从《XENIX 下误删文件的恢复方法》谈起

何崇杰 (河南省统计局)

《计算机系统应用》九五年第二期刊登的《XENIX 下误删文件的恢复方法》(以下简记作)《恢复》一文,介绍了一种在 XENIX 下手工恢复被误删文件的方法。从文章中可看出,作者对 XENIX 系统下普通文件的建/删原理尚欠了解,因而其介绍的方法是欠妥甚至有误。

为了便于越来越多的 XENIX/UNIX 用户了解和掌握有关原理,下边就对《恢复》一文中欠妥和有误的地方作一分析和说明。

1。"……但只要没人使用系统建立文件,删除的文件是可以恢复的。"

XENIX 系统的用户应该知道,XENIX 系统是一个多用户多任务的操作系统。它不仅支持多个用户同时上机,还可以(对用户来讲)同时执行不同的任务,这其中就包括系统本身的一些任务。因此,"在发生使用 rm 命令而不慎将文件删除的情况"时,即使系统中没有任何用户"使用系统建立新文件",也不能保证因删除文件而被释放的数据块不被系统重新分配使用。

原因一:XENIX 系统中有许多日志(LOG)文件,是由系统自动建立和维护的。例如增长较快的/etc/wtmp,只要有人注册进系统,wtmp 中就会有记录。换句话说,该文件就会增大(144 字节),就可能会申请新的数据块,就可能破坏"没人使用系统建立新文件"这一条件。

原因二:系统在删除文件时就已经把该文件释放的某一数据块(对小于 99K 的文件)或某几个数据块(对大于 99K 的文件)用掉了(原因后详)。

对于原因二,因系统设计就是如此,无法避免;对于原因一,可将根文件系统和用户文件系统分离。根文件系统下只安装系统文件和系统支持的各种软件;用户文件系统下安装应用程序和存放数据文件。可能的话,让不同类型的用户使用不同的文件系统。这样一旦用户报告有误删除文件的情况发生时,管理员(超级用户)就可将含有该文件的文件系统卸(umount)下来,以保证刚删除文件释放的数据块不被重新分配使用。

2."在 2 号专用块上找刚压栈和磁盘块……刚压入的块在索引表的最底部。"

XENIX 系统在构造文件系统时,将文件存储区中的数据块按块号从大到小每 100 块一组为单位链接起来。第一组(也是系统中最后用到的一组)只有 99 块,在链头块的位置上填上 0,表示其后再没有数据块可用。其它各组的 100 块中都有一块称作链头块,它记录了下 100 个空闲数据块的块号。最后≤100 个数据块的块数和块号则放在超级块中的空闲块表中。示意图如图 1:

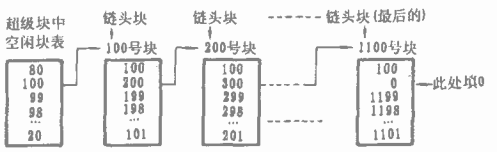

图 1 空闲块链表

图中链头块中,第 1～2 字节的值恒为 100,表示本组有 100 个空闲块;第 3～6 字节的值表示下一个链头块的块号;随后的 4×99=396 个字节分别表示其余 99 个空闲块的块号。随着文件的建、删,链头块中的块号就不会这么有序了。

XENIX 系统下的/usr/include/sys/filsys.h 中有超级块结构的详细说明。链头块中的这 402 个字节,就对应着超级块中的指针 s-free[s-nfree]和数组 s-free[]。更明白地说,s-nfree 的值在 0 到 99 之间变化,s-free[s-nfree]则取遍 s-free[0]到 s-free[99]。特别地,当 s-nfree=0 时,s-free[0]表示的是链头块的块号。

XENIX 系统建/删文件时的流程图如图 2 及图 3 图中用 SN 代表 s-nfree,用 SF[ ]代表 s-free[ ]。

从上面的流程图可看出,XENIX 系统建、删文件时 s-nfree[]的作用并不是通常意义下所以"刚压入的块在索引表的最底部"的说法是不对的。从概率的角度来看,"刚压入的块在索引表的最底部"与"在索引表的最上部且被破坏"的几率相同。正确的说法:由 s-nfree 即《恢复》文中所说"第七/八两字节 0024(十六进制)"可知,最

© 中国科学院软件研究所 http://www.c-s-a.org.cn

后释放的块号是 s-free[24-1] = 18D43H, 在超级块中的偏移地址是 0494H ～0497H。若按《恢复》一文的假定,文件长 4096 字节,那么刚删除的文件释放的数据块号应为:

18D43H, 18D44H, 18D45H 和 18D46H

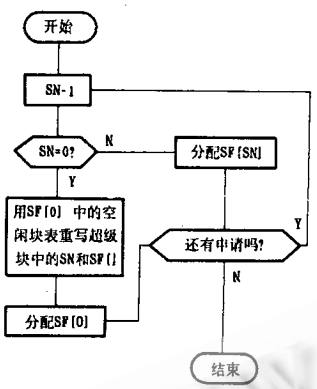

图 2　建立文件时的流程图

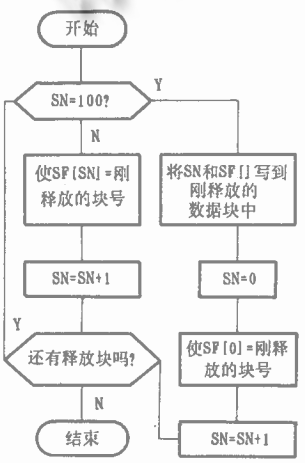

图 3　删除文件时的流程图

这里,《恢复》一文的作者混淆了系统逻辑块和物理块的概念。从《恢复》文图 2 中的偏移 07FCH～07FFH 处的值 00020000 可知,该 XENIX 系统的逻辑块长为 1023 字节, 也就是说 s-free[]中的每一个块号代表两个 512 字节的磁盘物理块。

3.i 节点的问题

《恢复》一文图 2 中偏移 0598H～0599H 及其后的 200 个字节, 在超级块中分别称为 s-ninode 和 s-inode[],i 节点的分配和回收原理与数据块相似,不同的是当 s-inode[]s-inode[],同时使 s-ninode=99;若 s-inode[]满了,系统只用最后释放的 i 节点更新 s-inode[0]。

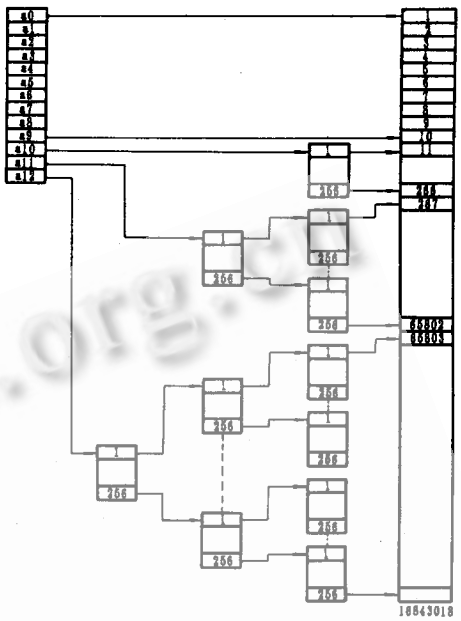

图 4

应强调的是:对于空闲块表,s-free[s-nfree]上面的数据块(例如《恢复》一文中的)s-free[24]=18D42H 左/上面的数据块)是空闲块;s-free[24] 及其下面的数据块的状态则未定,可能是空闲块,也可能是已分配过的数据块。故不能轻易分配使用。若从"索引表最底部"取 8 个块号,"好"的情况取到的是空闲块,徒劳无功;坏的情况取到的是已分配过的块,结果不可预料。

类似地,空闲 i 节点表中,s-inode[s-ninode]上面的 i 节点(例如《恢复》一文中的)s-inode[63]=9B5H 左/上面的 i 节点)是空闲 i 节点;s-inode[s-ninode]及其下面的 i 节点状态未定,不能如《恢复》一文所说,"从中任取一个",否则可能破坏现有的文件。另外,i 节点中只有 3×13=39 个字节用于表示文件使用的数据块。对于长度超过 10K(10 个逻辑块长)的文件,怎样重构 i 节点呢?文中没说。有兴趣的读者可参考九一年第 9 期里面有 i 节点结构的详细说明和有关数据块地址的变量 a0～a12 的简单说明。图 4 是它们的用法的示意图。

图中最右列表示文件使用的示意性数据块号;a10～a12 的值依次表示一次间接块, 二次间接块和三次间接块。所谓一次间接块是指该块中的内容为(<256 个)数据块号;所谓二次间接块是指该块中的内容为(<256 个)

© 中国科学院软件研究所 http://www.c-s-a.org.cn

一次间接块;所谓三次间接块是指该块中的内容为(<256个)二次间接块。

ai (O≤i<12)的使用与可表示文件的大小关系如下:

a0～a9 <10K

a0～a10<10+256=266(K)

a0～a11<10+256+256×256+65802(K)

a0～a12<10+256+256×256+256×256×256=16843018(K)

4.首地址问题

《恢复》一文提到,"若文件长度超过索引表给出的空闲块数,则剩余块号就得从索引表最上端,即00018D66块号找出。"

此处的18D66号,就是一个链头块。由于文件系统的逻辑块长为1024字节,即十六进制的400H字节,所以该块在文件系统内距文件系统开始处的偏移量(或称"首地址")是 18D66×400=635980H。没有必要将18D66H换算成二进制后左移十位再换算回十六进制。至于18D66H块中第一字段的内容,如前所说,固定为100(64H),不可能是80(50H)。需要指出的是,在XENIX系统中,随着文件的建、删,最后释放的数据块在空闲块表中的位置是随机的,可能位于空闲块表的最下端(s-free[99]),也可能位于空闲块表最上端(s-free[0])。当位于(s-free[0])时,即使想恢复的文件只需一个逻辑块,也会用到"索引表最上端"的数据块。

对于i节点,因每一个i节点长64(40H)字节,i节点区由从计数的第2个逻辑块算起,i节点号从2开始,相对文件系统开始处的偏移量为 1024×2+64=2112=840H,所以由公式:偏移量=(i节点号-1)×40+800可得(《恢复》一文取的)9F3号i节点的"首地址"为:(9F3-1)×40+800=28480H。换言之,既然9F3号i节点开始于28480H处,那么用adb重构该i节点时,使用的"首地址"就应该是28480H,何来305C0呢?

附:/usr/include/sys/filsys.h 节选

```
/*
 * @(#)filsys.h 2.1 88/05/18
 * Copyright (C) the Santa Cruz Operation,1984,1985,1986,1987。
 * Copyright (C) Microsoft Corporation,1984,1985,1986,1987。
/*
 * Structure of the super-block        ;超级块结构
 */
struct filsys
{   ushort  s-isize;    /* size in blocks of i-list */
    daddr-t s-fsize;    /* size in blocks of entire volume */
    short   s-nfree;    /* number of addresses in s-free */
    daddr-t s-free[NICFREE]; /* free block list */
    short   s-ninode;   /* number of i-nodes in s-inode */
    ino-t   s-inode[NICINOD]; /* free i-node list */
    char    s-flock;    /* lock during free list manipulation */?
    char    s-ilock;    /* lock during i-list manipulation */
    char    s-fmod;     /* super block modified flag */
    char    s-ronly;    /* mounted read-only flag */
    time-t  s-time;     /* last super block update */
    daddr-t s-tfree;    /* total free blocks */
    ino-t   s-tinode;   /* total free inodes */
    short   s-dinfo[4]; /* device information */
    char    s-fname[6]; /* file system name */
    char    s-fpack[6]; /* file system pack name */
    /* remainder is maintained for xenix */
    char    s-clean;    /* if structure is properly closed */
    char    s-fill[NSBFILL]; /* space to make sizeof filsys
            be
                BSIZE */
    long    s-magic;    /* indicates version of filsys */
    long    s-type;     /* type of new file system */
};                      ↑                ↑ 新文件系统类型
.....
/* s-type,block size of file system */ ;s-type:文件系统块的大小
#define S-B512 1 /* 512 byte block */
#define S-B1024 2 /* 1024 byte block */
};                      ↑                ↑ 每块长1024字节
.....
```

·投稿须知·

- 内容开门见山,直接进入主题;
- 来稿请尽量用打印稿,并附软盘,插图必须描绘清晰;
- 程序不宜太长,如超过150行,请指出重要段落及可删略部分并注明运行环境;
- 参考文献只指明主要的2～3篇.

© 中国科学院软件研究所 http://www.c-s-a.org.cn