JFS 文件系统分析与实践

卢晋平　（中石化胜利油田有限公司物探研究院计算机室）

摘要：本文在 JFS 文件系统结构分析的基础上，重点对 LVCB 和 superblock 进行了论述，给出了修复 JFS 文件系统故障的实例。

关键词：JFS LVCB superblock magic 号

1　引言

文件系统是操作系统的重要组成部分，也是一个强有力的文件组织工具。文件系统故障时有发生，故障出现后，快速、准确恢复正常，是系统管理员的职责。要达到此目标，对文件系统的深入了解是必不可少的，经验也很重要。

2　JFS 结构分析

JFS 是 Journal File System 的简称，即日志文件系统。JFS 产生在逻辑卷的顶端，了解 JFS 的结构，有助于高效地信息检索。

一个日志文件系统是文件和目录的层结构，即文件树。这种结构类型与倒置的树即根在顶端、枝叶在低端相类似，枝代表目录，叶代表文件。这种文件树使用目录组织数据而且设计成组，允许一次管理多个目录和文件，在一个 JFS 中你能保存各种数据。

目录在内核里是基本数据，而在文件系统里，和基本文件所处的地位完全一样。目录可以像普通文件一样被读出，但是不能像普通文件那样生成和写入。目录文件的格式是由二进制数和正文数据组合而成的，一个目录项由 16 字节组成，最后的 14 字节保存文件名。当文件名不足 14 个字符时用 ASCII 码 NULL（全零值）填满，而开始两个字节表示文件管理信息的地址。每个目录文件均以‘.’或‘..’开始的，‘.’是进入本目录的目录入口，‘..’代表当前目录的双亲目录，也就是靠近此目录的上一层目录。

```
# od -cb /home
0000000  \0 002  .  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0
         000 002 056 000 000 000 000 000 000 000 000 000 000 000 000 000
0000020  \0 002  .  .  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0
         000 002 056 056 000 000 000 000 000 000 000 000 000 000 000 000
0000040  \0 020  l  o  s  t  +  f  o  u  n  d  \0  \0  \0  \0
         000 020 154 157 163 164 053 146 157 165 156 144 000 000 000 000
0000060  \b  \0  g  u  e  s  t  \0  \0  \0  \0  \0  \0  \0  \0  \0
         010 000 147 165 145 163 164 000 000 000 000 000 000 000 000 000
0000100  \b 001  n  e  t  i  n  s  t  \0  \0  \0  \0  \0  \0  \0  \0
         010 001 156 145 164 151 156 163 164 000 000 000 000 000 000 000
0000120  \0 021  l  o  a  d  l  \0  \0  \0  \0  \0  \0  \0  \0  \0
         000 021 154 157 141 144 154 000 000 000 000 000 000 000 000 000
0000140  \0  =  s  p  o  p  e  r  \0  \0  \0  \0  \0  \0  \0  \0
         000 075 163 160 157 160 145 162 000 000 000 000 000 000 000 000
```

文件是一个字节序列。对 JFS 文件系统来说，文件没有记录、类型的概念，文件系统对文件的管理并不涉及文件内部的数据结构，文件内部的数据结构视具体程序而定。正文文件以行为单位记录信息，每行以换行符为结束标志。JFS 文件系统中没有记录、记录总数、字节总数等信息。而是根据文件是否还有符号来表示文件是否结束。系统内核的文件系统记录每个文件长度，根据文件长度，系统可以判断文件的结束与否。确定文件类型，不能只根据文件名做出判断（虽然文件名可以有类型区别），命名的方法仅仅是一个惯例，并不能完全代表文件类型。有时，文件的标签是很明确的，可执行程序可以通过在文件开始的“二进制幻数”（magic number）来识别。在 od 命令不加选择项时，它以 16 位，即两个字节为单位打印文件八进制表示，便可看到文件的幻数。

```
# od -N 64 /bin/ed
0000000 000737 000004 033402 132274 000000 000000 000000 000000
0000020 000110 010007 000413 000001 000000 073454 000000 015650
0000040 000000 043135 000000 014764 010000 000400 000000 000000
0000060 000000 015150 000002 000001 000002 000002 000004 000003
# ls -l /bin/ed
-r-xr-xr-x  2 bin    bin      41390 Jul 21 1999  /bin/ed
```

八进制数 737 表示文件为纯粹的可执行代码程序。幻数 737 代表非 ASCII 码文件，所以不能由编辑程序产生。

一般文件有几个组成部分：名字、内容和管理信息，管理信息是指访问权限和修改时间等。管理信息存放在 i-node 中，文件系统基本数据，如文件长度，在磁盘中的存放地址等都存放在 i-node 中。在 i-node 里还存放三个时间信息：文件内容最后改变的时间（写）；文件最后使用的时

·间(读或执行);i－node 最后改变的时间,既由设置访问权限等原因而引起 i－node 改变的时间}。在某种意义上 i 节点就是文件。目录的层次结构是由通常的文件名构成的,而系统内部的文件名就是它的 i 节点数,从它可以找出文件的各种信息。i 节点数存放在目录项的开始二个字节里,随后是文件名。利用 od －d 命令可以显示 i 节点数据,－d 选择项表示以十进制数显示双字节的内容,而不是按八进制数显示。

```
# od －d /home
0000000 00002 11776 00000 00000 00000 00000 00000 00000
0000020 00002 11822 00000 00000 00000 00000 00000 00000
0000040 00016 27759 29556 11110 28533 28260 00000 00000
0000060 02048 26485 25971 29696 00000 00000 00000 00000
0000100 02049 28261 29801 28275 29696 00000 00000 00000
0000120 00017 27759 24932 27648 00000 00000 00000 00000
0000140 00061 29552 28528 25970 00000 00000 00000 00000
```

每个目录项开始两个字节表示文件名和内容的链接指针。目录里的一个文件名对应一个链接,它将目录结构里的名字链接到 i 节点,最后链接到文件数据。同一 i 节点数可能出现在多个目录中。rm 命令实际上并不删除 i 节点,它只删除目录项及其拉链。只当文件的所有链接都删除时系统才删除 i 节点,并最后删除文件本身。如果目录项入口处的 i 节点数是零,表示该链接已删除,但并非文件内容也一起被删掉,文件在其他地方仍可能还有链接。

JFS 文件系统空间分为 chuck(大块)称作分配组,每个分配组包含 i－nodes 和数据块。允许 i－nodes 和数据块被分散到文件系统中,而且允许文件数据分配在离它的 i－node 很近。当文件系统增大时,分配组数也增多,当分配组增加时,文件系统包含更多的 i－nodes 和数据块。第一个分配组开始于文件系统的开端,而且包含一个保留区,被一个 4096 * 2 字节的组所占用,这个区的第 0 个 4096 字节(0x0000)用来保存 LVCB(logical volume control block),第 1 个 4096 字节(0x1000)占据文件系统的主 superblock;第 31 个 4096 字节(0x1f000)保存次 superblock。第二个分配组始于 32 块,而且占用连续足够的块保存分配组的 inodes。如图 1 所示。

2.1 LVCB

下面以某一个物理卷如 /dev/hd1 上的 JFS 文件系统 /home 为例,用 od 工具对 LVCB 分别以 16 进制和 ASCII 码显示如下:

(其中－x 表示以十六进制格式显示输出,－c 表示以 ASCII 码格式显示输出,－N 256 表示输出 256 字节信息)

```
# od －x －N 256 /dev/hd1 0x0000
0000000 4149 5820 4c56 4342 0000 6a66 7300 0000
0000010 0000 0000 0000 0000 0000 0000 0000 0000
0000020 0000 0000 0000 0000 0000 3030 3032 3535
0000030 3939 3532 3366 3038 3234 2e38 0000 0000
0000040 0000 0068 6431 0000 0000 0000 0000 0000
0000050 0000 0000 0000 0000 0000 0000 0000 0000
*
0000080 0000 0057 6564 204a 616e 2031 3320 3036
0000090 3a31 323a 3532 2031 3939 390a 0000 0000
00000a0 0054 6875 2053 6570 2031 3620 3039 3a30
00000b0 303a 3238 2031 3939 390a 0000 0000 0032
00000c0 3535 3939 3537 3030 0079 6d6d 0079 0020
00000d0 0001 0001 2f68 6f6d 6500 0000 0000 0000
00000e0 0000 0000 0000 0000 0000 0000 0000 0000
*
000150 0000 0000 6c6f 673d 2f64 6576 2f68 6438
0000160 0000 0000 0000 0000 0000 0000 0000 0000
*
0000400 a5a5 a5a5 a5a5 a5a5 a5a5 a5a5 a5a5 a5a5
# od －c －N 256 /dev/hd1 0x0000
0000000   A   I   X       L   V   C   B  \0  \0   j   f   s  \0  \0  \0
0000010  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0
0000020  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0   0   0   0   2   5   5
0000030   9   9   5   2   3   f   0   8   2   4   .   8  \0  \0  \0  \0
0000040  \0  \0  \0   h   d   1  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0
0000050  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0
*
0000080  \0  \0  \0   W   e   d       J   a   n       1   3       0   6
0000090   :   1   2   :   5   2       1   9   9   9  \n  \0  \0  \0  \0
00000a0  \0   T   h   u       S   e   p       1   6       0   9   :   0
00000b0   0   :   2   8       1   9   9   9  \n  \0  \0  \0  \0  \0   2
00000c0   5   5   9   9   5   7   0   0  \0   y   m   m  \0   y   V  \0
00000d0  \0 001  \0 001   /   h   o   m   e  \0  \0  \0  \0  \0  \0  \0
00000e0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0
*
0000150  \0  \0  \0  \0   l   o   g   =   /   d   e   v   /   h   d   8
0000160  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0
*
0000400   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *
```

从以上显示,可知道,LVCB 开始于零地址,最前面存放的内容为:AIX、LVCB、jfs 和关于块特殊文件的一些信息。

2.2 superblock

下面以某一个物理卷如 /dev/hd1 上的 JFS 文件系统 /home 为例,用 od 工具对 superblock 以 16 进制码显示如下:

```
# od －x －N 64 /dev/hd1 0x0001000
0001000 6587 2143 0000 0000 0000 4000 0000 00e1
0001010 0000 8000 1000 0000 2f68 6f6d 6500 2f68
0001020 6f6d 6500 000a 0003 0100 0000 369c 397b
0001030 0000 0001 0000 0200 0000 0800 0000 0000
```

下面是块特殊文件 /dev/hd1 的文件系统的 dump 信息:

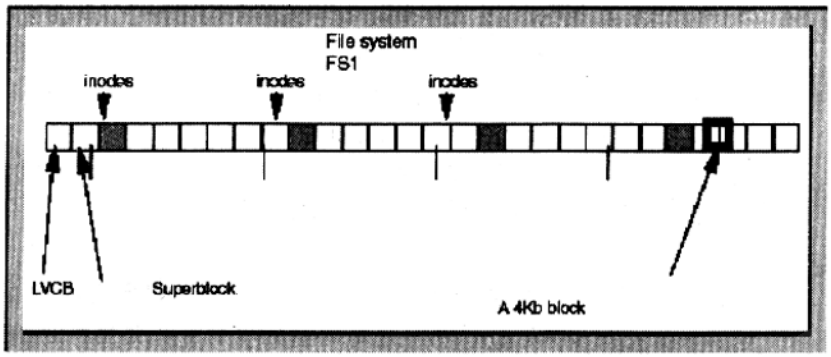

图 1

```
# dumpfs /dev/hd1
/dev/hd1:
```

magic	0x65872143	cpu type	0x0
file system type	0	file system version	1
file system size	32768	fragment size	512
inodes/allocation grp	2048	compress	0
file system name	/home	volume name	/home
log device	0xa0003	log serial number	0xe1
file system state	1	read only	0
last change	Wed Jan 13 14:13:15 TAIST 1999		

从以上显示，可看出，superblock 开始于十六进制 0001000 地址，各字节的含义为：

0x65872143	magic 号
0x00000000	cpu 和文件系统的类型
0x00004000	分配组大小 16384(十进制)
0x000000e1	当文件系统 mount 上时，log 的序列号即 0xe1
0x00008000	文件系统的大小即 32768
0x1000	块大小即 4096
0x0000	暂时没用
0x2f686f6d6500	文件系统名(六位)/home
0x2f686f6d6500	卷名(六位)/home
0x000a0003	log 的设备地址
0x01	文件系统 mount 时设置
0x00	文件系统是只读
0x0000	非压缩文件系统
0x369c397b	文件系统创建的时间
0x00000001	版本号

3 实例

一般的文件系统故障用 fsck 命令可修复，但如出现如下故障，用 fsck 就无法修复了。

3.1 修复 superblock 受损的 magic 号

当一个文件系统的 superblock 受损坏时，文件系统就无法访问，大多数对 superblock 的损坏无法修复，下面将描述在一个 JFS 文件系统中，当故障出现的原因是 magic 号受损时，怎样修复主 superblock。如果在一个 JFS2 文件系统中，主 superblock 受损，则使用 fsck 命令就能自动拷贝次 superblock 对主 superblock 进行修复。

假定 /home 是建立在物理卷 /dev/hd1 上的一个 JFS 文件系统。

3.1.1 对受损的文件系统 /home 进行卸载，使用如下命令

```
# umount /home
```

3.1.2 对文件系统运行 fsck 命令

```
# fsck - p /dev/hd1
```

如果 superblock 受损，fsck 命令返回以下信息之一：

fsck:Not an AIXV5 file system

或

Not a recognized filesystem type

3.1.3 使用 root 身份，使用 od 命令显示文件系统的 superblock

```
# od - x - N 64 /dev/hd1 0x1000
```

输出信息为：

```
0001000 1234 0234 0000 0000 0000 4000 0000 00e1
0001010 0000 8000 1000 0000 2f68 6f6d 6500 2f68
0001020 6f6d 6500 000a 0003 0100 0000 369c 397b
0001030 0000 0001 0000 0200 0000 0800 0000 0000
```

从上面的输出，可注意到：在 0x1000 地址处受损的 magic 值为：(12340234)。如前所述如果文件系统产生时都使用默认值，则 magic 号应该为：0x43218765；如果有默认值被改动，则 magic 号应该为：0x65872143。

3.1.4 使用 od 命令显示文件系统的次 superblock

```
# od - x - N 64 /dev/hd1 0x1f000
0001f000 6587 2143 0000 0000 0000 4000 0000 00e1
0001f010 0000 8000 1000 0000 2f68 6f6d 6500 2f68
0001f020 6f6d 6500 000a 0003 0100 0000 369c 397b
0001f030 0000 0001 0000 0200 0000 0800 0000 0000
```

从上面的输出，可看出在 0x1f000 地址处有正确的 magic 号。

3.1.5 使用 dd 命令把次 superblock 拷贝到主 superblock

```
# dd count=1 bs=4k skip=31 seek=1 if= /dev/hd1 of= /dev/hd1
```

命令执行完后显示：

dd:1+0 records in.

dd:1+0 records out.

3.1.6 运行 fsck 命令确保一致性

fsck /dev/hd1

3.2 文件太大导致节点死机

节点 ssp01n04 死机，三字显示为：888—102—700—0c4

错误报告如下：

```
# errpt |more
C0AA5338 0609075704 U S SYSDUMP SYSTEM DUMP
684A365B 0609064804 U U SYSPFS FILE SYSTEM CORRUPTION
# errpt -aj 684A365B|more
---------------------------------------
.....
MAJOR/MINOR DEVICE NUMBER
0027 000D
ADDITIONAL INFORMATION
4A46 5345 40C6 4248 0000 002D 0000 003A 0000 1211 0000 0000 0000 0000
0000 0006
14D4 7530 0000 2003 0022 4B89 00FF 8400 0003 036C 0000 0045 0000 0000
0000 0000
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0000 0000
```

分析以上报告之后，进行了以下处理：

3.2.1 根据 MAJOR/MINOR DEVICE NUMBER

0027 000D

这两行信息，执行如下命令：

Root ssp01n04：/ # ls -l /dev

brw-rw---- 1 root system 39, 13 Dec 02 2003 ssp01n04lv30

判定错误出在 /dev/ssp01n04lv30 卷上。

3.2.2 卸载文件系统

Root ssp01n04：/ # umount /UDD/ssp01n04d30

3.2.3 运行 fsck

Root ssp01n04：/ # fsck /dev/ssp01n04lv30

没发现异常也没做任何修复。

3.2.4 根据 ADDITIONAL INFORMATION

4A46 5345 40C6 4248 0000 002D 0000 003A 0000 1211 0000 0000 0000 0000 0000 0006

14D4 7530 0000 2003 0022 4B89 00FF 8400 0003 036C 0000 0045 0000 0000 0000 0000

中的 FF8400000 判定此文件系统中有一个文件达到了 JFS 文件系统允许的文件大小最大值。虽然大多数参考资料中说明 JFS 文件系统中最大的文件大小为 64G，但实际上 JFS 文件系统允许的最大文件大小为：十六进制 0xFF8400000，即十进制 68589453312。

执行以下命令：

Root ssp01n04：/UDD/ssp01n04d30 # ls - laR

结果中确实有一行的显示结果为：

-rw-r--- 1 eslly ershi 68589453312 Jun 06 20：16 tnb_8_1126-2301.sgy

判定有文件达到了最大值。

3.2.5 删除该文件后，系统正常

以上两个实例揭示出 JFS 文件系统的缺点和局限性，目前，JFS2 文件系统具有它的优越性。

4 JFS 文件系统和 JFS2 文件系统的对比

JFS2 是 Journal File System 2 的简称，是建立在 AIX Version 5 版本以上。JFS 只能在 POWER 系统上使用，而 JFS2 既可支持 POWER 系统，又支持 Itanium-based 系统。JFS2 文件系统的文件布局采用 B-树数据结构，B-树是一种平衡的多路查找树，提高了读写盘区的性能。JFS2 具有许多新的功能。下面是 JFS 和 JFS2 的功能对照表：

JFS 和 JFS2 的功能对照表

Function	JFS2	JFS
Fragments/Block Size	512-4096 Block Sizes	512-4096 Fragments
Architectural Maximum File	1 PB	64 GB
Architectural Maximum File System Size	4 PB	1 TB
Maximum File Size Tested	1 TB	64 GB
Maximum File System Size	1 TB	1 TB
Number of Inodes	Dynamic, limited by disk space	Fixed, set at file system creation
Directory Organization	B-tree	Linear
Online Defragmentation	Yes	Yes
Compression	No	Yes
Default Ownership at Creation	root.system	sys.sys
SGID of Default File Mode	SGID=off	SGID=on
Quotas	No	Yes
PB stands for PetaBytes, which is equal to 1,048,576 GigaBytes. TB stands for TeraBytes, which is equal to 1,024 GigaBytes.		

5 结束语

JFS 是可靠的、高性能的文件系统。JFS2 文件系统的文件布局采用 B-树数据结构，更加提高了读写盘区的性能。希望能和同行们交流，使文件系统更加可靠、完善。

参考资料

1 AIX 5L Differences Guide Version 5.1 June 2001 SG24-5765-01.

2 AIX Logical Volume Manager, from A to Z: Introduction and Concepts January 2000 SG24-5432-00.