Exy4412 产品发布

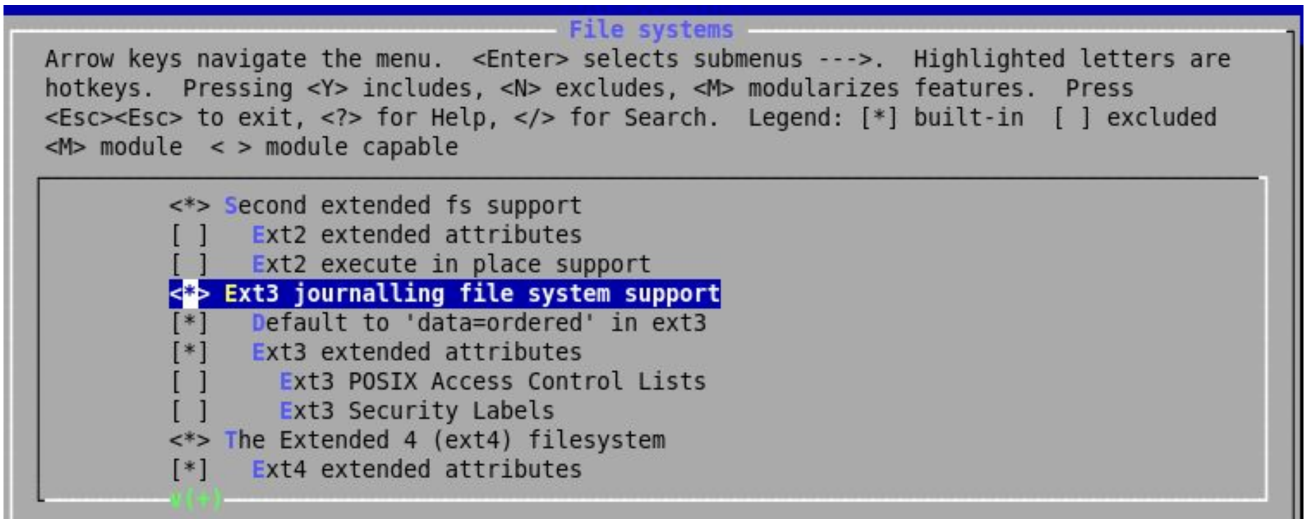
1、挂载方式

第一种：nfs 挂载（网络文件系统挂载），用来调试内核或者驱动。
第二种：本地挂载（u-boot kernel root）烧写到板子自带的 emmc 或 sd 卡。

2、配置内核支持 ext3 文件系统和 sd 卡驱动

配置内核

修改内核配置
确保 ext3 的文件系统支持,和 SD 卡驱动都被静态编译到内核,我使用的是 Linux 3.5
在内核源码目录执行 make menuconfig
其中 ext3 在 File system 下.成功的配置如下.



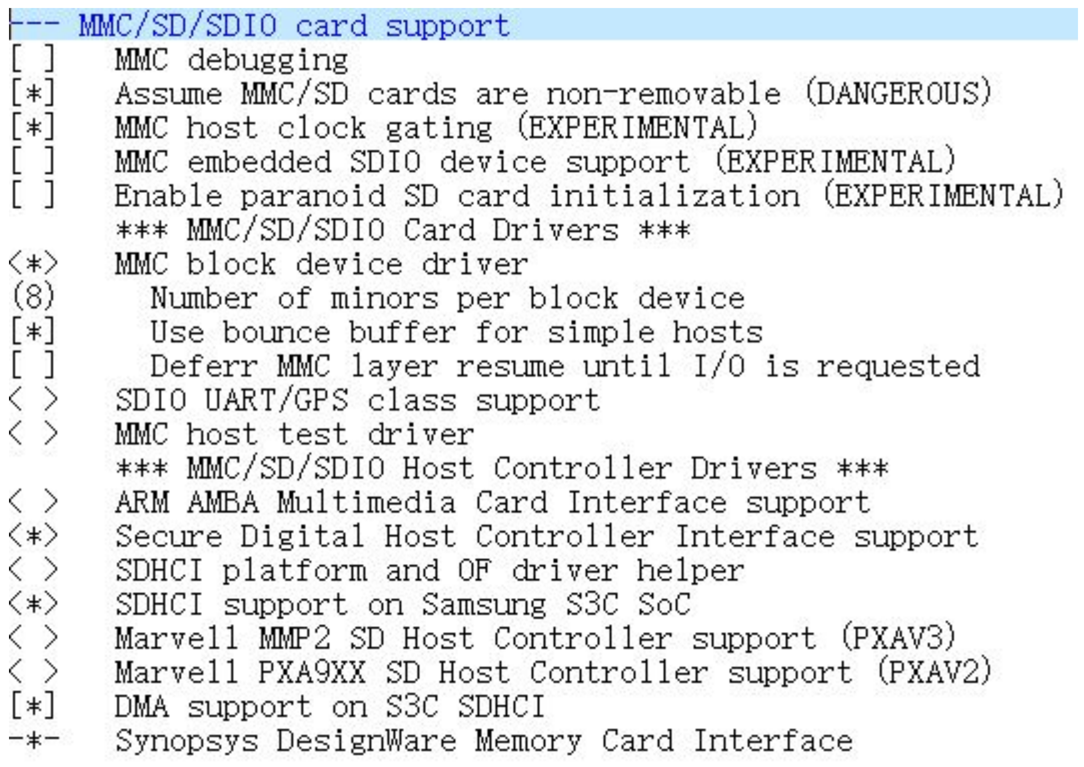
```
                              File systems
 Arrow keys navigate the menu.  <Enter> selects submenus --->.  Highlighted letters are
 hotkeys.  Pressing <Y> includes, <N> excludes, <M> modularizes features.  Press
 <Esc><Esc> to exit, <?> for Help, </> for Search.  Legend: [*] built-in  [ ] excluded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

          <*> Second extended fs support
          [ ]   Ext2 extended attributes
          [ ]   Ext2 execute in place support
          <*> Ext3 journalling file system support
          [*]   Default to 'data=ordered' in ext3
          [*]   Ext3 extended attributes
          [ ]     Ext3 POSIX Access Control Lists
          [ ]     Ext3 Security Labels
          <*> The Extended 4 (ext4) filesystem
          [*]   Ext4 extended attributes
          v(+)
```

```
(available at <http://sourceforge.net/projects/e2fsprogs/>).
 |
 | To compile this file system support as a module, choose M here: the
 | module will be called ext3.
 |
 | Symbol: EXT3_FS [=y]
 | Type   : tristate
 | Prompt: Ext3 journalling file system support
 |   Defined at fs/ext3/Kconfig:1
 |   Depends on: BLOCK [=y]
 |   Location:
 |     -> File systems
 |   Selects: JBD [=y]
 |
```

SD 卡驱动支持在 **Device Drivers ->MMC/SD/SDIO card support** 下**,**成功的配置如下：

```
--- MMC/SD/SDIO card support
[ ]    MMC debugging
[*]    Assume MMC/SD cards are non-removable (DANGEROUS)
[*]    MMC host clock gating (EXPERIMENTAL)
[ ]    MMC embedded SDIO device support (EXPERIMENTAL)
[ ]    Enable paranoid SD card initialization (EXPERIMENTAL)
        *** MMC/SD/SDIO Card Drivers ***
<*>     MMC block device driver
(8)       Number of minors per block device
[*]       Use bounce buffer for simple hosts
[ ]       Deferr MMC layer resume until I/O is requested
< >     SDIO UART/GPS class support
< >     MMC host test driver
        *** MMC/SD/SDIO Host Controller Drivers ***
< >     ARM AMBA Multimedia Card Interface support
<*>     Secure Digital Host Controller Interface support
< >     SDHCI platform and OF driver helper
<*>     SDHCI support on Samsung S3C SoC
< >     Marvell MMP2 SD Host Controller support (PXAV3)
< >     Marvell PXA9XX SD Host Controller support (PXAV2)
[*]    DMA support on S3C SDHCI
-*-    Synopsys DesignWare Memory Card Interface
```

```
--- MMC/SD/SDIO card support
[*]   MMC debugging
[ ]   Allow unsafe resume (DANGEROUS)
      *** MMC/SD/SDIO Card Drivers ***
<*>   MMC block device driver
[*]     Use bounce buffer for simple hosts
<*>   SDIO UART/GPS class support
<*>   MMC host test driver
      *** MMC/SD/SDIO Host Controller Drivers ***
<*>   Secure Digital Host Controller Interface support
<*>   Samsung S3C SD/MMC Card Interface support
```


```
├-- MMC/SD/SDIO card support
[ ]   MMC debugging
[*]   Assume MMC/SD cards are non-removable (DANGEROUS)
[*]   MMC host clock gating (EXPERIMENTAL)
[ ]   MMC embedded SDIO device support (EXPERIMENTAL)
[ ]   Enable paranoid SD card initialization (EXPERIMENTAL)
      *** MMC/SD/SDIO Card Drivers ***
<*>   MMC block device driver
(8)     Number of minors per block device
[*]     Use bounce buffer for simple hosts
[ ]     Deferr MMC layer resume until I/O is requested
< >   SDIO UART/GPS class support
< >   MMC host test driver
      *** MMC/SD/SDIO Host Controller Drivers ***
< >   ARM AMBA Multimedia Card Interface support
<*>   Secure Digital Host Controller Interface support
< >   SDHCI platform and OF driver helper
<*>   SDHCI support on Samsung S3C SoC
< >   Marvell MMP2 SD Host Controller support (PXAV3)
< >   Marvell PXA9XX SD Host Controller support (PXAV2)
[*]   DMA support on S3C SDHCI
-*-   Synopsys DesignWare Memory Card Interface
```

3. 烧写 u-boot 和 kernel 到 sd 卡的步骤

1) 把 sd 卡插入 pc 机在虚拟机 redhat 识别出来

```
[root@localhost boot]# ls /dev/sd*
/dev/sda    /dev/sda1    /dev/sda2    /dev/sdc    /dev/sdc1
```

上面的的/dev/sdc 就是 sd 卡

烧写 uboot 到 sd 卡（步骤不在解释）

使用 uboot 烧写脚本：

```
[root@localhost xyd4412]# ./sd_fusing.sh /dev/sdb
```

把 kernel 烧写进 sd 卡（步骤不在解释）

```
[root@localhost boot]# ./fush_uimage
```

把 sd 卡插入 tiny4412，启动。

4. 烧写根文件系统到 ext3 格式的文件

4.1 分区，并且格式化

进入 uboot 命令，格式化 sd 卡，并且分为四个区，其他第一个分区为 FAT 格式，主要用来存放 u-boot.bin uImage 等。其他分区使用 ext3 格式文件。

把 emmc 分 4 个区：

SMDK4412 # **fdisk -c 0 320 2057 520**

把 sd 第一个分区初始化为 fat。

SMDK4412 # **fatformat mmc 0:1**

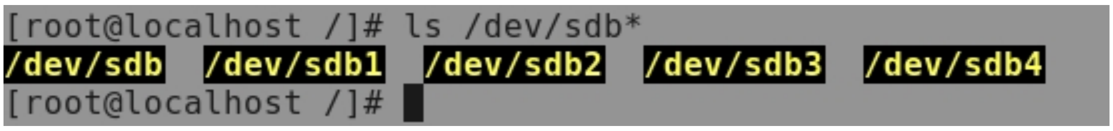
把 sd 第二、三、四个分区初始化为 ext3。
SMDK4412 # **ext3format mmc 0:2**
SMDK4412 # **ext3format mmc 0:3**
SMDK4412 # **ext3format mmc 0:4**

4.3 复制根文件系统到系统 sd 卡存放根文件系统的分区

把 sd 卡拔出放到虚拟机上，把之前制作好的根文件系统拷贝到 **ext3** 格式文件的分区（注：fat 格式文件不直接链接文件）。

查看 sd 卡分区，

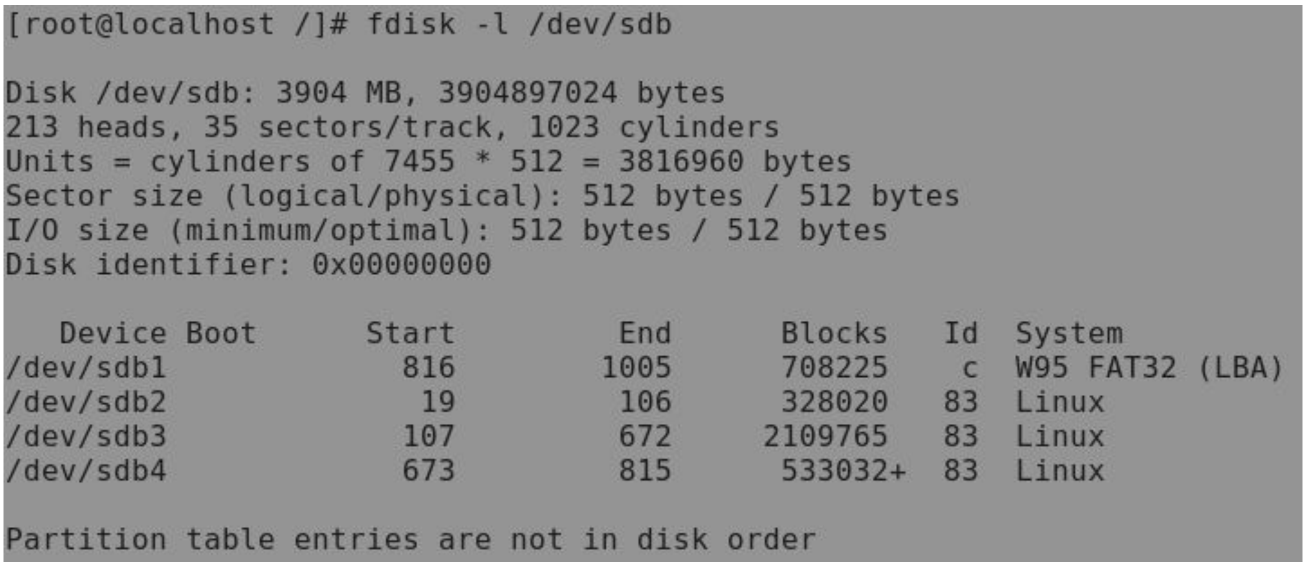
```
[root@localhost /]# ls /dev/sdb*
/dev/sdb    /dev/sdb1    /dev/sdb2    /dev/sdb3    /dev/sdb4
```



```
[root@localhost /]# ls /dev/sdb*
/dev/sdb  /dev/sdb1  /dev/sdb2  /dev/sdb3  /dev/sdb4
[root@localhost /]#
```

查看 sd 卡详细分区：

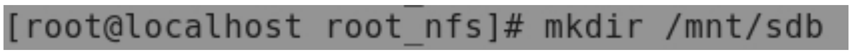
```
[root@localhost /]# fdisk -l /dev/sdb
Disk /dev/sdb: 3904 MB, 3904897024 bytes
213 heads, 35 sectors/track, 1023 cylinders
Units = cylinders of 7455 * 512 = 3816960 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000
   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1             816        1005      708225    c  W95 FAT32 (LBA)
/dev/sdb2              19         106      328020   83  Linux
/dev/sdb3             107         672     2109765   83  Linux
/dev/sdb4             673         815      533032+  83  Linux

Partition table entries are not in disk order
[root@localhost /]#
```



```
[root@localhost /]# fdisk -l /dev/sdb

Disk /dev/sdb: 3904 MB, 3904897024 bytes
213 heads, 35 sectors/track, 1023 cylinders
Units = cylinders of 7455 * 512 = 3816960 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000

   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1             816        1005      708225    c  W95 FAT32 (LBA)
/dev/sdb2              19         106      328020   83  Linux
/dev/sdb3             107         672     2109765   83  Linux
/dev/sdb4             673         815      533032+  83  Linux

Partition table entries are not in disk order
```

4.3 把 sd 卡分区 2 的设备节点挂载到/mnt 目录

[root@localhost root_nfs]# **mkdir /mnt/sdb**



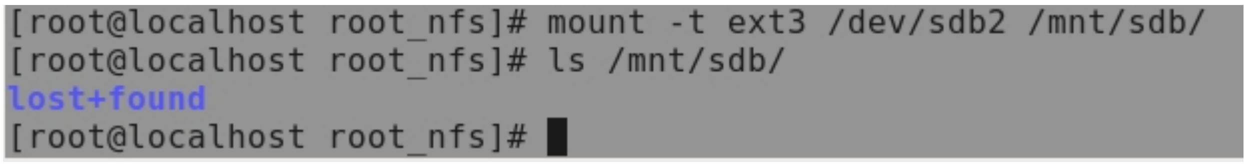
```
[root@localhost root_nfs]# mkdir /mnt/sdb
```

[root@localhost root_nfs]# **mount -t ext3 /dev/sdb2 /mnt/sdb/**

[root@localhost root_nfs]# **ls /mnt/sdb/**

lost+found

[root@localhost root_nfs]#



```
[root@localhost root_nfs]# mount -t ext3 /dev/sdb2 /mnt/sdb/
[root@localhost root_nfs]# ls /mnt/sdb/
lost+found
[root@localhost root_nfs]# 
```

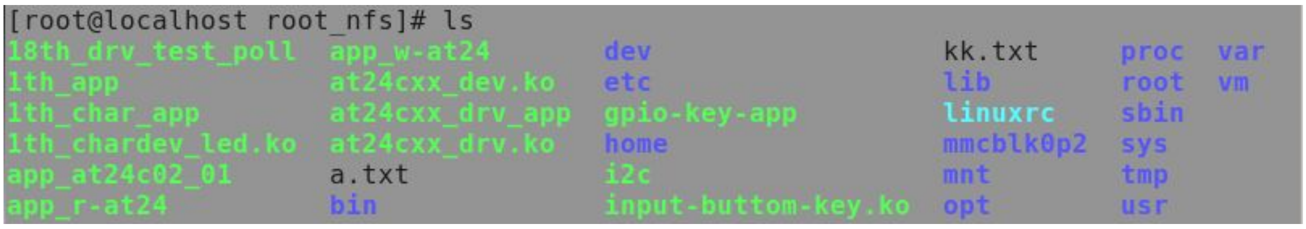
4.4 把制作好的根文件系统拷贝到/mnt（注：此时操作/mnt 就是操作/dev/sdb2）

进入根文件系统目录：

[root@localhost /]# **cd /xyd/rootfs/**

[root@localhost root_nfs]# **ls**

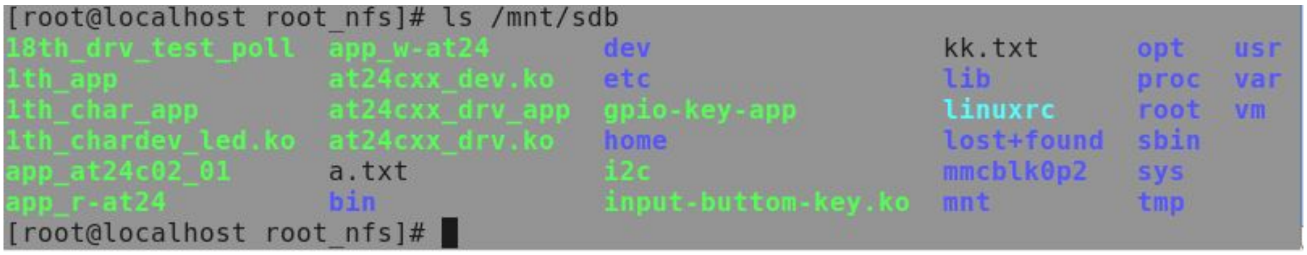
```
18th_drv_test_poll   app_w-at24        dev                    kk.txt       proc   var
1th_app              at24cxx_dev.ko    etc                    lib          root   vm
1th_char_app         at24cxx_drv_app   gpio-key-app           linuxrc      sbin
1th_chardev_led.ko   at24cxx_drv.ko    home                   mmcblk0p2    sys
app_at24c02_01       a.txt             i2c                    mnt          tmp
app_r-at24           bin               input-buttom-key.ko    opt          usr
```



```
[root@localhost root_nfs]# ls
18th_drv_test_poll  app_w-at24       dev                  kk.txt     proc  var
1th_app             at24cxx_dev.ko   etc                  lib        root  vm
1th_char_app        at24cxx_drv_app  gpio-key-app         linuxrc    sbin
1th_chardev_led.ko  at24cxx_drv.ko   home                 mmcblk0p2  sys
app_at24c02_01      a.txt            i2c                  mnt        tmp
app_r-at24          bin              input-buttom-key.ko  opt        usr
```

复制全部文件到 sd 卡分区 2 所接的目录：

```
[root@localhost root_nfs]# cp * -R /mnt/sdb/
[root@localhost root_nfs]# ls   /mnt/sdb
18th_drv_test_poll   app_w-at24        dev                  kk.txt      opt    usr
1th_app              at24cxx_dev.ko    etc                  lib         proc   var
1th_char_app         at24cxx_drv_app   gpio-key-app         linuxrc     root   vm
1th_chardev_led.ko   at24cxx_drv.ko    home                 lost+found  sbin
app_at24c02_01       a.txt             i2c                  mmcblk0p2   sys
app_r-at24           bin               input-buttom-key.ko  mnt         tmp
[root@localhost root_nfs]#
```

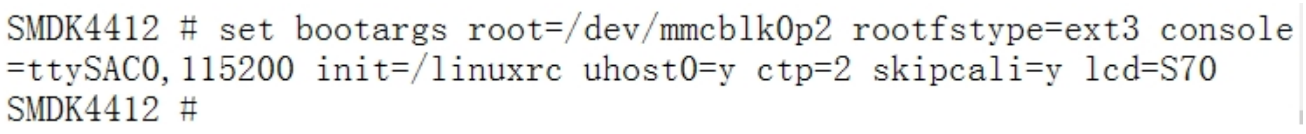


```
[root@localhost root_nfs]# ls /mnt/sdb
18th_drv_test_poll  app_w-at24       dev                  kk.txt      opt   usr
1th_app             at24cxx_dev.ko   etc                  lib         proc  var
1th_char_app        at24cxx_drv_app  gpio-key-app         linuxrc     root  vm
1th_chardev_led.ko  at24cxx_drv.ko   home                 lost+found  sbin
app_at24c02_01      a.txt            i2c                  mmcblk0p2   sys
app_r-at24          bin              input-buttom-key.ko  mnt         tmp
[root@localhost root_nfs]#
```

4.5 重启开发板，修改 u-boot 环境变量

拷贝完成后，放回 tiny4412 中，启动 u-boot，修改 bootargs。

```
SMDK4412 # set bootargs root=/dev/mmcblk0p2 rootfstype=ext3 console=ttySAC0,115200 init=/linuxrc uhost0=y ctp=2 skipcali=y lcd=S70
```

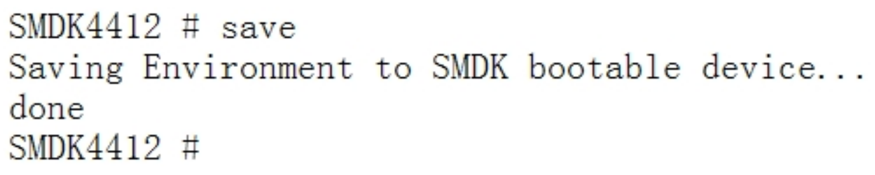


```
SMDK4412 # set bootargs root=/dev/mmcblk0p2 rootfstype=ext3 console
=ttySAC0,115200 init=/linuxrc uhost0=y ctp=2 skipcali=y lcd=S70
SMDK4412 #
```

（注:其中 mmcblk0p3:表示 blk0 表示第一个设备，也就是 sd 卡。p2 表示第二个分区）

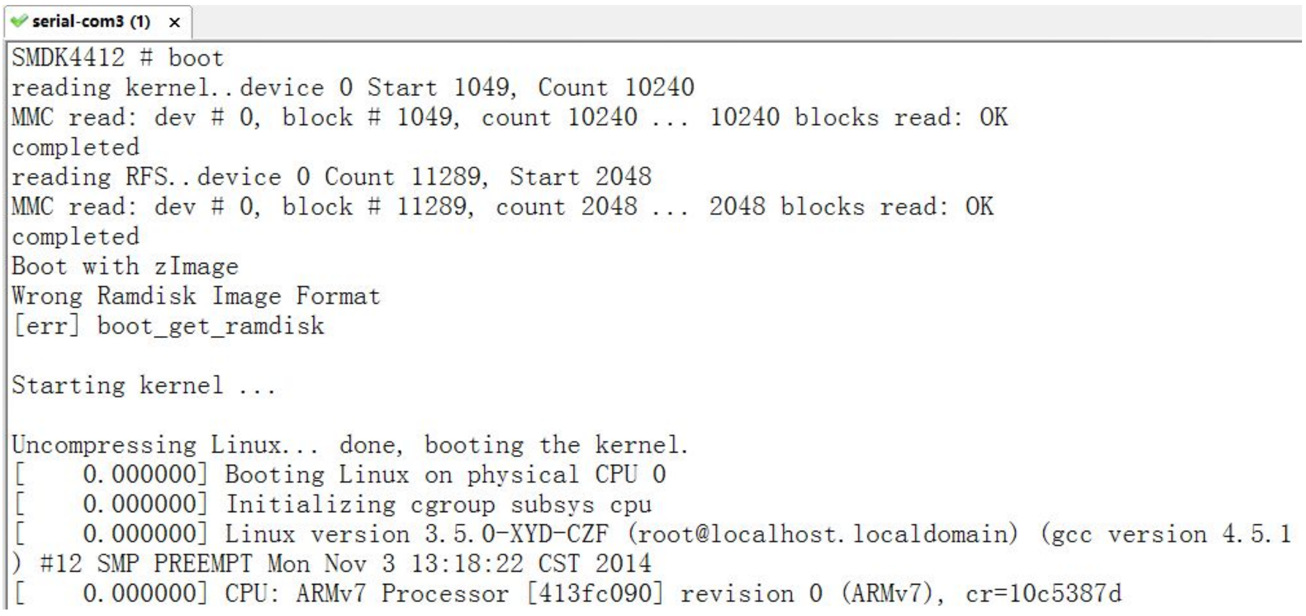
保存环境变量：

```
SMDK4412 # save
Saving Environment to SMDK bootable device...
done
SMDK4412 #
```



```
SMDK4412 # save
Saving Environment to SMDK bootable device...
done
SMDK4412 #
```

```
SMDK4412 #boot
reading kernel..device 0 Start 1049, Count 10240
MMC read: dev # 0, block # 1049, count 10240 ... 10240 blocks read: OK
completed
reading RFS..device 0 Count 11289, Start 2048
MMC read: dev # 0, block # 11289, count 2048 ... 2048 blocks read: OK
completed
Boot with zImage
Wrong Ramdisk Image Format
[err] boot_get_ramdisk

Starting kernel ...

Uncompressing Linux... done, booting the kernel.
[    0.000000] Booting Linux on physical CPU 0
[    0.000000] Initializing cgroup subsys cpu
[    0.000000] Linux version 3.5.0-XYD-CZF (root@localhost.localdomain) (gcc version 4.5.1 (ctng-1.8.1-FA) )
#12 SMP PREEMPT Mon Nov 3 13:18:22 CST 2014
[    0.000000] CPU: ARMv7 Processor [413fc090] revision 0 (ARMv7), cr=10c5387d
[    0.000000] CPU: PIPT / VIPT nonaliasing data cache, VIPT aliasing instruction cache
[    0.000000] Machine: TINY4412
[    0.000000] USB PHY0 configured as HOST mode
[    0.000000] TINY4412: S70 selected
[    0.000000] cma: CMA: reserved 32 MiB at 6d800000
[    0.000000] Memory policy: ECC disabled, Data cache writealloc
```

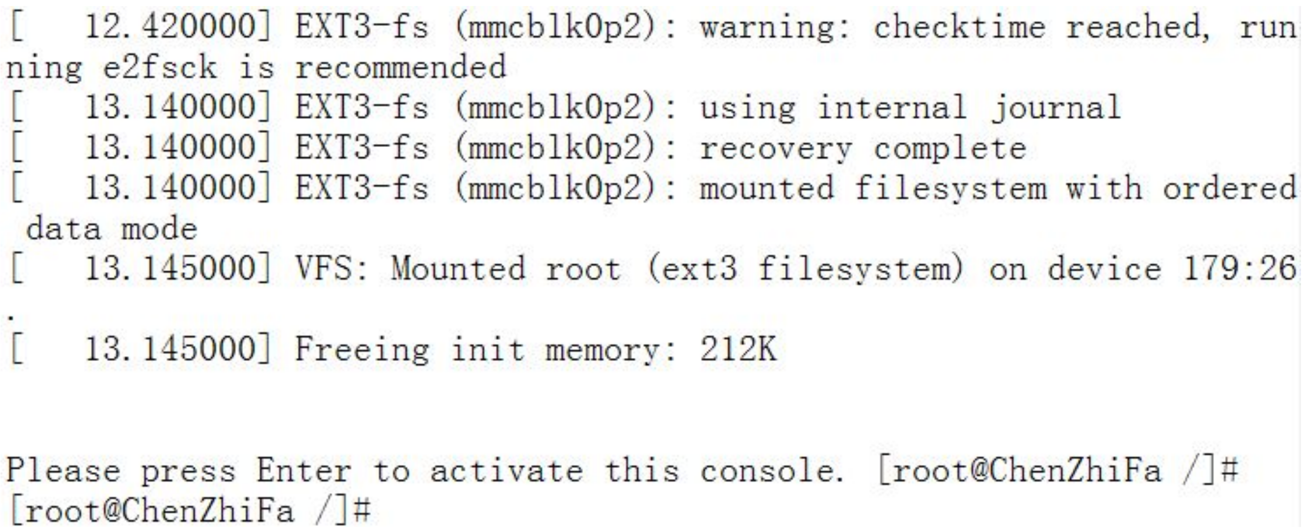

serial-com3 (1)

```
SMDK4412 # boot
reading kernel..device 0 Start 1049, Count 10240
MMC read: dev # 0, block # 1049, count 10240 ... 10240 blocks read: OK
completed
reading RFS..device 0 Count 11289, Start 2048
MMC read: dev # 0, block # 11289, count 2048 ... 2048 blocks read: OK
completed
Boot with zImage
Wrong Ramdisk Image Format
[err] boot_get_ramdisk

Starting kernel ...

Uncompressing Linux... done, booting the kernel.
[    0.000000] Booting Linux on physical CPU 0
[    0.000000] Initializing cgroup subsys cpu
[    0.000000] Linux version 3.5.0-XYD-CZF (root@localhost.localdomain) (gcc version 4.5.1
) #12 SMP PREEMPT Mon Nov 3 13:18:22 CST 2014
[    0.000000] CPU: ARMv7 Processor [413fc090] revision 0 (ARMv7), cr=10c5387d
```

…… 其他省略。。。。。

```
[    3.350000] ALSA device list:
```

```
[       3.350000]      No soundcards found.
[       3.380000] kjournald starting.    Commit interval 5 seconds
[       3.380000] EXT3-fs (mmcblk0p2): warning: checktime reached, running e2fsck is recommended
[       3.380000] EXT3-fs (mmcblk0p2): using internal journal
[       3.380000] EXT3-fs (mmcblk0p2): recovery complete
[       3.385000] EXT3-fs (mmcblk0p2): mounted filesystem with ordered data mode
[       3.385000] VFS: Mounted root (ext3 filesystem) on device 179:26.
[       3.390000] Freeing init memory: 212K
[       3.420000] usb 1-2.1: New USB device found, idVendor=0424, idProduct=4040
[       3.420000] usb 1-2.1: New USB device strings: Mfr=1, Product=2, SerialNumber=3
[       3.420000] usb 1-2.1: Product: Ultra Fast Media Reader
[       3.420000] usb 1-2.1: Manufacturer: Generic
[       3.420000] usb 1-2.1: SerialNumber: 000000264001
[       3.425000] scsi0 : usb-storage 1-2.1:1.0
[       3.515000] usb 1-2.2: new full-speed USB device number 4 using s5p-ehci
[       3.620000] usb 1-2.2: not running at top speed; connect to a high speed hub
[       3.620000] usb 1-2.2: config 1 interface 0 altsetting 0 endpoint 0x83 has an invalid bInterval 0, changing to 32
[       3.695000] usb 1-2.2: New USB device found, idVendor=0a46, idProduct=9621
[       3.695000] usb 1-2.2: New USB device strings: Mfr=1, Product=2, SerialNumber=3
[       3.695000] dm962x: dm_read_reg() 0x29 0x0a
[       3.695000] dm962x: dm_read_reg() 0x28 0x46
[       3.695000] dm962x: dm_read_reg() 0x2b 0x96
[       3.700000] dm962x: dm_read_reg() 0x2a 0x21
[       3.705000] dm962x: dm_read_reg() 0xF2 0x00
[       3.705000] dm962x:    [Analysis.2] 0xF2, D[7] 0 OK
[       3.710000] dm962x:    [Analysis.2] 0xF2, D[6] 0 OK
[       3.715000] dm962x:    [Analysis.2] 0xF2, D[5] 0 EP1: Empty
[       3.720000] dm962x:    [Analysis.2] 0xF2, D[3] 0 OK
[       3.725000] dm962x:    [Analysis.2] 0xF2, D[2] 0 OK
[       3.730000] dm962x:    [Analysis.2] 0xF2, D[1] 0 OK
[       3.735000] dm962x:    [Analysis.2] 0xF2, D[0] 0 Status: TX buffer 0 pkts
[       3.750000] dm962x: ethernet MAC address 00:00:ff:ff:00:00 (chip)
[       3.750000] dm962x: 9620 Mode = 128
[       3.765000] dm9620 1-2.2:1.0: eth0: register 'dm9620' at usb-s5p-ehci-2.2, Davicom DM9620 USB Ethernet,
00:00:ff:ff:00:00

Please press Enter to activate this console. [       3.850000] usb 1-2.3: new high-speed USB device number 5 using
s5p-ehci
[       3.955000] usb 1-2.3: New USB device found, idVendor=1a40, idProduct=0101
[       3.955000] usb 1-2.3: New USB device strings: Mfr=0, Product=1, SerialNumber=0
[       3.955000] usb 1-2.3: Product: USB 2.0 Hub
[       3.960000] hub 1-2.3:1.0: USB hub found
[       3.960000] hub 1-2.3:1.0: 4 ports detected
[       4.430000] scsi 0:0:0:0: Direct-Access         Generic    Ultra HS-COMBO      2.01 PQ: 0 ANSI: 0
[       4.445000] sd 0:0:0:0: Attached scsi generic sg0 type 0
[       4.450000] sd 0:0:0:0: [sda] Attached SCSI removable disk

[root@ChenZhiFa /]#
```



```
[   12.420000] EXT3-fs (mmcblk0p2): warning: checktime reached, run
ning e2fsck is recommended
[   13.140000] EXT3-fs (mmcblk0p2): using internal journal
[   13.140000] EXT3-fs (mmcblk0p2): recovery complete
[   13.140000] EXT3-fs (mmcblk0p2): mounted filesystem with ordered
 data mode
[   13.145000] VFS: Mounted root (ext3 filesystem) on device 179:26
.
[   13.145000] Freeing init memory: 212K


Please press Enter to activate this console. [root@ChenZhiFa /]#
[root@ChenZhiFa /]#
```

那么像要在 tiny4412 自带的 movinand 启动挂载根文件系统。也是同样的道理，把 movinand 看做 sd 卡即可。下面是操作步骤：

一、使用 sd 卡（必须有 uboot 和 kernel），把 sd 卡的 uboot 和 kernel 拷贝到 movinand。

从 SD 启动 uboot，进入 uboot 命令：

把 sd 卡中 u-boot 的第一阶段的 bl0 数据复制到内存，然后再写入 emmc 对应位置

movi r f 0 40000000;emmc open 1;movi w z f 1 40000000;emmc close 1;

把 sd 卡中 u-boot 密数据复制到内存，然后再写入 emmc 对应位置

movi r b 0 40000000;emmc open 1;movi w z b 1 40000000;emmc close 1;

把 sd 卡中 u-boot 密数据复制到内存，然后再写入 emmc 对应位置

movi r u 0 40000000;emmc open 1;movi w z u 1 40000000;emmc close 1;

把 sd 卡中 u-boot 安全加密数据复制到内存，然后再写入 emmc 对应位置

movi r t 0 40000000;emmc open 1;movi w z t 1 40000000;emmc close 1;

把 sd 卡中内核数据复制到内存，然后再写入 emmc 对应位置

movi r k 0 40000000;movi w k 1 40000000;

二、烧写根文件系统到 ext3 格式的文件

开关切换到 eMMC 启动。

1、进入 uboot 命令，格式化 emmc 卡，并且分为四个区，其他第一个分区为 FAT 格式，主要用来存放 u-boot.bin uImage 等。其他分区使用 ext3 格式文件。

```
# fdisk -c 0 320 2057 520
# fatformat mmc 0:1          //把 emmc 第一个分区初始化为 fat
# ext3format mmc 0:2         //把 emmc 第二、三、四个分区初始化为 ext3
# ext3format mmc 0:3
# ext3format mmc 0:4
```

2.1、使用 nfs 挂载方式，在 emmc 的分区 2 的设备节点挂载到/mnt 目录

先挂接 NFS，然后进入 Linux 系统后，开发板终端输入：

```
#set bootargs noinitrd root=/dev/nfs
nfsroot=192.168.0.101:/xyd/rootfs/ip=192.168.0.99:192.168.0.101:192.168.0.1:255.255.255.0::eth0:off
init=/linuxrc console=ttySAC0 lcd=S70
#save
#boot
```

进入到 Linux 系统后，在开发板终端执行以下命令：

创建 NFS 文件系统挂载点：

```
# mkdir /mnt/nfs -p
```

挂载虚拟机的 NFS 文件系统到开发板的/mnt/nfs 目录

```
# mount -o nolock,proto=tcp,nfsvers=3 192.168.0.101:/xyd/rootfs/   /mnt/nfs
```

创建 EMMC 分区 2 挂载点：

```
#mkdir /mnt/mmcblk0p2 -p
```

挂载虚拟机的 EMMC 分区 2 到开发板的/mnt/mmcblk0p2 **目录**

```
#mount -t ext3 /dev/mmcblk0p2 /mnt/mmcblk0p2/
```

2.2、把制作好的根文件系统拷贝到/mnt（注：此时操作/mnt 就是操作/dev/sdc2）

复制虚拟机上的 NFS 文件系统到 EMMC 分区 2

```
#cp -Rf /mnt/nfs/* /mnt/mmcblk0p2/
```

卸载 eMMC 分区 2 挂载

```
#umount /mnt/mmcblk0p2/
```

2.3、拷贝完成后，重启开发板，从 eMMC 启动，修改 bootargs

```
#set bootargs root=/dev/mmcblk0p2 rootfstype=ext3 rootwait console=ttySAC0,115200 init=/linuxrc
uhost0=y ctp=2 skipcali=y lcd=S70
#save
```

（注:其中 mmcblk0p2:表示 blk0 表示第一个设备，也就是 sd 卡。p2 表示第二个分区）