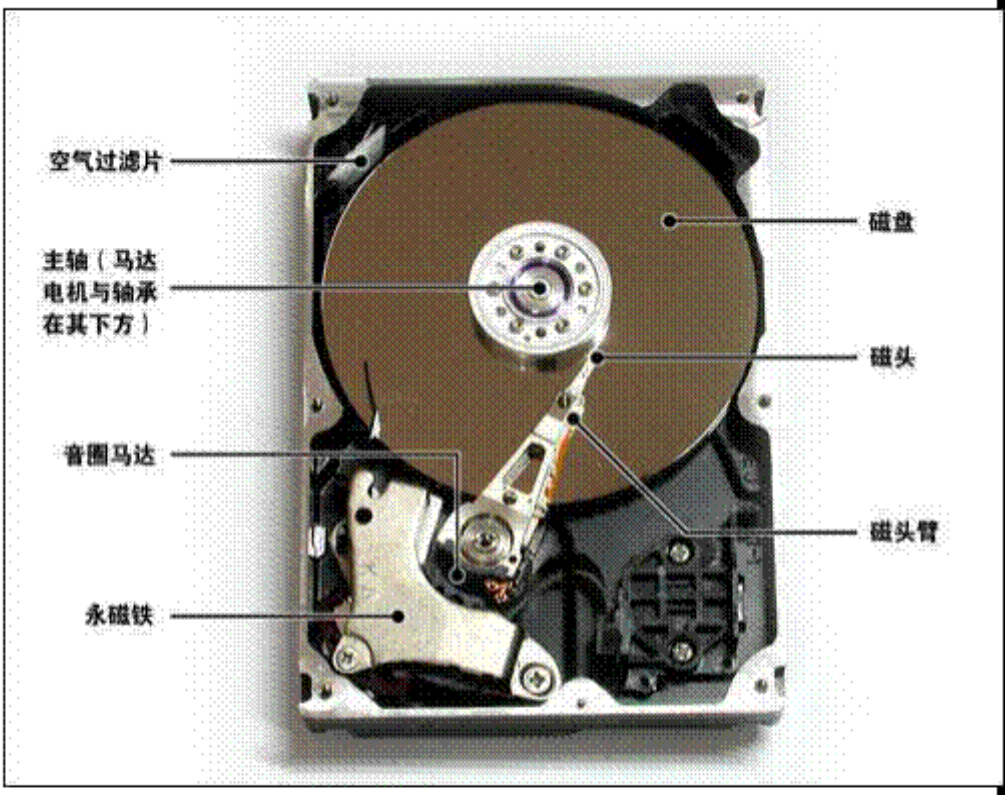
1.硬盘结构

文件系统结构，理解文件系统，要从文件储存说起。

硬盘结构：



硬盘相关专业术语：

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。**

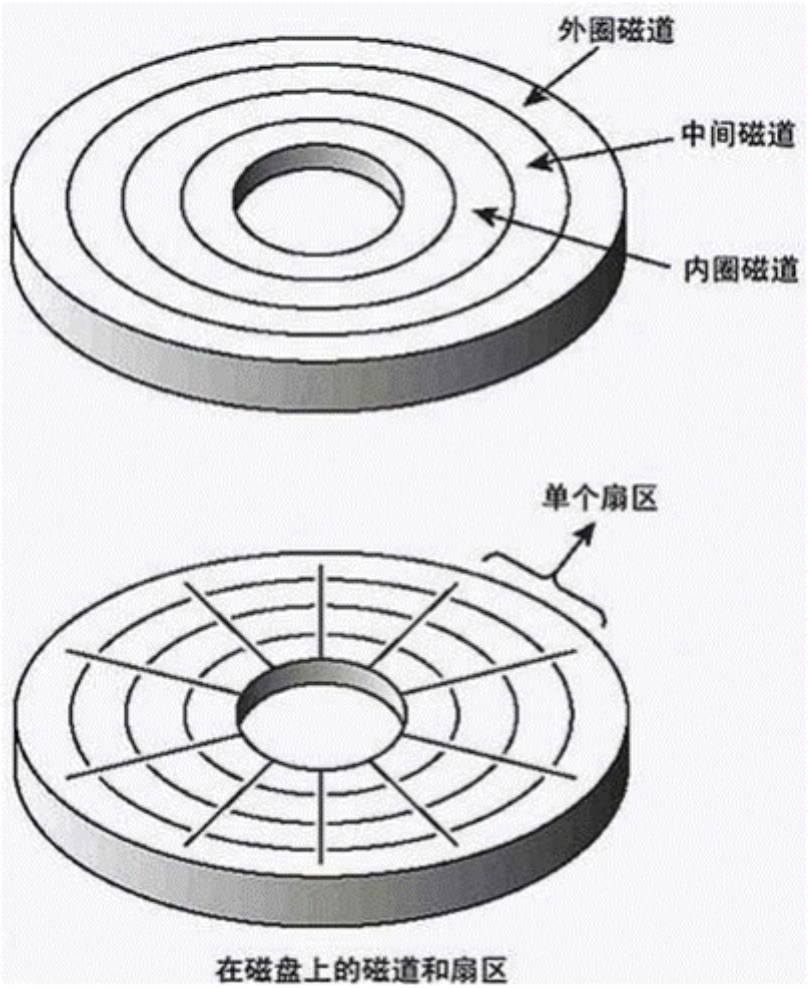
硬盘内的盘片有两个面，都可以储存数据，而硬盘内的盘片往往不止一张，常见的有两张，那么，两张盘片中相同位置的磁道，就组成一个**“柱面”**，盘片中有多少个磁道，就有多少个柱面。盘片两面都能存数据，要读取它，必须有磁头，所以，每一个面，都有一个磁头，一张盘片就有两个磁头。

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？？？

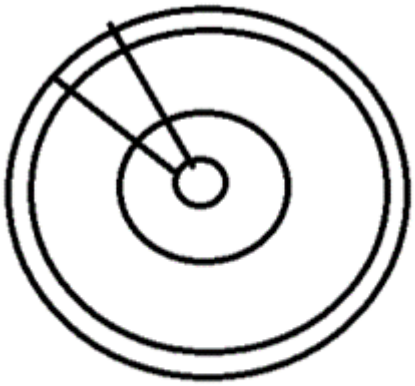
这种结构有问题：

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。

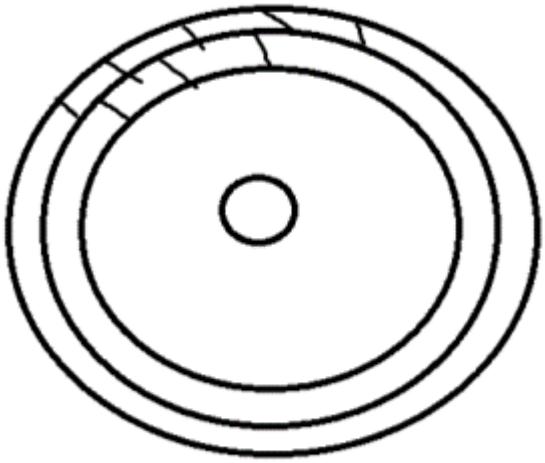
如果你是磁盘设计工程师，你打算怎么解决？你选择下面哪种方法？

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

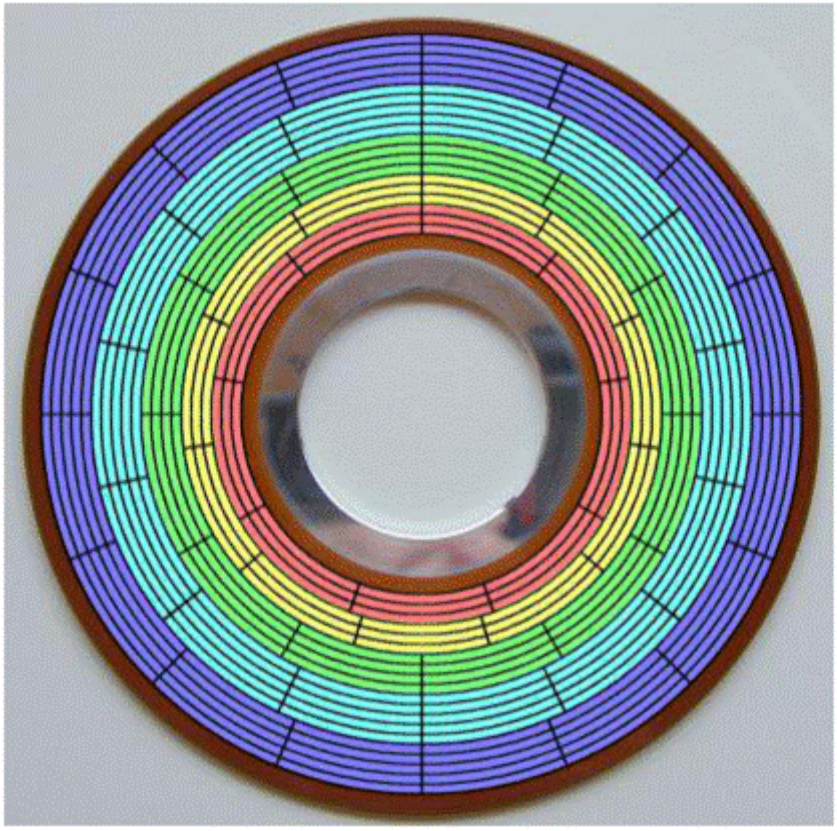


方法：2

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

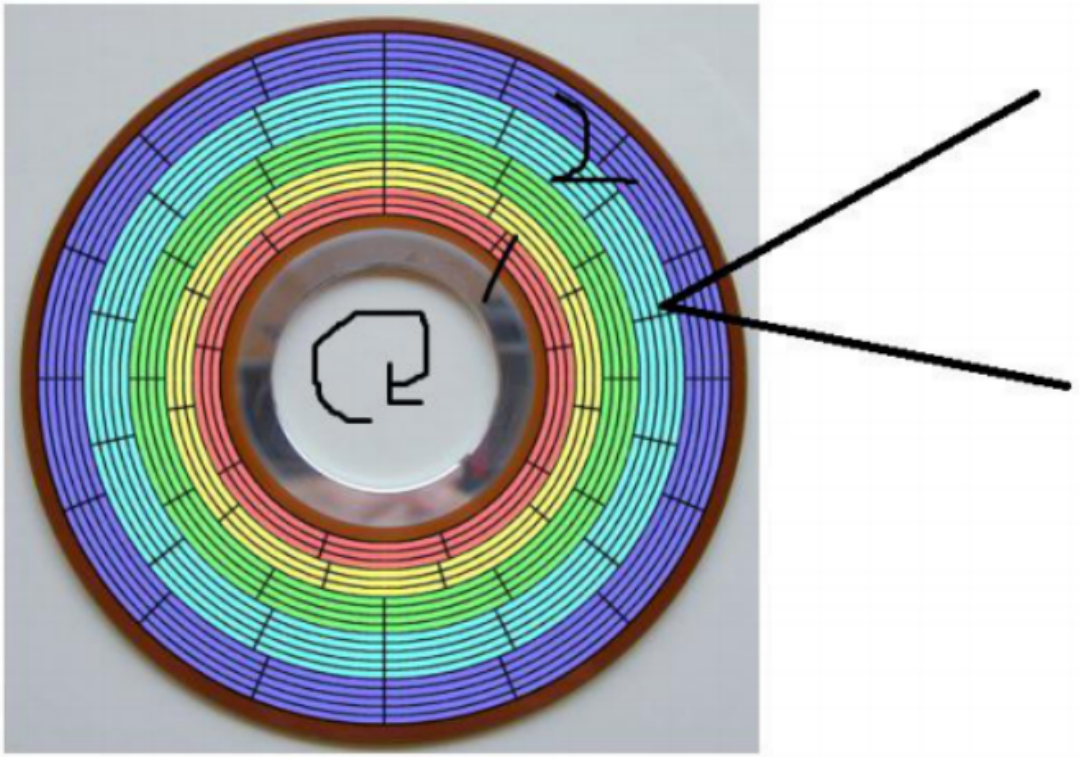
Zoned-bit recording（ZBR 区位记录）是一种物理优化硬盘存储空间的方法，此方法通过将更多 的扇区放到磁盘的外部磁道而获取更多存储空间。

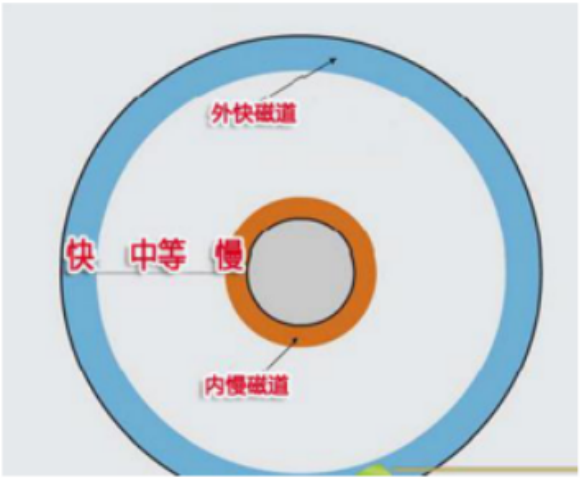
ZBR磁盘扇区结构示意图

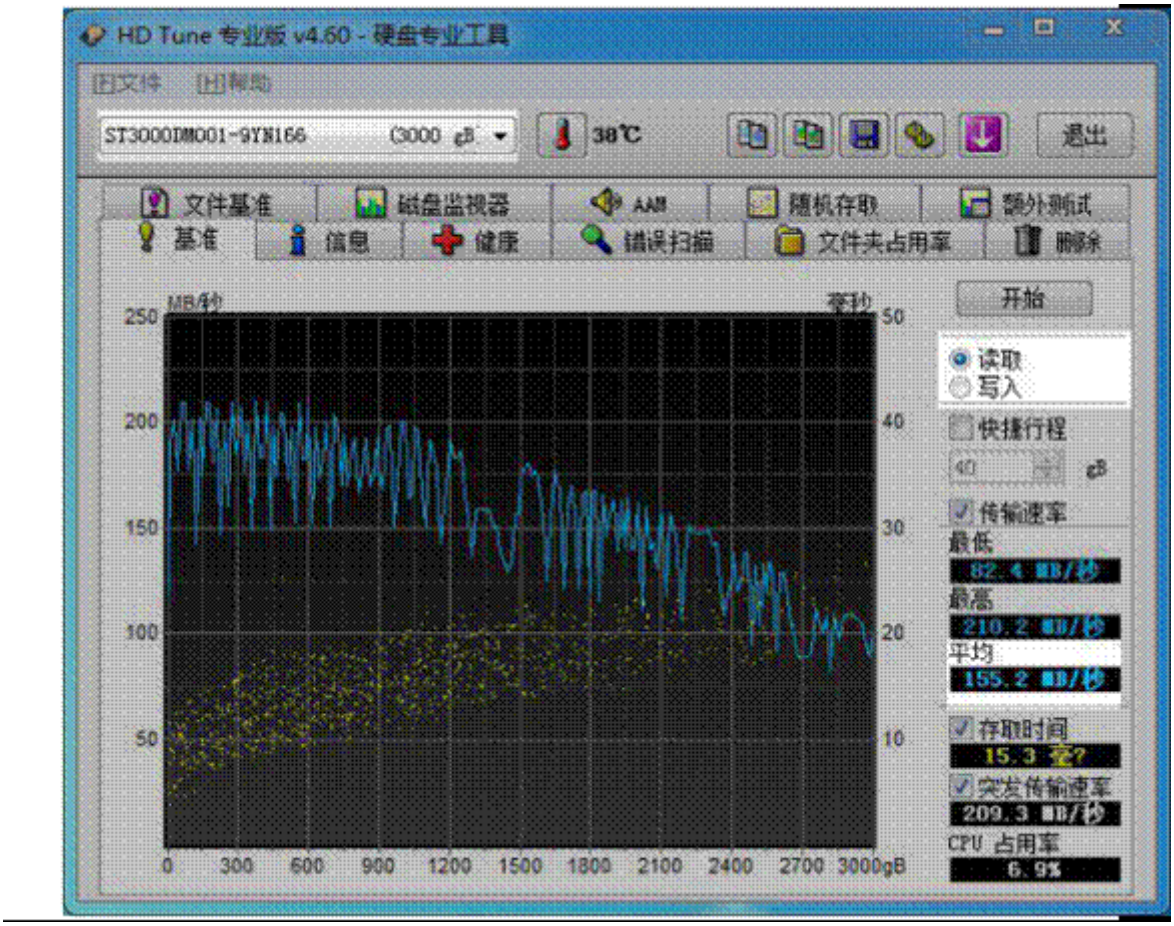


互动:从外面读数据快？还是从里面快?

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。




外快磁道
快
中等
慢
内慢磁道



互动：windows 安装系统的 C 盘或 Linux boot

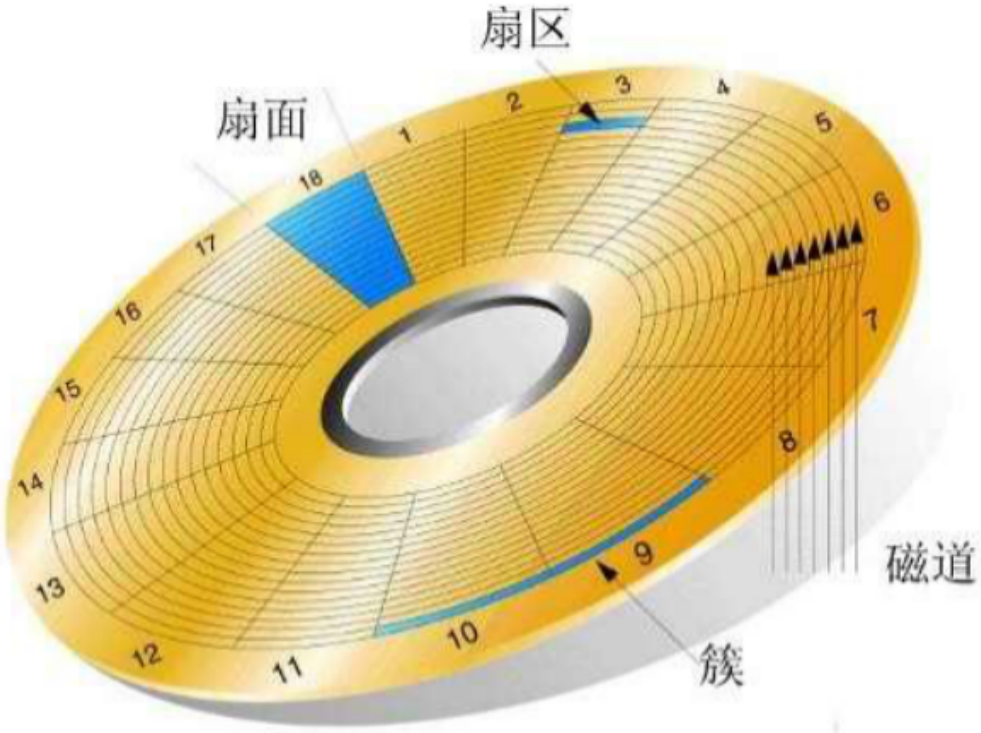
分区一般安装在磁盘最外面还是最里面?

windows ： C 盘安装最外，速度也是最快

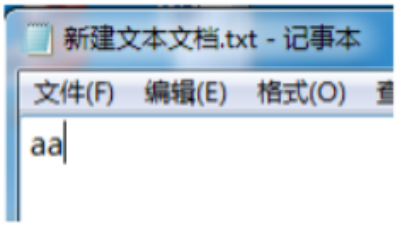
Linux ： boot 分区和 swap 分区，装最外面

磁盘写数据时，先从外面往里。

2.簇和block



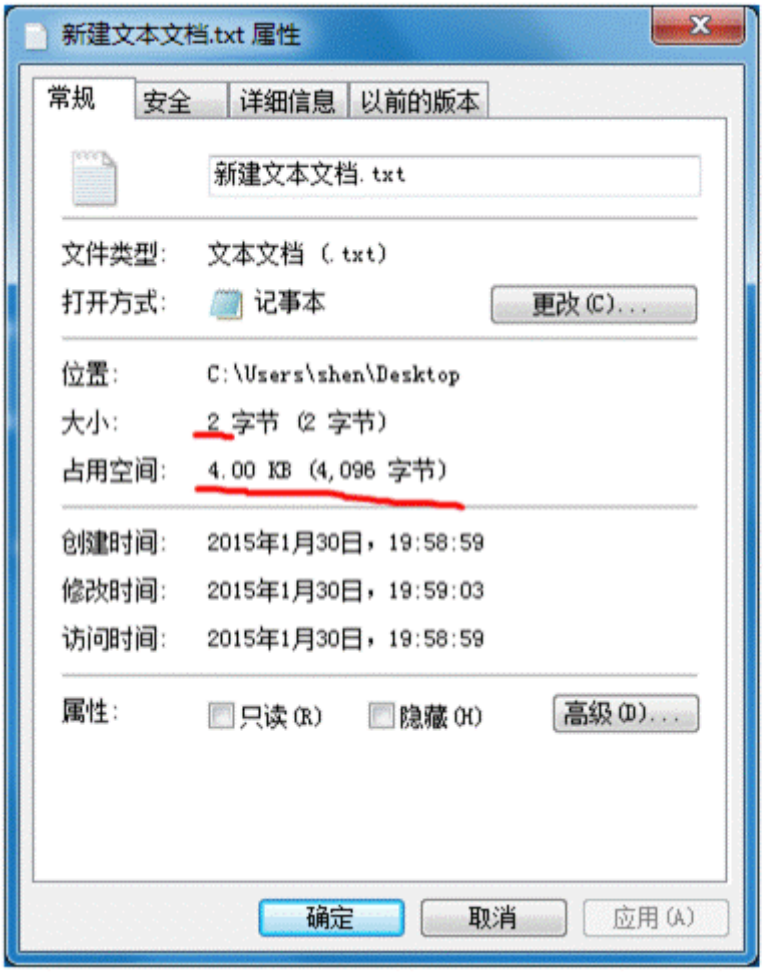
磁盘上的磁道、扇区和簇



windows上的簇类似于Linux系统中的block

可以理解为簇是Windows上的说法，而块是Linux上的说法

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

3.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

3.1、文件名

```
[root@exercise1 ~]# cp /etc/passwd /opt/a.txt

[root@exercise1 ~]# ls /opt/a.txt   #a.txt就是文件名
/opt/a.txt
```

3.2、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UserID**

***文件的GroupID**

***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@exercise1 ~]# stat /opt/a.txt
  文件："/opt/a.txt"
  大小：933         块：8          IO 块：4096   普通文件
设备：803h/2051d   Inode：16784851    硬链接：1
权限：(0644/-rw-r--r--)  Uid：(    0/    root)   Gid：(    0/    root)
环境：unconfined_u:object_r:usr_t:s0
最近访问：2022-02-02 09:37:44.598359017 +0800
最近更改：2022-02-02 09:37:44.598359017 +0800
最近改动：2022-02-02 09:37:44.598359017 +0800
创建时间：-
```

---

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 表的大小就会达到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。

例子：

例1：使用ls -i命令，可以看到文件名对应的inode号码

```
[root@exercise1 ~]# ls -i /opt/a.txt
16784851 /opt/a.txt
[root@exercise1 ~]#
```

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@exercise1 ~]# df -i
文件系统            Inode 已用(I) 可用(I) 已用(I)% 挂载点
/dev/sda3       9334272   39225 9295047        1% /
devtmpfs         122504     390  122114        1% /dev
tmpfs            124962       1  124961        1% /dev/shm
tmpfs            124962     477  124485        1% /run
tmpfs            124962      16  124946        1% /sys/fs/cgroup
/dev/sr0              0       0       0         - /mnt
/dev/sda1        102400     327  102073        1% /boot
tmpfs            124962       1  124961        1% /run/user/0
[root@exercise1 ~]#
```

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。

3.3、目录文件

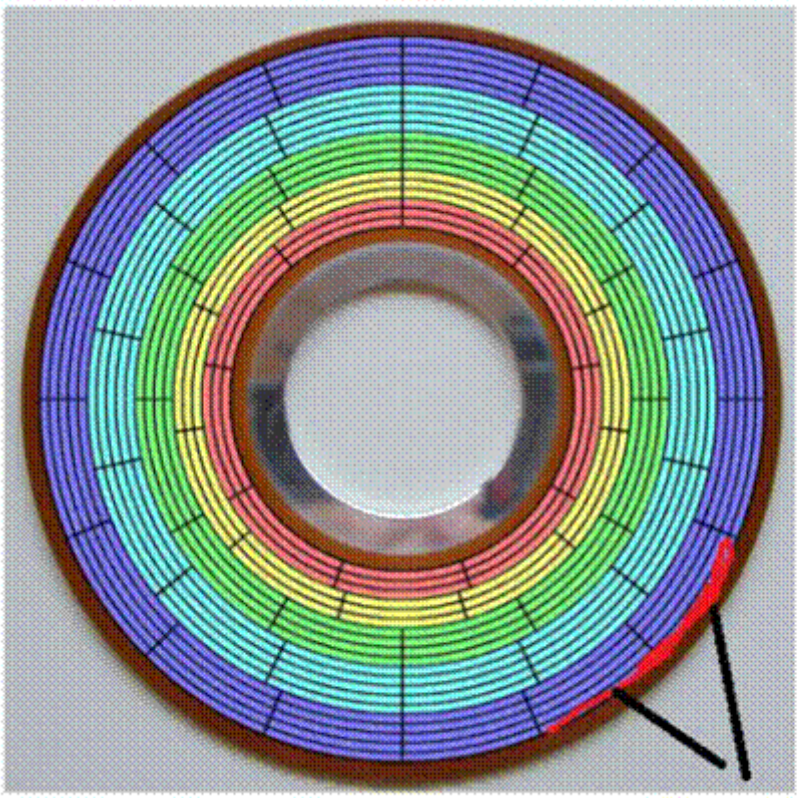
Unix/Linux系统中，目录（directory）也是一种文件。打开目录，实际上就是打开目录文件。目录文件的结构非常简单，就是一系列目录项的列表。每个目录项，由两部分组成：所包含文件的文件名，以及该文件名对应的inode号码。

例：ls-i命令列出整个目录文件，即文件名和inode号码

```
[root@exercise1 ~]# ls -i /opt/
16784851 a.txt  16784824 grub2.tar.gz
```

3.4、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



两个扇区称为：簇/block

在linux下中叫：block，在windows中叫：簇

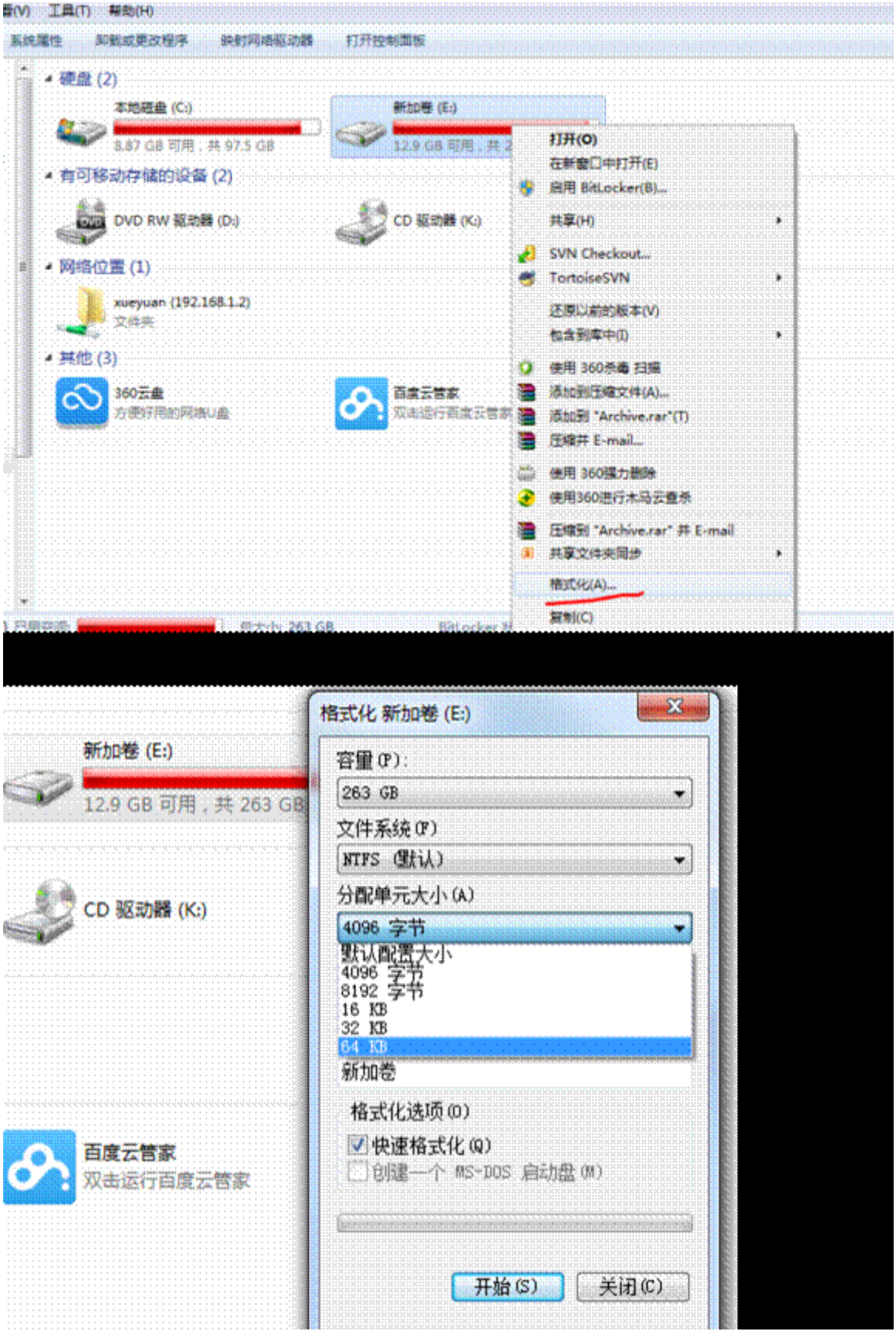
互动：为什么要有block，直接使用扇区可以吗？，

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。

情景：如果没有block？会怎么样？夜深人静，下了3.6G 的电影，一次读512KB，寻址次太多，太慢了。。。结果。。。你懂得。。。？
结果:电影看成了jpg

案例

例1：格式化修改磁盘，修改簇大小



实战：簇和 block 大小设定

簇和block调大：

优点：速度快，节约寻找时间，缺点：空间浪费

比如：2T硬盘,前1.5 T，使用4K簇，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@exercise1 ~]# stat /etc/passwd |grep IO
  大小：933         块：8           IO 块：4096    普通文件
[root@exercise1 ~]#
#block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。

3.5使用du命令查看文件大小

-a 列出所有的文件与目录容量
-s 列出目录容量(只列出目录以及当前目录下的内容，不会把下一级的内容列出)
-h 较易读的容量格式显示

```
[root@exercise1 opt]# du    #显示目录或者文件所占空间
3048    .
```

[root@exercise1 opt]# du -a #显示目录占用的磁盘空间大小，还要显示其下目录和文件占用磁盘空间的大小
3044 ./grub2.tar.gz
4 ./a.txt
3048 .

[root@exercise1 opt]# du -ah #查看所有文件和目录大小以较易读的容量格式显示

3.0M ./grub2.tar.gz

4.0K ./a.txt

3.0M .

[root@exercise1 opt]# du -sh #查看当前目录总共占的容量。而不单独列出各子项占用的容量

3.0M .

[root@exercise1 opt]# du -sh /* #查看指定文件大小

0 /bin

87M /boot

0 /dev

60M /etc

36K /home

0 /lib

0 /lib64

0 /media

4.3G /mnt

3.0M /opt

du: 无法访问"/proc/3635/task/3635/fd/4": 没有那个文件或目录

du: 无法访问"/proc/3635/task/3635/fdinfo/4": 没有那个文件或目录

du: 无法访问"/proc/3635/fd/4": 没有那个文件或目录

du: 无法访问"/proc/3635/fdinfo/4": 没有那个文件或目录

0 /proc

104K /root

13M /run

0 /sbin

0 /srv

0 /sys

4.0K /tmp

983M /usr

1.2G /var

4、文件的硬链接和软链接

4.1、Linux链接概念

Linux链接分两种，一种被称为硬链接（HardLink），另一种被称为软链接，即符号链接（SymbolicLink）。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
```

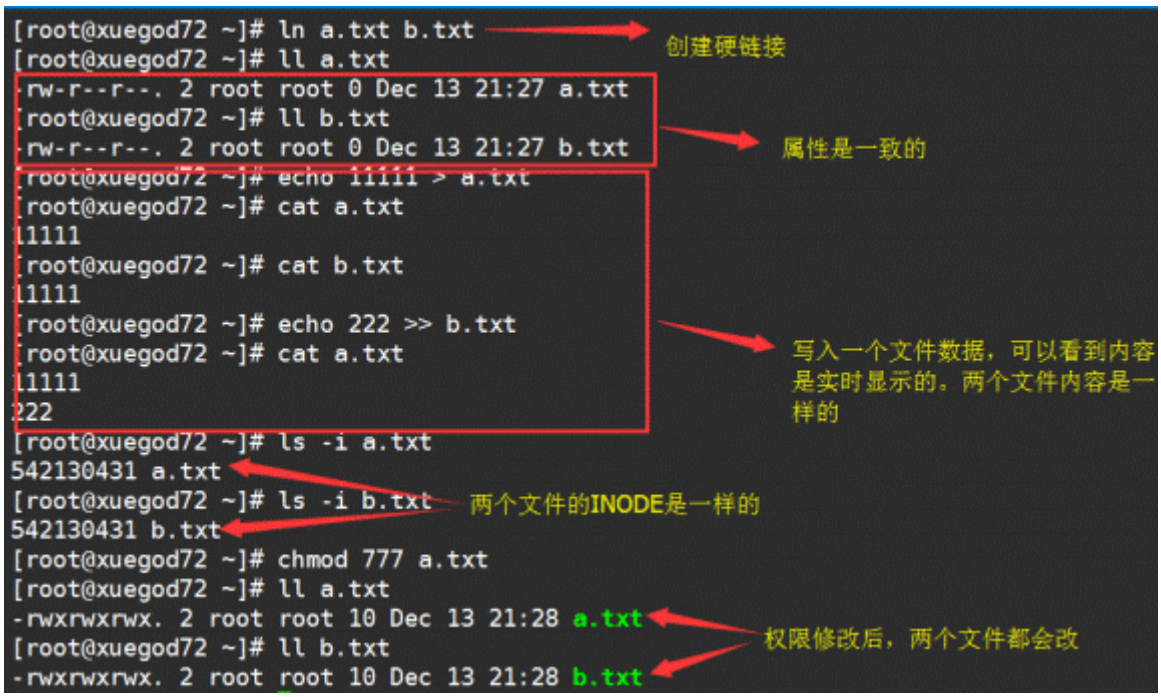
在Linux的文件系统中，保存在磁盘分区中的文件不管是什么类型都给它分配一个编号，称为索引节点号(InodeIndex)。
在Linux中，多个文件名可以指向同一索引节点。一般这种连接就是硬连接。
硬连接的作用是允许一个文件拥有多个有效路径名，这样用户就可以建立硬连接到重要文件，以防止“误删”的功能。
只删除一个连接并不影响索引节点本身和其它的连接，只有当最后一个连接被删除后，文件的数据块及目录的连接才会被释放。也就是说，文件真正删除的条件是与之相关的所有硬连接文件均被删除。

【软连接】：另外一种连接称之为符号连接（SymbolicLink），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

4.2、实战-1:ln命令创建硬链接

语法格式：ln　源文件　目标文件

```
[root@exercise1 opt]# 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

```
[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@xuegod72 boot]# ln b.txt /mnt/bb.txt
ln: failed to create hard link '/mnt/bb.txt' => 'b.txt': Invalid cross-device link
```

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

4.3、互动：为什么刚创建的一个目录，链接数就是2?

```
[root@exercise1 opt]# ln a.txt b.txt
```

[root@exercise1 opt]# ll
-rw-r--r--. 2 root root 0 2月 2 14:49 b.txt

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".."），所以隐藏目录.是一个链接，隐藏目录..是第二个链接

[root@exercise1 opt]# ll -id test/ #两个inode号是一样的
2453723 drwxr-xr-x 2 root root 6 2月 2 15:55 test/

[root@exercise1 opt]# ll -id test/.
2453723 drwxr-xr-x 2 root root 6 2月 2 15:55 test/.

4.4、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

```
[root@exercise1 opt]# cp /etc/passwd /opt/a.txt

[root@exercise1 opt]# ln -s /opt/a.txt /opt/a-link.txt

[root@exercise1 opt]# ll /opt/a-link.txt
lrwxrwxrwx. 1 root root 10 2月  2 20:04 /opt/a-link.txt -> /opt/a.txt

[root@exercise1 opt]# rm -rf /opt/a.txt

[root@exercise1 opt]# ll /opt/a-link.txt
lrwxrwxrwx. 1 root root 10 2月  2 20:04 /opt/a-link.txt -> /opt/a.txt
```

```
[root@exercise1 opt]# cp /etc/passwd /opt/a.txt
[root@exercise1 opt]# ln -s /opt/a.txt /opt/a-link.txt
[root@exercise1 opt]# ll /opt/a-link.txt
lrwxrwxrwx. 1 root root 10 2月    2 20:04 /opt/a-link.txt -> /opt/a.txt
[root@exercise1 opt]# rm -rf /opt/a.txt
[root@exercise1 opt]# ll /opt/a-link.txt
lrwxrwxrwx. 1 root root 10 2月    2 20:04 /opt/a-link.txt -> /opt/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@exercise1 opt]# ln -s /boot/grub /opt/grub-link
[root@exercise1 opt]# ll -d /opt/grub-link
lrwxrwxrwx. 1 root root 10 2月    2 20:08 /opt/grub-link
-> /boot/grub
```

```
[root@exercise1 opt]# ll -d /opt/grub-link
lrwxrwxrwx. 1 root root 10 2月    2 20:08 /opt/grub-link -> /boot/grub
```

能跨分区创建（源文件必须写绝对路径）

```
[root@exercise1 opt]# cd /boot/
[root@exercise1 opt]# ln -s ./grub/root/aaa
[root@exercise1 opt]# ll /root/aaa
lrwxrwxrwx. 1 root root 10 2月    2 20:08 /root/aaa-
>./grub  #报错了
```

5.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(vi) 每次修改完后，Inode 号都会变。

```
[root@exercise1 opt]# cp /etc/passwd /opt/passwd
[root@exercise1 opt]# ls -i /opt/passwd
16784824 /opt/passwd
[root@exercise1 opt]# vim /opt/passwd   #添加一些内容
[root@exercise1 opt]# ls -i /opt/passwd
16784851 /opt/passwd
```

这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。

注：当生成硬链接时，修改文件，不会改变inode号码

```
[root@home abc]# ll -i
总用量 8
67 -rwxr-xr-x 2 root root 4 7月  26 14:52 a.txt
67 -rwxr-xr-x 2 root root 4 7月  26 14:52 a.txt.bak
68 -rw-r--r-- 1 root root 0 7月  26 15:18 b.txt
[root@home abc]# vim a.txt
[root@home abc]# ll -i
总用量 8
67 -rwxr-xr-x 2 root root 40 7月  26 15:19 a.txt
67 -rwxr-xr-x 2 root root 40 7月  26 15:19 a.txt.bak
68 -rw-r--r-- 1 root root  0 7月  26 15:18 b.txt

```

总结：

硬链接与软链接的区别：

1.硬链接文件与源文件的inode节点号相同，而软链接文件的inode节点号与源文件不同

2.软链接能针对目录和跨分区创建软链接，硬链接只能针对文件 ==> 软链接可以跨文件系统，硬链接不可跨文件系统

3.删除软链接文件，对源文件及硬链接文件无任何影响;删除文件的硬链接文件，对源文件及软链接文件无任何影响;删除链接文件的源文件，对硬链接文件无影响，会导致其软链接失效;同时删除源文件及其硬链接文件，整个文件才会真正的删除

6.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

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@exercise1 opt]# df -h
文件系统            容量  已用  可用  已用% 挂载点
/dev/sda3            18G  2.3G   16G   13% /
devtmpfs            479M     0  479M    0% /dev
tmpfs               489M     0  489M    0% /dev/shm
tmpfs               489M   19M  470M    4% /run
tmpfs               489M     0  489M    0% /sys/fs/cgroup
/dev/sr0            4.3G  4.3G     0  100% /mnt
/dev/sda1           197M   97M  100M   50% /boot
tmpfs                98M     0   98M    0% /run/user/0
/dev/sdb1             5G    4G    1G   80% /opt/abc
```

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：

```
模拟情况
[root@base  ~]#df  -i
文件系统          Inode      已用(I)        可用(I)         已用(I)%       挂载点
/dev/sdb1      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
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。

格式化给文件系统的时候可以指定大小

```
[root@base  ~]# mkfs.xfs  -I  500000000000  /dev/sda1#可以指定大小
参数：
[-ibytes-per-inode][-I  inode-size]
```