1、Linux 最传统的磁盘文件系统 EXT2

每种操作系统能够使用的文件系统不相同。

win98 以前用 FAT、win2000 以后用 NTFS，Linux 用 EXT2，一般 windows 操作系统不能识别 Linux 的 EXT2

2、文件系统将文件的权限与属性放置在 inode 区块中，将实际数据放置在 data block 区块中，另外还有一个超级区块（superblock）会记录文件系统的整体信息，包括 inode 与 block 的总量、使用量、、剩余量等。

每个 inode 与 block 都有编号，每个文件都会占用一个 inode，inode 里有该文件数据放置的 block 号码，所以只要能找到文件的 inode，就能得知文件数据所放置的 block 号码，也就能读取文件数据了。

比如一个文件的属性与权限数据放置到 inode 4 号，这个 inode 记录了文件数据实际放置在 2,7,13,15 这几个 block 号码处，文件系统能一次性把所有 block 的内容都读取出来。这种数据存取的方法叫索引式文件系统。

3、U 盘（是闪存的一种）的文件系统是 FAT 格式。没有 inode，每个 block 都记录着此文件的下一个 block 号码，由此一个个的读取 block，如果 block 过于分散，则读取效能很差，也就需要“碎片整理”了，而 EXT2 这种索引式文件系统一般不需要碎片整理，但缺点是索引控件占用率比较大。

win7 用的是 NTFS，NTFS 与 EXT4 都是使用 extent 文件系统，只记录第一个 block 的位置和 extent 的 block 数。所以会存在碎片问题。

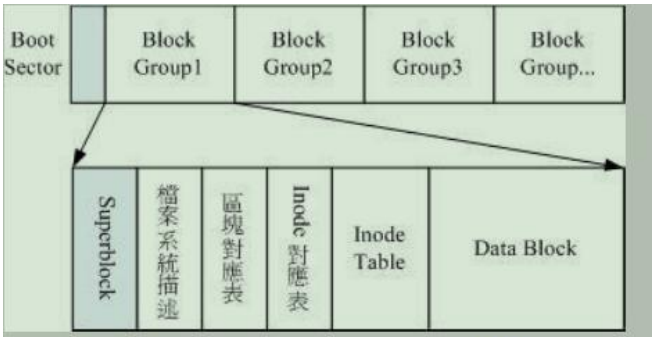


图 1.3.1、ext2 文件系统示意图(注 1)

EXT2 中支持的 block 大小有 1KB、2KB、4KB 三种。

Block 大小	1KB	2KB	4KB
最大单一档案限制	16GB	256GB	2TB
最大文件系统总容量	2TB	8TB	16TB

block 的限制还有：

1、block 的大小和数量在只有格式化才能改变

2、每个 block 最多只能放一个文件的数据

3、如果文件比较大，则会占用多个 block

4、如果文件小于一个 block，则这个 block 的剩余容量不能再被使用

因此如果文件都比较小，则最好不要选用 block 是 4KB 那种，不然会浪费较多空间

但也不是说 block 选的越小越好，因为越小代表大型文件占用 block 数量越多，inode 要记录的 block 也越多，影响读写性能。

inode 记录的文件数据至少有：

该档案的存取模式(read/write/excute)；
该档案的拥有者与群组(owner/group)；
该档案的容量；
该档案建立或状态改变的时间(ctime)；
最近一次的读取时间(atime)；
最近修改的时间(mtime)；
定义档案特性的旗标(flag)，如 SetUID...；
该档案真正内容的指向 (pointer)；

inode 数量大小在格式化时已固定，此外 inode 的特色还有：

每个 inode 大小均固定为 128 bytes；
每个档案都仅会占用一个 inode 而已；
承上，因此文件系统能够建立的档案数量与 inode 的数量有关；
系统读取档案时需要先找到 inode，并分析 inode 所记录的权限与用户是否符合，若符合才能够开始实际读取 block 的内容。

inode 只有 128bytes，而 inode 记录一个 block 号码占用 4bytes。因此如果全部是直接映射则 inode 不可能够用。
因此 inode 将记录 block 号码的区域定义为 12 个直接、一个间接、一个双间接和一个三间接记录区。
12 个直接能直接取得 block 号码，而一个间接就是连接到一个 block，这个 block 用来记录 block 号码（这样就能节省 inode 空间），通常太大的文件是采用间接的。
同理，双间接就是第一个 block 再指出另外的 block，这些 block 里记录着文件 block 号码。如此类推，三间接同理。

superblock 记录的信息：

block 与 inode 的总量；
未使用与已使用的 inode / block 数量；
block 与 inode 的大小 (block 为 1, 2, 4K，inode 为 128 bytes)；
filesystem 的挂载时间、最近一次写入数据的时间、最近一次检验磁盘 (fsck) 的时间等文件系统的相关信息；
一个 valid bit 数值，若此文件系统已被挂载，则 valid bit 为 0 ，若未被挂载，则 valid bit 为 1 。

superblock 大小为 1024bytes

文件系统描述 filesystem description

描述每个 block group 的开始与结束的 block 号码，说明 superblock、inodemap 等分别位于哪些 2block 号码之间

block bitmap 区块对照表
用来记录哪些 block 是空的，哪些是已被占用的。新增文件时，通过查询 block bitmap 可以快速的找到空闲 block 来放置文件。
inode bitmap 同理，是记录已经使用和未使用的 inode 号码。

每个区段与 superblock 的信息都可以用 dumpe2fs 指令来查询
df 指令显示当前挂载的装置

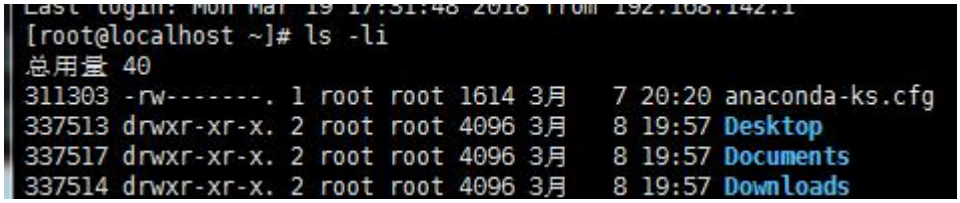
```
[root@localhost ~]# df
文件系统                 1K-块    已用    可用 已用% 挂载点
/dev/mapper/centos-root 10190136 3665292 6000556   38% /
devtmpfs                  922584       0  922584    0% /dev
tmpfs                     933512       0  933512    0% /dev/shm
tmpfs                     933512    9112  924400    1% /run
tmpfs                     933512       0  933512    0% /sys/fs/cgroup
/dev/sda1                 487652  114659  347393   25% /boot
/dev/mapper/centos-home  4908544   33280 4619264    1% /home
tmpfs                     186704       0  186704    0% /run/user/0
```

试一下查看根目录的信息：
。。。然而信息一大串，根本看不完

```
934319, 934325-934351, 934361-934383, 934392-934415, 934
-934543, 934545-934575, 934579-934607, 934656-934911, 93
  可用inode数: 232957, 232959-237568
Group 29: (Blocks 950272-983039)
  Block bitmap at 950272 (+0), Inode bitmap at 950273 (+
  Inode表位于 950274-950785 (+2)
  10655 free blocks, 4655 free inodes, 225 directories
  可用块数: 952880-952895, 952928-952943, 953024-953039,
3344-953359, 953376-953391, 953408-953423, 953434-953455
```

鸟哥输入上述指令后的输出显示：
（1）Group0 占用的 block 号码从 0 到 32767 号
（2）superblock 在第 0 号的 block 内
（3）文件系统描述在第 1 号的 block 号码中
（4）block bitmap 627　　inode bitmap 628
（5）inode table 629~1641
。。。

文件系统与目录树的关系：
在 ext2 文件系统下新建一个目录时，会分配一个 inode 和至少一个 block 给该目录（如果文件过多，会多给 block）。inode 记录目录的权限与属性，以及分配到的那块 block 号码。block 记录该目录下的文件名及其对应的 inode 号码。
显示目录下每个文件的 inode 号码（最左边的数字）：



新建一个文件时，ext2 会分配一个 inode 与相对于该档案大小的 block 数量给该文件。

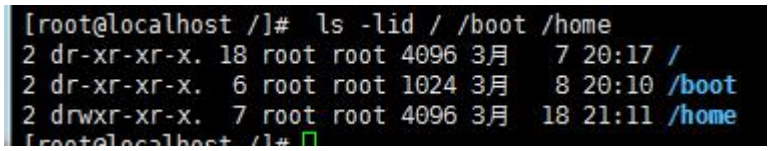
目录树由根目录开始读起，首先找到根目录 inode 内容，然后读取到根目录 block 内容，得知每个子文件夹的名字和 inode 号码，然后一层层往下读取。

新增文件过程：
1、确认用户对目录是否有 w、x 权限
2、根据 inode bitmap 找到空闲 inode 号码，将新文件属性、权限写入
3、根据 block bitmap 找到空闲 block 号码，将实际数据写入 block，更新 inode 指向此 block
4、将 inode、block 同步更新 inode bitmap,block bitmap，并更新 superblock

数据不一致状态 Inconsistent
万一新建文件时，系统在同步更新中介数据之前就被中断了，就会导致数据不一致，此时需要有：
日志式文件系统：
专门记录写入或修改文件时的步骤。万一数据记录发生问题，系统只需要检查日志记录区块，就能知道那个文件出了问题，而不需要整个 filesystem 检查，这个日志功能是 ext3 才加上去的

Linux 异步处理 asynchronously
系统加载一个文件到内存后，如果文件没有被更改过，则是 clean（干净）的，如果更改过，就设定为 dirty（脏）的，系统不定时将脏数据从内存写回磁盘。
正常关机时，系统会呼叫 sync 来将内存的数据写回磁盘内

将文件系统和目录树相结合成为挂载，挂载点一定是目录，它是进入文件系统的入口。
我们的 Linux 系统有三个挂载点 ： / , /boot, /home



filesystem 最顶层的目录的 inode 一般是 2 号，因此以上三个目录作为挂载点能进入三个不同的文件系统。
同一个文件系统中的某一个 inode 只会对应一个文件，因此可以观察 inode 判断不同文件名是否为同一文件。

磁盘与目录的容量：
磁盘的整体是数据在 superblock 中，各文件容量记录在 inode 中，如何显示它们的数据呢
df：显示文件系统的整体磁盘使用量，显示系统中所有 filesystem

df -h　:将结果以易读的容量格式显示出来

```
[root@localhost /]# df -h
文件系统                容量  已用  可用 已用% 挂载点
/dev/mapper/centos-root  9.8G  3.5G  5.8G   38% /
devtmpfs                 901M     0  901M    0% /dev
tmpfs                    912M     0  912M    0% /dev/shm
tmpfs                    912M  9.0M  903M    1% /run
tmpfs                    912M     0  912M    0% /sys/fs/cgroup
/dev/sda1                477M  112M  340M   25% /boot
/dev/mapper/centos-home  4.7G   33M  4.5G    1% /home
tmpfs                    183M     0  183M    0% /run/user/0
[root@localhost /]#
```

df -h 目录　显示目标目录所在的挂载点的磁盘容量使用情况

```
[root@localhost /]# df -h /etc
文件系统                容量  已用  可用 已用% 挂载点
/dev/mapper/centos-root  9.8G  3.5G  5.8G   38% /
[root@localhost /]# df -h /home/Ray
文件系统                容量  已用  可用 已用% 挂载点
/dev/mapper/centos-home  4.7G   33M  4.5G    1% /home
```

df -ih　显示各个文件系统中 inode 使用情况

```
[root@localhost /]# df -ih
文件系统                Inode 已用(I) 可用(I) 已用(I)% 挂载点
/dev/mapper/centos-root  640K    121K    520K      19% /
devtmpfs                 226K     399    225K       1% /dev
tmpfs                    228K       1    228K       1% /dev/shm
tmpfs                    228K     563    228K       1% /run
tmpfs                    228K      16    228K       1% /sys/fs/cgroup
/dev/sda1                126K     336    125K       1% /boot
/dev/mapper/centos-home  313K     445    313K       1% /home
tmpfs                    228K       1    228K       1% /run/user/0
```

du　：列出当前目录下的所有文件容量（单位为 1KB），实际只显示目录容量

du -a　：目录、文件容量都显示出来

再谈连接文件 link。有两种：

1、类似快捷方式，可以快速链接到目标文件/目录

2、通过文件系统 inode 连结来产生新文件名（not 新文件），成为实体链接

hardlink(实体链接)就是某个目录下新增的一个文件名链接到某个 inode 号码的关联记录。(这个关联写在目录的 block 里)

创建实体链接

```
[root@localhost ~]# ln /etc/crontab
```

```
[root@localhost ~]# ll -i /etc/crontab /root/crontab
197081 -rw-r--r--. 2 root root 451 6月  10 2014 /etc/crontab
197081 -rw-r--r--. 2 root root 451 6月  10 2014 /root/crontab
```

可见两个文件名都连结到同一个 inode 上 数字 2 代表有 2 个文件名链接到这个 inode 号码上。hard link 只是在某个目录的 block 里多写入了一个关联数据，既不会增加 inode 也不会曾佳佳

block 数量（除非这个 block 满了）
但是 hardlink 不能跨 filesystem，比如/home/Ray 目录下就不能建根目录下文件的实体链接，且不支持目录的实体链接

符号链接，它是建立一个新的文件，这个文件让数据的读取指向他 link 的那个文件的文件名。因此如果源文件被删除后，此符号链接会打开不了
建立符号链接：ln -s 源文件名 符号链接文件名
ln -s /etc/crontab crontab2

/etc/crontab 不 /root/crontab 指向同一个文件，如果我删除了 /etc/crontab 这个文件，该删除的动作其实只是将 /etc 目录下关于 crontab 的关连数据拿掉而已， crontab 所在的 inode 和 block 其实都没有被变动

磁盘的分割、格式化、检验和挂载（这个一看起来就很麻烦的样子）
如果想新增一个硬盘，要做什么：
1、分割硬盘，从而建立可用的 partition
2、格式化 partition，建立系统可用的 filesystem
3、建立挂载点（即目录），并将 filesystem 挂载上来
还要考虑诸如 磁盘分区槽 partition 要多大，inode、block 数量如何规划等问题

查阅目前系统内所有的 partition
df

```
[root@localhost ~]# df
文件系统                 1K-块    已用    可用 已用% 挂载点
/dev/mapper/centos-root 10190136 3665708 6000140   38% /
devtmpfs                  922584       0  922584    0% /dev
tmpfs                     933512       0  933512    0% /dev/shm
tmpfs                     933512    9112  924400    1% /run
tmpfs                     933512       0  933512    0% /sys/fs/cgroup
/dev/mapper/centos-home  4908544   33280 4619264    1% /home
/dev/sda1                 487652  114659  347393   25% /boot
tmpfs                     186704       0  186704    0% /run/user/0
```

然后进入 centos-root 这个分区里，练习新增分割区和删除分割区：
新增分割区：
新增之前：

```
命令(输入 m 获取帮助)：p

磁盘 /dev/mapper/centos-root：10.7 GB, 10737418240 字节, 20971520 个扇区
Units = 扇区 of 1 * 512 = 512 bytes
扇区大小(逻辑/物理)：512 字节 / 512 字节
I/O 大小(最小/最佳)：512 字节 / 512 字节
磁盘标签类型：dos
磁盘标识符：0x5c76dbe2

                设备 Boot      Start         End      Blocks   Id  System
```

没有分割区

```
命令(输入 m 获取帮助)：n
Partition type:
   p   primary (0 primary, 0 extended, 4 free)
   e   extended
Select (default p): p
分区号 (1-4，默认 1)：4
起始 扇区 (2048-20971519，默认为 2048)：
将使用默认值 2048
Last 扇区，+扇区 or +size{K,M,G} (2048-20971519，默认为 20971519)：+512M
分区 4 已设置为 Linux 类型，大小设为 512 MiB
```

新增分割区的命令为 n，然后要选择分区的类型，我们先新增一个 primary 类型的分区，分区号设置为 4，起始扇区取默认分配的起始扇区，然后要设置分区大小，我们设置为 512MBytes，如最后一行所示，分区新建完毕

```
命令(输入 m 获取帮助)：p

磁盘 /dev/mapper/centos-root：10.7 GB，10737418240 字节，20971520 个扇区
Units = 扇区 of 1 * 512 = 512 bytes
扇区大小(逻辑/物理)：512 字节 / 512 字节
I/O 大小(最小/最佳)：512 字节 / 512 字节
磁盘标签类型：dos
磁盘标识符：0x5c76dbe2

                设备 Boot      Start         End      Blocks   Id  System
/dev/mapper/centos-root4            2048     1050623      524288   83  Linux
```

看到出现了一个分割区，名字为 centos-root4

现在再新增一个 extended 类型的分区：

```
命令(输入 m 获取帮助)：n
Partition type:
   p   primary (1 primary, 0 extended, 3 free)
   e   extended
Select (default p): e
分区号 (1-3，默认 1)：1
起始 扇区 (1050624-20971519，默认为 1050624)：
将使用默认值 1050624
Last 扇区，+扇区 or +size{K,M,G} (1050624-20971519，默认为 20971519)：
将使用默认值 20971519
分区 1 已设置为 Extended 类型，大小设为 9.5 GiB
```

延伸分区最好包含所有为分割的磁盘区间，因此这里把剩余所有磁柱都分配给它了

```
命令(输入 m 获取帮助)：p

磁盘 /dev/mapper/centos-root：10.7 GB，10737418240 字节，20971520 个扇区
Units = 扇区 of 1 * 512 = 512 bytes
扇区大小(逻辑/物理)：512 字节 / 512 字节
I/O 大小(最小/最佳)：512 字节 / 512 字节
磁盘标签类型：dos
磁盘标识符：0x5c76dbe2

                设备 Boot      Start         End      Blocks   Id  System
/dev/mapper/centos-root1         1050624    20971519     9960448    5  Extended
/dev/mapper/centos-root4            2048     1050623      524288   83  Linux
```

又尝试新增一个 2GB 的分割区：

```
命令(输入 m 获取帮助)：n
Partition type:
   p   primary (1 primary, 1 extended, 2 free)
   l   logical (numbered from 5)
Select (default p): l
添加逻辑分区 5
起始 扇区 (1052672-20971519，默认为 1052672)：
将使用默认值 1052672
Last 扇区，+扇区 or +size{K,M,G} (1052672-20971519，默认为 20971519)：+2048M
分区 5 已设置为 Linux 类型，大小设为 2 GiB
```

可见，出现了 logical 类型的分割区选项，而且不能再新建 primary 类型分割区了，因为没有空闲磁柱。而之前建的 extended 分区的空间里可以新建 logical 类型分割区

```
磁盘标识符：0x5c76dbe2

                  设备 Boot      Start         End      Blocks   Id  System
/dev/mapper/centos-root1         1050624    20971519     9960448    5  Extended
/dev/mapper/centos-root4            2048     1050623      524288   83  Linux
/dev/mapper/centos-root5         1052672     5246975     2097152   83  Linux
```

centos-root5 就是这次新建的分割区

下面尝试删除分区槽：

```
命令(输入 m 获取帮助)：d
分区号 (1,4,5，默认 5)：
```

可见系统已经知道这个 partition 里有 1、4、5 这三个分区了，这里尝试删除 1 号分区(extended)

```
                  设备 Boot      Start         End      Blocks   Id  System
/dev/mapper/centos-root4            2048     1050623      524288   83  Linux
```

看到依附在 1 号延伸分区上的 5 号 logical 分区也不见了

再输入 d，系统知道现在只有一个分区了，所以直接删除，都不用我们选择哪一个分区来删除

再重新建一个 primary 分区，重启，卧槽，分区又不见了，也就是说新增的分区并没有写进文件系统中

```
磁盘标签类型：dos
磁盘标识符：0x775e2a80

                  设备 Boot      Start         End      Blocks   Id  System
```

即分割完毕后必须进行文件系统的格式化 ： mkfs 指令

如果新建完分区后直接退出的话，再次进入 fdisk 并 p 查看时又是空的，因为还有先执行 w 命令

先执行 w 命令：

```
命令(输入 m 获取帮助)：w
The partition table has been altered!

Calling ioctl() to re-read partition table.

WARNING: Re-reading the partition table failed with error 22: 无效的参数.
The kernel still uses the old table. The new table will be used at
the next reboot or after you run partprobe(8) or kpartx(8)
正在同步磁盘。
```

此时再进入 fdisk 查看时，分区 2 还在，而且重启之后这个分区也还在

```
                设备 Boot      Start         End      Blocks   Id  System
/dev/mapper/centos-root2            2048     1050623      524288   83  Linux
```

查看当前分区信息：

```
[root@localhost mapper]# fdisk -l

磁盘 /dev/sda：21.5 GB, 21474836480 字节，41943040 个扇区
Units = 扇区 of 1 * 512 = 512 bytes
扇区大小(逻辑/物理)：512 字节 / 512 字节
I/O 大小(最小/最佳)：512 字节 / 512 字节
磁盘标签类型：dos
磁盘标识符：0x000014fa

   设备 Boot      Start         End      Blocks   Id  System
/dev/sda1   *        2048     1026047      512000   83  Linux
/dev/sda2         1026048    34283519    16628736   8e  Linux LVM
```

```
磁盘 /dev/mapper/centos-root：10.7 GB, 10737418240 字节，20971520 个扇区
Units = 扇区 of 1 * 512 = 512 bytes
扇区大小(逻辑/物理)：512 字节 / 512 字节
I/O 大小(最小/最佳)：512 字节 / 512 字节
磁盘标签类型：dos
磁盘标识符：0x2df83d93

                设备 Boot      Start         End      Blocks   Id  System
/dev/mapper/centos-root2            2048     1050623      524288   83  Linux
```

```
磁盘 /dev/mapper/centos-swap：1044 MB, 1044381696 字节，2039808 个扇区
Units = 扇区 of 1 * 512 = 512 bytes
扇区大小(逻辑/物理)：512 字节 / 512 字节
I/O 大小(最小/最佳)：512 字节 / 512 字节
```

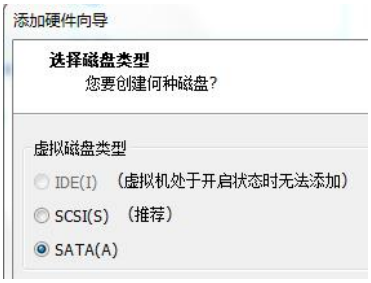
```
磁盘 /dev/mapper/centos-home：5242 MB, 5242880000 字节，10240000 个扇区
Units = 扇区 of 1 * 512 = 512 bytes
扇区大小(逻辑/物理)：512 字节 / 512 字节
I/O 大小(最小/最佳)：512 字节 / 512 字节

[root@localhost mapper]#
```

然而还是找不到新的分区文件，唉不管了

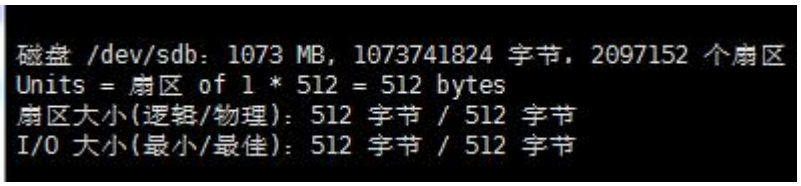
不行，再尝试！

https://www.cnblogs.com/chenmh/p/5096592.html

看到了添加新硬盘，就这个了

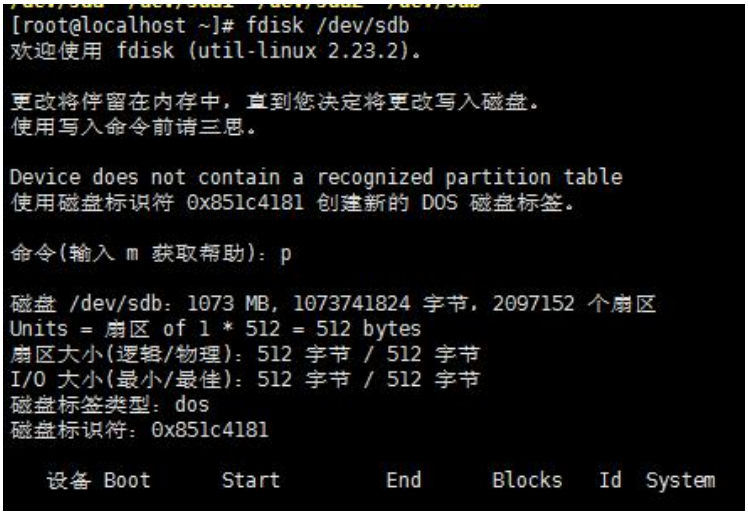


查看分区情况，发现多了这个



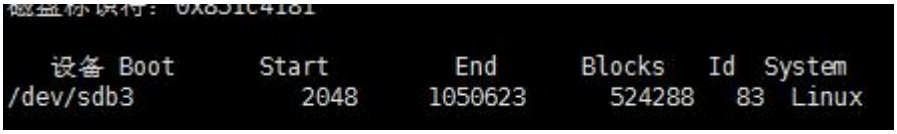
而 sda 是 20G，说明 sda 是原有硬盘，sdb 是刚刚新增的硬盘

好了，在这块新硬盘上新增分区：



可见，原本是没有分区的

新增一个 3 号分区，大小为 512M



w 命令保存

找到了新分区的文件/dev/sdb3，基本成功一半了

将制作出来的分区 3 格式化成 ext3 文件系统

但是很烦，我输入 df 里面还是没有 centos-root2，只有 centos-root

而鸟哥是有 hdv6（他新增的分区）的，自然我就没办法格式化新分区，因为根本就系统找不到

那个新的分区
鸟哥的格式化： mkfs -t ext3 /dev/hdc6
我的格式化应该是： mkfs -t ext3 /dev/mapper/centos-root2

以上的是失败的尝试，以下继续在新硬盘上，格式化分区 3：

```
[root@localhost ~]# mkfs -t ext3 /dev/sdb3
mke2fs 1.42.9 (28-Dec-2013)
文件系统标签=
OS type: Linux
块大小=4096 (log=2)
分块大小=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
32768 inodes, 131072 blocks
6553 blocks (5.00%) reserved for the super user
第一个数据块=0
Maximum filesystem blocks=134217728
4 block groups
32768 blocks per group, 32768 fragments per group
8192 inodes per group
Superblock backups stored on blocks:
        32768, 98304

Allocating group tables: 完成
正在写入inode表: 完成
Creating journal (4096 blocks): 完成
Writing superblocks and filesystem accounting information: 完成
```

没有报错，舒畅

磁盘检验：fsck、badblocks
宕机等问题会导致文件系统的错乱，此时可以使用 fsck 命令：
fsck -C -f -t ext3 /dev/hdc6
-C：直方图显示检查进度；
-f：强制检查
-t：后面接指定的文件系统

注意，执行 fsck 时，被检查的 partition 不能挂载在系统上，尴尬的是，我好像只设置了 3 个 partition：boot、home 和 root，都挂载了

磁盘的挂载与卸除
挂载点是目录，此目录是进入磁盘分区槽（文件系统）的入口。挂载要注意：
1、单一文件系统不应被重复挂载正在不同的挂载点（目录）
2、单一目录不应重复挂载多个文件系统
3、目录应是空目录

挂载指令： mount
mount 装置文件名 挂载点
用预设的方法，将新建立的/dev/sdb3 挂载到/mnt/sd3_entrance 上面
mkdir /mnt/sd3_entrance

```
[root@localhost ~]# mkdir /mnt/sdb3_entrance
[root@localhost ~]# mount /dev/sdb3 /mnt/sdb3_entrance
[root@localhost ~]# df
文件系统                 1K-块    已用    可用 已用% 挂载点
/dev/mapper/centos-root 10190136 3666576 5999272   38% /
devtmpfs                  922584       0  922584    0% /dev
tmpfs                     933512       0  933512    0% /dev/shm
tmpfs                     933512    9156  924356    1% /run
tmpfs                     933512       0  933512    0% /sys/fs/cgroup
/dev/sda1                 487652  114659  347393   25% /boot
/dev/mapper/centos-home  4908544   33280 4619264    1% /home
tmpfs                     186704       0  186704    0% /run/user/0
/dev/sdb3                 499656     416  473028    1% /mnt/sdb3_entrance
```

最后一行，已经挂载成功。

检查已挂载的文件系统状况：mount -l

输出有点多，截取一部分

```
/dev/sda1 on /boot type ext3 (rw,relatime,seclabel,data=ordered)
/dev/mapper/centos-home on /home type ext3 (rw,relatime,seclabel,data=ordered)
sunrpc on /var/lib/nfs/rpc_pipefs type rpc_pipefs (rw,relatime)
tmpfs on /run/user/0 type tmpfs (rw,nosuid,nodev,relatime,seclabel,size=186704k,mode=
/dev/sdb3 on /mnt/sdb3_entrance type ext3 (rw,relatime,seclabel,data=ordered)
```

可见/dev/sdb3 是 ext3 类型的文件系统

重新挂载根目录与挂载不特定目录

根目录很重要，不能被卸载，但如果想要改变挂载参数，或者根目录变成只读状态，此时可以重启或重新挂载根目录：

mount -o remount,rw,auto /

以上指令将 / 重新挂载，并加入参数为 rw 和 auto，好像也没什么用

也可以用 mount 指令将某个目录挂载到另一个目录去

```
[root@localhost /]# mkdir /mnt/home
[root@localhost /]# mount --bind /hom /mnt/home
mount: 特殊设备 /hom 不存在
[root@localhost /]# mount --bind /home /mnt/home
[root@localhost /]# ls -lid /home/ /mnt/home
2 drwxr-xr-x. 7 root root 4096 3月  18 21:11 /home/
2 drwxr-xr-x. 7 root root 4096 3月  18 21:11 /mnt/home
```

可见两个目录的 inode 号相同，通过 mount --bind，将目录/home 挂载到目录/mnt/home 上，从此进入/mnt/home 就是进入/home

```
[root@localhost /]# cd /mnt/home
[root@localhost home]# touch newFile
```

umount 将已经挂载的文件系统卸载（可以用 df 或 mount -l 查看是否还存在于目录树中）

可以用 mount +文件系统安装点或 mount + 挂载点删除，即

mount /dev/sdb3 OR mount /mnt/sdb3_entrance

一般用挂载点

```
[root@localhost /]# mount /dev/sdb3 /mnt/sdb3_entrance/
[root@localhost /]# df
文件系统                 1K-块    已用    可用 已用% 挂载点
/dev/mapper/centos-root 10190136 3666916 5998932   38% /
devtmpfs                  922584       0  922584    0% /dev
tmpfs                     933512       0  933512    0% /dev/shm
tmpfs                     933512    9124  924388    1% /run
tmpfs                     933512       0  933512    0% /sys/fs/cgroup
/dev/sda1                 487652  114659  347393   25% /boot
/dev/mapper/centos-home  4908544   33280 4619264    1% /home
tmpfs                     186704       0  186704    0% /run/user/0
/dev/sdb3                 499656     416  473028    1% /mnt/sdb3_entrance
[root@localhost /]# umount /mnt/sdb3_entrance/
[root@localhost /]# df
文件系统                 1K-块    已用    可用 已用% 挂载点
/dev/mapper/centos-root 10190136 3666916 5998932   38% /
devtmpfs                  922584       0  922584    0% /dev
tmpfs                     933512       0  933512    0% /dev/shm
tmpfs                     933512    9124  924388    1% /run
tmpfs                     933512       0  933512    0% /sys/fs/cgroup
/dev/sda1                 487652  114659  347393   25% /boot
/dev/mapper/centos-home  4908544   33280 4619264    1% /home
tmpfs                     186704       0  186704    0% /run/user/0
```

如果正处于挂载点目录内，是卸载不了的，可以先退回根目录再卸载

调整硬盘参数：mknod 等，但是比较难

设定开机挂载：

手动 mount 挂载不是很人性化，事实上，mount 挂载的文件系统在重启之后就自动卸载了

```
[root@localhost ~]# cat /etc/fstab

#
# /etc/fstab
# Created by anaconda on Wed Mar  7 20:00:25 2018
#
# Accessible filesystems, by reference, are maintained under '/dev/disk'
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info
#
/dev/mapper/centos-root /                       ext3    defaults        1 1
UUID=4ef668b5-0bc1-40a2-b426-715aff59c8ec /boot                   ext3    defaults        1 2
/dev/mapper/centos-home /home                   ext3    defaults        1 2
/dev/mapper/centos-swap swap                    swap    defaults        0 0
```

第一栏：Label，文件系统的装置文件名

第二栏：挂载点

第三栏：磁盘分区的文件系统类型

第四栏：文件系统参数，defaults 表示同时具有 rw, suid, dev, exec, auto, nouser, async 等参数，一般设置为 defaults 即可

第五栏：能否被 dump 备份指令作用，0：不作 dump 备份；1：每天 dump；2：不定期 dump

第六栏：是否以 fsck 检验扇区。开机时系统默认以 fsck 检验 filesystem 是否 clean。0：不检验，1：优先检验，2：检验。一般根目录 filesystem 设为 1，其他 filesystem 设为 2

每次开机都将/dev/sdb3 挂载到/mnt/sdb3_entrance 上：

/etc/fstab 是开机时的配置文件

nano /etc/fstab

打开文件并在最后加上一行

```
#
/dev/mapper/centos-root /                       ext3    defaults        1 1
UUID=4ef668b5-0bc1-40a2-b426-715aff59c8ec /boot                   ext3    defaults        1 2
/dev/mapper/centos-home /home                   ext3    defaults        1 2
/dev/mapper/centos-swap swap                    swap    defaults        0 0
/dev/sdb3 /mnt/sdb3_entrance ext3 defaults 1 2
```

然后再把 sdb3 卸载，重启

```
[root@localhost ~]# df
文件系统                 1K-块    已用    可用 已用% 挂载点
/dev/mapper/centos-root 10190136 3667248 5998600   38% /
devtmpfs                922584       0  922584    0% /dev
tmpfs                   933512       0  933512    0% /dev/shm
tmpfs                   933512    9136  924376    1% /run
tmpfs                   933512       0  933512    0% /sys/fs/cgroup
/dev/sdb3               499656     416  473028    1% /mnt/sdb3_entrance
/dev/sda1               487652  114690  347362   25% /boot
/dev/mapper/centos-home 4908544   33280 4619264    1% /home
tmpfs                   186704       0  186704    0% /run/user/0
```

成功。其实不用重启，mount -a 然后再 df，如果看见 sdb3 挂载了就相当于那条加上去的语句没有语法错误了

最后再将那一行注释掉（前面加上#）

建立大文件以制作 loop 装置文件：

如果当初在分割时，只分割出一个根目录，且没有多余的容量进额外的分割。但根目录剩余容量还很大，此时可以挂载一个大文件，相当于多了一个分割槽

现在在/home 目录下建立一个 512MB 的大文件，将其格式化并挂载

1、建立大文件：dd 指令建立空文件

if (input file) /dev/zero 是输出 0 的装置

of (output file) 将一堆 0 写进输出文件中

bs (block size)

count (block count)

```
[root@localhost ~]# dd if=/dev/zero of=/home/loopdev bs=1M count=512
记录了512+0 的读入
记录了512+0 的写出
536870912字节(537 MB)已复制，17.9563 秒，29.9 MB/秒
```

2、格式化

```
[root@localhost ~]# mkfs -t ext3 /home/loopdev
mke2fs 1.42.9 (28-Dec-2013)
/home/loopdev is not a block special device.
无论如何也要继续? (y,n) y
Discarding device blocks: 完成
```

提示不是正常装置，但是还是强行格式化

3、挂载

使用 mount 的 -o loop 参数

```
[root@localhost ~]# mkdir /media/cdrom
[root@localhost ~]# ls /media
cdrom
[root@localhost ~]# mount -o loop /home/loopdev /media/cdrom
[root@localhost ~]# df
文件系统                 1K-块    已用    可用 已用% 挂载点
/dev/mapper/centos-root 10190136 3667620 5998228   38% /
devtmpfs                 922584       0  922584    0% /dev
tmpfs                    933512       0  933512    0% /dev/shm
tmpfs                    933512    9136  924376    1% /run
tmpfs                    933512       0  933512    0% /sys/fs/cgroup
/dev/sda1                487652  114692  347360   25% /boot
/dev/mapper/centos-home 4908544   58196 4594348    2% /home
tmpfs                    186704       0  186704    0% /run/user/0
/dev/loop0               499656     416  473028    1% /media/cdrom
[root@localhost ~]#
```

挂载成功

swap（内存置换空间）

作用是应付物理内存不足时所造成内存延伸记录的功能

CPU 读取的数据来自于内存，当内存不足时，内存中暂不使用的程序和数据就会被移动到 swap 中，此时内存就有空间给需要执行的程序加载。

使用实体分割槽建立 swap：

1、分割：我选择继续在 sdb 上分割，新增 128M 的 sdb4

```
   设备 Boot      Start         End      Blocks   Id  System
/dev/sdb3            2048     1050623      524288   83  Linux
/dev/sdb4         1050624     1312767      131072   83  Linux
```

```
st cdrom]# ls /dev/s*
ev/sda2  /dev/sdb3  /
ev/sdb   /dev/sdb4  /
```

2、设置 swap 格式：

```
[root@localhost cdrom]# mkswap /dev/sdb4
正在设置交换空间版本 1，大小 = 131068 KiB
无标签，UUID=4017c427-0929-4b81-a88f-bc21509b7541
[root@localhost cdrom]#
```

3、观察与加载：

```
[root@localhost cdrom]# free
              total        used        free      shared  buff/cache   available
Mem:        1867024      164776     1373308        9140      328940     1504500
Swap:       1019900           0     1019900
```

表示现在总共 1867024 内存，已用 164776 内存。而 Swap 共 1029900，还没有用过

```
[root@localhost cdrom]# swapon /dev/sdb4
[root@localhost cdrom]# free
              total        used        free      shared  buff/cache   available
Mem:        1867024      165076     1371588        9140      330360     1504196
Swap:       1150968           0     1150968
[root@localhost cdrom]#
```

可见 Swap 增加了

使用文件建 swap：

1、建一个 128M 的空文件

```
[root@localhost ~]# dd if=/dev/zero of=/tmp/swap bs=1M count=128
记录了128+0 的读入
记录了128+0 的写出
134217728字节(134 MB)已复制，9.05836 秒，14.8 MB/秒
```

2、mkswap 格式化

```
[root@localhost ~]# mkswap /tmp/swap
正在设置交换空间版本 1，大小 = 131068 KiB
无标签，UUID=ff2c985a-a8a5-4b20-b667-d94574a544c5
```

3、用 swapon 启动/tmp/swap

```
[root@localhost ~]# free
              total        used        free      shared  buff/cache   available
Mem:        1867024      167996     1180512        9172      518516     1488236
Swap:       1150968           0     1150968
[root@localhost ~]# swapon /tmp/swap
swapon: /tmp/swap：不安全的权限 0644，建议使用 0600。
[root@localhost ~]# free
              total        used        free      shared  buff/cache   available
Mem:        1867024      168028     1180480        9172      518516     1488204
Swap:       1282036           0     1282036
```

4、Finally，用 swapoff 关掉 swap file

```
[root@localhost ~]# free
              total        used        free      shared  buff/cache   available
Mem:        1867024      168028     1180480        9172      518516     148820
Swap:       1282036           0     1282036
[root@localhost ~]# swapoff /tmp/swap
[root@localhost ~]# free
              total        used        free      shared  buff/cache   available
Mem:        1867024      167984     1180512        9172      518528     148824
Swap:       1150968           0     1150968
[root@localhost ~]# swapoff /dev/sdb4
[root@localhost ~]# free
              total        used        free      shared  buff/cache   available
Mem:        1867024      169608     1178884        9172      518532     148662
Swap:       1019900           0     1019900
```

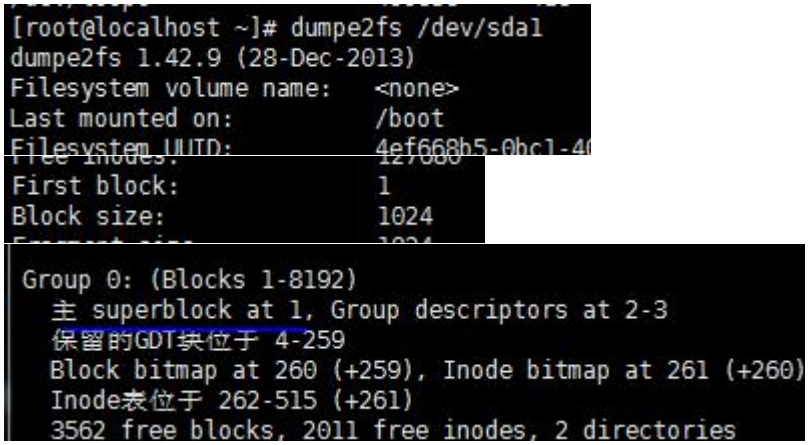
文件系统的观察：

1、block 为 1KB 时，boot sector 与 superblock 各占一个 block

首先在 df 中看到/boot 挂载的是/dev/sda1

```
tmpfs                   933512        0  933512    0% /sys/fs/c
/dev/sda1               487652   114692  347360   25% /boot
/dev/mapper/centos-home 4908544   58196 4594348    2% /home
```

因此使用 dumpe2fs 查询这个文件系统：



可见 block0 是保留下来的，那就是 boot sector 的所在了

2、block 大于 1KB

则 superblock 会在 block 0 ,因为其实 superblock 就只需要 1KB 大小 ,为了省空间 ,因此 block 里同时含有 boot sector 和 superblock

比如 block 是 4KB 大小的情况下，0~1K 是 boot sector，1~2K 是 superblock，剩下的 2K 是空闲的

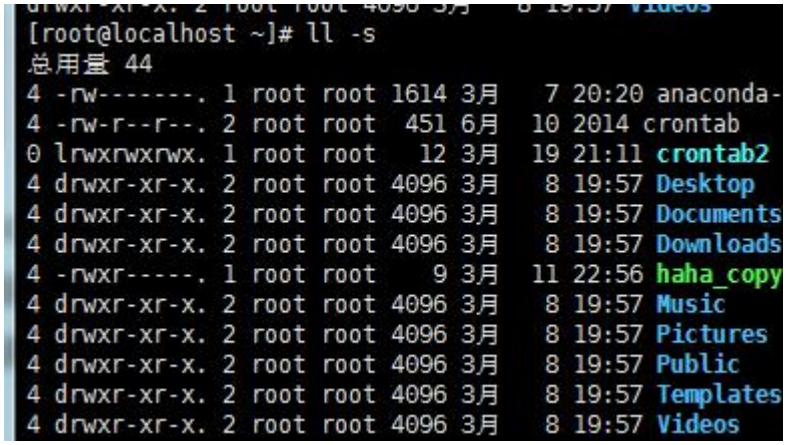
因此比较正确的说法 ,应该是 boot sector 安装到该 filesystem 最前面的 1024 bytes 内的区域

磁盘浪费：

一个 block 只能放一个文件的东西，因此太多小文件会浪费很多磁盘空间；

一个文件系统包括 superblock,inode table 等中介数据，也算是占据了磁盘空间，所以一旦在磁盘上建立其文件系统并挂载，就 like 有很多容量被占用了

每次使用 ll 指令查看目录下的文件 ,第一行会有一个 total：,代表这个目录下所有数据所耗用的实际 block 数量*block 大小



每个 block 大小为 4K，大多数目录都只用到了一个 block，但也不少了，可以看到一个文本文件 haha.txt 也只有 9Bytes 大小，却也占用了一个 block

使用 parted 指令进行分割：

fdisk 不支持高于 2TB 以上的分隔槽，但 parted 可以，而且 parted 直接一行指令就能完成分

割：

parted 列出分割情况

```
[root@localhost ~]# parted /dev/sdb print
Model: ATA VMware Virtual S (scsi)
Disk /dev/sdb: 1074MB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start   End    Size   Type     File system     标志
 3      1049kB  538MB  537MB  primary  ext3
 4      538MB   672MB  134MB  primary  linux-swap(v1)
```

可以看到分区 3 从该磁盘的 1048KB 处开始 ,到 538MB 处结束 ,前 1048KB 应该是留给了 boot sector。但是我 TM 一看，sdb3BLOCK SIZE 怎么是 4K 的 ! ! ! 根本对不上啊，前 1048Bytes 到底是干什么用的

```
[root@localhost ~]# dumpe2fs /dev/sdb3 |grep "Block size"
dumpe2fs 1.42.9 (28-Dec-2013)
Block size:               4096
[root@localhost ~]# dumpe2fs /dev/sdb4 |grep "Block size"
dumpe2fs 1.42.9 (28-Dec-2013)
dumpe2fs: Bad magic number in super-block 当尝试打开 /dev/sdb4 时
[root@localhost ~]# dumpe2fs /dev/sda1 |grep "Block size"
dumpe2fs 1.42.9 (28-Dec-2013)
Block size:               1024
```

开始分割：

```
[root@localhost ~]# parted /dev/sdb mkpart primary ext3 672MB 700MB
信息: You may need to update /etc/fstab.
```

/etc/fstab 就是开机自动挂载那个配置文件

```
Number  Start   End    Size    Type     File system     标志
3       1049kB  538MB  537MB   primary  ext3
4       538MB   672MB  134MB   primary  linux-swap(v1)
1       672MB   700MB  28.3MB  primary
```

磁盘乩使用必需要经过：分割、格式化不挂载，分别惯用乩挃令为：fdisk, mkfs, mount 三个挃
令