UBI文件系统及相关工具的使用

**产生的背景**

在linux-2.6.27以前，谈到[Flash](http://www.dzsc.com/stock-ic/FLASH.html" \t "_blank)文件系统，大家很多时候多会想到cramfs、jffs2、yaffs2等文件系统。它们也都是基于文件系统+mtd+[flash](http://www.dzsc.com/stock-ic/FLASH.html)设备的架构。linux-2.6.27后，内核加入了一种新型的[FLASH](http://www.dzsc.com/stock-ic/FLASH.html" \t "_blank)文件系统UBI(Unsor[te](http://www.dzsc.com/product/searchfile/10366.html)d Block Images)。

FLASH具有的“先擦除再写入”、坏块、“有限的读写次数”等特性，目前管理FLASH的方法主要有：

　　1、采用MTD+FTL／NFTL（flash 转换层／nand flash转换层）＋ 传统文件系统，如：FAT、ext2等。FTL／NFTL的使用就是针对FLASH的特有属性，通过软件的方式来实现日志管理、坏块管理、损益均衡等技术。但实践证明，由于知识产权、效率等各方面因素导致本方案有一定的局限性。

　　2、采用硬件翻译层+传统文件系统的方案。这种方法被很多[存储卡](http://www.dzsc.com/product/searchfile/4531.html)产品采用，如：[SD卡](http://www.dzsc.com/product/searchfile/6339.html)、[U盘](http://www.dzsc.com/product/searchfile/18089.html)等。这种方案对于一些产品来说，成本较高。

　　3、采用MTD+ FLASH专用文件系统，如JFFS1／2，YAFFS1/2等。它们大大提高了FLASH的管理能力，并被广泛应用。

　　JFFS2、YAFFS2等专用文件系统也存在着一些技术瓶颈，如：[内存](http://www.dzsc.com/product/searchfile/4072.html" \t "_blank)消耗大，对FLASH容量、文件系统大小、内容、访问模式等的线性依赖，损益均衡能力差或过渡损益等。在此背景下内核加入了UBI文件系统的支持。

**STH239平台的使用背景**

由于STiH239芯片使用了BCH NAND控制器，该控制器可以达到18Bits ECC，使用OOB区域。由于jffs2,yaffs等文件系统将同样使用OOB区域而导致冲突。UBI文件系统不使用OOB区域，所以就需要更新文件系统为UBI文件系统。

**Kenrel配置**

在以前的基础上配置内核支持UBIFS

配置mtd支持UBI[接口](http://www.dzsc.com/product/searchfile/5540.html)

[Device](http://www.dzsc.com/icstock/857/DEVICE.html) Drivers  --->[Memory](http://www.dzsc.com/icstock/933/MEMORY.html) [Technology](http://www.dzsc.com/stock-ic/TECHNOLOGY.html) [device](http://www.dzsc.com/icstock/857/DEVICE.html) (MTD) support  --->UBI - Unsorted block images  --->Enable UBI   
配置内核支持UBIFS文件系统

 File [system](http://www.dzsc.com/stock-ic/SYSTEM.html)s  --->Miscellaneous file[System](http://www.dzsc.com/stock-ic/SYSTEM.html)s  --->UBIFS file system support 

**UBI相关工具支持**

ubiattach ubiformat ubimkvol ubirename ubiupdatevol

ubicrc32 ubigen ubinfo ubirmvol

ubidetach ubimirror ubinize ubirsvol

flash\_erase /dev/mtd4 0 0 //擦除mtd4

ubiformat /dev/mtd4 -s 2048 -O 2048 -v –y //分区格式化成UBI格式

ubiattach /dev/ubi\_ctrl -m 4 //和mtd4关联,会指定ubi0,1,2…

ubimkvol /dev/ubi0 -N rootfs -m //设定volume 大小 -s（-s [100M](http://www.dzsc.com/stock-ic/100M.html)iB可以指定大小，可以用工具改变）及名称

ubiupdatevol /dev/ubi0\_0 CH\_STH239-ROOTFS-DXXXXX-XXXXX-R121024-V0302 //更新根文件系统文件

mount -t ubifs ubi0\_0 /mnt或mount -t ubifs ubi0:rootfs /mnt //mount根文件系统

注意，UBI引入了Volume的概念，一个分区(UBI0)可以分为多个Volume(可以用名称和ID访问),也可以一个分区(UBI0,UBI1…)一个Volume.如上面例子，mtd4只有一个分区一个Volume,所以可以通过ubi0\_0访问。

如果在ubiattach时指定为ubi1,分为两个Volume(rootfs app),那么访问volume app就需要使用ubi1\_1,同理，rootfs为ubi1\_0.

**UBI镜像的制作**

/opt/STM/STLinux-2.4/host/bin/mkfs.ubifs -r /home/zhangqiang/target24-new -o /home/zhangqiang/ CH\_STH239-ROOTFS-DXXXXX-XXXXX-R121024-V0302 -m 2048 -e 126976 -c 154

其中

-m 为NAND页大小 UBI: smallest flash I/O unit: 2048

-e UBI: logical eraseblock size: 126976 bytes

(ubiattach时会打印这些信息)

注意这里-c需要指定为volume的最大值。不然即使分区再大，这里指定的大小仅仅大于生成的文件大小的话，会导致分区使用量为100%，从而Volume的其他空余部分将不能使用。

#df -m

Filesystem 1M-blocks Used Available Use% Mounted on

**ubi0:rootfs 15 12 3 81% /**

**ubi0:rootfs 15 12 3 81% /dev/.static/dev**

udev 49 0 49 0% /dev

shmfs 49 0 49 0% /dev/shm

ubi10:app 42 7 36 16% /app

ubi11:flash\_dat 15 0 15 0% /ch\_flash\_data

tmpfs 49 0 49 0% /tmp

生成的文件可以在kernel中使用ubiupdatevol /dev/ubi0\_0 CH\_STH239-ROOTFS-DXXXXX-XXXXX-R121024-V0302进行更新。

**Bootargs的设置**

set bootargs console=ttyAS0,115200 $mtdinfo $meminfo **ubi.mtd=4 root=ubi0:rootfs rootfstype=ubifs** bigphysarea=3000