公告

昵称：加菲猫
园龄：4年5个月
粉丝：3
关注：0
+加关注

< 2011 年6月 >

日	一	二	三	四	五	六
29	30	31	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	1	2
3	4	5	6	7	8	9

搜索

找找看

谷歌搜索

常用链接

我的随笔
我的评论
我的参与
最新评论
我的标签

随笔分类

peon-存储(8)
peon-流媒体(17)
peon-软件使用(8)
peon-网络(11)

分析ext2文件系统磁盘分区结构

最近看了些kernel fs code, 从实际例子，简单分析一下ext2文件系统的结构，希望对大家有帮助

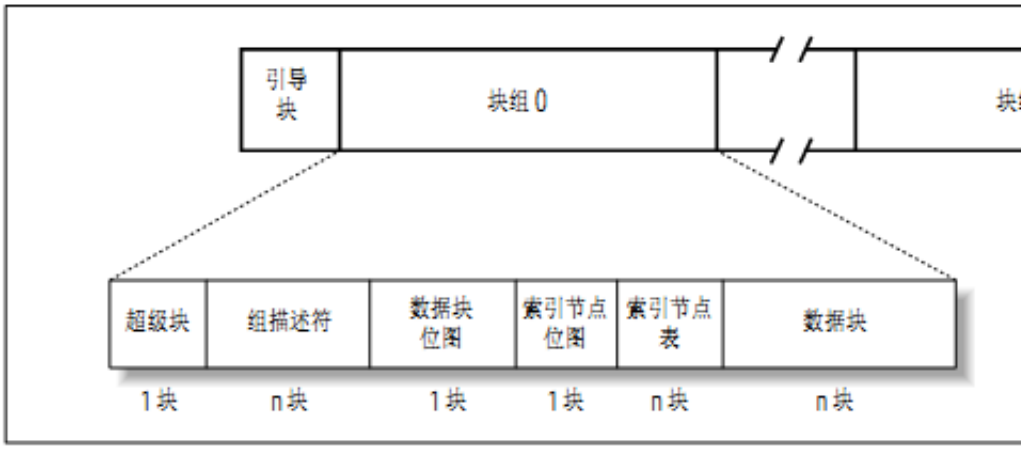

图 17-1：Ext2 分区和 Ext2 块组的分布图

本文涉及到一些结构，主要是：

超级块 struct ext2_super_block { }

组描述　struct ext2_group_desc { }

索引节点　struct ext2_inode { }

目录结构　struct ext2_dir_entry_2 { }

1. 准备工作

为了分析，特地格式化了一个100MB左右的ext2文件系统，block size 1024 Bytes

可以看一下这个分区的主要信息：

```
debugfs:  stats
Filesystem volume name:   <none>
Last mounted on:          <not available>
```

Filesystem UUID: 9c0e702c-f80e-4382-a95d-444fafaab34c
Filesystem magic number: **0xEF53**
Filesystem revision #: 1 (dynamic)
Filesystem features: resize_inode filetype sparse_super
Default mount options: (none)
Filesystem state: not clean
Errors behavior: Continue
Filesystem OS type: Linux
Inode count: **26104**
Block count: **104296**
Reserved block count: 5214
Free blocks: 99442
Free inodes: 26091
First block: 1
Block size: **1024**
Fragment size: 1024
Reserved GDT blocks: 256
Blocks per group: 8192
Fragments per group: 8192
Inodes per group: 2008
Inode blocks per group: 251
Filesystem created: Sun Feb 13 22:36:23 2011
Last mount time: Mon Feb 14 04:49:41 2011
Last write time: Mon Feb 14 04:49:41 2011
Mount count: 2
Maximum mount count: 34
Last checked: Sun Feb 13 22:36:23 2011
Check interval: 15552000 (6 months)
Next check after: Fri Aug 12 22:36:23 2011
Reserved blocks uid: 0 (user root)
Reserved blocks gid: 0 (group root)
First inode: 11
Inode size: 128
Default directory hash: tea
Directory Hash Seed: de62f19c-eaa2-4cb7-a2c6-84fca69baafd
Directories: 2
Group 0: block bitmap at 259, inode bitmap at 260,

peon-网络(11)
peon-自我提高(5)

随笔档案

2011年6月 (4)
2011年3月 (1)
2011年2月 (4)
2011年1月 (1)
2010年12月 (1)
2010年11月 (1)
2010年10月 (2)
2010年3月 (2)
2010年1月 (1)
2009年12月 (5)
2009年8月 (8)
2009年7月 (1)
2009年5月 (1)
2009年4月 (2)
2009年3月 (4)
2009年2月 (2)
2008年12月 (1)
2008年11月 (3)
2008年10月 (4)
2008年7月 (2)
2008年6月 (2)
2008年5月 (1)
2008年3月 (1)
2008年2月 (5)
2008年1月 (9)
2007年12月 (3)

文章分类

我的转贴(3)

相册

个人相册

最新评论

1. Re:Firefox IE 自动代理脚本
IE8下应该是 file://c:/.../ , 即file:后只有2个'/', 不是3个. Firefox下是3个.
--seizethetime

inode table at 261

7665 free blocks, 1995 free inodes, 2 used directories

Group 1: block bitmap at 8451, inode bitmap at 8452, inode table at 8453

7681 free blocks, 2008 free inodes, 0 used directories

Group 2: block bitmap at 16385, inode bitmap at 16386, inode table at 16387

7939 free blocks, 2008 free inodes, 0 used directories

Group 3: block bitmap at 24835, inode bitmap at 24836, inode table at 24837

7681 free blocks, 2008 free inodes, 0 used directories

Group 4: block bitmap at 32769, inode bitmap at 32770, inode table at 32771

7939 free blocks, 2008 free inodes, 0 used directories

Group 5: block bitmap at 41219, inode bitmap at 41220, inode table at 41221

7681 free blocks, 2008 free inodes, 0 used directories

Group 6: block bitmap at 49153, inode bitmap at 49154, inode table at 49155

7939 free blocks, 2008 free inodes, 0 used directories

Group 7: block bitmap at 57603, inode bitmap at 57604, inode table at 57605

7681 free blocks, 2008 free inodes, 0 used directories

Group 8: block bitmap at 65537, inode bitmap at 65538, inode table at 65539

7939 free blocks, 2008 free inodes, 0 used directories

Group 9: block bitmap at 73987, inode bitmap at 73988, inode table at 73989

7681 free blocks, 2008 free inodes, 0 used directories

Group 10: block bitmap at 81921, inode bitmap at 81922, inode table at 81923
        7939 free blocks, 2008 free inodes, 0 used directories
Group 11: block bitmap at 90113, inode bitmap at 90114, inode table at 90115
        7939 free blocks, 2008 free inodes, 0 used directories
Group 12: block bitmap at 98305, inode bitmap at 98306, inode table at 98307
        5738 free blocks, 2008 free inodes, 0 used directories

2. 查看 super block

先看超级块，因为第一个block(1024字节)是引导块，所以我们从1024 字节 开始
inode count 等都可以对得上 (注意是little endian)

```
struct ext2_super_block {
    __u32     s_inodes_count;         /* Inodes count */ //   f8 65 00 00  = 26140
    __u32     s_blocks_count;         /* Blocks count */  //   68 97 01 00 = 104296
......
    __u16 s_magic;  /* Magic signature */  // ef 53
......
```

```
[root@ms3003 ~]# dd if=/dev/hdb1 bs=1 count=1024 skip=1024 | od -t x1 -Ax
1024+0 records in
1024+0 records out
000000 f8 65 00 00 68 97 01 00 5e 14 00 00 72 84 01 00
000010 eb 65 00 00 01 00 00 00 00 00 00 00 00 00 00 00
000020 00 20 00 00 00 20 00 00 d8 07 00 00 e5 43 58 4d
000030 e5 43 58 4d 02 00 22 00 53 ef 00 00 01 00 00 00
000040 67 ec 57 4d 00 4e ed 00 00 00 00 00 01 00 00 00
000050 00 00 00 00 0b 00 00 00 80 00 00 00 10 00 00 00
000060 02 00 00 00 01 00 00 00 9c 0e 70 2c f8 0e 43 82
000070 a9 5d 44 4f af aa b3 4c 00 00 00 00 00 00 00 00
000080 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
......
```

3. 接下来我们看组描述

再看组描述符，32个字节一个，我们可以把debugfs的结果和实际的磁盘信息对照一下

```
/*
* Structure of a blocks group descriptor
*/
```

```
struct ext2_group_desc
{
    __u32    bg_block_bitmap;        /* Blocks bitmap block */
    __u32    bg_inode_bitmap;        /* Inodes bitmap block */
    __u32    bg_inode_table;        /* Inodes table block */
    __u16    bg_free_blocks_count;    /* Free blocks count */
    __u16    bg_free_inodes_count;    /* Free inodes count */
    __u16    bg_used_dirs_count;    /* Directories count */
    __u16    bg_pad;
    __u32    bg_reserved[3];
};
```

```
Group  0: block bitmap at 259, inode bitmap at 260, inode table at 261
        7665 free blocks, 1995 free inodes, 2 used directories
Group  1: block bitmap at 8451, inode bitmap at 8452, inode table at 8453
        7681 free blocks, 2008 free inodes, 0 used directories
......
```

观察磁盘上的信息，都可以对得上

```
03 01 00 00 = 259
04 01 00 00 = 260
F1 1D       = 7665
cb 07       = 1995
02 00       = 2
03 21 00 00 = 8451
04 21 00 00 = 8452
```

组描述分布在 super block 后面 (所以从2048开始) ，根据block group数量而有多个，32个字节一组(2行一组)，

磁盘100MB，每个block gourp 物理8192KB, 所以有有13个block group, 32*13=0x1A0 , 所以1A0 后面就都是0了

```
[root@ms3003 ~]# dd if=/dev/hdb1 bs=1 count=1024 skip=2048 | od -t x1
-Ax
1024+0 records in
1024+0 records out
000000 03 01 00 00 04 01 00 00 05 01 00 00 f1 1d cb 07
000010 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000020 03 21 00 00 04 21 00 00 05 21 00 00 01 1e d8 07
000030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000040 01 40 00 00 02 40 00 00 03 40 00 00 03 1f d8 07
000050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000060 03 61 00 00 04 61 00 00 05 61 00 00 01 1e d8 07
000070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000080 01 80 00 00 02 80 00 00 03 80 00 00 03 1f d8 07
000090 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000a0 03 a1 00 00 04 a1 00 00 05 a1 00 00 01 1e d8 07
0000b0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000c0 01 c0 00 00 02 c0 00 00 03 c0 00 00 03 1f d8 07
0000d0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000e0 03 e1 00 00 04 e1 00 00 05 e1 00 00 01 1e d8 07
0000f0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000100 01 00 01 00 02 00 01 00 03 00 01 00 03 1f d8 07
000110 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000120 03 21 01 00 04 21 01 00 05 21 01 00 01 1e d8 07
000130 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

000140 01 40 01 00 02 40 01 00 03 40 01 00 03 1f d8 07
000150 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000160 01 60 01 00 02 60 01 00 03 60 01 00 03 1f d8 07
000170 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000180 01 80 01 00 02 80 01 00 03 80 01 00 6a 16 d8 07
000190 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
*
000400

4. 查看根目录的inode

ext2 根目录的inode number = 2， 所以这个inode 在 group 0 ， 是第二个inode
Group 0: block bitmap at 259, inode bitmap at 260, inode table at 261
每个 inode 是一个 struct ext2_inode， 128 个字节，所以磁盘上的偏移就是 261*1024 + 128 = 267392

[root@ms3003 ext2]# dd if=/dev/hdb1 bs=1 count=256 skip=267392 | od -t x1 -Ax
256+0 records in
256+0 records out
000000 ed 41 00 00 00 04 00 00 52 13 5b 4d e1 82 58 4d
000010 e1 82 58 4d 00 00 00 00 00 00 03 00 02 00 00 00
000020 00 00 00 00 00 00 00 00 **00 02** 00 00 00 00 00 00
000030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

i_mode = 0x41ed = 0040755 (8进制)， 也就是 目录 + 755 权限(drwxr-xr-x)

block 索引第一个是 00 02 ，就是512块

对照 *struct ext2_inode ，*

```
{
__u16 i_mode;  /* File mode */
__u16 i_uid;  /* Low 16 bits of Owner Uid */
__u32 i_size;  /* Size in bytes */
.....
__u32 i_block[EXT2_N_BLOCKS];/* Pointers to blocks */ 从40个字节处开始
.....
}
```

5. 查看根目录的目录结构

我们看一下根目录有哪些文件(目录是特殊的文件）， 和磁盘分区信息对照一下
注意最左边的数字是inode number

```
[root@ms3003 ext2]# ll -ia
total 23
     2 drwxr-xr-x  3 root root  1024 Feb 14 09:18 .
812001 drwxr-xr-x  4 root root  4096 Feb 13 22:36 ..
    11 drwx------  2 root root 12288 Feb 13 22:36
lost+found
    15 -rw-r--r--  1 root root    72 Feb 14 09:18 test
    12 -rw-r--r--  1 root root    69 Feb 13 22:45 test2
[root@ms3003 ext2]#  dd if=/dev/hdb1 bs=1024 count=1
skip=512 | hexdump -C
1+0 records in
1+0 records out
00000000  02 00 00 00 0c 00 01 02  2e 00 00 00 02 00 00
00  |................|
00000010  0c 00 02 02 2e 2e 00 00  0b 00 00 00 14 00 0a
02  |................|
00000020  6c 6f 73 74 2b 66 6f 75  6e 64 00 00 0c 00 00
00  |lost+found......|
00000030  10 00 05 01 74 65 73 74  32 2e 73 77 0f 00 00
00  |....test2.sw....|
00000040  c4 03 04 01 74 65 73 74  74 65 73 74 00 00
00 00  |....testtest....|
00000050  b4 03 09 01 2e 74 65 73  74 2e 73 77 70 70 00
00  |.....test.swpp..|
00000060  00 00 00 00 a0 03 05 01  74 65 73 74 7e 2e 73
77  |........test~.sw|
00000070  70 78 78 00 00 00 00 00  00 00 00 00 00 00 00
00  |pxx.............|
00000080  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00
00  |................|
```

对照 ext2_dir_entry_2 分析，注意 **这是个可变长度的结构**，长度为rec_len , 而名字长度为 name_len

```
struct ext2_dir_entry_2 {
    __u32    inode;            /* Inode number */
    __u16    rec_len;         /* Directory entry length */
    __u8     name_len;         /* Name length */
    __u8     file_type;
    char     name[EXT2_NAME_LEN];    /* File name */
};
```

蓝色和红色分别是目录 . 和 .. inode id = 2

绿色是 lost+found , inode id = 11 (0b 00 00 00)

紫色是 test2 , inode id = 12 (0c 00 00 00)

深黄色 是 test , inode id = 15 (0f 00 00 00)

该块信息和 ext2 根目录文件都可以对应上

6.查看普通文件 test 的内容

test 文件的 inode 是 15，那么我们计算一下它的inode的位置，

是group 0 的 第15个inode, 磁盘偏移是

261*1024 + 128*14 = 269056

```
[root@ms3003 ~]# dd if=/dev/hdb1 bs=1 count=256
skip=269056 | od -t x1 -Ax
256+0 records in
256+0 records out
000000 a4 81 00 00 48 00 00 00 df 0d 5b 4d e1 82 58 4d
000010 e1 82 58 4d 00 00 00 00 00 00 01 00 02 00 00 00
000020 00 00 00 00 00 00 00 00 09 14 00 00 00 00 00 00
000030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

我们找到这个 inode , imode = 0x81a4 = 00100644 (8进制)，

也就是普通文件+权限644(-rw-r--r--)

第一个block号码是 0x 14 09 = 5192

那我们查看一下 5192 block 的内容， 和 cat test 的结果是一致的

```
[root@ms3003 ext2]# cat test
apeonaaaaaaaaaaaaaaaaaab
bbbbbbbbbbbbbbbbbbbb
ccccccccccccccccccc:q!
\
[root@ms3003 ~]# dd if=/dev/hdb1 bs=1024 count=1
skip=5129 | od -t x1 -aAx
1+0 records in
1+0 records out
000000 61 70 65 6f 6e 61 61 61 61 61 61 61 61 61 61 61
         a  p  e  o  n  a  a  a  a  a  a  a  a  a  a  a
000010 61 61 61 61 61 61 61 61 62 0a 62 62 62 62 62 62
         a  a  a  a  a  a  a  a  b nl  b  b  b  b  b  b
000020 62 62 62 62 62 62 62 62 62 62 62 62 62 62 0a 63
         b  b  b  b  b  b  b  b  b  b  b  b  b  b nl  c
000030 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63
         c  c  c  c  c  c  c  c  c  c  c  c  c  c  c  c
000040 63 63 3a 71 21 0a 5c 0a 00 00 00 00 00 00 00 00
         c  c  :  q  !  nl  \  nl nul nul nul nul nul nul nul nul
000050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

nul nul nul nul nul nul nul nul nul nul nul nul nul nul nul nul

本文用菊子曰发布

下面是我的其他博客：
博客园，写一些工作和学习的笔记： http://www.cnblogs.com/peon/
博客堂，开发方面的一些文章：http://blog.joycode.com/peon/
流媒体博客，流媒体方面的一些文章：http://blog.lmtw.com/b/peon/

加菲猫
关注 - 0
粉丝 - 3
+加关注

0 0

(请您对文章做出评价)

« 博主前一篇：Scaling Redis

posted on 2011-06-22 00:04 加菲猫 阅读(135) 评论(0) 编辑 收藏

最新知识库文章：

· 详图实证：再谈JavaScript的语源问题
· 还原JavaScript的真实历史
· 编程给你现实生活带来了哪些坏习惯
· 多疑到刚刚好：防御性编程
· 从经理的角度看技术债务
» 更多知识库文章...