**第13章-Linux文件系统结构**

**本节所讲内容：**

**13.1 硬盘结构**

**13.2 文件系统结构**

**13.2 硬链接和软链接**

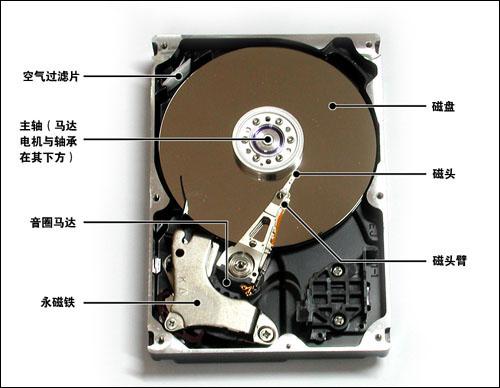
**13.4 实战：解决磁盘有空间但创建不了文件-修复服务器文件系统**

# 13.1 硬盘结构

## 13.1.1 硬盘结构

**文件系统结构，理解文件系统，要从文件储存说起。**

**硬盘结构：**



**互动：磁盘内部是真空的吗？ 是：1 ，不是：2**

**磁盘内部不是真空，只不过里面的空气很干净。如果是真空，还不利于散热**

**磁盘相关专业术语：**

**硬盘的内部是金属盘片，将圆形的盘片划分成若干个扇形区域，这就是扇区。若干个扇区就组成整个盘片。为什么要分扇区？是逻辑化数据的需要，能更好的管理硬盘空间。 以盘片中心为圆心，把盘片分成若干个同心圆，那每一个划分圆的“线条”，就称为**[**磁道**](http://baike.baidu.com/view/201106.htm)**。**

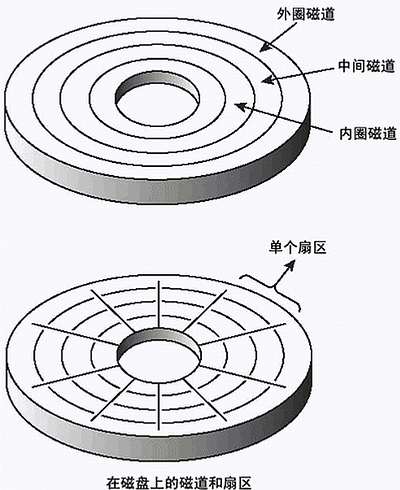
**硬盘内的盘片有两个面，都可以储存数据，而硬盘内的盘片往往不止一张，常见的有两张，那么，两张盘片中相同位置的磁道，就组成一个“柱面”，盘片中有多少个磁道，就有多少个柱面。盘片两面都能存数据，要读取它，必须有磁头，所以，每一个面，都有一个磁头，一张盘片就有两个磁头。**

**硬盘的存储容量=磁头数×磁道（柱面）数×每道扇区数×每道扇区字节数。**

**磁道从外向内自0开始顺序进行编号，各个磁道上的扇区数是在硬盘格式化时确定的。**

**文件储存在硬盘上，硬盘的最小存储单位叫做"扇区"（Sector）。每个扇区储存512字节（相当于0.5KB）。**

**比较古老的CHS (Cylinder/Head/Sector ：磁头(Heads)、柱面(Cylinder)、扇区(Sector)）结构体系. 因为很久以前，在硬盘的容量还非常小的时候，人们采用与软盘类似的结构生产硬盘。也就是硬盘盘片的每一条磁道都具有相同的扇区数，由此产生了所谓的3D参数，即是磁头数（Heads）、柱面数（Cylinders）、扇区数（Sectors）以及相应的3D寻址方式。**



**互动 ： 如上的磁盘结构有没有问题？？？**

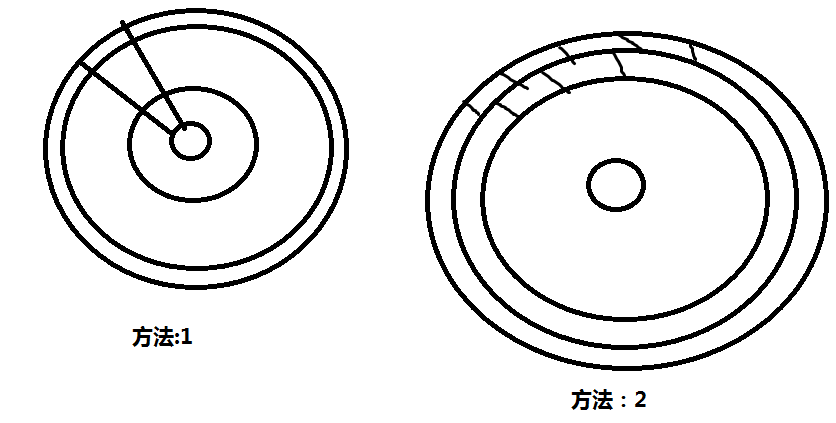
**这种结构有问题：**

**以前老式的磁盘，每个磁道的扇区都一样，这样外磁道整个弧长要大于内部的扇区弧长，因而其磁记录密度就要比内部磁道的密度要小。最终，导致了外部磁道的空间浪费。**

**如查你磁盘设计工程师，你打算怎么解决？ 你选择下面哪种方法？**

**方法1：每个磁道的宽度不一样，从而让每个扇区面积尽量一样**

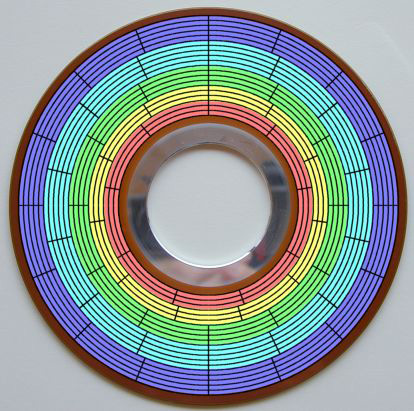
**方法2：不再一刀切，让磁道中的扇区数量可以不一样**

****

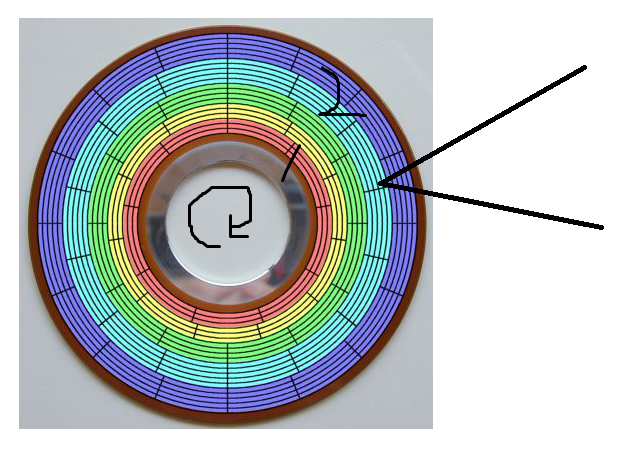
**现在硬盘都采用这种技术：ZBR（Zoned Bit Recording）区位记录 （Zoned zōnd ）**

**Zoned-bit recording（ZBR 区位记录）是一种物理优化硬盘存储空间的方法，此方法通过将更多的扇区放到磁盘的外部磁道而获取更多存储空间。**

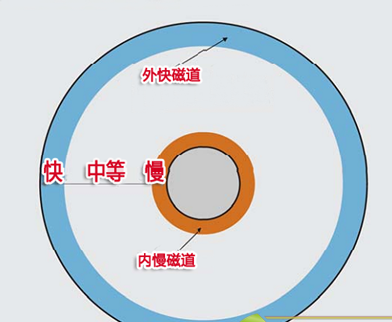
**ZBR磁盘扇区结构示意图**

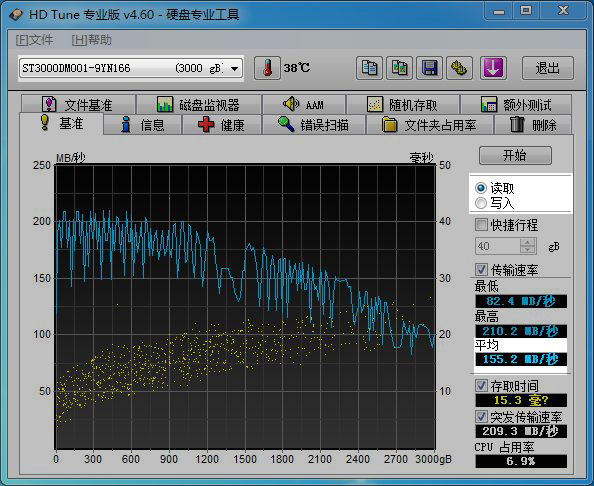


**互动: 从外面读数据快？ 还是从里面快？ 里：1 外：2**

****

**使用ZBR 区位记录法做的磁盘有以下特点：读外圈的数据快，读内圈的数据慢，所以测试硬盘经常看到读取速度越来越慢的曲线图就很正常了。**





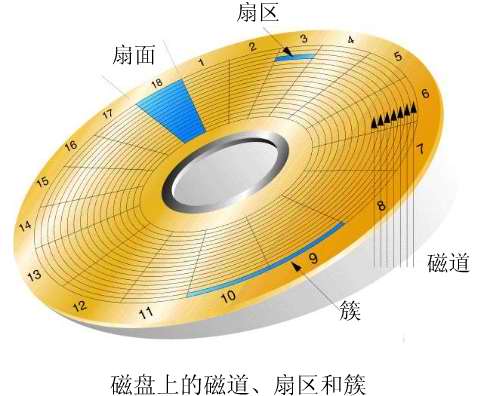
**互动：windows安装系统的C盘或Linux boot分区一般安装在磁盘最外面还是最里面？**

**windows ： C盘安装最外，速度也是最快**

**Linux ： boot分区和 swap分区，装最外面**

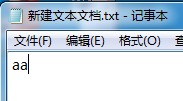
**磁盘写数据时，先从外面往里。**

## 13.1.2 簇和block

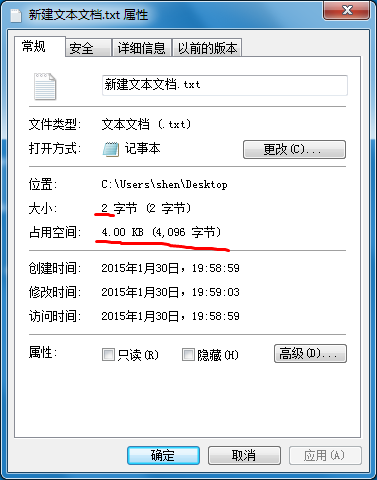


**簇类似于Linux系统中的block**

**例：在win10系统，新一个文本文件“新建文本文档.txt”，只输入aa两个字符。**



**右击属性查看大小： 说明我的NTFS文件系统中默认的簇大小为4KB**



**[root@xuegod63 ~]# stat /etc/passwd #查看Linux block 大小**

**文件："/etc/passwd"**

**大小：2053 块：8 IO 块：4096 =4KB 普通文件**

# 13.2 文件系统结构

**Linux文件系统由三部分组成 ： 文件名，inode，block**

**Linux文件系统： ext3,ext4，xfs**

**windows文件系统： FAT32，NTFS**

## 13.2.1 文件名：

**[root@xuegod63 ~]# cp /etc/passwd a.txt**

**[root@xuegod63 ~]# ls a.txt # a.txt 就是文件名**

## 13.2.2 inode的内容

**inode包含文件的元信息，具体来说有以下内容：**

**\* 文件的字节数**

**\* 文件拥有者的User ID**

**\* 文件的Group ID**

**\* 文件的读、写、执行权限**

**\* 文件的时间戳，共有三个：ctime指inode上一次变动的时间，mtime指文件内容上一次变动的时间，atime指文件上一次打开的时间。**

**\* 链接数，即有多少文件名指向这个inode**

**\* 文件数据block的位置**

**可以用stat命令，查看某个文件的inode信息：**

**[root@xuegod63 ~]# stat a.txt**

**File: ‘a.txt’**

**Size: 2053 Blocks: 8 IO Block: 4096 regular file**

**Device: 803h/2051d Inode: 18521833 Links: 1**

**Access: (0644/-rw-r--r--) Uid: ( 0/ root) Gid: ( 0/ root)Access最近访问时间: 2018-05-16 14:55:36.061095050 +0800**

**Modify最近更改时间: 2018-05-16 14:55:36.062095050 +0800**

**Change最近改动时间: 2018-05-16 14:55:36.062095050 +0800**

**Birth创建时间: -**

**[root@xuegod63 ~]# ll /etc/passwd #ll 其实就是查看passwd的inode信息**

**-rw-r--r--. 1 root root 2053 Sep 19 2017 /etc/passwd #ll查看到时间是ctime时间**

**互动：**

**ctime是什么？ 是创建时间吗？ 不会：1**

**mtime ： modify time 修改文件内容的时间**

**atime ： access time 访问文件内容的时间**

**ctime指inode上一次文件属性变动的时间，change time 。 比如： chmod +x a.sh**

**mtime指文件内容上一次变动的时间，modify time  。如：echo aa >> a.sh 或vim a.sh 修改内容**

**atime指文件上一次查看文件的时间，access time 。 如： cat a.sh**

**例2:测试mtime时间，黑客先修改时间，再植入木马程序，防止 find / -mtime 查看木马文件**

**[root@xuegod63 ~]# stat a.txt #查看时间**

**[root@xuegod63 ~]# date -s '13:42'**

**[root@xuegod63 ~]# vim a.txt #写入aaaa，vim会修改mtime和atime时间**

**[root@xuegod63 ~]# stat a.txt #查看时间**

**[root@xuegod63 ~]# chmod +x a.txt #修改ctime，有时黑客忘记修改ctime时间了，所以你可以find / -ctime 查看木马文件**

**[root@xuegod63 ~]# stat a.txt #查看时间**

**这个一招学到手：1**

## 13.2.3 inode的大小

**inode也会消耗硬盘空间，所以硬盘格式化的时候，操作系统自动将硬盘分成两个区域。一个是数据区，存放文件数据；另一个是inode区（inode table），存放inode所包含的信息。**

**每个inode节点的大小，一般是128字节或256字节。inode节点的总数，在格式化时就给定，假定在一块1GB的硬盘中，每个inode节点的大小为128字节，每1KB就设置一个inode，那么inode table的大小就会达到128MB，占整块硬盘的12.8%。**

**inode号码**

**每个inode都有一个号码，操作系统用inode号码来识别不同的文件。**

**Unix/Linux系统内部不使用文件名，而使用inode号码来识别文件。对于系统来说，文件名只是inode号码便于识别的别称或者绰号。表面上，用户通过文件名，打开文件。实际上，系统内部这个过程分成三步：首先，系统找到这个文件名对应的inode号码；其次，通过inode号码，获取inode信息；最后，根据inode信息，找到文件数据所在的block，读出数据。**

**例1： 使用ls -i命令，可以看到文件名对应的inode号码**

**[root@xuegod63 ~]# ls -i a.txt**

**440269 a.txt**

**例2：查看每个硬盘分区的inode总数和已经使用的数量，可以使用df命令。**

**[root@localhost ~]# df -i**

**Filesystem Inodes IUsed IFree IUse% Mounted on**

**/dev/sda2 640848 151010 489838 24% /**

**tmpfs 145579 1 145578 1% /dev/shm**

**/dev/sda1 51200 38 51162 1% /boot**

**注：由于每个文件都必须有一个inode，因此有可能发生inode已经用光，但是硬盘还未存满的情况。这时，就无法在硬盘上创建新文件。**

**13.2.4 目录文件**

**Unix/Linux系统中，目录（directory）也是一种文件。打开目录，实际上就是打开目录文件。**

**目录文件的结构非常简单，就是一系列目录项的列表。每个目录项，由两部分组成：所包含文件的文件名，以及该文件名对应的inode号码。**

**[root@xuegod63 ~]# ls -id /etc**

**8388673 /etc**

**例：ls -i命令列出整个目录文件，即文件名和inode号码：**

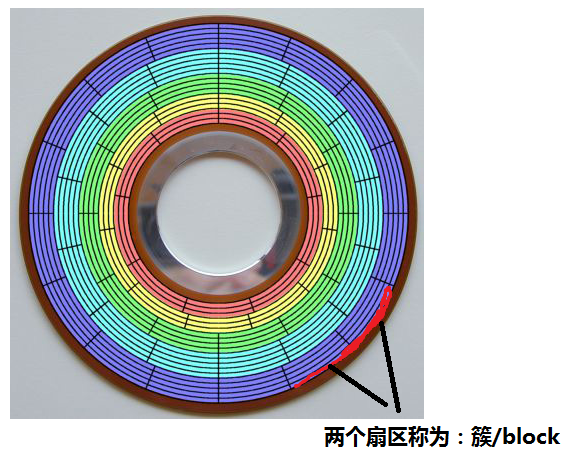
**[root@xuegod63 ~]# ls -i /etc**

## 13.2.5 block块大小

**block 是真正存储数据的地方。**

**block是 文件系统 中最小的存储单位**

**扇区 是 磁盘 中最小的存储单位**

****

**在linux下中叫：block，在windows中叫：簇**

**互动：为什么要有block，直接使用扇区可以吗？**

**操作系统读取硬盘的时候，不会一个个扇区（512字节）地读取，这样效率太低，而是一次性连续读取多个扇区，即一次性读取一个"块"（block）。这种由多个扇区组成的"块"，是文件存取的最小单位。"块"的大小，最常见的是1KB，即连2个 sector扇区组成一个 block。或4K。**

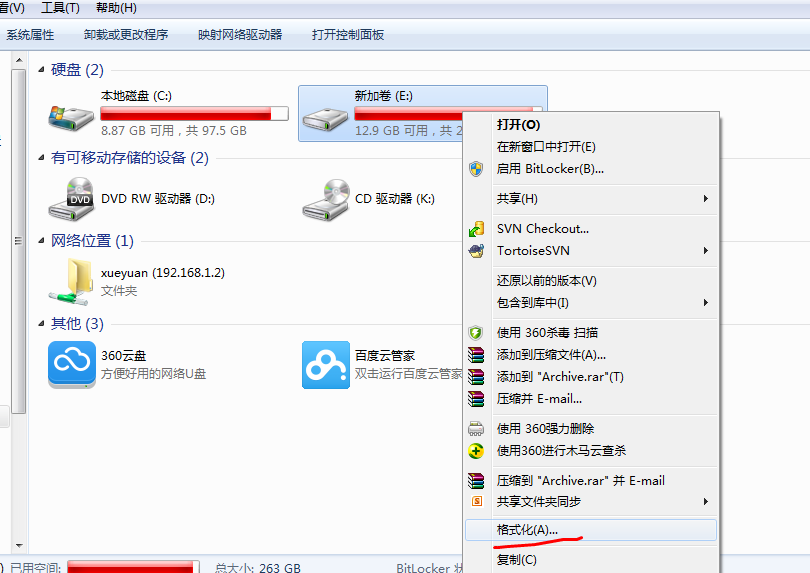
**情景： 如果没有block？ 会怎么样？**

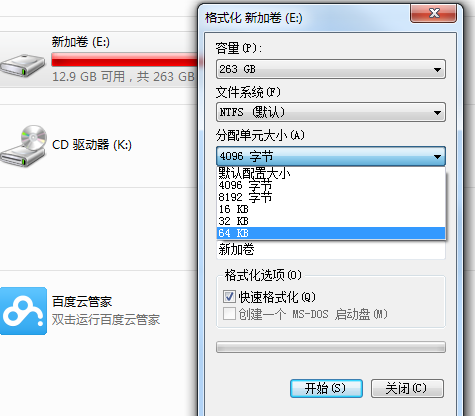
**夜深人静，下了3.6G cang.avi 的电影，一次读512B ， 寻址次太多，太慢了。。。**

**结果。。。你懂得 。。。？**

**结果:**  **avi看成了 jpg** C:\Users\shen\AppData\Local\Temp\LY6DR3ISJE0)6K)L)]~VIZK.gif

**例1：格式化修改磁盘，修改簇大小**





**实战：簇和block大小设定**

**你喜欢大一些，还是小一些？** C:\Users\shen\AppData\Local\Temp\X@8}U9MLE}EBUE273)]9PGF.gif



**簇和block调大：**

**优点： 速度快，节约寻址时间，缺点：空间浪费**

**比如： 2T硬盘, 前1.5Ｔ，使用4K， 把剩下的500G格式化成64K簇。用空间换时间**

**例2：查看Linux系统块大小**

**[root@xuegod63 ~]# stat /etc/passwd | grep IO**

**大小：2053 块：8 IO 块：4096 普通文件**

**#block到是4K**

**总结：**

**硬盘的结构：ZBR 区位记录**

**inode（inode表中主要看inode号）**

**inode号唯一标识一个文件（一个文件系统里面）**

**inode用完了，文件就不能创建了。**

**inode数据量设置大一些：可以创建多个文件。占用空间比较大**

**inode数据量设置小一些：可以创建很少文件。占用空间比较小**

**block**

**block设置大：效率高，利用率低。**

**block设置小：效率低，利用率高。**

# 13.3 文件的硬链接和软链接

## 13.3.1 Linux链接概念 Linux链接分两种，一种被称为硬链接（Hard Link），另一种被称为软链接，即符号链接（Symbolic Link）。默认情况下，ln命令产生硬链接。

**【硬连接】：硬连接指通过索引节点号来进行连接。inode是可以对应多个文件名的**

**在Linux的文件系统中，保存在磁盘分区中的文件不管是什么类型都给它分配一个编号，称为索引节点号(Inode Index)。**

**在Linux中，多个文件名可以指向同一索引节点。一般这种连接就是硬连接。**

**硬连接的作用是允许一个文件拥有多个有效路径名，这样用户就可以建立硬连接到重要文件，以防止“误删”的功能。**

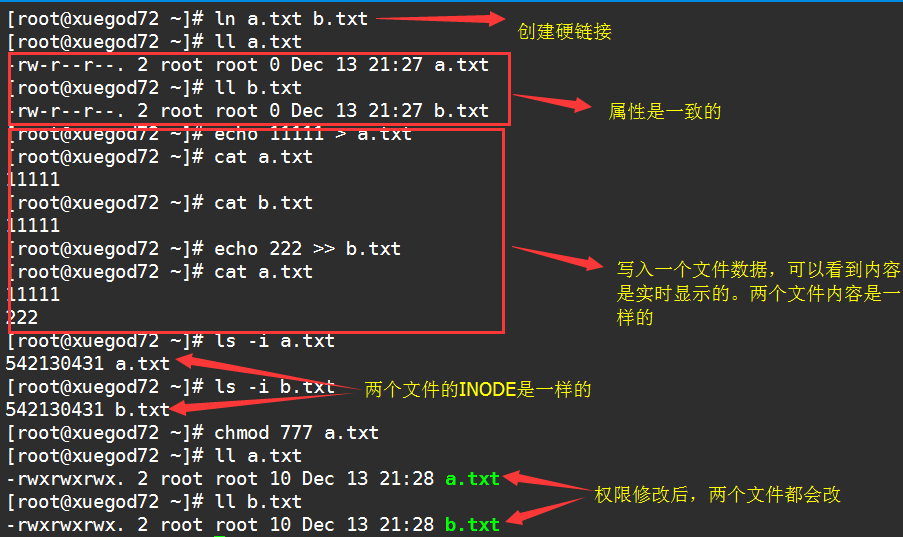
**只删除一个连接并不影响索引节点本身和其它的连接，只有当最后一个连接被删除后，文件的数据块及目录的连接才会被释放。也就是说，文件真正删除的条件是与之相关的所有硬连接文件均被删除。**

**【软连接】：另外一种连接称之为符号连接（Symbolic Link），也叫软连接。软链接文件有类似于Windows的快捷方式。它实际上是一个特殊的文件。在符号连接中，文件实际上是一个文本文件，其中包含的有另一文件的位置信息**

## 13.3.2 实战-1: ln命令创建硬链接

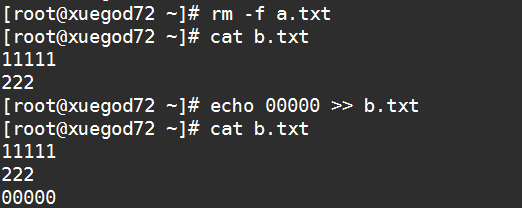
**语法格式：ln 源文件 目标文件**

**[root@xuegod63 ~]# echo 1111 > a.txt**

****

**硬链接的原理就是多个文件名指向同一个inode，因此多个文件名共用一个inode号，达到共享与备份的目的**

**注意：源文件被删除，不影响链接文件的正常使用**

****

**硬链接不能针对目录创建**

****

**硬链接不能跨分区进行创建**

****

**硬链接的特点: 无法针对目录,跨分区无法实现。因为每个分区都有自己独立的INDOE编号**

**互动：为什么刚创建的一个目录，链接数就是2？**

**[root@xuegod63 ~]# mkdir test**

**[root@xuegod63 ~]# ll -d test/**

**drwxr-xr-x 2 root root 6 5月 16 15:55 test/**

**默认新一个空目录，此目录的第二字段就是2（包含两个隐藏目录，因为每一个目录都有一个指向它本身的子目录"." 和指向它上级目录的子目录".."），所以test是一个链接， 隐藏目录. 是第二个链接**

**[root@xuegod63 ~]# ll -id test/ #两个inode号是一样的**

**2453723 drwxr-xr-x 2 root root 6 5月 16 15:55 test/**

**[root@xuegod63 ~]# ll -id test/.**

**2453723 drwxr-xr-x 2 root root 6 5月 16 15:55 test/.**

## 13.3.3 ln -s 创建软连接

**软链接：相当于windows中的快捷方式**

**语法：ln -s 源文件 软链接的名字**

**例：**

**[root@xuegod63 ~]# cp /etc/passwd a.txt**

**[root@xuegod63 ~]# ln -s a.txt a-link.txt**

**[root@xuegod63 ~]# ll a-link.txt**

**lrwxrwxrwx 1 root root 5 5月 16 16:10 a-link.txt -> a.txt**

**[root@xuegod63 ~]# rm -rf a.txt**

**[root@xuegod63 ~]# ll a-link.txt**

**lrwxrwxrwx 1 root root 5 5月 16 16:10 a-link.txt -> a.txt**



**注：源文件被删除，链接文件失效**

**例2：能针对目录和跨分区创建软链接**

**[root@xuegod63 ~]# ln -s /boot/grub grub-link**



**能跨分区创建（源文件必须写绝对路径）**

**[root@xuegod63 ~]# cd /boot/**

**[root@xuegod63 boot]# ln -s ./grub /root/aaa**

**[root@xuegod63 boot]# ll /root/aaa**

**lrwxrwxrwx 1 root root 6 5月 16 16:21 /root/aaa -> ./grub #报错了**

## 13.3.4 inode的特殊作用

**由于inode号码与文件名分离，这种机制导致了一些Unix/Linux系统特有的现象。**

**1. 有时，文件名包含特殊字符，无法正常删除。这时，直接删除inode节点，就能起到删除文件的作用。**

**2. 移动文件或重命名文件，只是改变文件名，不影响inode号码。**

**3. 打开一个文件以后，系统就以inode号码来识别这个文件，不再考虑文件名。因此，通常来说，系统无法从inode号码得知文件名。**

**互动：为什么每次修改完服务器配置文件后，都需要重新加载一下配置文件？**

**因为vim每次修改完后，Inode号都会变。**

**[root@xuegod63 ~]# cp /etc/passwd passwd**

**[root@localhost ~]# ls -i passwd**

**393418 passwd**

**[root@localhost ~]# vim passwd #添加一些内容**

**[root@localhost ~]# ll -i passwd**

**440252 -rw-r--r-- 1 root root 1813 Dec 29 12:04 passwd**

**就是为什么每次修改完服务器的配置文件，都要重启服务，重新读一下配置文件。**

# 13.4 实战：解决磁盘有空间但创建不了文件-修复服务器文件系统

## 13.4.1 解决磁盘有空间但创建不了文件

**实战场景：在一台配置较低的Linux服务器（内存、硬盘比较小）的/data分区内创建文件时，系统提示磁盘空间不足，用df -h命令查看了一下磁盘使用情况，发现/data分区只使用了80%，还有1.9G的剩余空间，但是无法创建新的文件。当时使用的是root用户。服务器没有被黑。**

**[root@xuegod63 ~]# df -h**

**文件系统 容量 已用 可用 已用% 挂载点**

**/dev/sda3 10G 8.0G 1.9G 80% /**

**常识： 只要权限够，磁盘上有空间一定可以创建文件。 这个是错的。**

**排查：**

**[root@xuegod63 ~]# df -i**

**文件系统 Inode 已用(I) 可用(I) 已用(I)% 挂载点**

**/dev/sda3 5242880 5242880 0 100% /**

**#后来用df -i查看了一下/data所在的分区的索引节点(inode)，发现已经用满(IUsed=100%)，导致系统无法创建新目录和文件。**

**查找原因：**

**/data/cache目录中存在数量非常多的小字节缓存文件，占用的Block不多，但是占用了大量的inode。**

**解决方案1：删除/data/cache目录中的部分文件，释放出/data分区的一部分inode。**

**解决方案2 : 在/data备份好一些文件，然后删除这些文件，释放一些inode，然后创建一个文件夹/data/cache2。在cache2下挂载一个新分区： sda4 ，下次写数据需要写到新分区cache2目录下。**

**inode分区完后，可以增加吗？ 不可以。 inode总数是在格式化时定下来。**

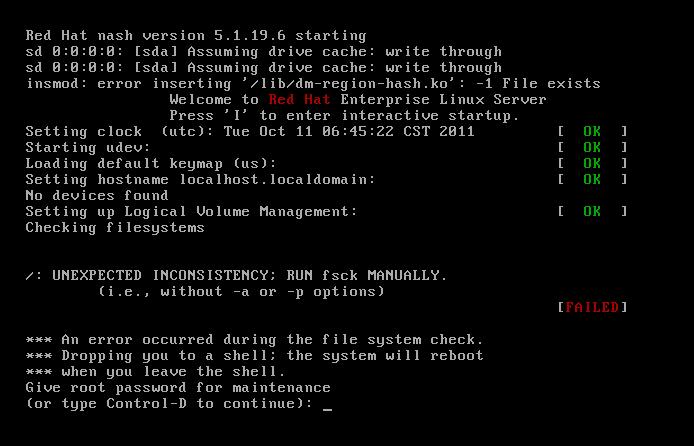
**[root@xuegod63 ~]# mkfs.ext4 -I 500000000000 /dev/sda1 #可以指定大小**

**参数：**

**[-i bytes-per-inode] [-I inode-size]**

## 13.4.2 实战：修复服务器文件系统

**实战场景：公司服务器突然断电后，再次启动后，报如下错误。**



**解决方法：**

**输入root 密码**

**fsck -f -y /dev/sda1 #把引导分区文件系统修复一下 # 慎用，给领导说一声**

**fsck -f -y /dev/sda3 #把根分区文件系统修复一下 # 慎用，给领导说一声**

**reboot 重启**

**fsck参数：**

**-y 对所有问题都回答 "yes"**

**-f 即使文件系统标记为 clean 也强制进行检查**

**总结：**

**13.1 硬盘结构**

**13.2 文件系统结构**

**13.2 硬链接和软链接**

**13.4 实战：解决磁盘有空间但创建不了文件-修复服务器文件系统**