如何正确的在 PC 上挂载 ubi 镜像文件及反向重新制作镜像

注:文中的 PC 环境为 ubuntu 13.04 32 位

其中经历的各种曲折就不表述了，下面直接说正确的方法步骤。

一，工具准备

从 http://elinux.org/UBIFS#Mounting_UBI_Image_on_PC_using_nandsim 这个网站得知，要挂载 ubi 镜像，需要使用到 nandsim，ubidetach，ubiattach 等工具。nandsim 系统自带，ubidetach 和 ubiattach 需要另外安装一个叫 mtd-utils 的工具集。mtd-utils 工具集里还有制作镜像所需的 mkfs.ubifs 和 ubinize。

mtd-utils 的安装：
方法 1：
$ sudo apt-get install mtd-utils
如果提示在库里找不到 mtd-utils，请参照
http://packages.ubuntu.com/zh-cn/lucid-updates/i386/mtd-utils/download，将 mtd-utils 的最近的源加入到 sources.list 中，然后再安装。比如：
deb http://ubuntu.stu.edu.tw/ubuntu precise main universe

方法 2：
通过源码安装。(我用的是方法 1，方法 2 没有验证)
$ git clone git://git.infradead.org/mtd-utils.git
$ cd mtd-utils/
$ make

二，挂载分析步骤

首先要用 nandsim 模拟出一个 mtd 设备。而且这个 mtd 设备要与 ubi 镜像的参数保存一致，否则后面的步骤会失败。这些参数包括 mtd 设备的物理块擦除大小 (Physical Erase Block， PEB) 和 页大小 (Page Size)。网上的资料对于这一点完全没有提及。

这两个参数可以从 ubi 镜像中分析出来。

ubi 镜像有多个 PEB 组成，每个 PEB 包括以下三部分内容
[UBI_EC_HDR，UBI_VID_HDR，DATA (LEB)]

$ xxd userdata.img | less
0000000: 5542 4923 0100 0000 0000 0000 0000 0000 UBI#.....
0000010: 0000 1000 0000 2000 2240 9f1f 0000 0000 ..... ."@.....
0000020: 0000 0000 0000 0000 0000 0000 0000 0000 ....
0000030: 0000 0000 0000 0000 0000 0000 2956 d5be ....)V..
0000040: ffff ffff ffff ffff ffff ffff ffff ffff ....

这是 ubi 镜像的头部，从 ubi-header.h 中可以了解到这个头部各个字节的含义。

```
struct ubi_ec_hdr {
    uint32_t magic;    //#define UBI_EC_HDR_MAGIC    0x55424923
    uint8_t    version;
    uint8_t    padding1[3];
    uint64_t ec; /* Warning: the current limit is 31-bit anyway! */
    uint32_t vid_hdr_offset;
    uint32_t data_offset;
    uint8_t    padding2[36];
    uint32_t hdr_crc;
} __attribute__ ((packed));
```

上面的 UBI_EC_HDR 告诉了我们 UBI_VID_HDR 的存储位置 (红色字部分 0000 1000) 是偏移 4KB，DATA 区的存储位置是偏移 8KB (绿色字部分 0000 2000)。

我们来验证一下：

```
$ xxd userdata.img | less
0001000: 5542 4921 0101 0005 7fff efff 0000 0000    UBI!............
0001010: 0000 0000 0000 0000 0000 0000 0000 0000    ................
0001020: 0000 0000 0000 0000 0000 0000 0000 0000    ................
0001030: 0000 0000 0000 0000 0000 0000 b825 64a8    ..............%d.
0001040: ffff ffff ffff ffff ffff ffff ffff ffff    ................

0002000: 0000 02fd 0000 0001 0000 0000 0100 0008    ................
0002010: 7573 6572 6461 7461 0000 0000 0000 0000    userdata........
0002020: 0000 0000 0000 0000 0000 0000 0000 0000    ................
0002030: 0000 0000 0000 0000 0000 0000 0000 0000    ................
```

在 4KB 的偏移处，发现了 UBI_VID_HDR 头 (头部标志是 UBI!)。其各字节定义也可以在 ubi-header.h 里找到，这里就不贴了它的数据结构了。

根据网上获取的知识，UBI_EC_HDR 和 UBI_VID_HDR 要么在每个 PEB 的头部各占一页大小，要么都在第一页。如果是前者，那么可以推算出该镜像的页大小为 4KB，如果是后者，则页大小为 8KB。如今的 nand flash 常见的页大小是 512byte 和 2KB，4KB 的也比较少见，8KB 的据了解还没有。所以排除后者，确定页大小为 4KB。

确定了页大小后，再确定 PEB 的大小。

```
$ xxd userdata.img | grep “5542 4923” | less
0000000: 5542 4923 0100 0000 0000 0000 0000 0000    UBI#............
0040000: 5542 4923 0100 0000 0000 0000 0000 0000    UBI#............
0080000: 5542 4923 0100 0000 0000 0000 0000 0000    UBI#............
00c0000: 5542 4923 0100 0000 0000 0000 0000 0000    UBI#............
0100000: 5542 4923 0100 0000 0000 0000 0000 0000    UBI#............
0140000: 5542 4923 0100 0000 0000 0000 0000 0000    UBI#............
0180000: 5542 4923 0100 0000 0000 0000 0000 0000    UBI#............
```

01c0000: 5542 4923 0100 0000 0000 0000 0000 0000 UBI#.....

由上得知，每个 UBI_EC_HDR 相距 256KB，这就是一个 PEB 的大小。同时可以计算出 LEB (Logical Erase Block) 的大小，是 PEB 减去 UBI_EC_HDR 和 UBI_VID_HDR 的大小，这里算出来后是 256-4-4 = 248 KB。这个值在重新制作镜像时有用。

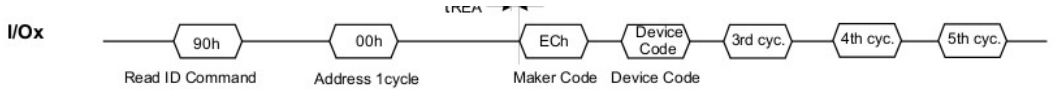
确定了 页大小和 PEB 的大小后，就可以用 nandsim 来模拟出相应的 mtd 设备了。
但是，遍寻 google 也没有 nandsim 详细的使用，没有任何地方说明 nandsim 的四个参数的含义。甚至在 ubi 的官网上也是如此：
http://www.linux-mtd.infradead.org/faq/nand.html#L_nand_nandsim

后来在 mtd-utils 的源码目录中发现了一个名为 load_nandsim.sh 的脚本，这个脚本可以使用 nandsim 模拟出 PEB 大小为 256KB 的 mtd 设备，但页大小只有 512byte 和 2KB 两种选择，没有 4KB。怀疑 nandsim 是不是不能模拟页大小为 4KB 的 mtd。

无奈之下，下载了 nandsim.c 开始看源码，从源码中来看，nandsim 是支持页大小为 4KB 的。

```
if (ns->geom.pgsz == 256) {
    ns->options |= OPT_PAGE256;
}
else if (ns->geom.pgsz == 512) {
    ns->options |= (OPT_PAGE512 | OPT_AUTOINCR);
    if (ns->busw == 8)
        ns->options |= OPT_PAGE512_8BIT;
} else if (ns->geom.pgsz == 2048) {
    ns->options |= OPT_PAGE2048;
} else if (ns->geom.pgsz == 4096) {
    ns->options |= OPT_PAGE4096;
} else {
    NS_ERR("init_nandsim: unknown page size %u\n", ns->geom.pgsz);
    return -EIO;
}
```

用关键词 “4k page size nand flash“ 进行 google，得到一篇页大小为 4KB 的 nand flash 芯片的数据手册。根据网上的描述，nandsim 后面跟的 4 个参数是 nand flash 芯片的 ID。于是重点阅读 Read ID 命令，果然有发现，正是第 4 个参数决定了生成的 mtd 设备的 PEB 和 页大小。



Device	Device Code (2nd Cycle)	3rd Cycle	4th Cycle	5th Cycle
K9F8G08B0M	Same as K9F8G08U0M			
K9F8G08U0M	D3	10h	A6h	64h

4th ID Data

	Description	I/O7	I/O6	I/O5 I/	I/O3	I/O2	I/O1 I/O0
Page Size (w/o redundant area)	1KB 2KB 4KB 8KB						0 0 0 1 1 0 1 1
Block Size (w/o redundant area)	64KB 128KB 256KB 512KB			0 0 0 1 1 0 1 1			
Redundant Area Size (byte/512byte)	8 16					0 1	
Organization	x8 x16		0 1				
Serial Access Minimum	50ns/30ns 25ns Reserved Reserved	0 1 0 1			0 0 1 1		

如上图所示，如果 nandsim 的第 4 个参数为 0xA6，则说明 Page Size 为 4KB，PEB 为 256KB。前三个参数不是很重要，按照手册上默认的即可。

于是使用下面的命令可以模拟出 Page Size 为 4KB，PEB 为 256KB 的 mtd：
$ sudo modprobe nandsim first_id_byte=0xec second_id_byte=0xd3 third_id_byte=0x10 fourth_id_byte=0xa6

$ sudo cat /proc/mtd
dev: size erasesize name
mtd0: 40000000 00040000 "NAND simulator partition 0"

$ mtdinfo /dev/mtd0
mtd0
Name: NAND simulator partition 0
Type: nand
Eraseblock size: 262144 bytes, 256.0 KiB
Amount of eraseblocks: 4096 (1073741824 bytes, 1024.0 MiB)
Minimum input/output unit size: 4096 bytes
Sub-page size: 1024 bytes
OOB size: 128 bytes
Character device major/minor: 90:0
Bad blocks are allowed: true
Device is writable: true

接下来的步骤：
将 ubi 与 /dev/mtd0 关联
$ sudo modprobe ubi mtd=0

格式化前先解绑定
$ sudo ubidetach /dev/ubi_ctrl -m 0

格式化
注意：这里要加上 -O 4096 的选项，显式表明 UBI_VID_HDR 的偏移位置是 4KB，而不是默认值。从上面 mtdinfo /dev/mtd0 的输出结果中，有一项 Sub-page size 的选项，如果不用 －O 显示指定，默认偏移值则是 sub-page size。
$ sudo ubiformat /dev/mtd0 -f userdata.img -O 4096

绑定
注意：仍然要显式加上 -O 4096 的选项
$ sudo ubiattach /dev/ubi_ctrl -m 0 －O 4096
UBI device number 0, total 4096 LEBs (1040187392 bytes, 992.0 MiB), available 0 LEBs (0 bytes), **LEB size 253952 bytes (248.0 KiB)**
输出结果中显示 LEB size 为 248KB，与我们之前计算的一致。

挂载
$ sudo mkdir /mnt/ubi
$ sudo mount -t ubifs ubi0 /mnt/ubi
$ ls /mnt/ubi
app　local.prop

至此，挂载成功！

三，挂载分析总结要点
1，使用 xxd 分析 ubi 镜像文件，得到 Page Size，PEB Size，UBI_VID_HDR 的偏移；
2，阅读相对应的 nand flash 的数据手册，找到 Read ID 命令，确定 nandsim 的 4 个参数的值；
3，在使用 ubiformat 和 ubiattach 时使用 －O 选项显示指定 UBI_VID_HDR 的偏移。

四，反向制作 ubi 镜像
在挂载 ubi 镜像成功后，我们往里面添加一些自己的文件，再重新制作回和原始镜像兼容的镜像。主要使用两个命令 mkfs.ubifs 和 unbnize，包含在 mtd-utils 工具集中。

1，往挂载目录 /mnt/ubi 中添加自己的文件

2，mkfs.ubifs
$ sudo mkfs.ubifs -m 4096 -e 253952 -c 4096 -r /mnt/ubi ubifs.img
-m - Minimum I/O unit size. 即页大小，由前面得知为 4KB。
-e - Logical Erase Block (LEB) size. 由前面计算得为 248KB，即 253952。
-c - Max LEB count. (vol_size/LEB). 通过 mtdinfo /dev/mtd0 输出结果中的 Amount of eraseblocks 可得。
-r - Path.
ubifs.img - Temporary image file.

3，ubinize

首先要准备一个配置文件，内容如下，文件名为 ubi.ini：

```
[ubi_rfs]
mode=ubi
image=ubifs.img
vol_id=0
vol_size=6856704     // ubifs.img 的大小
vol_type=dynamic
vol_name=userdata    //分区卷标名，可以随便取，但最好与原来的镜像保持一致
vol_alignment=1
vol_flags=autoresize
```

这里面最重要的是 vol_size 和 image 选项，其它保持默认即可。

然后使用下面的命令：

$ sudo ubinize -o userdata.ubi -p 262144 -m 4096 -s 1024 -O 4096 ubi.ini

-o - Output file.

-p - Physical Erase Block (PEB) size. 由前面分析得 PEB 为 256KB，即 262144。

-m - Minimum I/O unit size. 即页大小 4KB。

-s - Minimum I/O size for UBI headers, eg. sub-page size. Sub-page size，从 mtdinfo /dev/mtd0 的结果中可以得知。

-O - VID header offset from start of PEB. UBI_VID_HDR 的偏移，由前面分析得为 4KB。

ubi.ini - UBI image configuration file.

至此，镜像反向制作成功，稍候测试能否挂载成功。

五，反向制作 ubi 镜像要点总结

1，仍然是要保证 Page Size，PEB Size，LEB Size，Sub-page Size，UBI_VID_HDR 的偏移这些参数的正确性。

六，挂载反向制作的 ubi 镜像

1，umount

$ sudo umount /mnt/ubi

2，解绑定

$ sudo ubidetach /dev/ubi_ctrl -m 0

3，重新格式化

$ sudo ubiformat /dev/mtd0 -f userdata.ubi -O 4096

4，绑定

$ sudo ubiattach /dev/ubi_ctrl -m 0 -O 4096

5，挂载

$ sudo mount -t ubifs ubi0 /mnt/ubi
$ ls /mnt/ubi
app local.prop self_add_file

验证挂载反向制作的 ubi 镜像成功！

七，删除模拟的 mtd
$ sudo umount /mnt/ubi
$ sudo ubidetach /dev/ubi_ctrl -m 0
$ sudo rmmod nandsim
$ cat /proc/mtd

八，自动脚本
为了方便操作，写了以下自动化脚本，下面会详细讲解如何使用
cal_img_para.sh
load_nandsim.sh
auto_mount_ubi.sh
auto_make_ubi.sh

mount_system.sh
mount_userdata.sh
make_system.sh
make_userdata.sh

cal_img_para.sh
用于从得到的 ubi 镜像文件中解析出关键的参数，比如 UBI_VID_HDR_Offset，PEB Size，LEB Size，vol name。
示例：
$ ./cal_img_para.sh userdata.img
vid_hdr_offset : 0001000 (4 KB)
data_offset : 00002000 (8 KB)
PEB_size : 0040000 (256 KB)
LEB_size : 3E000 (248 KB)
vol_name : userdata
Page size : is not sure, depend by vid_hdr_offset and data_offset, commonly, it equal vid_hdr_offset or data_offset

load_nandsim.sh
是 mtd-utils 里自带的脚本，但自带的脚本不能模拟生成 page size 为 4KB 的 mtd，我做了修改。这个脚本被后面的 auto_mount_ubi.sh 和 auto_make_ubi.sh 使用。

auto_mount_ubi.sh
将 ubi 镜像挂载到 pc 上。其内部原理是先调用 load_nandsim.sh 模拟生成 mtd 设备，再依次调用 ubiformat 和 ubiattach，mount 等命令。需要自己提供很多参数。使用方法如下：

$ sudo ./auto_mount_ubi.sh

Usage: auto_mount_ubi.sh <total mtd size in MiB> <PEB size in KiB> \
<page size in Byte> <UBI_VID_HDR offset in Byte> \
<ubi img> <mount path>

Example:
auto_mount_ubi.sh 1024 256 4096 4096 userdata.img /mnt/ubi

$ sudo ./auto_mount_ubi.sh 1024 256 4096 4096 userdata.img /mnt/ubi

Loaded NAND simulator (1024MiB, 256KiB eraseblock, 4096 bytes NAND page)
ubidetach: error!: cannot detach mtd0
error 19 (No such device)
ubiformat: mtd0 (nand), size 1073741824 bytes (1024.0 MiB), 4096 eraseblocks of 262144 bytes (256.0 KiB), min. I/O size 4096 bytes
libscan: scanning eraseblock 4095 -- 100 % complete
ubiformat: 4096 eraseblocks are supposedly empty
ubiformat: flashing eraseblock 28 -- 100 % complete
ubiformat: formatting eraseblock 4095 -- 100 % complete
UBI device number 0, total 4096 LEBs (1040187392 bytes, 992.0 MiB), available 0 LEBs (0 bytes), LEB size 253952 bytes (248.0 KiB)
mkdir: 无法创建目录"/mnt/ubi": 文件已存在
app local.prop

为了避免每次都输这么多的参数，因此又写了以下两个脚本
mount_system.sh
mount_userdata.sh
里面各自都只有一条调用 auto_mount_ubi.sh 的命令
所以只需执行以下命令就可以自动挂载 system.img 和 userdata.img 了。但是注意，如果你拿到了其它手机的 system.img 和 userdata.img，是要先用 cal_img_para.sh 计算出相应的参数，修改这两个 .sh 中的参数后再执行。
$ ./mount_system.sh
$ ./mount_userdata.sh

auto_make_ubi.sh
挂载 ubi 镜像后，放入自己的文件或修改后，重新制作成与原来镜像参数一致的新镜像。
使用方法：
$ sudo ./auto_make_ubi.sh -h

Usage: auto_make_ubi.sh -m <page size in byte> -e <LEB size> \
-c <Max LEB count> \
-p <PEB size> －s <sub-page size> \
-O <ubi_vid_hdr_offset> -v <vol_name> \
-o <output ubi img> -r <mount path>

Example:
auto_make_ubi.sh -m 4096 -e 248KiB -c 4096 -p 256KiB -s 1024 \

-O 4096 -v userdata -o userdata.ubi -r /mnt/ubi

**$ sudo ./auto_make_ubi.sh -m 4096 -e 248KiB -c 4096 -p 256KiB -s 1024 **
-O 4096 -v userdata -o userdata.ubi -r /mnt/ubi
start working, please wait...
userdata.ubi has created!

为了避免每次输入一长串的参数，写了以下两个脚本
make_system.sh
make_userdata.sh
里面就是直接调用了 auto_make_ubi.sh 脚本
$ ./make_userdata.sh
$ ./make_system.sh