载内核镜像uImage到地址0x40007000，设备树文件exynos4412-itop-elite.dtb到0x40007000，并通过命令bootm加载启动内核。

bootargs

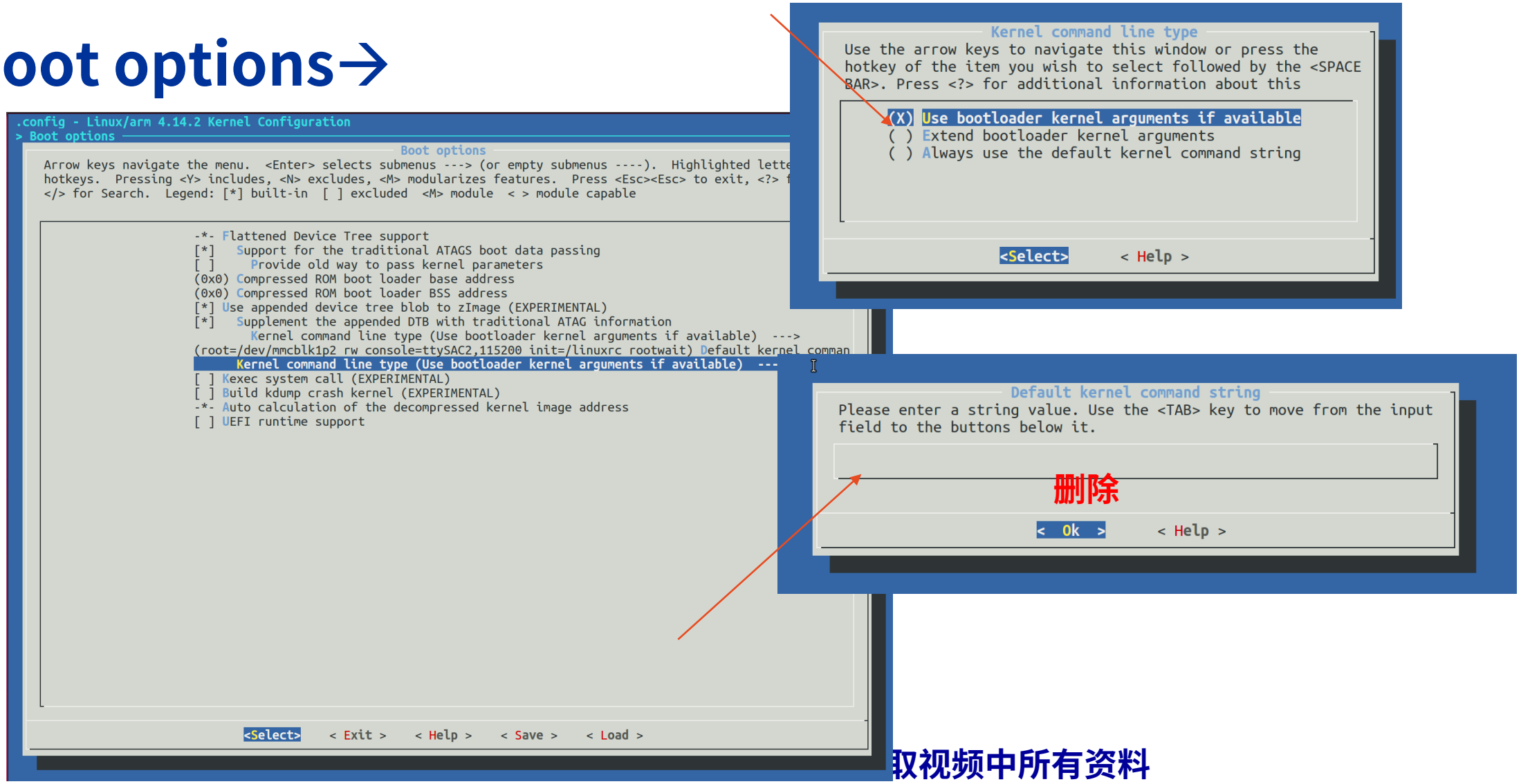
- setenv bootargs root=/dev/nfs nfsroot=192.168.0.112:/rootfs,proto=tcp rw ip=192.168.0.111:192.168.0.112:192.168.0.1:255.255.255.0::eth0:off init=/linuxrc
- 参数含义如下：
  - bootargs
    - 引导内核启动后，内核会去解析该启动参数
  - root=/dev/nfs
    - 通知内核linux内核，指定根文件系统采用NFS网络服务
  - nfsroot=192.168.0.112:/rootfs
    - nfs服务器地址192.168. 0.112，目录为/rootfs，
  - rw
    - 文件系统操作权限为可续写
  - console=ttySAC2,115200
    - 串口名称和波特率
  - ip=192.168. 0.111
    - 开发板地址192.168.0.111　服务器地址192.168.0.112　网关192.168.0.1　掩码255.255.255.0　网卡名eth0　自动配置off
  - init=/linuxrc
    - 内核启动后运行的进程为linuxrc

root=/dev/nfs语法

- root=/dev/nfs nfsroot=[<server-ip>:]<root-dir>[,<nfs-options>] ip=<client-ip>:<server-ip>:<gw-ip>:<netmask>:<hostname>:<device>:<autoconf>:<dns0-ip>:<dns1-ip>
  - <server-ip>:
    - 服务器 IP 地址，也就是存放根文件系统主机的 IP 地址，那就是 Ubuntu 的 IP地址，比如我的 Ubuntu 主机 IP 地址为 192.168.1.250。
  - <root-dir>：
    - 根文件系统的存放路径，/rootfs
  - <nfs-options>:
    - NFS 的其他可选选项，一般不设置。
  - <client-ip>：
    - 客户端 IP 地址，也就是我们开发板的 IP 地址192.168.1.251
    - Linux 内核启动以后就会使用此 IP 地址来配置开发板。此地址一定要和 Ubuntu 主机在同一个网段内，并且没有被其他的设备使用，。
  - <server-ip>：
    - 服务器 IP 地址。
  - <gw-ip>：
    - 网关地址，192.168.1.1。
  - <netmask>:
    - 子网掩码，255.255.255.0。
  - <hostname>:
    - 客户机的名字，一般不设置，此值可以空着。
  - <device>：
    - 设备名，也就是网卡名，一般是 eth0，eth1…
  - <autoconf>：
    - 自动配置，一般不使用，所以设置为 off。
  - <dns0-ip>：
    - DNS0 服务器 IP 地址，不使用。
  - <dns1-ip>：
    - DNS1 服务器 IP 地址，不使用。

4.内核修改

- Boot options→



内核命令行类型说明

- Use bootloader kernel arguments if available ： 使用Uboot提供的参数
- Extend bootloader kernel arguments ： 通过内核自身和UBoot一起
- Always use the default kernel command string：只使用内核自己

例1 linux内核根目录挂载到nfs服务器

- 讯为sdk中的uboot不支持网络，
  - 如果修改了内核或者设备树暂时用fastboot烧录到emmc，
  - 本实验设置bootcmd从emmc读取镜像

- 修改bootargs，root=/dev/nfs……

Uboot配置环境变量

- ubuntu ip：
  - 192.168.0.112
- 开发板ip：
  - 192.168.0.168
- U-Boot配置命令：
  - setenv ipaddr 192.168.0.111
  - setenv serverip 192.168.0.112
  - setenv gatewayip 192.168.0.1
  -
  - setenv bootargs root=/dev/nfs nfsroot=192.168.0.112:/rootfs,proto=tcp rw ip=192.168.0.111:192.168.0.112:192.168.0.1:255.255.255.0::eth0:off init=/linuxrc
  - setenv bootcmd mmc read 0x40007000 0x460 0x3000;mmc read 0x41000000 0x3460 0xa0;bootm 0x40007000 - 0x41000000

例2 直接挂载某个目录到nfs服务器

- 进入linux之后，mount某个目录到nfs服务器
  - ifconfig eth0 192.168.0.111
  - mkdir /mnt/nfs
  - mount -t nfs -o intr,nolock,rsize=1024,wsize=1024 192.168.0.112:/rootfs /mnt/nfs

后续驱动课程采用这种方式挂载

ARM

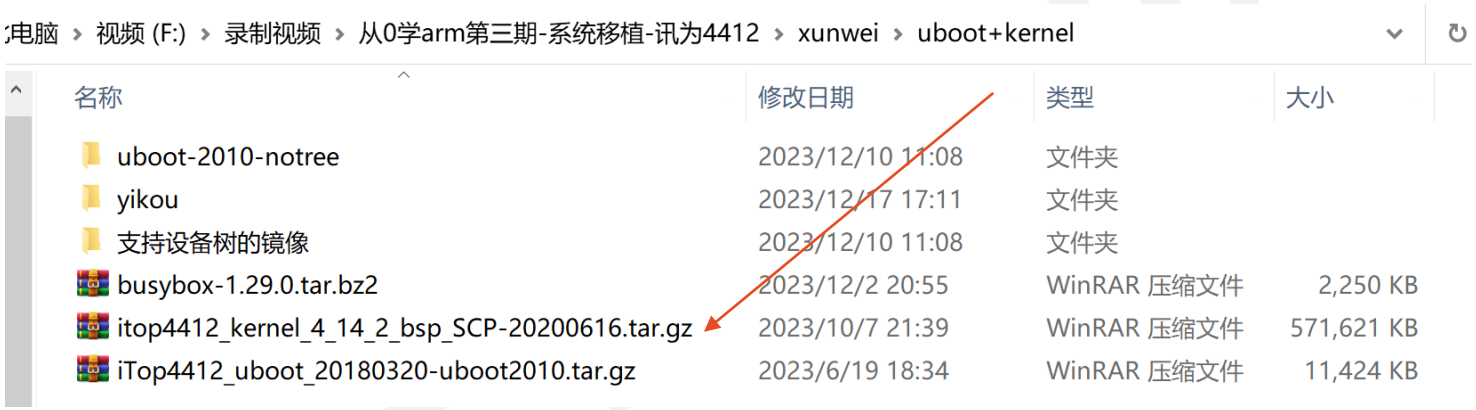
06

Uboot源码编译

Uboot编译说明和演示

讯为提供包含设备树的uboot+内核

- SoC原厂和开发板厂商都会进行定制



解压缩

- 拷贝到Ubuntu /home/peng/work/itop/
- 解压缩

```
peng@ubuntu:~/work/itop$ tar -zxvf itop4412_kernel_4_14_2_bsp_SCP-20200616.tar.gz
```

```
peng@ubuntu:~/work/itop$ cd itop4412_kernel_4_14_2_bsp/
peng@ubuntu:~/work/itop/itop4412_kernel_4_14_2_bsp$ ls
build_all.sh  gcc-4.6.2-glibc-2.13-linaro-multilib-2011.12.tar.gz  system.img      User guide.txt
burnimage.sh  linux-4.14.2_iTop-4412_scp                           u-boot-2017.11
```

镜像文件说明

```
peng@ubuntu:~/work/itop$ cd itop4412_kernel_4_14_2_bsp/
peng@ubuntu:~/work/itop/itop4412_kernel_4_14_2_bsp$ ls
build_all.sh  gcc-4.6.2-glibc-2.13-linaro-multilib-2011.12.tar.gz  system.img      User guide.txt
burnimage.sh  linux-4.14.2_iTop-4412_scp                           u-boot-2017.11
```

- buid_all.sh
  - 编译uboot和内核的自动化脚本
- burnimage.sh
  - 烧录uboot和内核镜像到sd卡脚本
- gcc-4.6.2-glibc-2.13-linaro-multilib-2011.12.tar.gz
  - 交叉编译工具链
- linux-4.14.2_iTop-4412_scp
  - 内核源码
- u-boot-2017.11
  - uboot
- system.img
  - 讯为制作的qt的文件系统，ext4格式
- User guide.txt
  - 使用说明手册

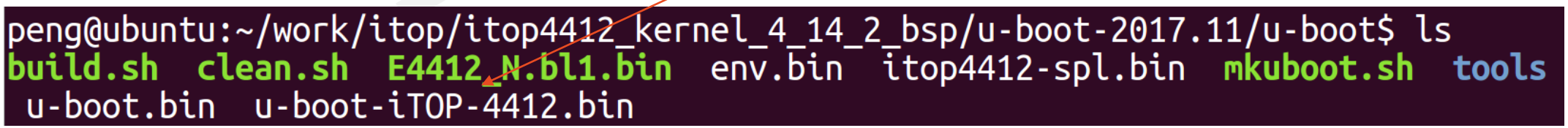
Uboot编译

- 分步编译
  - make itop4412_defconfig
  - make -j4

u-boot-2017.11/configs/itop4412_defconfig

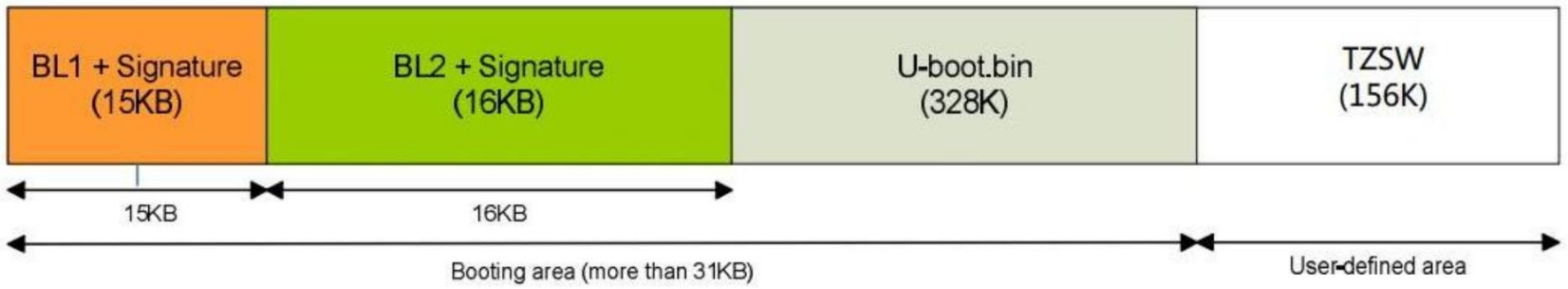
uboot-iTop-4412.bin制作

- 制作准备
  - cp u-boot.bin ../u-boot/
  - cp spl/itop4412-spl.bin ../u-boot/
- 制作
  - cat E4412_N.bl1.bin itop4412-spl.bin u-boot.bin > u-boot-iTOP-4412.bin



最终uboot-iTop-4412.bin制作

文件名	说明
E4412_N.bl1.bin	三星提供
itop4412-spl.bin	三星提供，讯为做了封装
u-boot.bin	编译好的U-Boot镜像
tzsw	三星提供



自动化编译脚本说明

itop4412_kernel_4_14_2_bsp\u-boot-2017.11\u-boot\build.sh

```
#!/bin/bash

cd ../u-boot-2017.11/

if [ ! -f .config ]
then
    make itop4412_defconfig
fi

make -j4

cp u-boot.bin ../u-boot/
echo "copy u-boot.bin done."

cd spl/
if [ ! -f itop4412-spl.bin ] ; then
    echo "notice: not found itop4412-spl.bin !"
    exit 0
else
    echo "copying itop4412-spl.bin..."
fi

cp itop4412-spl.bin ../../u-boot/
echo "copy u-boot-spl.bin done."

cd ../../u-boot/

echo "fusing u-boot-iTOP-4412.bin......"

################ for sd MMC boot ##################
#cat E4412_N.bl1.bin itop4412-spl.bin env.bin u-boot.bin > u-boot-iTOP-4412.bin
################ for eMMC boot ##################
cat E4412_N.bl1.bin itop4412-spl.bin u-boot.bin > u-boot-iTOP-4412.bin
#################################################


echo "build success !!!"
```

-f 如果 文件存在，则为真

取视频中所有资料

自动化编译脚本

- ./build.sh

Uboot源码简介

Uboot源码获取

- https://ftp.denx.de/pub/u-boot/

u-boot-2023.04-rc2.tar.bz2	14-Feb-2023 00:39	18M
u-boot-2023.04-rc2.tar.bz2.sig	14-Feb-2023 00:39	458
u-boot-2023.04-rc3.tar.bz2	27-Feb-2023 23:19	18M
u-boot-2023.04-rc3.tar.bz2.sig	27-Feb-2023 23:19	458
u-boot-2023.04-rc4.tar.bz2	14-Mar-2023 01:53	18M
u-boot-2023.04-rc4.tar.bz2.sig	14-Mar-2023 01:53	458
u-boot-2023.04-rc5.tar.bz2	27-Mar-2023 19:23	18M
u-boot-2023.04-rc5.tar.bz2.sig	27-Mar-2023 19:24	458
u-boot-2023.04.tar.bz2	03-Apr-2023 21:39	18M
u-boot-2023.04.tar.bz2.sig	03-Apr-2023 21:39	458
u-boot-2023.07-rc1.tar.bz2	01-May-2023 17:02	18M
u-boot-2023.07-rc1.tar.bz2.sig	01-May-2023 17:02	458
u-boot-2023.07-rc2.tar.bz2	08-May-2023 19:17	18M
u-boot-2023.07-rc2.tar.bz2.sig	08-May-2023 19:17	458
u-boot-2023.07-rc3.tar.bz2	29-May-2023 16:00	19M
u-boot-2023.07-rc3.tar.bz2.sig	29-May-2023 16:00	458
u-boot-2023.07-rc4.tar.bz2	12-Jun-2023 18:46	19M
u-boot-2023.07-rc4.tar.bz2.sig	12-Jun-2023 18:46	458
u-boot-2023.07-rc5.tar.bz2	26-Jun-2023 16:44	19M
u-boot-2023.07-rc5.tar.bz2.sig	26-Jun-2023 16:44	458
u-boot-2023.07-rc6.tar.bz2	03-Jul-2023 19:27	19M
u-boot-2023.07-rc6.tar.bz2.sig	03-Jul-2023 19:27	458
u-boot-2023.07.01.tar.bz2	11-Jul-2023 14:53	19M
u-boot-2023.07.01.tar.bz2.sig	11-Jul-2023 14:54	458
u-boot-2023.07.02.tar.bz2	11-Jul-2023 17:01	19M
u-boot-2023.07.02.tar.bz2.sig	11-Jul-2023 17:01	458
u-boot-2023.07.tar.bz2	10-Jul-2023 19:14	19M
u-boot-2023.07.tar.bz2.sig	10-Jul-2023 19:14	458
u-boot-2023.10-rc1.tar.bz2	25-Jul-2023 22:20	19M
u-boot-2023.10-rc1.tar.bz2.sig	25-Jul-2023 22:20	458
u-boot-2023.10-rc2.tar.bz2	07-Aug-2023 20:27	19M
u-boot-2023.10-rc2.tar.bz2.sig	07-Aug-2023 20:27	458
u-boot-2023.10-rc3.tar.bz2	21-Aug-2023 21:20	19M
u-boot-2023.10-rc3.tar.bz2.sig	21-Aug-2023 21:20	458
u-boot-2023.10-rc4.tar.bz2	04-Sep-2023 15:40	19M
u-boot-2023.10-rc4.tar.bz2.sig	04-Sep-2023 15:40	458
u-boot-2023.10.tar.bz2	02-Oct-2023 15:40	19M
u-boot-2023.10.tar.bz2.sig	02-Oct-2023 15:40	458
u-boot-2024.01-rc1.tar.bz2	23-Oct-2023 21:30	19M
u-boot-2024.01-rc1.tar.bz2.sig	23-Oct-2023 21:30	458
u-boot-2024.01-rc2.tar.bz2	06-Nov-2023 20:48	19M
u-boot-2024.01-rc2.tar.bz2.sig	06-Nov-2023 20:48	458
u-boot-2024.01-rc3.tar.bz2	20-Nov-2023 15:04	19M
u-boot-2024.01-rc3.tar.bz2.sig	20-Nov-2023 15:05	458
u-boot-2202.04-rc5.tar.bz2	28-Mar-2022 15:15	17M
u-boot-2202.04-rc5.tar.bz2.sig	28-Mar-2022 15:15	458

解压缩

```
tar -jxvf u-boot-2017.11.tar.bz2
```

Uboot源码目录

名字	功能描述
api	硬件无关的功能函数的API，是uboot本身使用的
arch	各种CPU架构平台例如：arm .mips. powerpc. x86. riscv .etc.
board	已经支持的开发板相关文件，板级相关配置文件，针对不同平台的功能下具体的实现
cmd	实现uboot命令行下支持的命令，每一条命令都对应一个文件。
common	通用启动相关初始化文件，board.r board_init_f的实现common/main.c是整个u-boot程序的主函数，主要负责运行维护uboot的shell命令行，这个文件夹以前是cmd的合集，后边版本变动了
configs	各个板卡平台的默认配置文件 make 的时候可以编译使用默认配置比如 make xxx_defconfig 就设置了按照默认配置编译
disk	与磁盘有关的文件
doc	文档目录，里面存放了很多uboot相关文档，这些文档可以帮助理解uboot代码。
drivers	板级的驱动。 这里面放的就是从linux源代码中移植过来的linux设备驱动，主要是开发板上必须用到的一些驱动，如网卡驱动、Inand/SD卡、NandFlash等的驱动。
dts	存放不同开发板的设备树源码文件，目前该目录为空，只有Makefile 和Kconfig

uboot命令宏定义

```
194: U_BOOT_CMD(
195:     bootm,  CONFIG_SYS_MAXARGS, 1,  do_bootm,
196:     "boot application image from memory", bootm_help_text
197: );
```

- U_BOOT_CMD
- 例如bootm命令对应就是bootm.c 而xxx命令，对应do_xxx函数
- cmd_process最终会调用cmd_tbl_s结构体中的成员变量cmd函数指针，
- 比如执行xxx命令，最终是会调用你do_xxx函数 uboot源码的cmd文件夹下xxx.c文件对应的是xxx命令

查找所有定义的命令：grep U_BOOT_CMD ./* -nr

Uboot源码目录

env	环境相关文件
examples	uboot示例代码
fs	文件系统相关，linux系统移植而来的
include	头文件目录。
lib	各类算法库的实现，比如crc，aes bzip系列，这类文件夹中的内容移植时基本不用管
Licenses	开源协议(BSD, GPL, LGPL,MIT)许可证书，uboot使用的开源许可协议
post	有些需要上电自检程序放在这里
scripts	常用脚本
test	测试程序
tools	里面是一些工具类的代码。譬如mkimage
Makefile	uboot的顶层makefile
config.mk	某个Makefile会调用此配置文件,用来处理一些编译过程中的环境变量。
Kconfig	这个文件是 Kconfig 系统的菜单项，当我们使用命令：make menuconfig 时，Kconfig 系统读取该文件，根据该文件的内容生成各级菜单。 U-Boot 源码根目录下的 Kconfig 就是顶级的配置菜单，其中会在引入其他目录下的 Kconfig 作为二级菜单，依次类推
Kbuild	是 Kbuild 系统使用的文件，该文件用于定义一些源码使用的需要根据编译环境产生的中间文件。

Makefile

- CROSS_CPMPILE
- ARCH

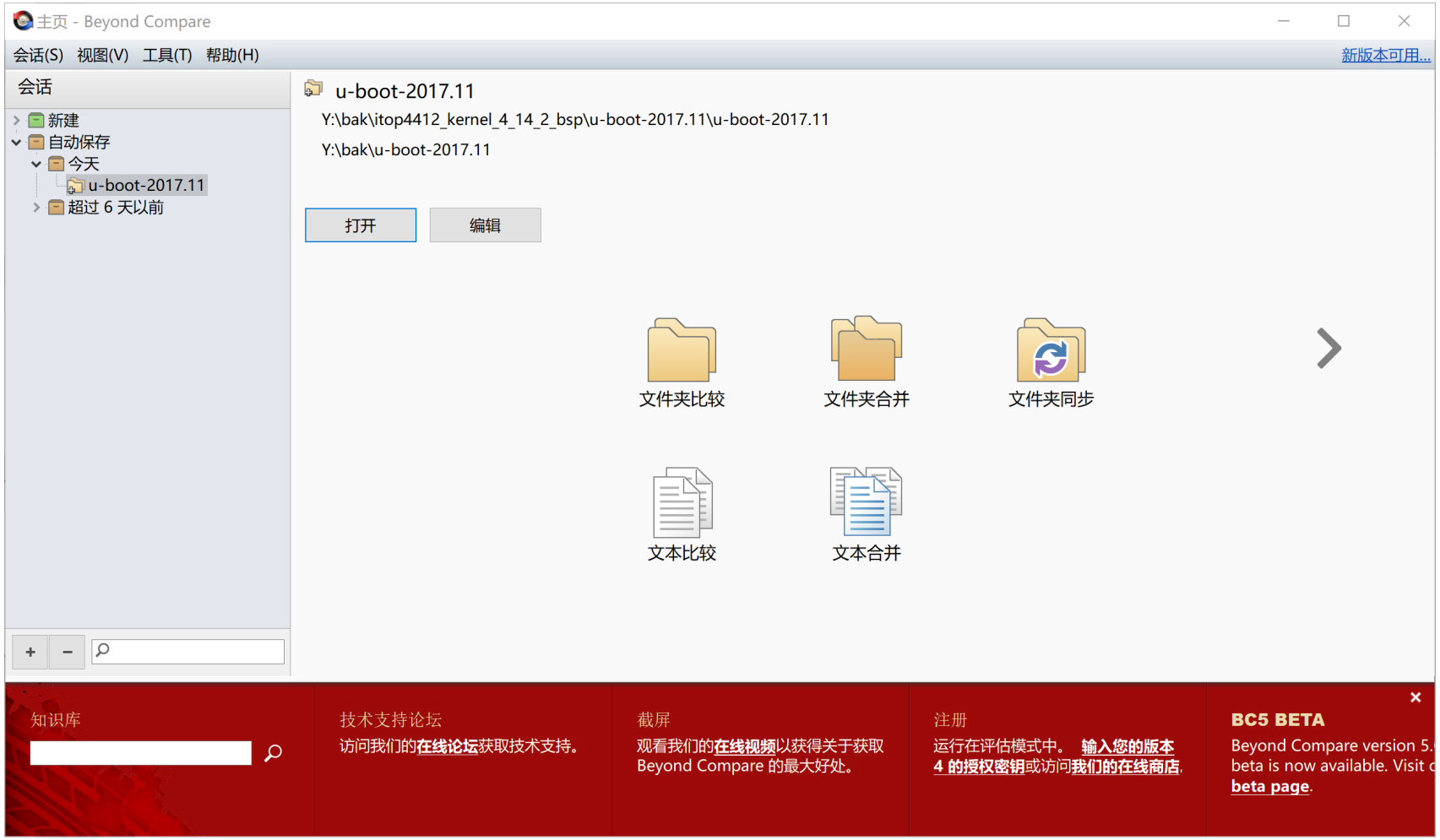
```
244 # set default to nothing for native builds
245 ifeq ($(HOSTARCH),$(ARCH))
246 CROSS_COMPILE ?=
247 endif
248
249 #CROSS_COMPILE := arm-linux-gnueabi-
250 CROSS_COMPILE := arm-none-linux-gnueabi-
251
252 KCONFIG_CONFIG  ?= .config
253 export KCONFIG_CONFIG
254
255 # SHELL used by kbuild
256 CONFIG_SHELL := $(shell if [ -x "$$BASH" ]; then echo $$BASH; \
257       else if [ -x /bin/bash ]; then echo /bin/bash; \
258       else echo sh; fi ; fi)
```

编译生成文件

名字	功能描述
u-boot	编译出的ELF格式的uboot镜像文件
u-boot.bin	编译出来的二进制格式的uboot可执行镜像文件
u-boot.cfg	uboot的另一种配置文件
u-boot.lds	链接脚本
u-boot.map	uboot映射文件
u-boot.srec	S-Record格式的镜像文件
u-boot.sym	uboot符号文件

如何知道厂家做了哪些移植工作?

- **Beyond compare**



u-boot-2017.11 - 文件夹比较(F) - Beyond Compare

会话(S) 操作(A) 编辑(E) 搜索(R) 视图(V) 工具(T) 帮助(H) 新版本可用...

主页 会话 全部 差别 相同 结构 次要 规则 复制 展开 折叠 选择 文件 刷新 交换 停止 过滤: *.* 过滤 不过滤

Y:\bak\itop4412_kernel_4_14_2_bsp\u-boot-2017.11\u-boot-2017.11

名称(N)	大小(Z)	已修改(M)
api	31,329	2019/2/16 9:21:08
arch	21,105,249	2019/2/16 9:21:30
board	11,110,528	2019/2/16 9:21:51
cmd	1,020,879	2024/1/1 20:39:39
common	1,090,263	2024/1/1 20:39:39
configs	1,229,922	2019/2/16 9:21:59
disk	92,365	2024/1/1 20:39:40
doc	1,252,023	2019/2/16 9:21:09
Documentation	1,111	2019/2/16 9:21:31
drivers	15,959,091	2024/1/1 20:39:39
dts	11,447	2024/1/1 20:39:40
env	111,434	2024/1/1 20:39:40
examples	71,313	2019/2/16 9:21:33
fs	1,760,903	2024/1/1 20:39:39
include	7,762,562	2020/6/22 10:19:44
lib	2,206,962	2024/1/1 20:39:39
Licenses	86,616	2019/2/16 9:21:32
net	186,277	2024/1/1 20:39:39
post	123,912	2019/2/16 9:20:58
scripts	2,186,491	2019/2/16 9:21:08
spl		2024/1/1 20:39:39
test	419,447	2024/1/1 20:39:40
tools	2,037,353	2024/1/1 20:39:40
.checkpatch.conf (h)	610	2019/2/16 9:21:59
.config (h)	25,101	2020/6/16 18:06:56
.gitignore (h)	839	2019/2/16 9:22:00
.mailmap (h)	1,323	2019/2/16 9:21:08
.travis.yml (h)	11,367	2019/2/16 9:21:13
CHANGELOG	0	2019/2/16 9:20:59
config.mk	2,260	2019/2/16 9:21:32
itop4412-5.6.3_defconfig	1,645	2019/2/16 9:21:59
Kbuild	1,863	2019/2/16 9:20:59
Kconfig	14,455	2019/2/16 9:21:08
MAINTAINERS	12,022	2019/2/16 9:22:01
Makefile	56,684	2019/2/16 9:21:59
MyUnderstandProject.udb	98,349,636	2019/2/16 9:21:57
README	185,894	2019/2/16 9:22:00
snapshot.commit	17	2019/2/16 9:21:32

Y:\bak\u-boot-2017.11

名称(N)	大小(Z)	已修改(M)
api	31,329	2017/11/14 9:08:06
arch	24,351,867	2017/11/14 9:08:06
board	11,105,958	2017/11/14 9:08:06
cmd	1,019,398	2017/11/14 9:08:06
common	1,090,267	2017/11/14 9:08:06
configs	1,204,821	2017/11/14 9:08:06
disk	92,365	2017/11/14 9:08:06
doc	1,252,023	2017/11/14 9:08:06
Documentation	1,111	2017/11/14 9:08:06
drivers	15,932,220	2017/11/14 9:08:06
dts	11,447	2017/11/14 9:08:06
env	111,253	2017/11/14 9:08:06
examples	71,313	2017/11/14 9:08:06
fs	1,760,903	2017/11/14 9:08:06
include	7,711,020	2017/11/14 9:08:06
lib	2,202,626	2017/11/14 9:08:06
Licenses	86,616	2017/11/14 9:08:06
net	186,277	2017/11/14 9:08:06
post	123,912	2017/11/14 9:08:06
scripts	1,655,151	2017/11/14 9:08:06
test	419,447	2017/11/14 9:08:06
tools	2,037,353	2017/11/14 9:08:06
.checkpatch.conf (h)	610	2017/11/14 9:08:06
.gitignore (h)	839	2017/11/14 9:08:06
.mailmap (h)	1,323	2017/11/14 9:08:06
.travis.yml (h)	11,367	2017/11/14 9:08:06
config.mk	2,260	2017/11/14 9:08:06
Kbuild	1,863	2017/11/14 9:08:06
Kconfig	14,455	2017/11/14 9:08:06
MAINTAINERS	12,022	2017/11/14 9:08:06
Makefile	56,602	2017/11/14 9:08:06
README	185,894	2017/11/14 9:08:06
snapshot.commit	17	2017/11/14 9:08:06

2024/1/1 20:38:45 加载比较: Y:\bak\itop4412_kernel_4_14_2_bsp\u-boot-2017.11\u-boot-2017.11 <-> Y:\bak\u-boot-2017.11

2024/1/1 20:39:43 快速刷新

15 个文件, 94.1 MB 448 GB 可用，在 Y:\ 上 11 个文件, 281 KB 448 GB 可用，在 Y:\ 上

- include\configs\itop4412.h
- arch\arm\include\asm\mach-types.h
- arch\arm\mach-exynos\clock.c
- arch\arm\mach-exynos\clock_init_exynos4.c
- arch\arm\mach-exynos\dmc_init_exynos4.c
- arch\arm\mach-exynos\itop4412_setup.h
- arch\arm\mach-exynos\lowlevel_init.c
- arch\arm\mach-exynos\power.c
- arch\arm\mach-exynos\spl_boot.c
- board\samsung\itop4412
- cmd\mmc.c

Uboot模块定制

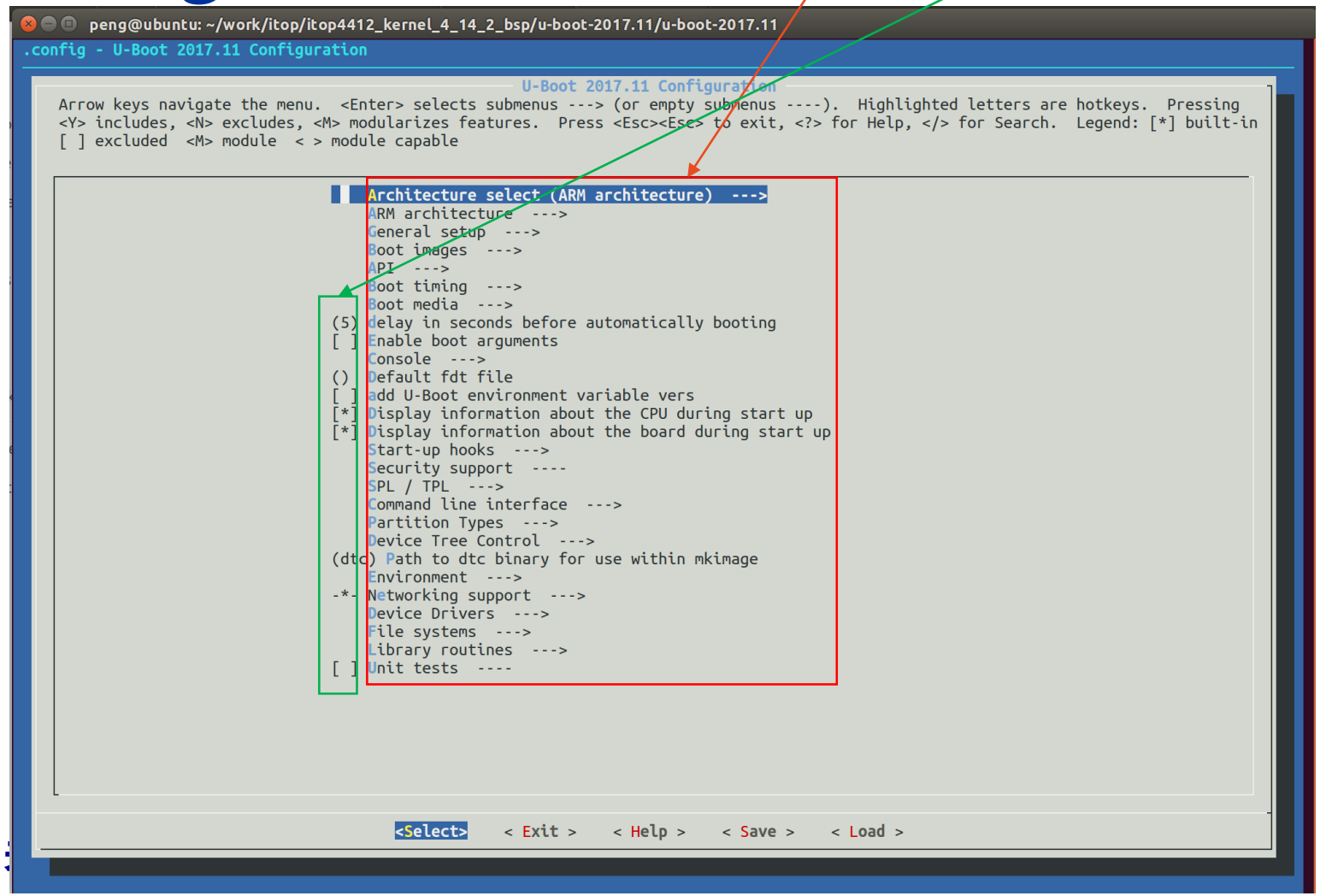
make menuconfig

几个重要文件

- Kconfig
  - make menuconfig 图形菜单界面 依赖该文件
- Makefile
  - 决定目录下源文件是否编译
- .config
  - 最终决定模块是否编译的开关依赖文件
- configs/itop4412_defconfig
  - 厂家出厂的uboot模块开关配置文件【备份用】

Uboot模块定制

- make menuconfig



选择说明

- 子菜单--->

```
    Command line interface  --->
```

- 中括号[]

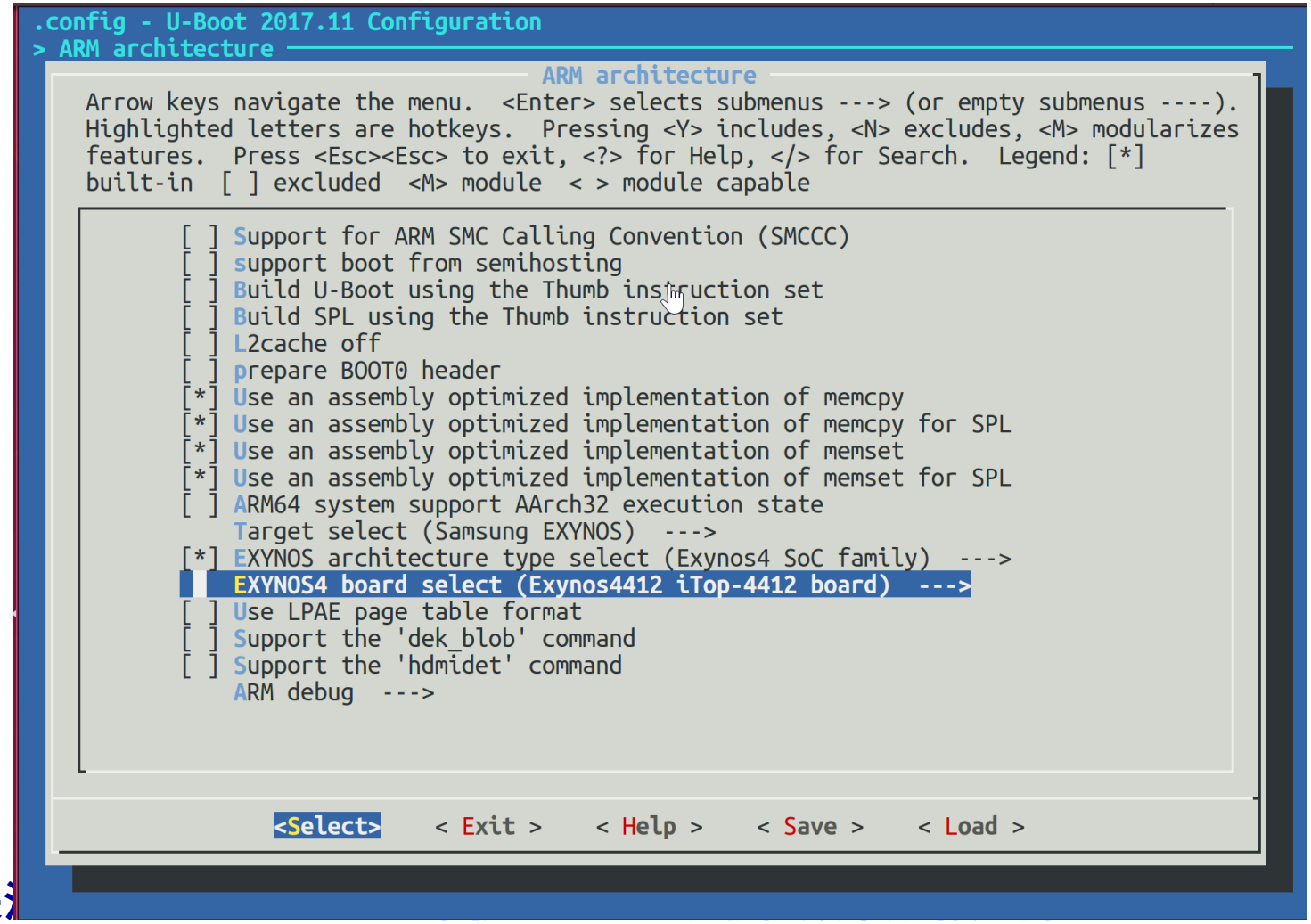
```
[*] Display information about the CPU during start up
```

  - 表示该选项只有两种选项，中括号中要么是空，要么是“*”
- 圆括号()
  - 圆括号的内容是可以输入一个值

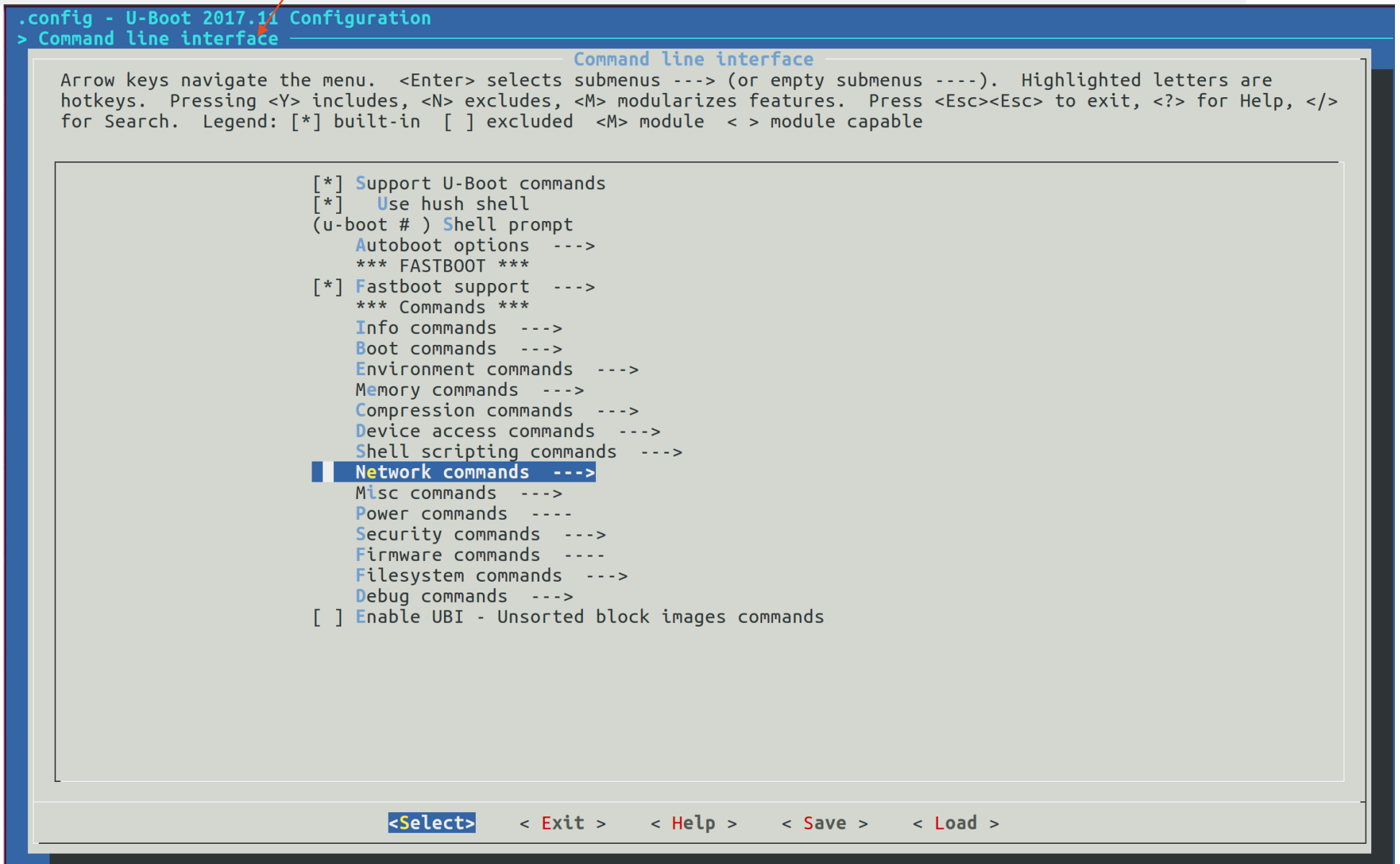
```
[ ] Boot timing and reporting
(30) Number of boot stage records to store
(5) Number of boot stage records to store for SPL
(0) Address to stash boot timing information
(0x1000) Size of boot timing stash region
```

Architecture select (ARM architecture) --->

- **Architecture select (ARM architecture) --->**
- **ARM architecture --->**

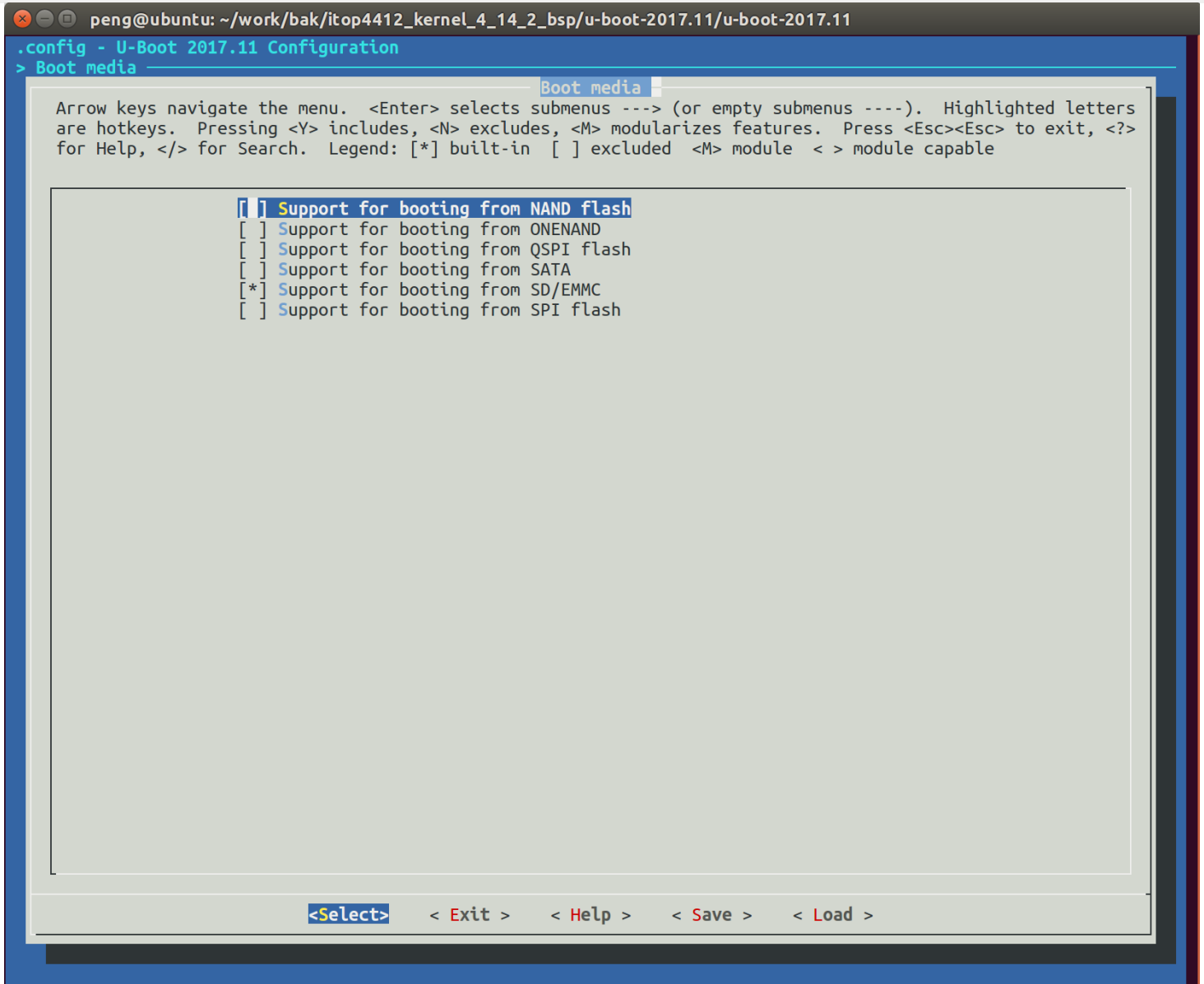


Uboot命令

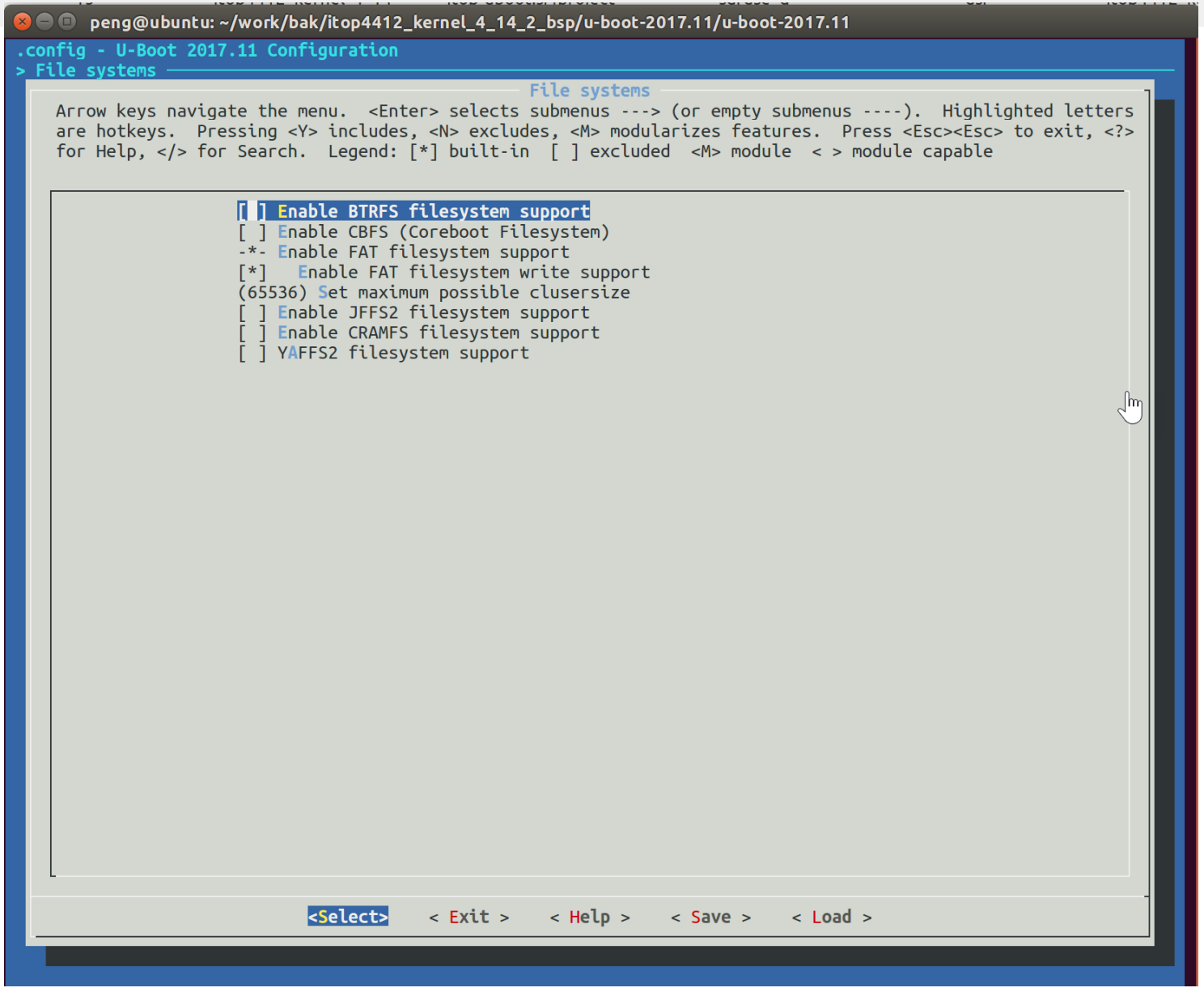


后面演示一下

Boot media


peng@ubuntu: ~/work/bak/itop4412_kernel_4_14_2_bsp/u-boot-2017.11/u-boot-2017.11
.config - U-Boot 2017.11 Configuration
> Boot media
Boot media
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty submenus ----). Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module < > module capable
[ ] Support for booting from NAND flash
[ ] Support for booting from ONENAND
[ ] Support for booting from QSPI flash
[ ] Support for booting from SATA
[*] Support for booting from SD/EMMC
[ ] Support for booting from SPI flash
<Select>
< Exit >
< Help >
< Save >
< Load >

文件系统


peng@ubuntu: ~/work/bak/itop4412_kernel_4_14_2_bsp/u-boot-2017.11/u-boot-2017.11
.config - U-Boot 2017.11 Configuration
> File systems
File systems
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty submenus ----). Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module < > module capable
[ ] Enable BTRFS filesystem support
[ ] Enable CBFS (Coreboot Filesystem)
-*- Enable FAT filesystem support
[*] Enable FAT filesystem write support
(65536) Set maximum possible clusersize
[ ] Enable JFFS2 filesystem support
[ ] Enable CRAMFS filesystem support
[ ] YAFFS2 filesystem support
<Select> < Exit > < Help > < Save > < Load >

查看.config

- 最终会将所有选择开关保存到.config
- 最好将配置保存到configs下 itop4412_defconfig
  - make itop4412_defconfig

演示

Uboot新增一个cmd

ping2

net/ping2.c

```c
#include "ping.h"
#include "arp.h"


#if defined(CONFIG_CMD_PING2)
static int do_ping2(struct cmd_tbl *cmdtp, int flag, int argc,
           char *const argv[])
{
    if (argc < 2)
        return CMD_RET_USAGE;

    printf("host %s is alive\n", argv[1]);

    return CMD_RET_SUCCESS;
}

U_BOOT_CMD(
    ping2,  2,  1,  do_ping2,
    "send ICMP ECHO_REQUEST to network host",
    "pingAddress"
);
#endif
```

net/Makefile

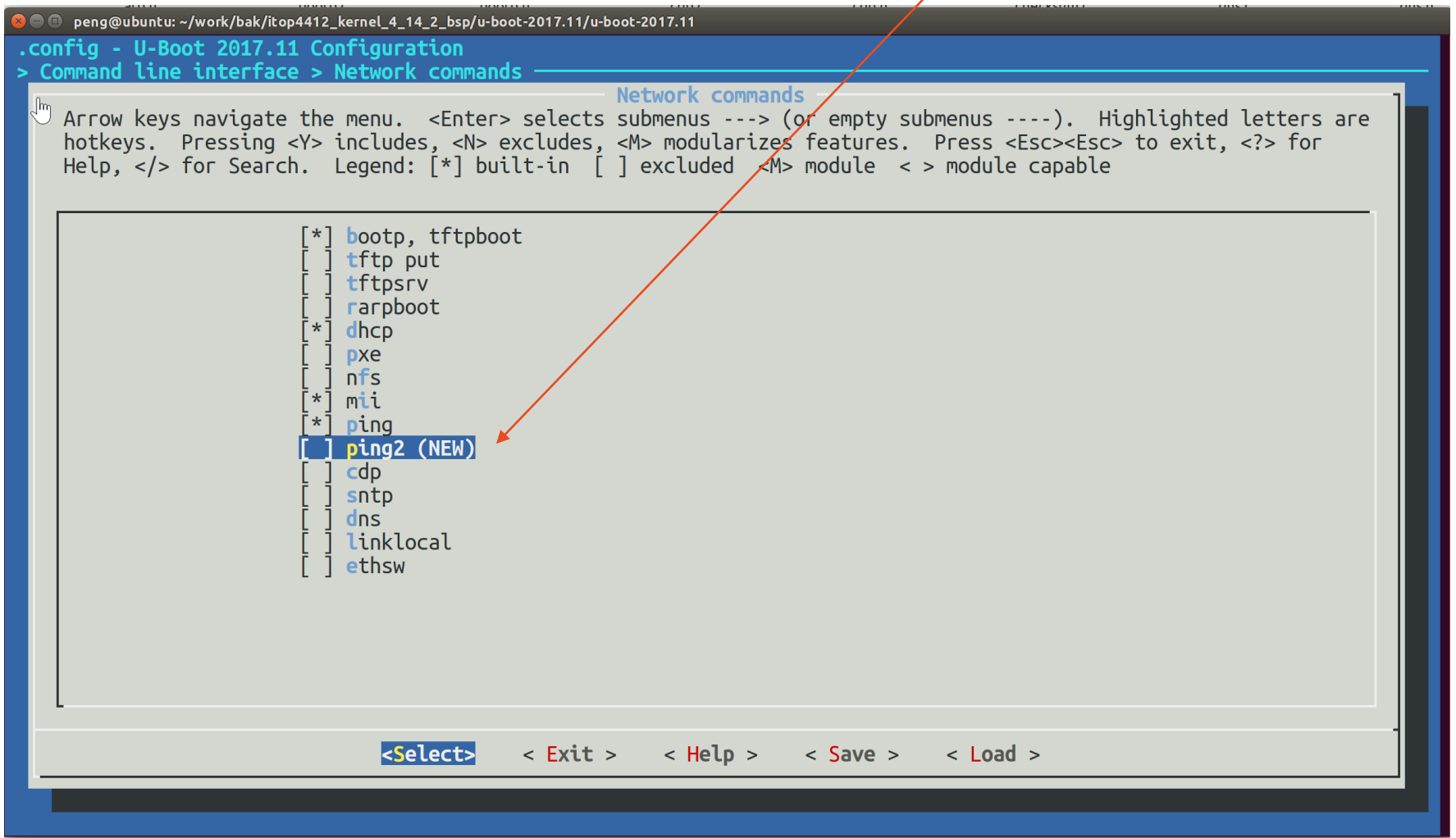
```
10 obj-y += checksum.o
11 obj-$(CONFIG_CMD_NET)  += arp.o
12 obj-$(CONFIG_CMD_NET)  += bootp.o
13 obj-$(CONFIG_CMD_CDP)  += cdp.o
14 obj-$(CONFIG_CMD_DNS)  += dns.o
15 ifdef CONFIG_DM_ETH
16 obj-$(CONFIG_CMD_NET)  += eth-uclass.o
17 else
18 obj-$(CONFIG_CMD_NET)  += eth_legacy.o
19 endif
20 obj-$(CONFIG_CMD_NET)  += eth_common.o
21 obj-$(CONFIG_CMD_LINK_LOCAL) += link_local.o
22 obj-$(CONFIG_CMD_NET)  += net.o
23 obj-$(CONFIG_CMD_NFS)  += nfs.o
24 obj-$(CONFIG_CMD_PING) += ping.o
25 obj-$(CONFIG_CMD_PING2) += ping2.o
26 obj-$(CONFIG_CMD_RARP) += rarp.o
27 obj-$(CONFIG_CMD_SNTP) += sntp.o
28 obj-$(CONFIG_CMD_NET)  += tftp.o
```

cmd/Kconfig

```
1010 config CMD_PING2
1011     bool "ping2"
1012         help
1013                 Send ICMP ECHO_REQUEST to network host
1014
```

Make menuconfig

- Command line interface --->
- Network commands --->



.config

```
459 #
460 # Network commands
461 #
462 CONFIG_CMD_NET=y
463 # CONFIG_CMD_TFTPPUT is not set
464 # CONFIG_CMD_TFTPSRV is not set
465 # CONFIG_CMD_RARP is not set
466 CONFIG_CMD_DHCP=y
467 # CONFIG_CMD_PXE is not set
468 # CONFIG_CMD_NFS is not set
469 CONFIG_CMD_MII=y
470 CONFIG_CMD_PING=y
471 CONFIG_CMD_PING2=y
472 # CONFIG_CMD_CDP is not set
473 # CONFIG_CMD_SNTP is not set
474 # CONFIG_CMD_DNS is not set
475 # CONFIG_CMD_LINK_LOCAL is not set
476 # CONFIG_CMD_ETHSW is not set
477
```

执行结果

```
u-boot # ping2
ping2 - send ICMP ECHO_REQUEST to network host

Usage:
ping2 pingAddress
u-boot # <INTERRUPT>
u-boot #
u-boot #
u-boot #
u-boot # ping2 1.1.1.1
host 1.1.1.1 is alive
```

演示一下，增加uboot命令

ARM

07

Uboot源代码流程详解

• 参考 B站视频

- https://www.bilibili.com/video/BV1Fp4y1x7L3/?vd_source=07570058a62e0e8a6cf489efac35cfec
- https://www.bilibili.com/video/BV1f5411n7tf/?vd_source=07570058a62e0e8a6cf489efac35cfec
- Pdf
- 《uboot启动源码分析.pdf》

ARM

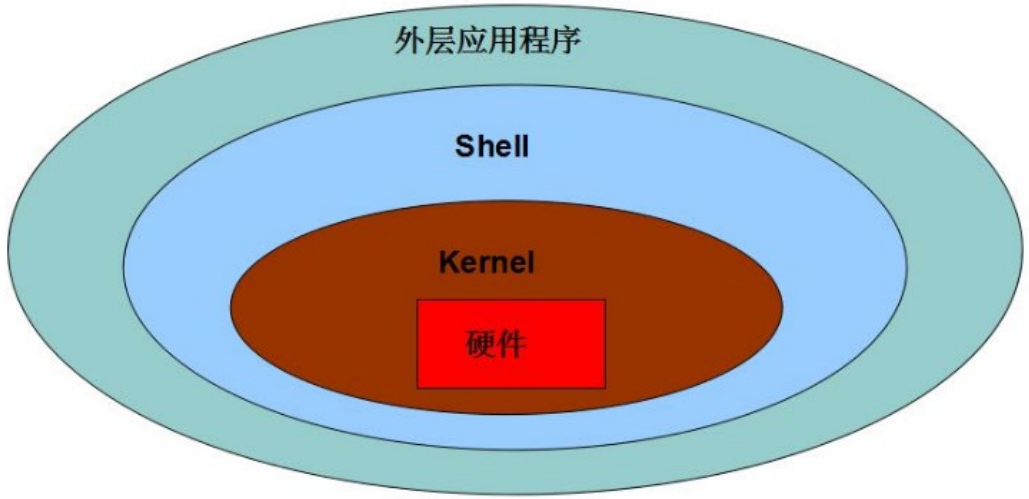
08

内核编译与驱动移植

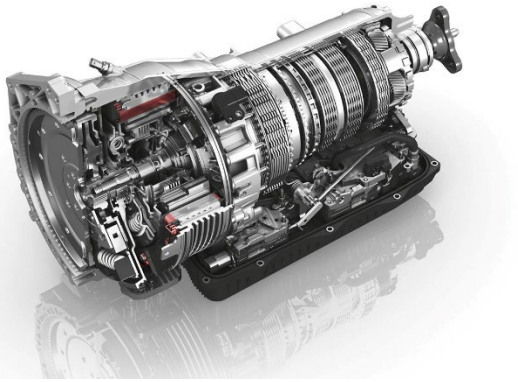
Linux内核基础知识

1. 什么是内核?

- 在计算机科学中是一个用来管理软件发出的数据I/O（输入与输出）要求的计算机程序，
- 将这些要求转译为数据处理的指令并交由中央处理器（CPU）及计算机中其他电子组件进行处理，是现代操作系统中最基本的部分。



2.操作系统与汽车



- 内　　核----------发动机
- 图形界面----------底盘、变速箱、真皮座椅……
- Os发行版----------整车

ubuntu

统信UOS

redhat LINUX

中标麒麟 NeoKylin

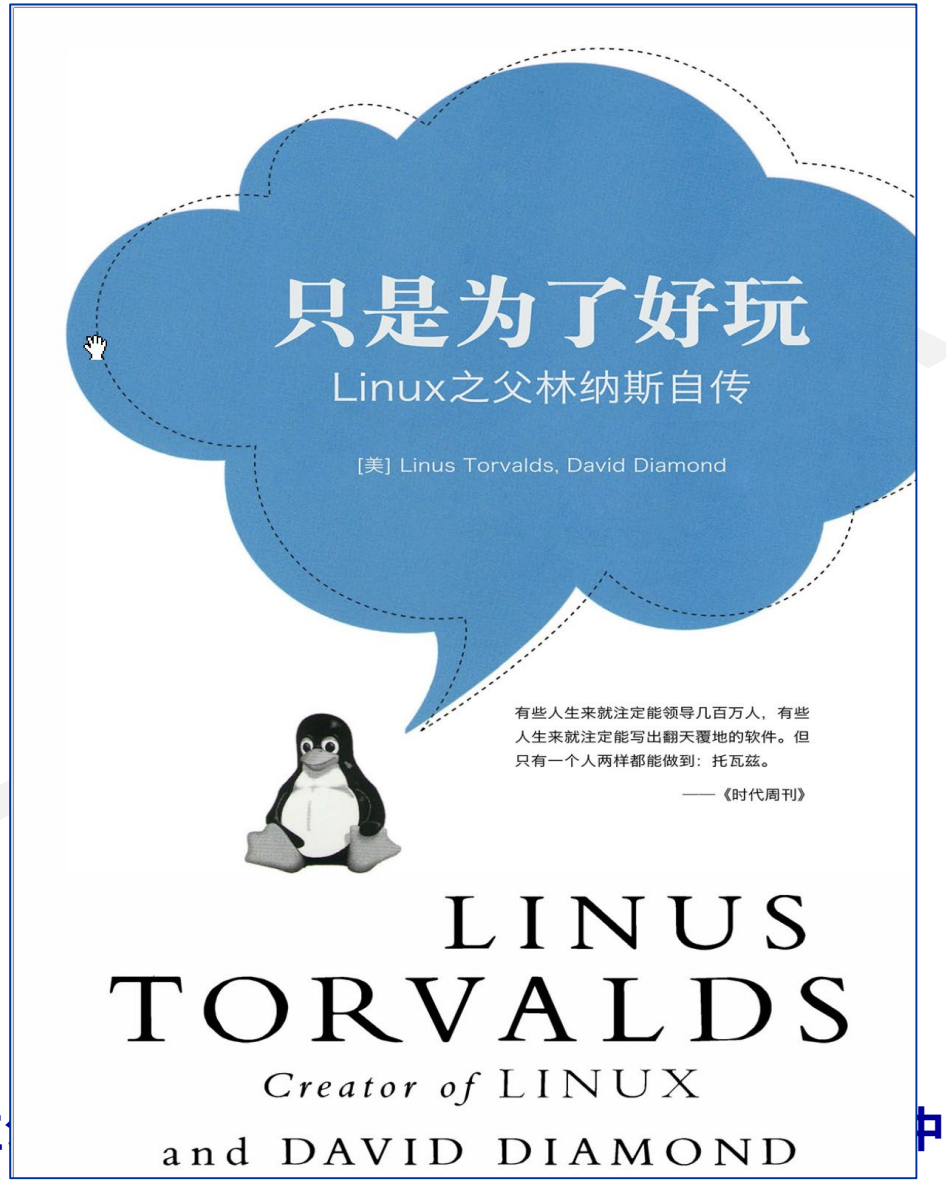
银河麒麟 KYLIN

SUSE

3.linux

- Linux是一种开源电脑操作系统内核。它是一个用C语言写成，符合POSIX标准的类Unix操作系统。
- Linux最早是由芬兰 Linus Torvalds为尝试在英特尔x86架构上提供自由的类Unix操作系统而开发的。
- 该计划开始于1991年，在计划的早期有一些 Minix 黑客提供了协助，而如今全球无数程序员正在为该计划无偿提供帮助。

4. Just for fun



公众号：

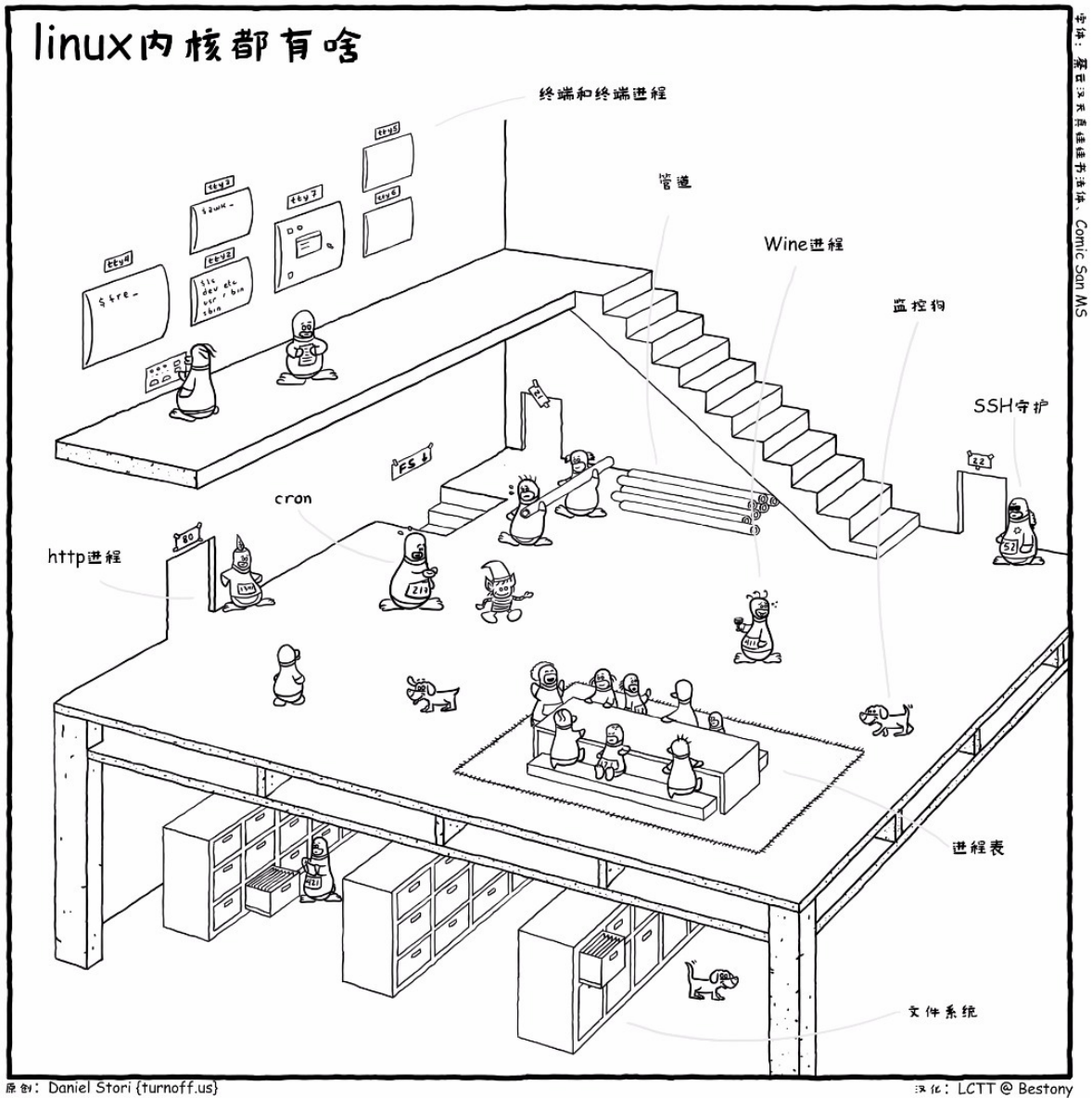
一口Linux

后台回复：

linus自传

关注 中所有资料

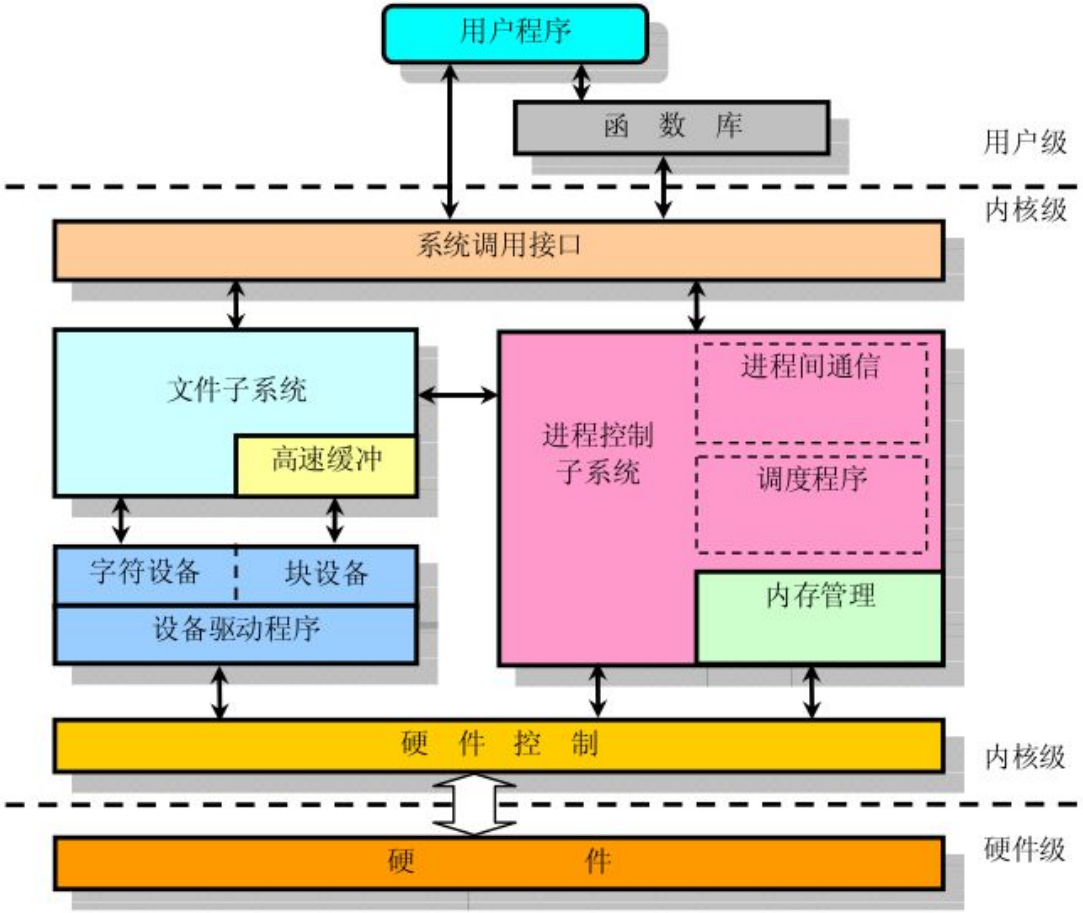
5. Linux内核里有什么



关注公

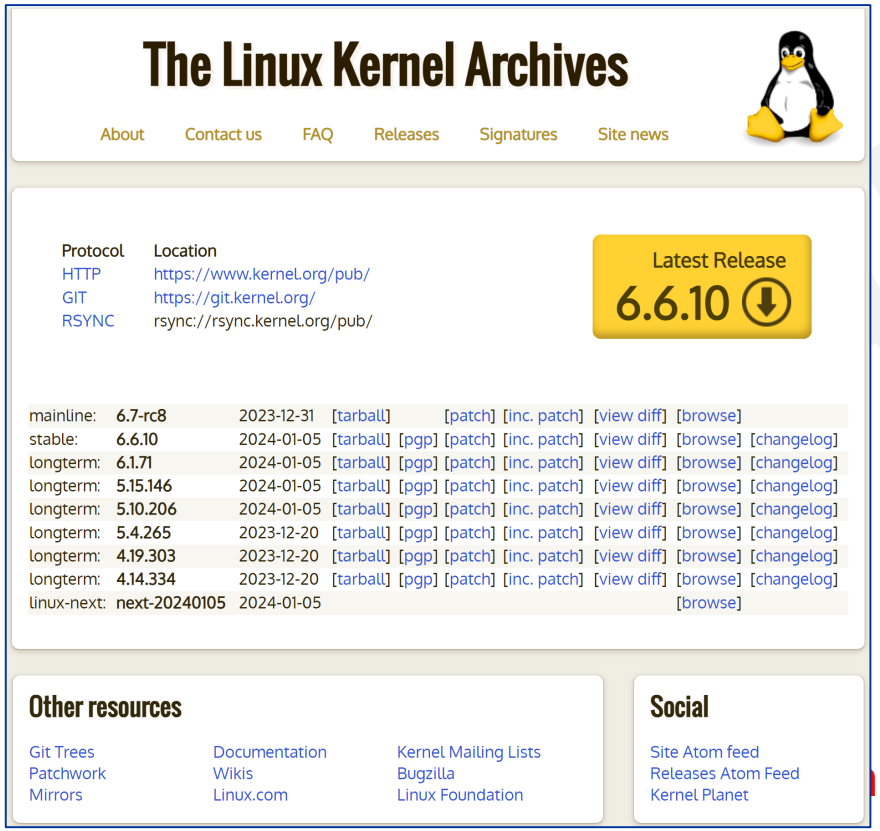
6.内核子系统

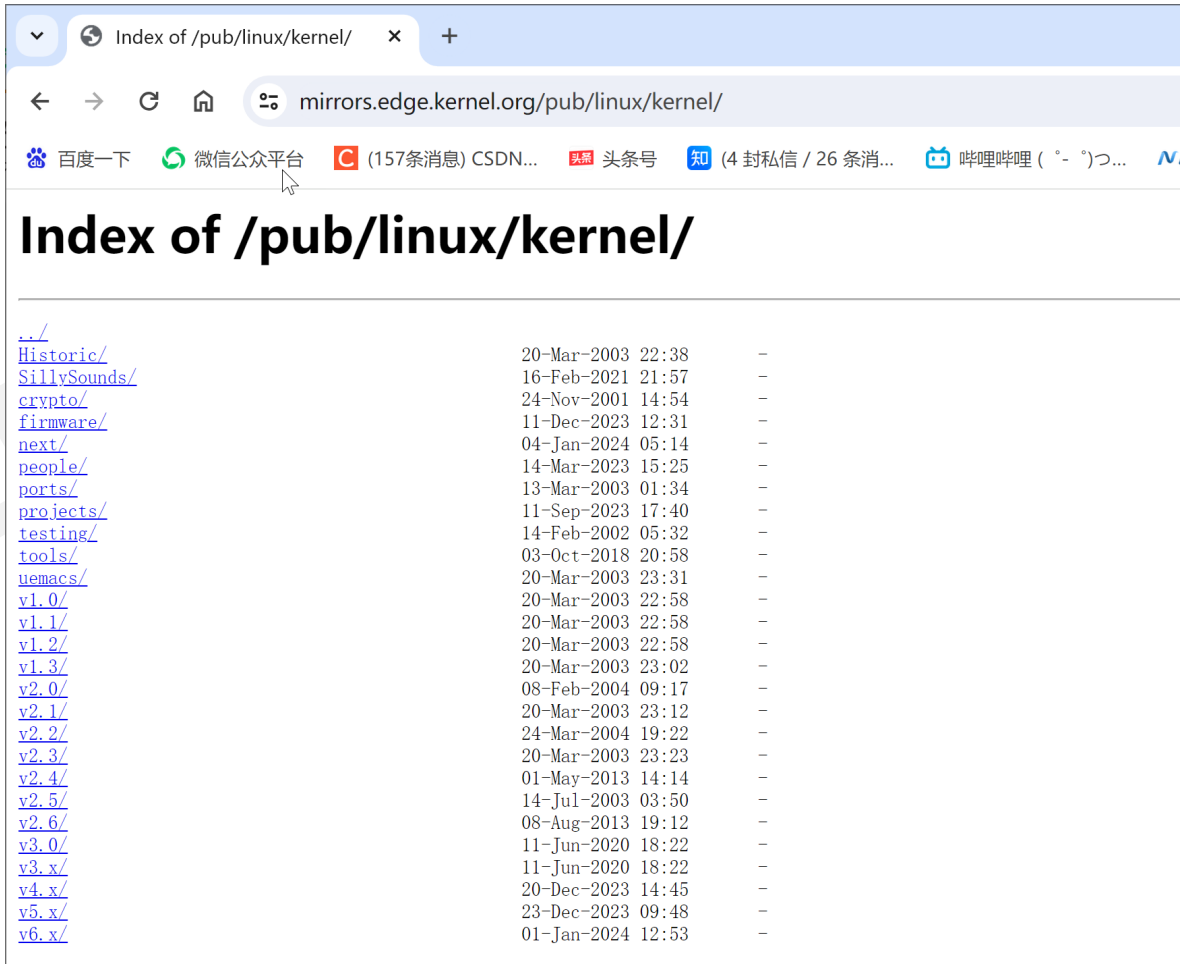
- 最上面是用户（或应用程序）空间
  - GNU C Library （glibc）也在这里。
  - 它提供了连接内核的系统调用接口，还提供了在用户空间应用程序和内核之间进行转换的机制。
- 内核主要系统包括：
  - SCI：系统调用接口
  - PM：进程管理
  - VFS：虚拟文件系统
  - MM：内存管理
  - Network Stack：内核协议栈
  - Arch：体系架构
  - DD：设备驱动



7.内核源码获取

- https://www.kernel.org/pub/linux/kernel/
- HTTP https://www.kernel.org/pub/
- GIT https://git.kernel.org/
- RSYNC rsync://rsync.kernel.org/pub/





8. Linux内核代码量

GitStats - linux

General | Activity | Authors | Files | Lines | Tags

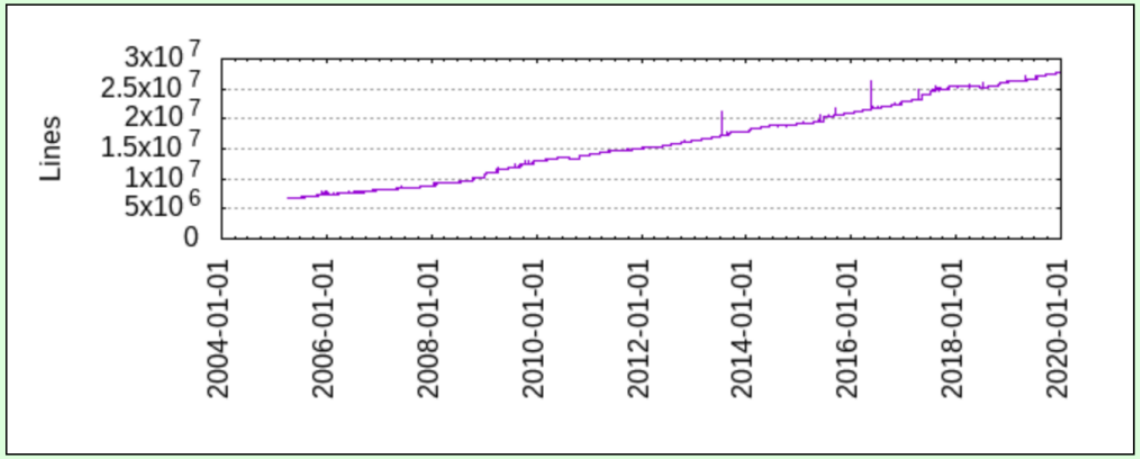
Project name:
linux
Generated:
2020-01-01 09:00:10 (in 8567 seconds)
Generator:
GitStats (version 55c5c28), git version 2.20.1, gnuplot 5.2 patchlevel 6
Report Period:
1969-12-31 19:00:01 to 2085-06-18 11:57:19
Age:
42173 days, 5474 active days (12.98%)
Total Files:
66492
Total Lines of Code:
27852148 (52733303 added, 24881155 removed)
Total Commits:
887925 (average 162.2 commits per active day, 21.1 per all days)
Authors:
21074 (average 42.1 commits per author)

Lines

General | Activity | Authors | Files | Lines | Tags

Total lines:
27852148

Lines of Code



5.内核目录

目录	内容
arch/	体系结构相关的代码，如arch/i386、arch/arm、arch/ppc
crypto	常用加密和散列算法（如AES、SHA等），以及一些压缩和CRC校验算法
drivers/	各种设备驱动程序，如drivers/char、drivers/block……
Documentation/	内核文档
fs/	文件系统，如fs/ext3、fs/jffs2……
include/	内核头文件：include/asm是体系结构相关的头文件，它是include/asm-arm、include/asm-i386等目录的链接； include/linux是Linux内核基本的头文件
init/	Linux初始化，如main.c
ipc/	进程间通信的代码
kernel/	Linux内核核心代码（这部分比较小）
lib/	各种库子程序，如zlib、crc32
mm/	内存管理代码
net/	网络支持代码，主要是网络协议
sound	声音驱动的支持
scripts/	内部或者外部使用的脚本
usr/	用户的代码

- Linux内核其他说明，请参考
- 《从0学Linux驱动第一期》视频

Linux内核编译操作

Linux内核编译常用命令

- make mrproper
  - 清除内核生成的配置文件与目标文件等，一般在第一次编译时使用
- 导入默认配置信息(在内核根目录中)
  - make xxx_deconfig
  - cp arch/arm/configs/xx_deconfig .config
- 配置命令
  - make menuconfig (常用 libncurses库)
- 编译内核
  - make uImage ---生成内核镜像 /arch/arm/boot/uImage
- 编译设备树
  - make dtbs ---生成设备树文件 /arch/arm/boot/dtb/xxxxxx.dtb
- 编译生成模块文件
  - make modules ---把配置值选成M的代码编译生成模块文件。（.ko）放在对应的源码目录下。
- make clean
  -

演示：编译命令

- 设置环境变量
  - export ARCH=arm
- 导入内核配置文件
  - make iTop-4412_scp_defconfig
  - 或者
  - cp config_for_iTop-4412_scp .config
- 编译内核
  - make uImage LOADADDR=0x40007000 -j4
- 编译设备树
  - make dtbs

自动化编译脚本说明

```
#cp config_for_iTop-4412_scp .config
export ARCH=arm

make iTop-4412_scp_defconfig

make uImage LOADADDR=0x40007000 -j4

make dtbs
```

编译举例

- 1.模块文件中增加打印log

linux驱动模块移植

led驱动移植

目标：移植led灯驱动

- 从0学arm第三期-系统移植-讯为4412\xunwei\code\driver\led\led_drv.c

Linux内核中驱动模块

- 1. 内核源码已经包含驱动模块
  - 只需要通过make menuconfig定制即可
- 2. 新增内核模块
  - 外部编译
    - 独立编译成ko文件，内核启动之后，
    - 通过开启启动脚本加载，或者手动加载
  - 直接编译进内核
    - 编程内核镜像的额一部分，开机就自动加载驱动模块

1. 外部编译

- Makefile

```
1  ifneq ($(KERNELRELEASE),)
2  #$(info "2nd")
3  obj-m:=led_drv.o
4  else
5  #KDIR :=/lib/modules/$(shell uname -r)/build
6  KDIR :=/home/peng/work/itop/itop4412_kernel_4_14_2_bsp/linux-4.14.2_iTop-4412_scp
7  PWD  :=$(shell pwd)
8  all:
9  #   $(info "1st")
10     make -C $(KDIR) M=$(PWD) modules
11 clean:
12     rm -f *.ko *.o *.mod.o *.symvers *.cmd  *.mod.c *.order
13 endif
```

模块操作命令

- insmod
  - 安装模块
- lsmod
  - 显示模块
- rmmod
  - 卸载模块

驱动测试操作

- U盘拷贝
  - mkdir /mnt/test
  - mount /dev/mmcblk0p2   /mnt/test
  - cd /mnt/test
- dmesg
  - dmesg –c
  - dmesg | tail -n 20
  - dmesg | grep ping
- mknod /dev/led c 500 0
  - 创建字符设备文件 /dev/led
  - 主设备号 500
  - 次设备号 0

2. 编译进内核

拷贝驱动文件到 drivers/char 下

- drivers/char/Kconfig

```
10 config YIKOU_LED
11     tristate "yikou led test"
12     default y
13     help
14       yikou linux test
```

- drivers/char/Makefile

```
 6 obj-$(CONFIG_YIKOU_LED) += led_drv.o
```

.config

```
1651 CONFIG_YIKOU_LED=y
```

- 如果有设备树文件
  - arch/arm/boot/dts/exynos4412-itop-elite.dts

Linux内核设备树编译举例

make dtbs

设备树编译命令

- 驱动源文件
  - xunwei\code\driver\beep\platform-beep-tree
  -
- 设备树文件
  - arch/arm/boot/dts/exynos4412-itop-elite.dts
- 设备树编译命令
  - make dtbs

设备树信息

```
2       yikou-beep{
3           compatible = "yikou,beep";
4           reg = <0x114000a0 0x4 0x139D0000 0x20>;
5       };
```

```
static struct of_device_id beep_table[] = {
        {.compatible = "yikou,beep"},
};
static struct platform_driver hello_driver =
{
    .probe = hello_probe,
    .driver.name = "duang",
    .remove = hello_remove,
    .driver = {
        .name = "yikou-led",
        .of_match_table = beep_table,
    },
};
```

举例

linux驱动模块移植
相关文件再整理

Kconfig Makefile

.config xxxx_defconfig

1. 驱动移植几个重要文件

- Kconfig → 所有的模块，有模块要管理的目录下均有该文件
  - make menuconfig 图形菜单界面 依赖该文件
- Makefile → 决定所有C源文件编译行为，有源文件的目录下均有该文件
  - 决定目录下源文件编译行为
- .config → 只有根目录有
  - 最终决定模块是否编译的开关依赖文件
- arch/arm/configs/itop4412_defconfig
  - 厂家出厂的内核模块开关配置文件【备份用】
- arch/arm/boot/dts/xxxx.dts　xxxx.dtsi
  - 设备树文件

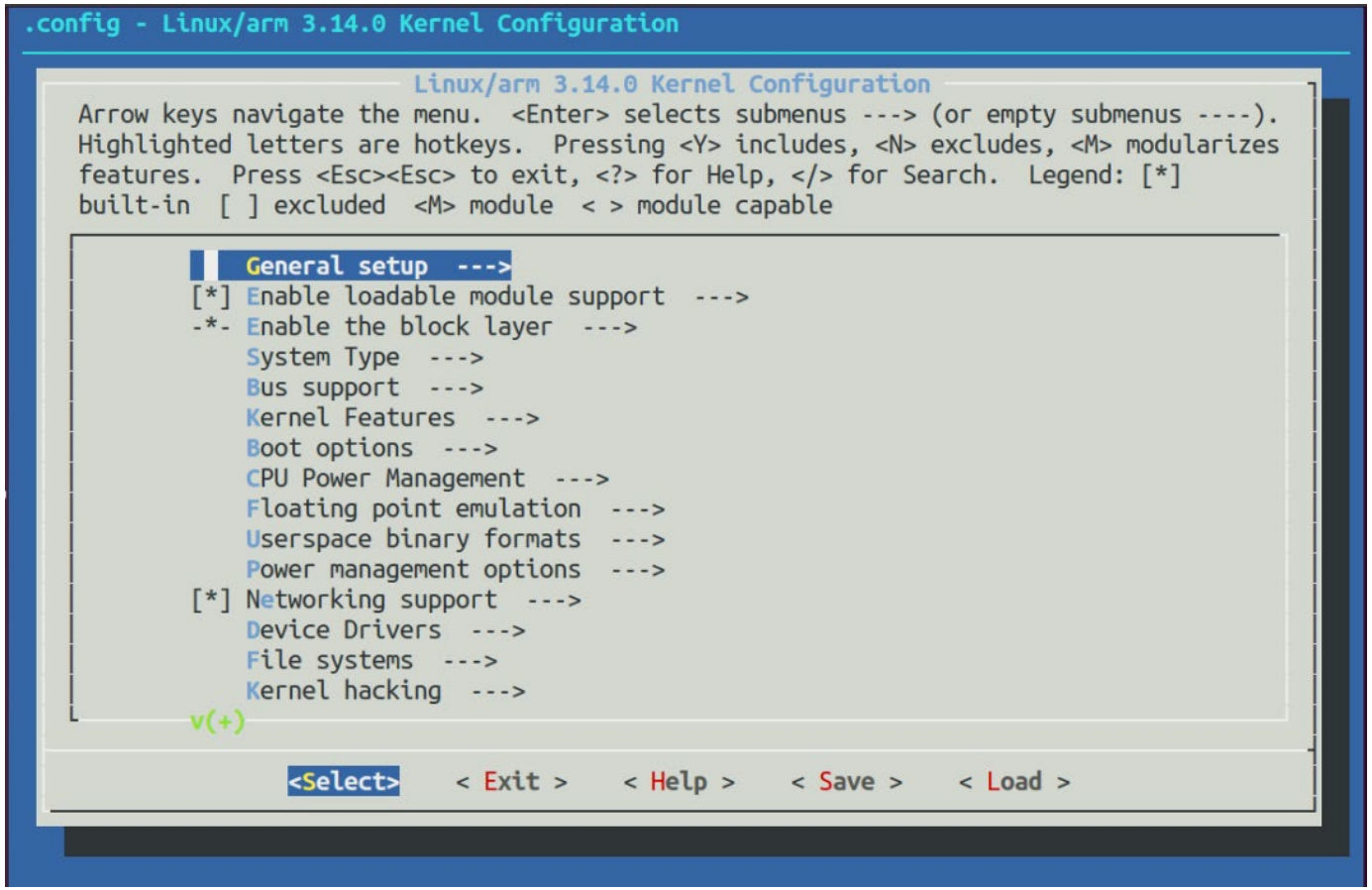
2. Kconfig

- Kconfig用来配置内核，它就是各种配置界面的源文件，
- 内核的配置工具读取各个目录下Kconfig文件，生成配置界面供开发人员配置内核，
- 最后生成配置项，保存在文件.config
- 包含子目录 source *subpath*/Kconfig

- Kconfig的语法可以参考
  - Documentation/kbuild/kconfig-language.txt

3. make menuconfig

依赖Kconfig



选择说明

- 子菜单--->

```
Power management options  --->
```

- 中括号[]

```
[*] Network device support  --->
```

  - 表示该选项只有两种选项，中括号中要么是空，要么是“*”

- 尖括号<>

```
< > Serial ATA and Parallel ATA drivers  ----
```

  - 有3种选择，它们代表的含义
    - *：将该功能编译进内核。
    - 空：不将该功能编译进内核。
    - M：将该功能编译成可以在需要时动态插入到内核中的模块。

- 圆括号()

```
(1024)  Horizontal screen resolution
(768)   Vertical screen resolution
```

  - 圆括号的内容是输入一个值

这些属性全部由Kconfig描述

4.config文件

- 界面选中之后，会在.config下增加对应的环境变量
- Kconfig中的配置项基础上，前面增加 CONFIG_

```
10 config YIKOU_LED
11     tristate "yikou led test"
12     default y
13     help
14       yikou linux test
```

drivers/char/Kconfig

```
1655 # CONFIG_YIKOU_LED is not set

1651 CONFIG_YIKOU_LED=y
```

.config

5. Makefile

- Makefile决定C源文件编译行为
- Makefile
  - obj-y　编译到内核
  - obj-m　编译成模块

```
6 obj-$(CONFIG_YIKOU_LED) += led_drv.o
```

来源于.config

```
1651 CONFIG_YIKOU_LED=y
```

驱动模块移植整理

- drivers/char/Kconfig

```
10 config YIKOU_LED
11     tristate "yikou led test"
12     default y
13     help
14       yikou linux test
```

执行make menuconfig
前面添加CONFIG_

.config

```
1651 CONFIG_YIKOU_LED=y
```

- drivers/char/Makefile

```
 6 obj-$(CONFIG_YIKOU_LED) += led_drv.o
```

执行make uImage
依赖这些环境变量

- 如果有设备树文件
  - arch/arm/boot/dts/exynos4412-itop-elite.dts

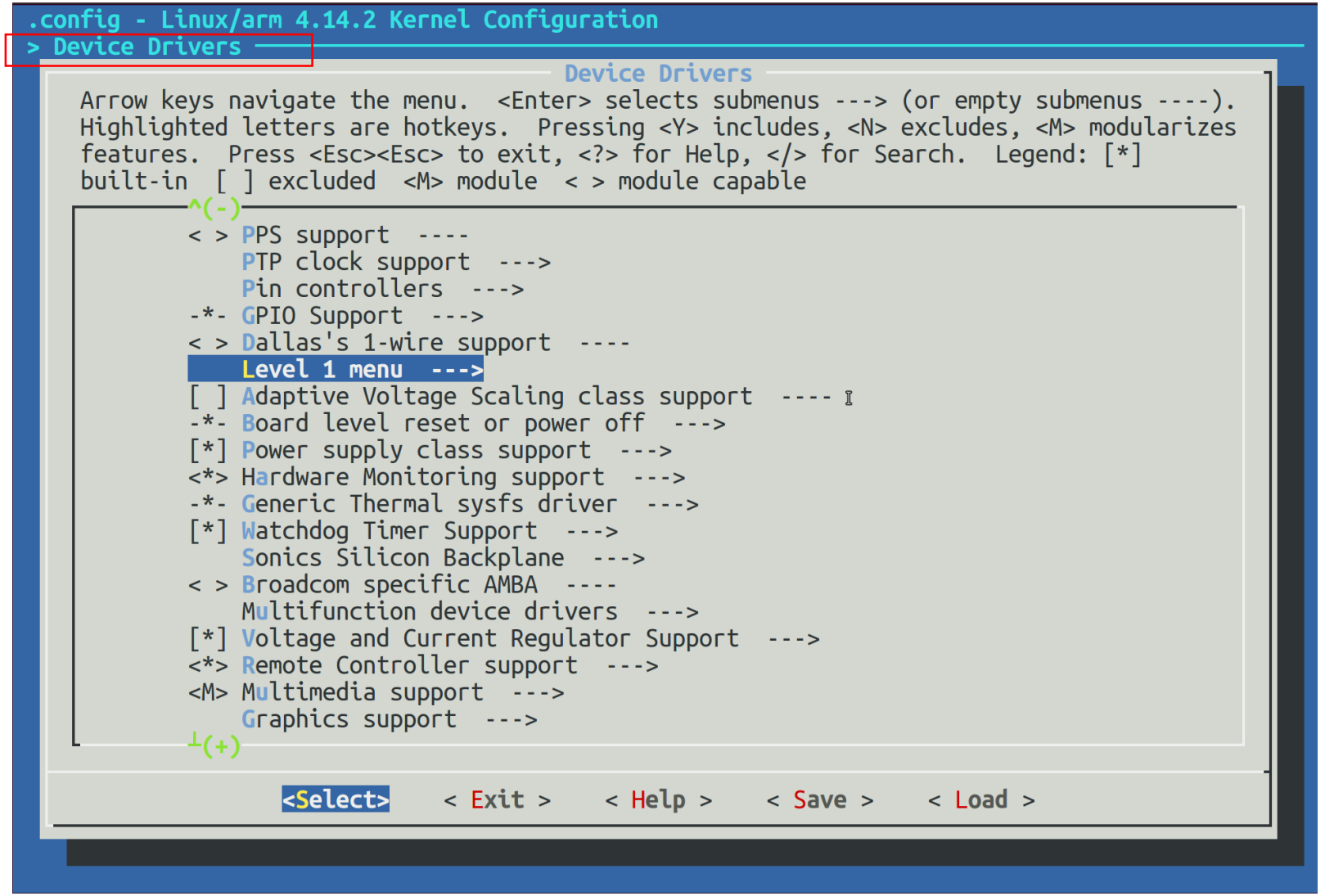
Linux驱动模块移植文件
Kconfig语法详解

Kconfig实例移植

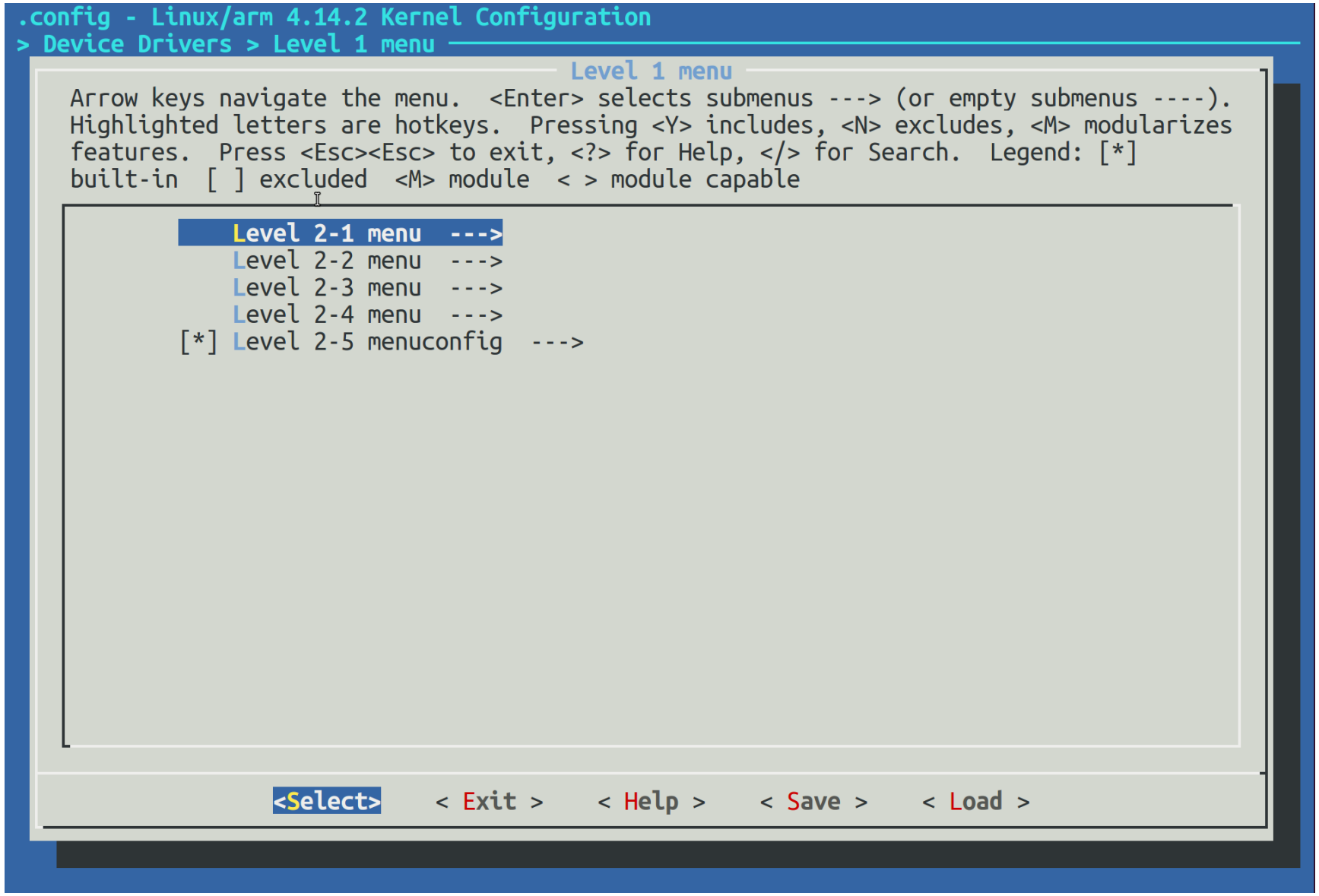
从0学arm第三期-系统移植-讯为4412\xunwei\code\Kconfig

- 1）添加 Kconfig 文件
  - 在 kernel/drivers目录下创建 test文件夹
  - 然后添加文件 Kconfig并把上面的示例代码拷贝进去。
- 2）修改上级Kconfig 文件kernel/drivers/Kconfig
  - 在第一行添加:
    - source "drivers/test/Kconfig"

测试界面


.config - Linux/arm 4.14.2 Kernel Configuration
> Device Drivers
Device Drivers
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty submenus ----).
Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes
features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*]
built-in [ ] excluded <M> module < > module capable
^(-)
< > PPS support ----
PTP clock support --->
Pin controllers --->
-*- GPIO Support --->
< > Dallas's 1-wire support ----
Level 1 menu --->
[ ] Adaptive Voltage Scaling class support ----
-*- Board level reset or power off --->
[*] Power supply class support --->
<*> Hardware Monitoring support --->
-*- Generic Thermal sysfs driver --->
[*] Watchdog Timer Support --->
Sonics Silicon Backplane --->
< > Broadcom specific AMBA ----
Multifunction device drivers --->
[*] Voltage and Current Regulator Support --->
<*> Remote Controller support --->
<M> Multimedia support --->
Graphics support --->
v(+)
<Select> < Exit > < Help > < Save > < Load >

测试界面


.config - Linux/arm 4.14.2 Kernel Configuration
> Device Drivers > Level 1 menu
Level 1 menu
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty submenus ----).
Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes
features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*]
built-in [ ] excluded <M> module < > module capable
Level 2-1 menu --->
Level 2-2 menu --->
Level 2-3 menu --->
Level 2-4 menu --->
[*] Level 2-5 menuconfig --->
<Select>
< Exit >
< Help >
< Save >
< Load >

1. config

- 配置项。可以定义一行配置选项。

```
[*] MENU_2_2_CONFIG_1
<M> MENU_2_2_CONFIG_2
(this is MENU_2_2_CONFIG_3) MENU_2_2_CONFIG_3 (NEW)
(0x456789) MENU_2_2_CONFIG_4 (NEW)
(666) MENU_2_2_CONFIG_5 (NEW)
```

```
config MENU_2_2_CONFIG_1
    bool "MENU_2_2_CONFIG_1"
    default n

config MENU_2_2_CONFIG_2
    tristate "MENU_2_2_CONFIG_2"
    default m
```

2. 选项类型

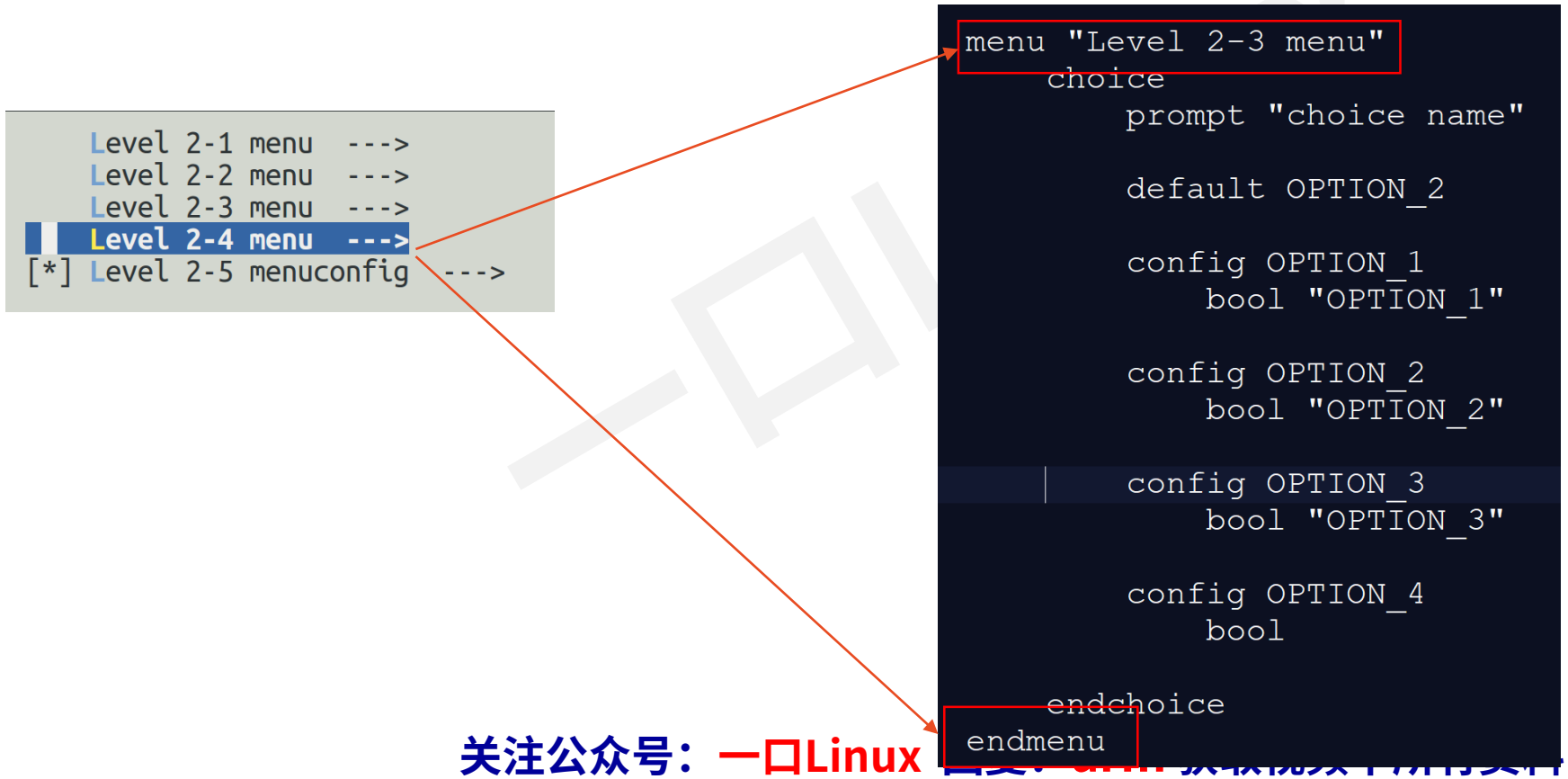
- 每个配置选项都有不同的选项类型。
  - - bool：
    - 布尔类型，可选参数（y/n）
  - - tristate三态：
    - 内建、模块、移除，可选参数（y/m/n）
  - - string：
    - 字符串，可填任意字符串内容
  - - hex：
    - 十六进制，可填任意16进制数值
  - - int：
    - 整型，可填任意10进制数值

```
[*] MENU_2_2_CONFIG_1
<M> MENU_2_2_CONFIG_2
(this is MENU_2_2_CONFIG_3) MENU_2_2_CONFIG_3 (NEW)
(0x456789) MENU_2_2_CONFIG_4 (NEW)
(666) MENU_2_2_CONFIG_5 (NEW)
```

```
menu "Level 2-2 menu"
    config MENU_2_2_CONFIG_1
        bool "MENU_2_2_CONFIG_1"
        default n

    config MENU_2_2_CONFIG_2
        tristate "MENU_2_2_CONFIG_2"
        default m

    config MENU_2_2_CONFIG_3
        string "MENU_2_2_CONFIG_3"
        default "this is MENU_2_2_CONFIG_3"

    config MENU_2_2_CONFIG_4
        hex "MENU_2_2_CONFIG_4"
        default 0x456789

    config MENU_2_2_CONFIG_5
        int "MENU_2_2_CONFIG_5"
        default 666
endmenu
```

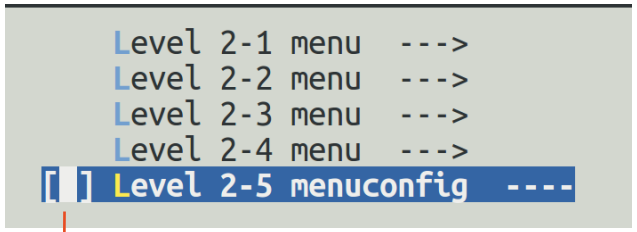
3. menu endmenu

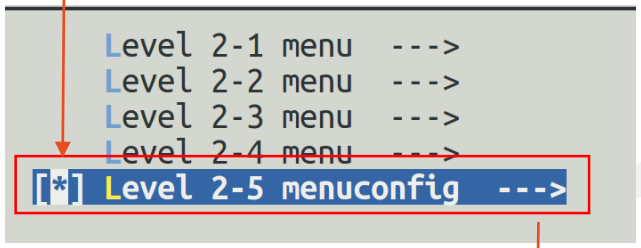
- 菜单。子内容会在子页面显示。要以 menu 开头，endmenu 结尾。菜单标题内容跟在 menu 后。



4. menuconfig

- 菜单配置项。可以定义一行配置选项，并且子选项会显示在子页面。



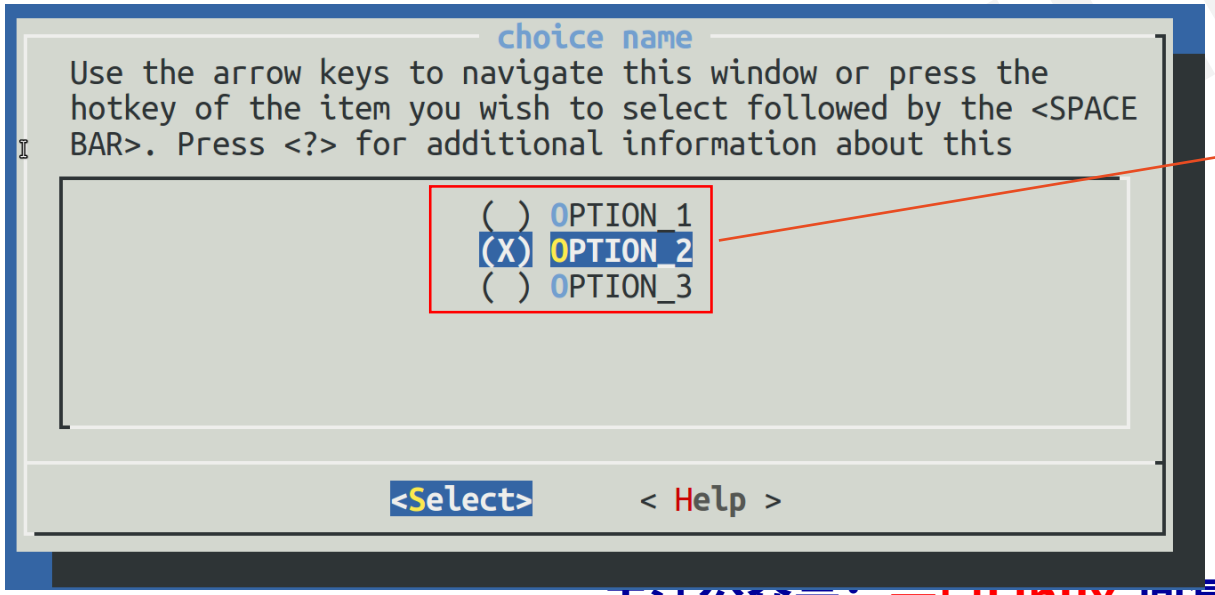



```
menuconfig Level_2-5_menu
    bool "Level 2-5 menuconfig"
    default n

    if Level_2-5_menu
        config MENU_2_5_CONFIG_1
            bool "MENU_2_5_CONFIG_1"
            default n

        config MENU_2_5_CONFIG_2
            bool "MENU_2_5_CONFIG_2"
            default y
    endif
```

5. choice endchoice

- 单选配置单。选项会在子页面显示。要以 choice 开头，endchoice 结尾。

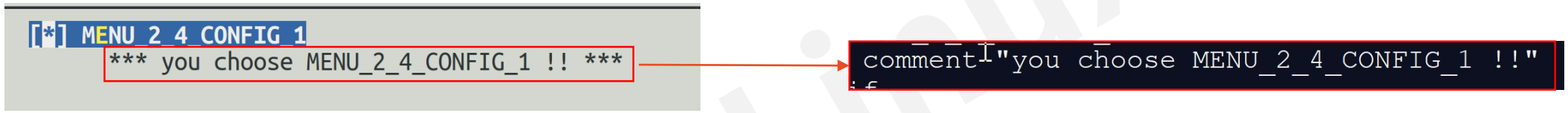
```
  choice name (OPTION_2)  --->
```




```
menu "Level 2-3 menu"
    choice
        prompt "choice name"

        default OPTION_2

        config OPTION_1
            bool "OPTION_1"

        config OPTION_2
            bool "OPTION_2"

        config OPTION_3
            bool "OPTION_3"

        config OPTION_4
            bool

    endchoice
endmenu
```

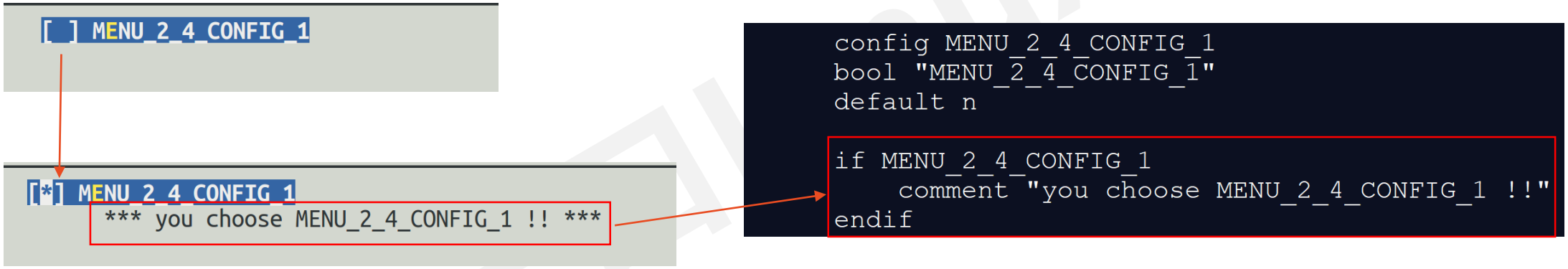
6. comment

- 注释。会显示一行注释信息



7. if endif

- if 判断。 if 后面跟着选项名，就是当该选项被选中时成立。



8. select

- 选中指定选项。 select 后面可以跟着已经定义了的配置选项，一般用作**如果XXX选项被选中则选中YYY选项。**

```
[ ] MENU_2_1_CONFIG_1
[ ] MENU_2_1_CONFIG_2 (NEW)
```

```
[*] MENU_2_1_CONFIG_1
-*- MENU_2_1_CONFIG_2
[*]   MENU_2_1_CONFIG_3
```

```
config MENU_2_1_CONFIG_1
    bool "MENU_2_1_CONFIG_1"
    select MENU_2_1_CONFIG_2
    select MENU_2_1_CONFIG_4
    default y

config MENU_2_1_CONFIG_2
    bool "MENU_2_1_CONFIG_2"
    default n

config MENU_2_1_CONFIG_3
    bool "MENU_2_1_CONFIG_3"
    depends on MENU_2_1_CONFIG_2
    default y

config MENU_2_1_CONFIG_4
    bool
    default n
```

为空，所以不显示

9. depends on

- 依赖于XXX选项。如果依赖的选项被选中了，则当前的选项才会显示，是连带关系。类似于子选项。

```
[ ] MENU_2_1_CONFIG_1
 [ ] MENU_2_1_CONFIG_2 (NEW)
```

```
[*] MENU_2_1_CONFIG_1
-*- MENU_2_1_CONFIG_2
[*]   MENU_2_1_CONFIG_3
```

```
config MENU_2_1_CONFIG_1
    bool "MENU_2_1_CONFIG_1"
    select MENU_2_1_CONFIG_2
    select MENU_2_1_CONFIG_4
    default y

config MENU_2_1_CONFIG_2
    bool "MENU_2_1_CONFIG_2"
    default n

config MENU_2_1_CONFIG_3
    bool "MENU_2_1_CONFIG_3"
    depends on MENU_2_1_CONFIG_2
    default y

config MENU_2_1_CONFIG_4
    bool
    default n
```

10. source

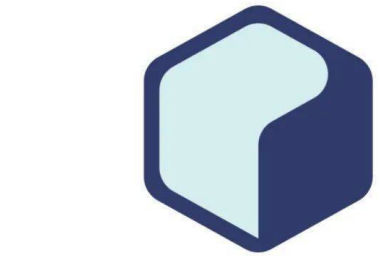
- 读取其他的配置文件。类似include
  - source "drivers/test/Kconfig"

```
74 source "drivers/test/Kconfig"
75
```

ARM

09

通过busybox制作文件系统



BusyBox

文件系统、Busybox入门

1. 常见嵌入式文件系统

- YAFFS（Yet Another Flash File System）：
  - 适用于NAND型Flash，支持JFFS2不支持YAFFS2。
- UBIFS（Unsorted Block Image File System）：
  - 适用于NAND型Flash，支持压缩和加密。
- JFFS（Journaling Flash File System）：
  - 适用于NOR型Flash，支持JFFS2。
- exFAT：
  - exFAT是微软为闪存U盘量身定制的，性能和技术支持先进，针对闪存优化，适合在移动设备中使用。Android11.0支持exFAT文件系统。
- Ext2/Ext3/Ext4：
  - 适用于硬盘等块设备，ext4是Linux操作系统中常见的文件系统，也是Android的默认文件系统。
- FAT/FAT32：
  - 诞生于1977年，最初为软盘设计的文件系统，后逐渐应用于硬盘上。 最大容量到2T，最大文件4G
- ntfs
  - Windows2000之后默认文件系统，支持更大的文件大小和分区容量。但是，NTFS在Android设备上通常只用于外部存储设备。
- APFs
  - 苹果公司发布的文件格式。

基于内存的文件系统

- ramdisk
  - 将一部分固定大小内存当做分区来使用。
- ramfs/Tmpfs
  - Linus开发的一种基于内存的文件系统。
- NFS
  - 由sun开发并发展起来的一向在不同机器、不同操作系统之间通过网络共享文件的技术。

重启后，目录下新建的内容就消失
适用于没有存储的环境

需要网络支持
调试环境采用nfs

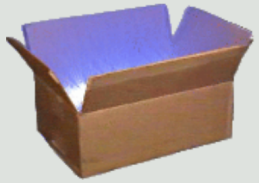
2. 什么是busybox?

- BusyBox是一个集成了大量的Linux命令（如ls、mv、ifconfig 等命令）和工具的软件。
- 借助BusyBox，进行配置和编译，就可以方便的构建一个嵌入Linux平台所需要的根文件系统。

3. Busybox获取

- https://busybox.net/

BUSYBOX



About

- About BusyBox
- BusyBox in VM
- Screenshot
- Announcements

Documentation

- FAQ
- Command Help

Get BusyBox

- Download Source
- Download Binaries
- License
- Products

Development

- Browse Source
- Source Control
- Mailing Lists
- Bug Tracking
- Use less RAM
- Contributing

Links

- Related Sites
- Tiny Utilities
- Sponsors

Developer Pages

- Rob

- The Software Freedom Conservancy acts as the GPL enforcement agent for various BusyBox c gpl@busybox.net.
- Life without systemd.
- **19 May 2023 -- BusyBox 1.36.1 (stable)**

  BusyBox 1.36.1. (git)

  Bug fix release. 1.36.1 has fixes for line editing, detection of hardware sha1/sha256 support, un SIGINT in sleep), ed.

- **3 January 2023 -- BusyBox 1.36.0 (unstable)**

  BusyBox 1.36.0. (git, patches, how to add a patch)

  Note: udhcpc6 now uses a different helper script by default (/usr/share/udhcpc/default6.script).

  Sizes of busybox-1.35.0 and busybox-1.36.0 (with equivalent config, static uclibc build):

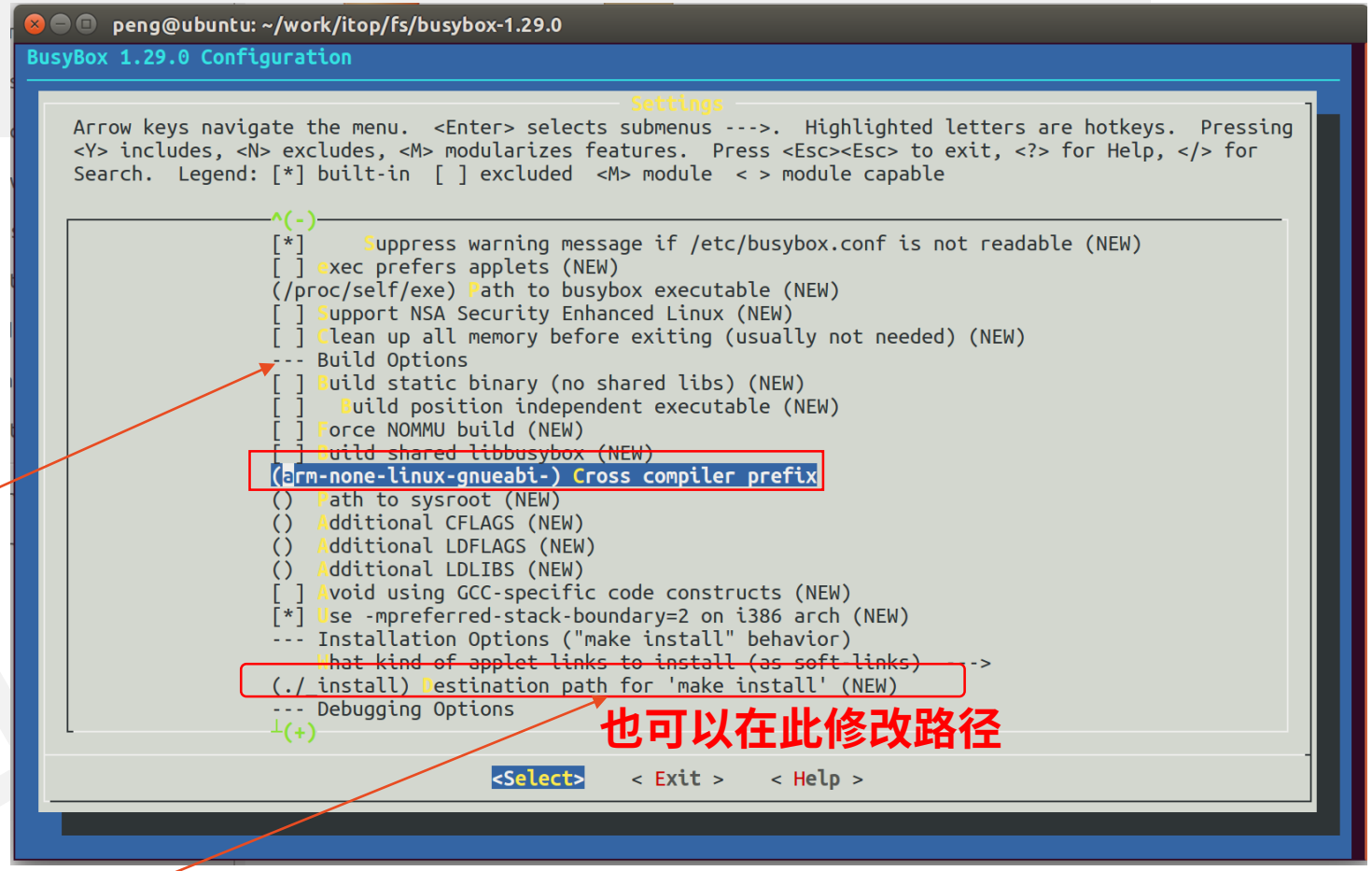
```
   text    data     bss     dec     hex filename
1044070     908   14328 1059306  1029ea busybox-1.35.0
1046317     908   14328 1061553  1032b1 busybox-1.36.0
```

  Changes since previous release:

```
Aaro Koskinen:
      devmem: add 128-bit width

Bernhard Reutner-Fischer (3):
      kbuild: fix building sha256
      kbuild: Prefer -Oz over -Os
      seedrng: manually inline seed_rng

Brandon Maier:
      xxd: fix typo in trivial usage

Dario Binacchi (2):
      fbset: abort on not handled options
      fbset: support setting pixel clock rate
```

4. Busybox编译

- 解压源码
  - tar xvf busybox-1.29.0.tar.bz2
- 配置源码
  - make menuconfig
    Settings --->
    ---Build Options
- 编译
  - make
- 安装
  - make install　　默认安装路径为源码目录下的_install



也可以在此修改路径

```
peng@ubuntu:~/work/itop/fs/busybox-1.29.0$ cd _install/
peng@ubuntu:~/work/itop/fs/busybox-1.29.0/_install$ ls
bin  linuxrc  sbin  usr
```

编译错误修改

```
==========
util-linux/lib.a(nsenter.o): In function `nsenter_main':
nsenter.c:(.text.nsenter_main+0x184): undefined reference to `setns'
coreutils/lib.a(sync.o): In function `sync_main':
sync.c:(.text.sync_main+0x70): undefined reference to `syncfs'
collect2: ld returned 1 exit status
```

Coreutils-→

peng@ubuntu: ~/work/itop/fs/busybox-1.29.0

BusyBox 1.29.0 Configuration

Linux System Utilities

Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted let
<Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exi
Search. Legend: [*] built-in [ ] excluded <M> module < > module capable

Linux System Utilities-→

```
                [*] mkswap (5.8 kb)
                [*]   UUID support
                [*] more (6.7 kb)
                [*] mount (30 kb)
                [*]   Support -f (fake mount)
                [*]   Support -v (verbose)
                [ ]   Support mount helpers
                [*]   Support specifying devices by label or UUID
                [ ]   Support mounting NFS file systems on Linux < 2.6.23
                [*]   Support mounting CIFS/SMB file systems
                [*]   Support lots of -o flags
                [*]   Support /etc/fstab and -a (mount all)
                [*]     Support -T <alt_fstab>
                [*] mountpoint (4.5 kb)
                [ ] nsenter (8.6 kb)
                [*] pivot_root (898 bytes)
                [*] rdate (6 kb)
                [*] rdev (1.4 kb)
                [*] readprofile (7.2 kb)
                [*] renice (3.8 kb)
                [*] rev (4.5 kb)
```

<Select> < Exit > < Help >

peng@ubuntu: ~/work/itop/fs/busybox-1.29.0

BusyBox 1.29.0 Configuration

Coreutils

Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are hotkeys. Pressing
<Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for
Search. Legend: [*] built-in [ ] excluded <M> module < > module capable

```
                [*] rm (4.9 kb)
                [*] rmdir (3.4 kb)
                [*] seq (3.6 kb)
                [*] shred (5 kb)
                [*] shuf (5.4 kb)
                [*] sleep (1.7 kb)
                [*]   Enable multiple arguments and s/m/h/d suffixes
                [*]     Enable fractional arguments
                [*] sort (7.4 kb)
                [*]   Full SuSv3 compliant sort (support -ktcbdfiogM)
                [ ]   Use less memory (but might be slower)
                [*] split (5.4 kb)
                [*]   Fancy extensions
                [*] stat (10 kb)
                [*]   Enable custom formats (-c)
                [*]   Enable display of filesystem status (-f)
                [*] stty (8.6 kb)
                [*] sum (4.3 kb)
                [ ] sync (769 bytes)
                [*] tac (4.1 kb)
                [*] tail (7.1 kb)
```

<Select> < Exit > < Help >

视频中所有资料

生成目录

```
peng@ubuntu:~/work/bak/fs/busybox-1.29.0/_install$ ls
bin  linuxrc  sbin  usr
```

rootfs配置文件定制

lib inittab fstab
init.d/rcS profile

1） 创建其他需要的目录

- mkdir dev etc mnt proc var tmp sys root

```
peng@ubuntu:~/work/itop/fs/busybox-1.29.0/_install$ ls
bin  dev  etc  mnt  proc  var  tmp  sys  root  linuxrc  sbin  usr
```

2）添加库

- 拷贝库
- cp /home/peng/toolchain/gcc-4.6.2-glibc-2.13-linaro-multilib-2011.12/fsl-linaro-toolchain/arm-fsl-linux-gnueabi/multi-libs/default/lib/ ./ -rf

```
peng@ubuntu:~/toolchain/gcc-4.6.2-glibc-2.13-linaro-multilib-2011.12/fsl-linaro-toolchain/arm-fsl-linux-gnueabi/multi-libs/default/lib$ ls
ld-2.13.so              libc-2.13.so      libmemusage.so        libnss_dns.so.2        libnss_nisplus.so.2  librt-2.13.so
ld-linux.so.3           libcrypt-2.13.so  libm.so.6             libnss_files-2.13.so   libnss_nis.so.2      librt.so.1
ldscripts               libcrypt.so.1     libnsl-2.13.so        libnss_files.so.2      libpcprofile.so      libSegFault.so
libanl-2.13.so          libc.so.6         libnsl.so.1           libnss_hesiod-2.13.so  libpthread-2.13.so   libthread_db-1.0.so
libanl.so.1             libdl-2.13.so     libnss_compat-2.13.so libnss_hesiod.so.2     libpthread.so.0      libthread_db.so.1
libBrokenLocale-2.13.so libdl.so.2        libnss_compat.so.2    libnss_nis-2.13.so     libresolv-2.13.so    libutil-2.13.so
libBrokenLocale.so.1    libm-2.13.so      libnss_dns-2.13.so    libnss_nisplus-2.13.so libresolv.so.2       libutil.so.1
```

- 修改文件权限并删除静态库和共享库文件中的符号表
  - chmod +w lib
    chmod +w lib/*
    rm lib/.a*
    *arm-none-linux-gnueabi-strip lib/**
- 删除不需要的库，确保所有库大小不超过8M
  - du -mh lib/

3）etc下添加文件inittab

```
1 #this is run first except when booting in single-user mode.
2 ::sysinit:/etc/init.d/rcS
3 # /bin/sh invocations on selected ttys
4 # start an "askfirst" shell on the console (whatever that may be)
5 ::askfirst:-/bin/sh
6 # stuff to do when restarting the init process
7 ::restart:/sbin/init
8 # stuff to do before rebooting
9 ::ctrlaltdel:/sbin/reboot
```

- 第 2 行，系统启动以后运行/etc/init.d/rcS 这个脚本文件
- 第 5 行，在运行 process 之前在控制台上显示“Please press Enter to activate this console.”。只要用户按下“Enter”键以后才会执行 process
- 第 7 行，重启的话运行/sbin/init
- 第 9 行，按下 ctrl+alt+del 组合键的话就运行/sbin/reboot，该组合键用于重启系统

Inittab格式说明

- inittab 的详细内容可以参考 busybox 下的文件 examples/inittab。
- init 程序会读取/etc/inittab这个文件， inittab 由若干条指令组成。
- 每条指令的结构都是一样的，由以“:”分隔的 4 个段组成，格式如下：
- <id>:<runlevels>:<action>:<process>
  - [id]：
    - 每个指令的标识符，不能重复。但是对于 busybox 的 init 来说， 有着特殊意义。对于 busybox 而言用来指定启动进程的控制 tty，一般我们将串口或者 LCD 屏幕设置为控制 tty
  - [runlevels]：
    - 对 busybox 来说此项完全没用，所以空着
  - [action]：
    - 动作，用于指定可能用到的动作
  - [process]：
    - 具体的动作，比如程序、脚本或命令等。

busybox 支持的动作

动作	描述
sysinit	在系统初始化的时候process才会执行一次
respawn	当 process 终止以后马上启动一个新的
askfirst	和 respawn 类似，在运行 process 之前在控制台上显示“Please press Enter to activatethis console.”。只要用户按下“Enter”键以后才会执行 process
wait	告诉 init，要等待相应的进程执行完以后才能继续执行
once	仅执行一次，而且不会等待 process 执行完成
restart	当 init 重启的时候才会执行 process
ctrlaltdel	当按下 ctrl+alt+del 组合键才会执行 process
shutdown	关机的时候执行 process

4）etc下添加文件fstab

peng@ubuntu: ~/work/itop/fs/busybox-1.29.0/_install

```
1 #device      mount-point    type       options          dump      fsck order
2 proc         /proc          proc       defaults         0         0
3 tmpfs        /tmp           tmpfs      defaults         0         0
4 sysfs        /sys           sysfs      defaults         0         0
5 tmpfs        /dev           tmpfs      defaults         0         0
```

这里我们挂载的文件系统有三个proc、sysfs和tmpfs。

在内核中proc和sysfs默认都支持，而tmpfs是没有支持的，我们需要确保内核有tmpfs的支持。

```
$ make menuconfig
File systems --->
  Pseudo filesystems --->
   [*] Virtual memory file system support (former shm fs)
   [*] Tmpfs POSIX Access Control Lists
```

fstab格式

- <file system> <mount point> <type> <options> <dump> <pass>
  - [file system]：
    - 要挂载的特殊的设备，也可以是块设备，比如/dev/sda 等等。
  - [mount point]：
    - 挂载点。
  - [type]：
    - 文件系统类型，比如 ext2、 ext3、 proc、 romfs、 tmpfs 等等。
  - [options]：
    - 挂载选项，在 Ubuntu 中输入“man mount”命令可以查看具体的选项。一般使用 defaults，也就是默认选项， defaults 包含了 rw、 suid、 dev、 exec、 auto、 nouser 和 async。
  - [dump]：
    - 为 1 的话表示允许备份，为 0 不备份，一般不备份，因此设置为 0。
  - [pass]：
    - 磁盘检查设置，为 0 表示不检查。根目录 ‘/’ 设置为 1，其他的都不能设置为 1，其他的分区从 2 开始。一般不在 fstab 中挂载根目录，因此这里一般设置为 0。

5）etc下创建init.d/rcS

- rcS 是个 shell 脚本， Linux 内核启动以后需要启动一些服务，而 rcS 就是规定启动哪些文件的脚本文件。

```
1 #!/bin/sh
2 # This is the first script called by init process
3 /bin/mount  -a     # 挂载fstab制定的所有文件系统
4 echo  /sbin/mdev  >  /proc/sys/kernel/hotplug
5 /sbin/mdev  -s
```

mount 命令来挂载所有的文件系统，这些文件系统由文件/etc/fstab 来指定

使用 mdev 来管理热插拔设备，通过这两行， Linux 内核就可以在/dev 目录下自动创建设备节点

增加权限

- chmod +x etc/init.d/rcS

6）etc下添加profile文件

- profile文件是是一个脚本文件，包含一系列的Shell命令，用于在用户登录时执行。
- linux启动文件profile的作用包括以下几个方面：
- 1.环境变量的设置：
  - profile文件中可以设置各种环境变量，如PATH等，用于告诉系统在哪里可以找到需要的命令或程序。
- 2.命令别名的设置：
  - 通常情况下，我们会把一些长的命令转换成一个短的别名，以方便我们的使用。profile文件中可以设置这些命令的别名。
- 3.启动程序的设置：
  - 有些程序需要在用户登录时启动，并且需要设置一些参数。这些参数可以存储在profile文件中，以便在用户登录时自动启动。
- 4.其他设置：
  - 其它设置还包括某些特定的操作系统的设置，启动定时任务等操作。

profile

```
1 #!/bin/sh
2 export HOSTNAME=yikoupeng
3 export USER=root
4 export HOME=root
5 export PS1="[$USER@$HOSTNAME \W]\# "
6 PATH=/bin:/sbin:/usr/bin:/usr/sbin
7 LD_LIBRARY_PATH=/lib:/usr/lib:$LD_LIBRARY_PATH
8 export PATH  LD_LIBRARY_PATH
9 mknod dev/console c  5  1
10
```

拷贝到 /home/peng/nfsroot

nfs挂载测试

nfs挂载根目录，环境变量

- ubuntu ip：
  - 192.168.0.112
- 开发板ip：
  - 192.168.0.111
- U-Boot配置命令：
  - setenv ipaddr 192.168.0.111
  - setenv bootargs root=/dev/nfs nfsroot=192.168.0.112:/home/peng/nfsroot,proto=tcp rw ip=192.168.0.111:192.168.0.112:192.168.0.1:255.255.255.0::eth0:off init=/linuxrc
  - mmc read 0x40007000 0x460 0x3000;mmc read 0x41000000 0x3460 0xa0;bootm 0x40007000 - 0x41000000

ext4格式文件系统制作

system.img

make_ext4fs

system.img制作 ext4

- xunwei\工具软件\ linux_tools.tgz

```
peng@ubuntu:~/work/itop$ tar xvf linux_tools.tgz
usr/
usr/local/
usr/local/bin/
usr/local/bin/mkimage
usr/local/bin/make_ext4fs
```

  - sudo cp usr/local/bin/make_ext4fs /usr/local/bin/
  - sudo chmod +x /usr/local/bin/make_ext4fs

make_ext4fs参数说明

```
peng@ubuntu:~/work/bak/fs$ make_ext4fs
Expected filename after options
make_ext4fs [ -l <len> ] [ -j <journal size> ] [ -b <block_size> ]
    [ -g <blocks per group> ] [ -i <inodes> ] [ -I <inode size> ]
    [ -L <label> ] [ -f ] [ -a <android mountpoint> ]
    [ -S file_contexts ]
    [ -z | -s ] [ -t ] [ -w ] [ -c ] [ -J ]
    <filename> [<directory>]
```

- **-l <len>**
  指定文件系统大小,单位为字节
- **-b <block_size>**
  指定文件系统的块大小
- **-g <blocks_per_group>**
  指定每个块组中的块数
- **-a <android mountpoint>**
  是指这个img用于android系统，设置挂载点目录
- **-L <label>**
  指定文件系统标签
- **-s**
  生成ext4的S模式制作

- **<filename>**
  要创建的镜像文件的路径和名称
- **[<directory>]**
  要将其内容添加到镜像中的目录;省略，则只会创建一个空的文件系统镜像

```
peng@ubuntu:~/work/itop/fs/busybox-1.29.0$ make_ext4fs -s -l 314572800 -a root -L linux system.img _install/
Creating filesystem with parameters:
    Size: 314572800
    Block size: 4096
    Blocks per group: 32768
    Inodes per group: 6400
    Inode size: 256
    Journal blocks: 1200
    Label: linux
    Blocks: 76800
    Block groups: 3
    Reserved block group size: 23
Created filesystem with 487/19200 inodes and 3429/76800 blocks
```

- make_ext4fs -s -l 314572800 -a root -L linux system.img _install/

测试

- fastboot.exe flash system system.img

ramdisk.img制作

什么是ramdisk?

- RAMDISK，就是内存磁盘，也就是在内存中开辟一块空间用于虚拟成为一个DISK。

- 对于一些经常被访问、并且不会被更改的文件，可以将它们通过ramDisk放在内存中，能够明显地提高系统性能。

ramdisk.img制作

- 1、制作一个大小为8M的镜像文件
  - dd if=/dev/zero of=ramdisk bs=1k count=8192 （ramdsik为8M）
- 2、格式化这个镜像文件为ext2
  - mkfs.ext2 -F ramdisk
- 3、在mount下面创建initrd目录作为挂载点
  - sudo mkdir /mnt/initrd
- 4、将这个磁盘镜像文件挂载到/mnt/initrd下
  - sudo mount -t ext2 ramdisk /mnt/initrd

ramdisk.img制作

- 5、将测试好的文件系统里的内容全部拷贝到 /mnt/initrd目录下面
  - sudo cp /home/peng/nfsroot/* /mnt/initrd –a
- 6、卸载/mnt/initrd
  - sudo umount /mnt/initrd
- 7、压缩ramdisk为ramdisk.gz
  - gzip --best -c ramdisk > ramdisk.gz
- 8、格式化为uboot识别的格式并拷贝到/tftpboot下
  - mkimage -n "ramdisk" -A arm -O linux -T ramdisk -C gzip -d ramdisk.gz ramdisk.img

测试

- uImage
  - mmc read 0x40007000 0x460 0x3000
- dtb
  - mmc read 0x41000000 0x3460 0xa0
- ramdisk
  - loadb 0x44000000

也可以用fatload

- 启动
  - bootm 0x40007000 0x44000000 0x41000000

如何设置开机自动启动程序

/etc/init.d/rcS

```
1 #!/bin/sh
2 # This is the first script called by init process
3 /bin/mount  -a     # 挂载fstab制定的所有文件系统
4 echo  /sbin/mdev  >  /proc/sys/kernel/hotplug
5 /sbin/mdev  -s
6
7 ifconfig eth0 192.168.0.111
```

Zynq平台

- /etc/rc.local

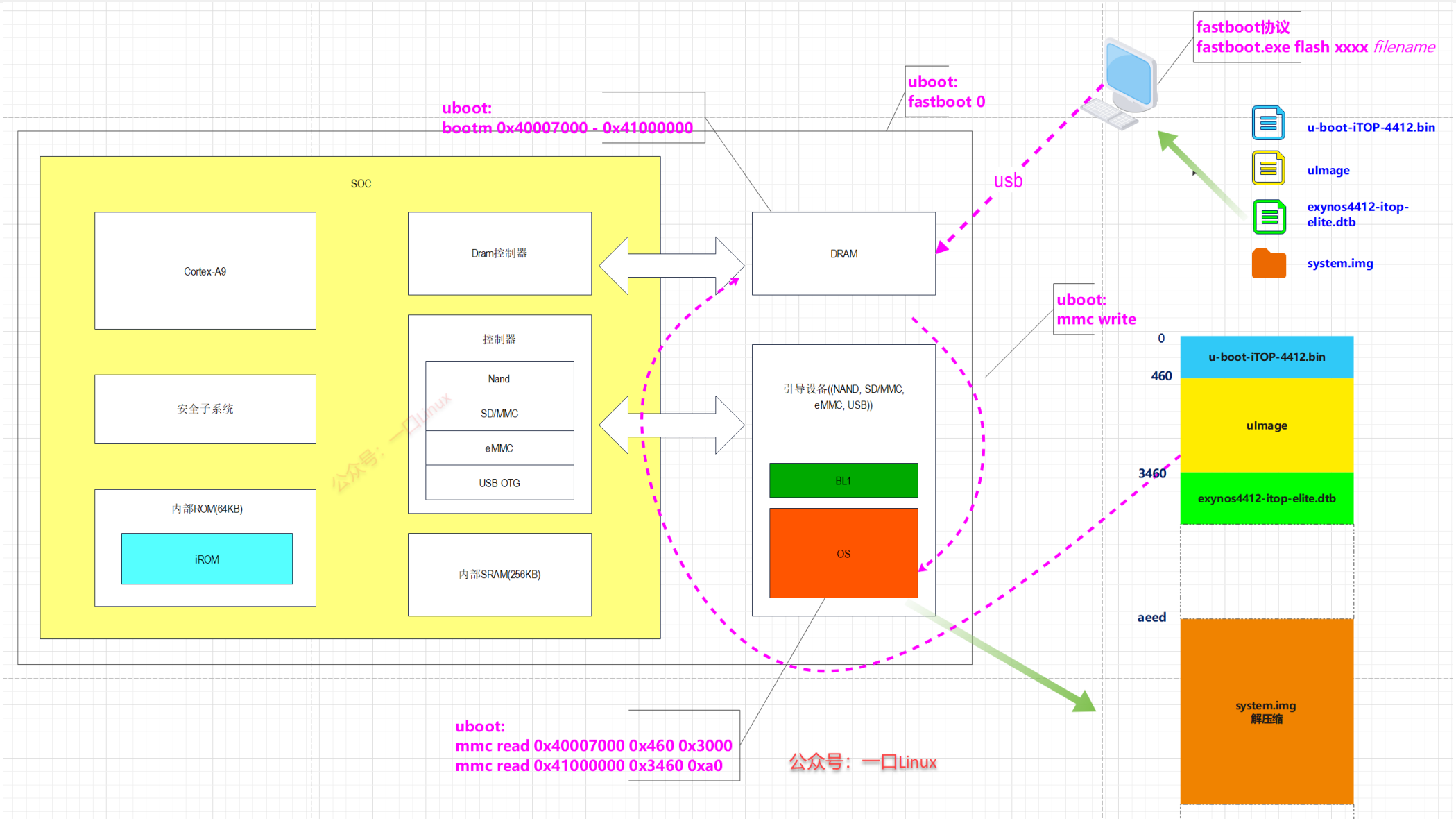
```
 1 #!/bin/sh -e
 2 #
 3 # rc.local
 4 #
 5 # This script is executed at the end of each multiuser runlevel.
 6 # Make sure that the script will "exit 0" on success or any other
 7 # value on error.
 8 #
 9 # In order to enable or disable this script just change the execution
10 # bits.
11 #
12 # By default this script does nothing.
13
14 exit 0
15 EOF
```

分区命令fdisk

- 只有uboot2010支持

  - fdisk -c 1　　慎用
  - fdisk -c 1 300 300 300

  - fatformat mmc 1:1
  - ext3format mmc 1:2
  - ext3format mmc 1:3
  - ext3format mmc 1:4


fastboot协议
fastboot.exe flash xxxx filename
uboot:
fastboot 0
uboot:
bootm 0x40007000 - 0x41000000
usb
u-boot-iTOP-4412.bin
uImage
exynos4412-itop-elite.dtb
system.img
SOC
Cortex-A9
Dram控制器
DRAM
控制器
Nand
SD/MMC
eMMC
USB OTG
安全子系统
内部ROM(64KB)
iROM
内部SRAM(256KB)
引导设备((NAND, SD/MMC, eMMC, USB))
BL1
OS
uboot:
mmc write
0
460
3460
aeed
u-boot-iTOP-4412.bin
uImage
exynos4412-itop-elite.dtb
system.img
解压缩
uboot:
mmc read 0x40007000 0x460 0x3000
mmc read 0x41000000 0x3460 0xa0
公众号：一口Linux
公众号：一口Linux

想入门和进阶ARM，
请加关注一口君的公众号：一口Linux