Linux 云计算集群架构师

学神 IT 教育：从零基础到实战，从入门到精通！

联系方式：

学神 IT 教育官方网站: http://www.xuegod.cn

Linux 云计算架构师进阶学习群 QQ 群: 1072932914

学习顾问：小语老师　　学习顾问：边边老师　　学神微信公众号

微信扫码添加学习顾问微信，同时扫码关注学神公众号了解最新行业动态，获取更多学习资料及答疑就业服务！

第十三章 Linux 文件系统结构

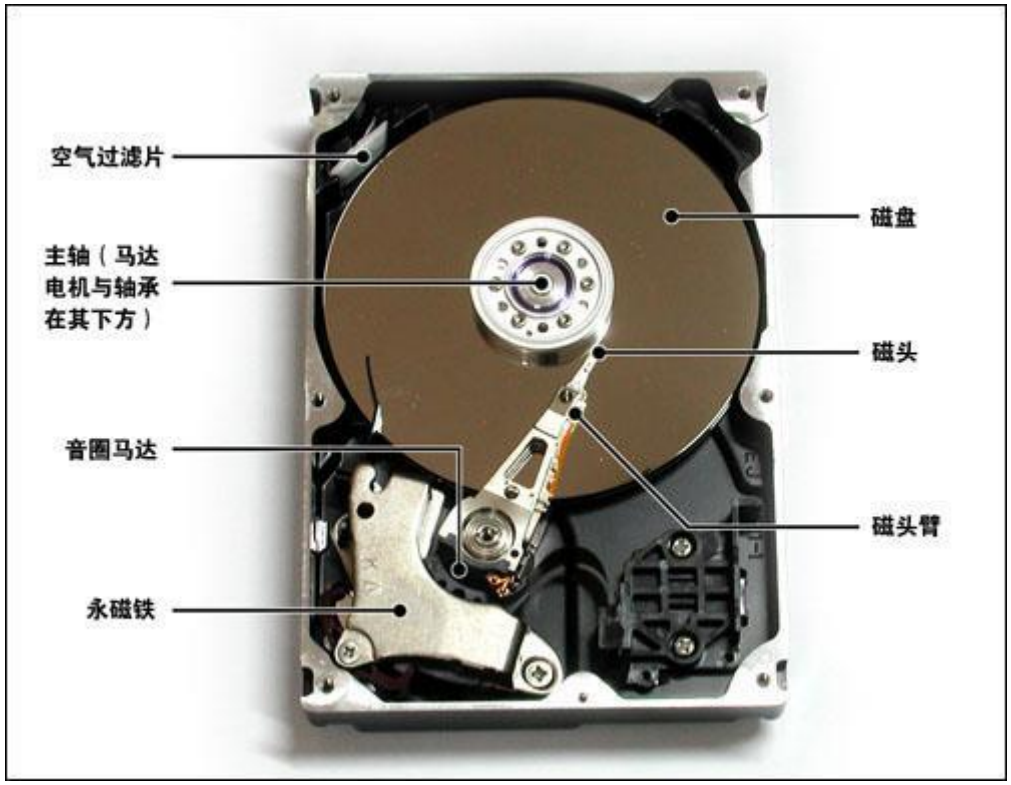
本节所讲内容：

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,会造成内部气体膨胀影响磁头的稳定性。

还有一些硬盘里面不是普通空气，而是惰性气体：氦（hai 害）气。它的好处是密度比空气小，可以减小磁盘转动阻力。但是充满氦气磁盘如果漏气会就损坏（常见于企业级）。

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。

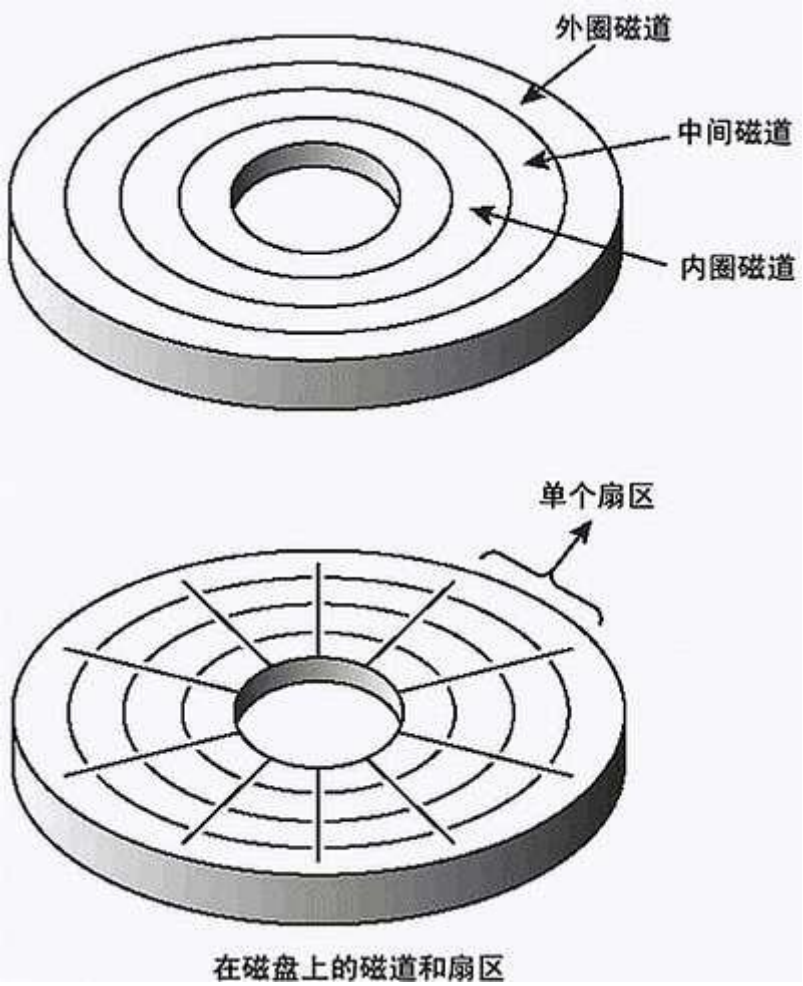
硬盘内的盘片有两个面，都可以储存数据，而硬盘内的盘片往往不止一张，常见的有两张，那么，两张盘片中相同位置的磁道，就组成一个“柱面”，盘片中有多少个磁道，就有多少个柱面。盘片两面都能存数据，要读取它，必须有磁头，所以，每一个面，都有一个磁头，一张盘片就有两个磁头。

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。



在磁盘上的磁道和扇区

互动 ： 如上的磁盘结构有没有问题？？？

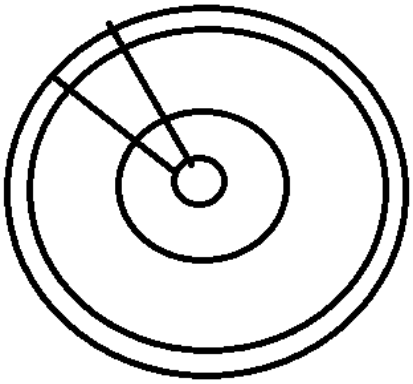
这种结构有问题：

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。

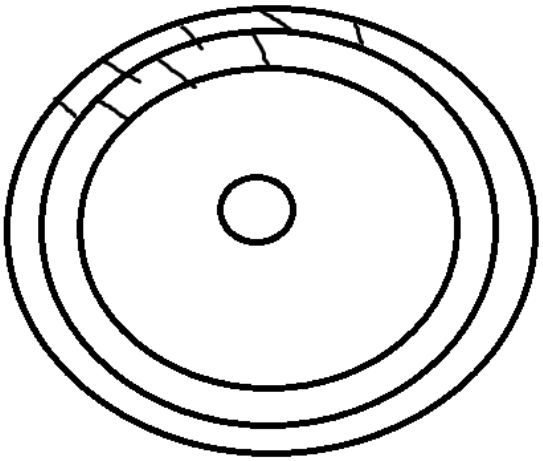
如查你磁盘设计工程师，你打算怎么解决？ 你选择下面哪种方法？

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



方法:1

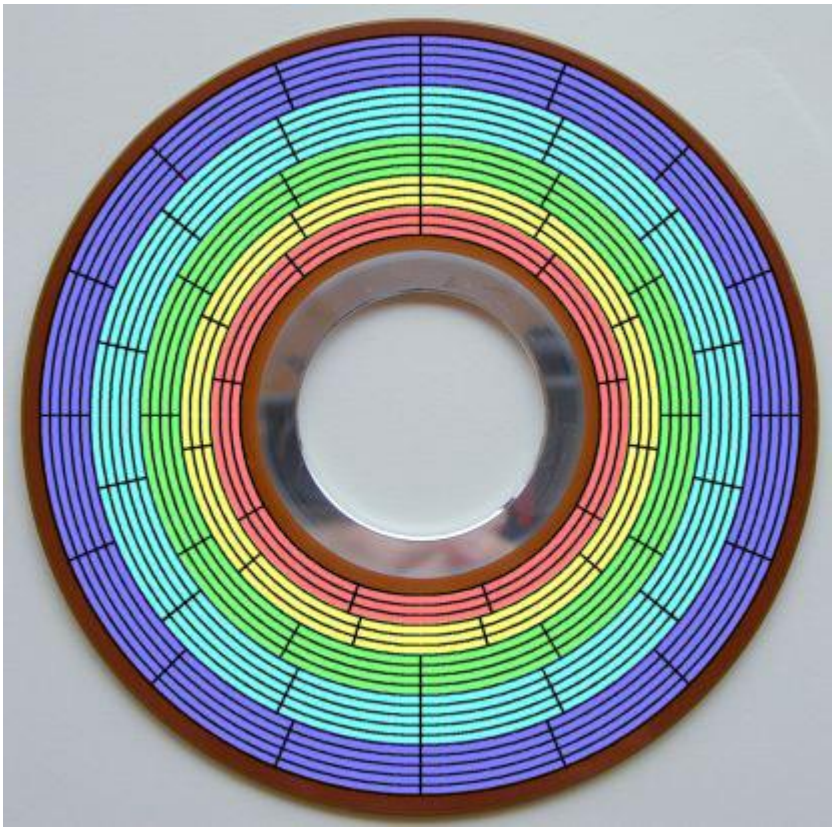


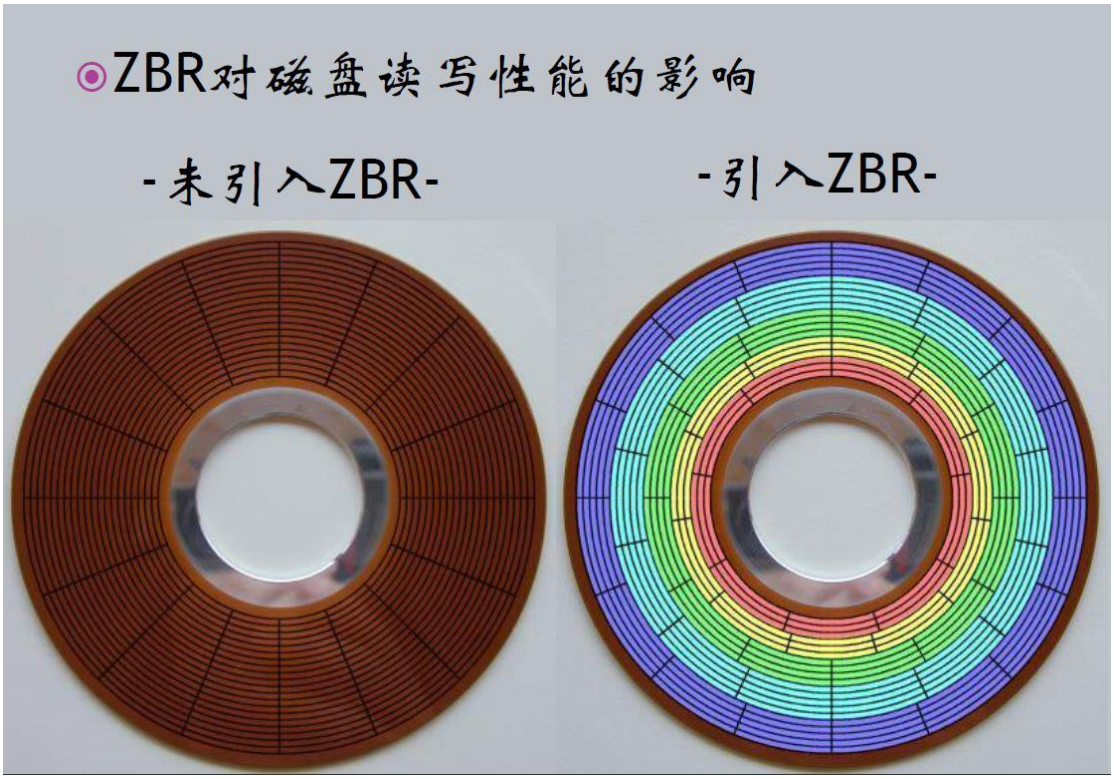
方法：2

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

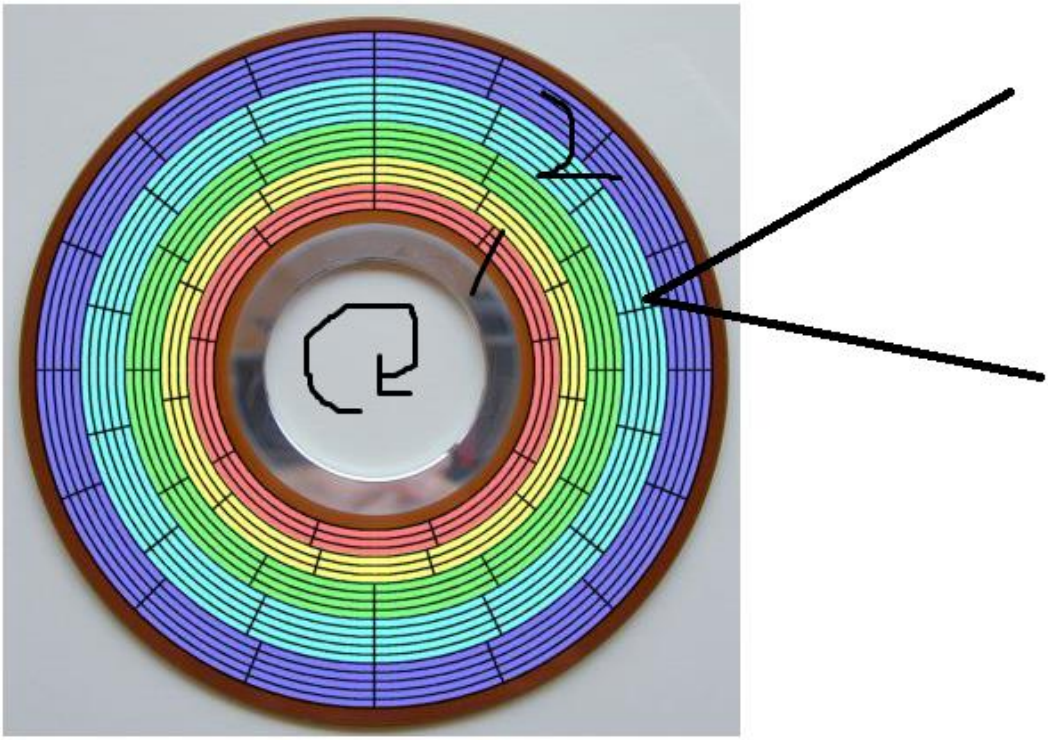
Zoned-bit recording（ZBR 区位记录）是一种物理优化硬盘存储空间的方法，此方法通过将更多的扇区放到磁盘的外部磁道而获取更多存储空间。

ZBR 磁盘扇区结构示意图

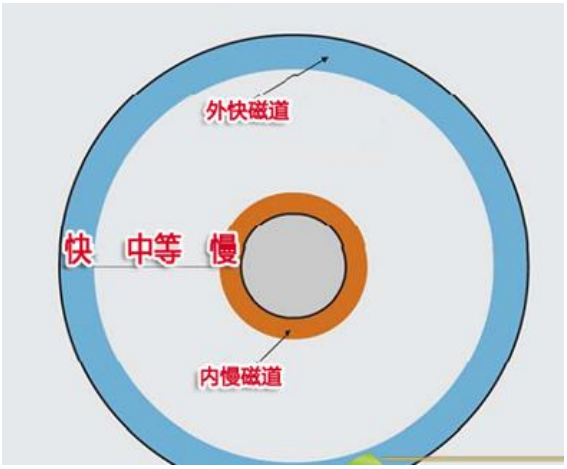


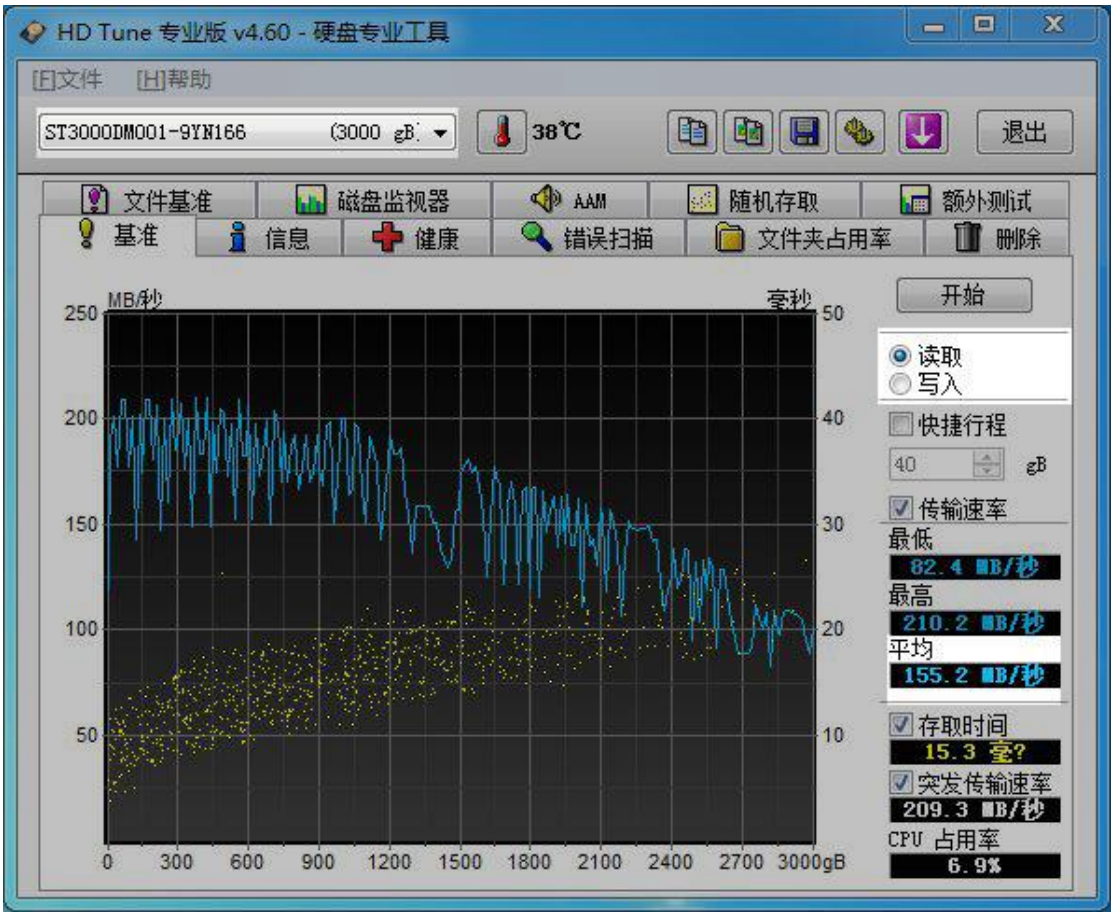


互动：从外面读数据快？ 还是从里面快？ 里：1 外：2



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。

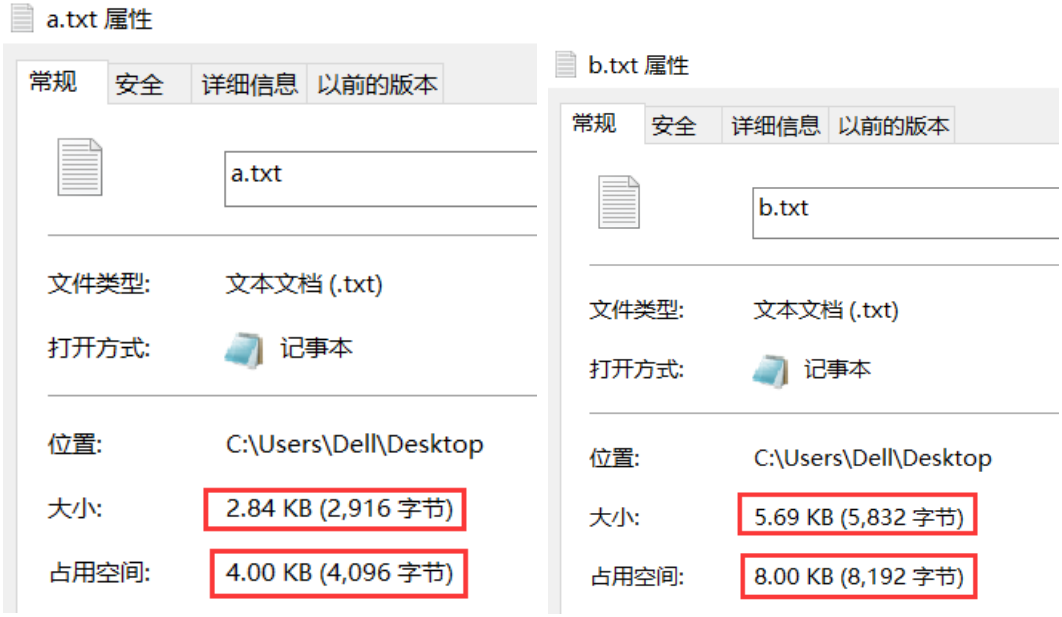
操作系统读取硬盘的时候，不会一个个扇区的读取，这样效率太低，而是一次性连续读取多个扇区，一次性读取一个“块”(block)。这种由多个扇区组成的“块”，是文件存取的最小单位。“块”的大小，最常见的是 1kb（连续 2 个“sector 扇区”组成一个“block 块”），(4kb 块 4096 字节 就是 8 个“sector 扇区”组成)，一个扇区是 512 字节。

[root@xuegod63 ~]# stat /etc/passwd #查看 Linux 系统的“块 block” 大小

文件："/etc/passwd"

大小：2053　　块：8　　IO 块：4096 字节 =4KB　普通文件

```
stat /etc/passwd
  文件："/etc/passwd"
  大小：946             块：8          IO 块：4096    普通文件
```



Windows 文件右击属性查看大小：　说明我的 windows 的 NTFS 文件系统中默认的簇大小为 4KB

Windows 里的簇相当于 linux 中的 block 块，都是文件的最小存取单位，由多个扇区组成。

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@centos60 ~]# echo $LANG                    #查看语言环境变量
[root@centos60 ~]# LANG=en_US.UTF-8              #语言环境变量改为英文
[root@centos60 ~]# LANG=zh_CN.UTF-8              #语言环境变量改为中文

[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 mtime 时间
(文件内容修改时间)
```

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'     #修改时间
[root@xuegod63 ~]# vim a.txt           #写入 aaaa
[root@xuegod63 ~]# stat a.txt          #查看时间
[root@xuegod63 ~]# chmod +x a.txt
```

注意：这里的思路是先改时间，然后修改文件，最后恢复时间，这样你按时间查找文件，就找不到

```
[root@xuegod63 ~]# hwclock -s          #以硬件时间为准，同步到系统时间

[root@xuegod63 ~]# yum -y install ntpdate
[root@xuegod63 ~]# ntpdate ntp1.aliyun.com #通过轮询指定正确时间的 NTP 服务器来校
正本地系统时间
```

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：

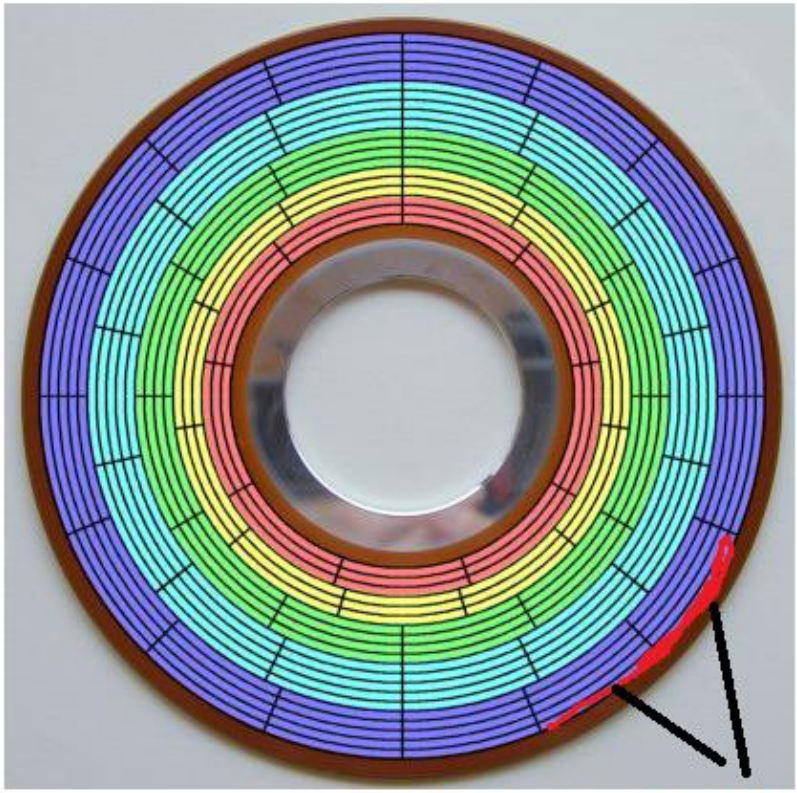
```
[root@xuegod63 ~]# ls -i /etc
```

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

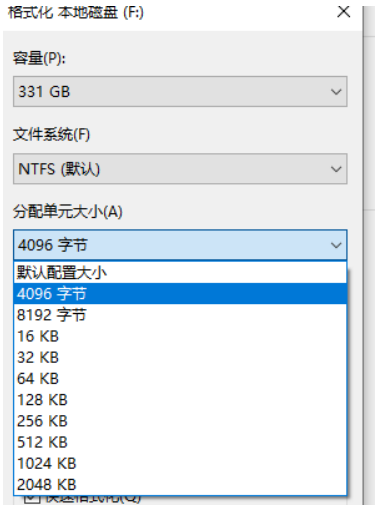


在 linux 下中叫：block，在 windows 中叫：簇

互动：为什么要有 block，直接使用扇区可以吗？直接往扇区上写数据就好了，为什么要搞 block?

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 ， 寻址次数太多，太慢了。。。
结果。。。你懂得 。。。？



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 1.5 T，使用 4K， 把剩下的 500G 格式化成 64K 簇。用空间换时间

例：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。(用空间换时间)
block 设置小：效率低，利用率高。（用时间换空间）

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。

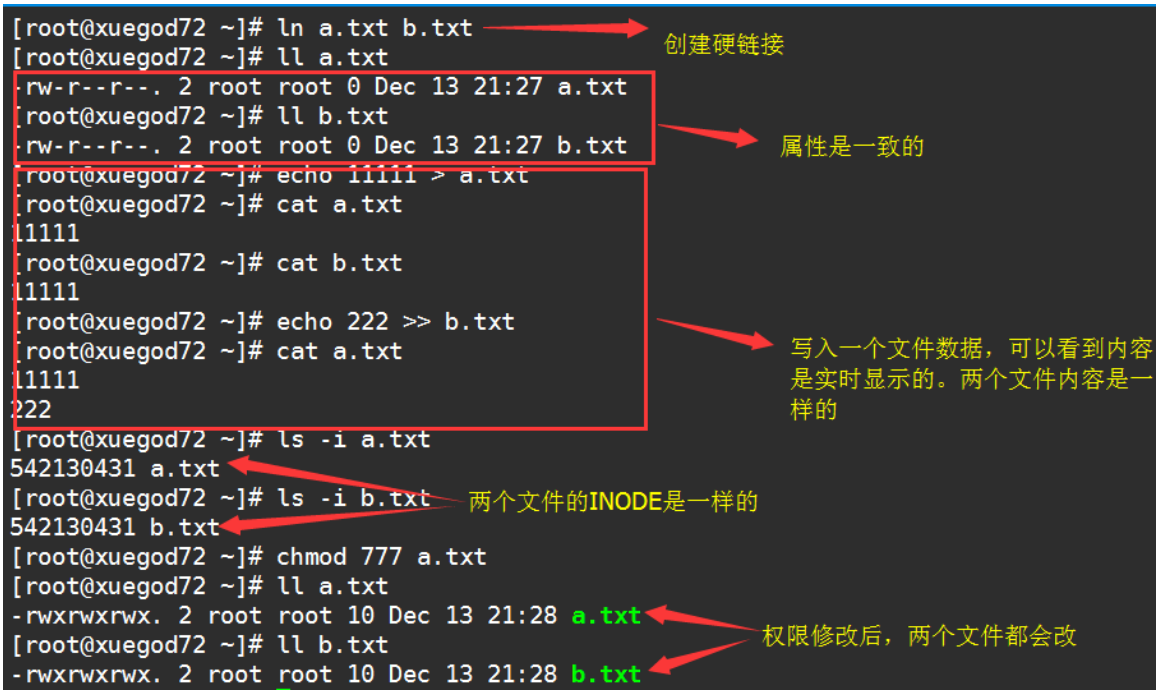
硬连接的作用是允许一个文件拥有多个有效路径名，这样用户就可以建立硬连接到重要文件，以防止“误删”的功能。

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，也可以理解为硬链接就是指向 inode 号的引用，删除一个硬链接并没有删除数据，只是删除了对这个 inode 的引用

注意：源文件被删除，不影响链接文件的正常使用，因为 2 个文件都只是对 inode 的引用

```
[root@xuegod72 ~]# rm -f a.txt
[root@xuegod72 ~]# cat b.txt
11111
222
[root@xuegod72 ~]# echo 00000 >> b.txt
[root@xuegod72 ~]# cat b.txt
11111
222
00000
```

硬链接不能针对目录创建

```
[root@xuegod72 ~]# ln /etc/ test
ln: '/etc/': hard link not allowed for directory
```

硬链接不能跨分区进行创建

```
[root@xuegod63 ~]# ln /dev/sr0 ./sr0
ln: 无法创建硬链接"./sr0" => "/dev/sr0": 无效的跨设备连接
```

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

每个分区在格式化之前就指定 inode 数据元信息存放区和文件数据存放区,所以 inode 和数据的对应关系就会在一个分区里面关联，而硬链接的文件是同分区下指向同一个 inode 的两个文件,故硬链接不能夸分区.那当我在 A 分区下为文件 aa 建立硬链接 bb 的时候,我编辑 bb 文件,aa 文件内容也会跟着改变,那

么当我把 bb 移动到另一个分区的时候 bb 的 inode 号发生了变化. bb 文件的变化已经不会对 aa 文件造成影响了,说明跨分区生成了 bb 新的 inode 元数据库,跟之前分区的 inode 元数据没有关系了

2《《

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 -a test #显示隐藏文件

[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/.

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

16797761 dr-xr-x---. 5 root root 282 12 月 12 14:31 test/..

[root@xuegod63 ~]# ll -id /root

16797761 dr-xr-x---. 5 root root 282 12 月 12 14:31 /root

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

```
[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

```
[root@xuegod63 ~]# ll -d grub-link
lrwxrwxrwx 1 root root 10 5月  16 16:18 grub-link -> /boot/grub
```

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。

（vim 打开文件时会有 swp 缓存文件，关闭时会用 swp 重命名，所以 inode 会变）

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。

```
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：

```
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。

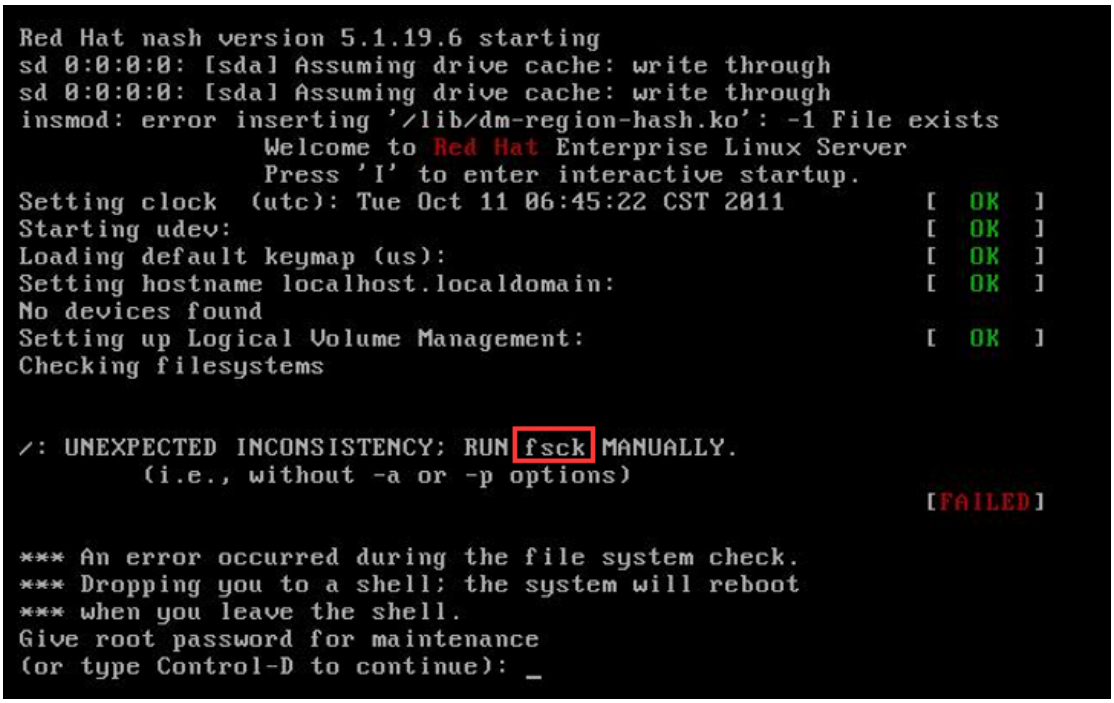
解决方案 1：删除/data/cache 目录中的部分文件，释放出/data 分区的一部分 inode。

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。有兴趣的可以自行扩展

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：

fsck（英文全拼：file system check)

命令用于检查与修复 Linux 档案系统，可以同时检查一个或多个 Linux 档案系统。

输入 root 密码

输入密码后，报错是哪个分区坏了，就修复哪个分区，修复的时候，可能会丢失数据，如果有重要数据的话，一定要和领导说一声。

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