磁盘管理、挂载和逻辑卷LVM

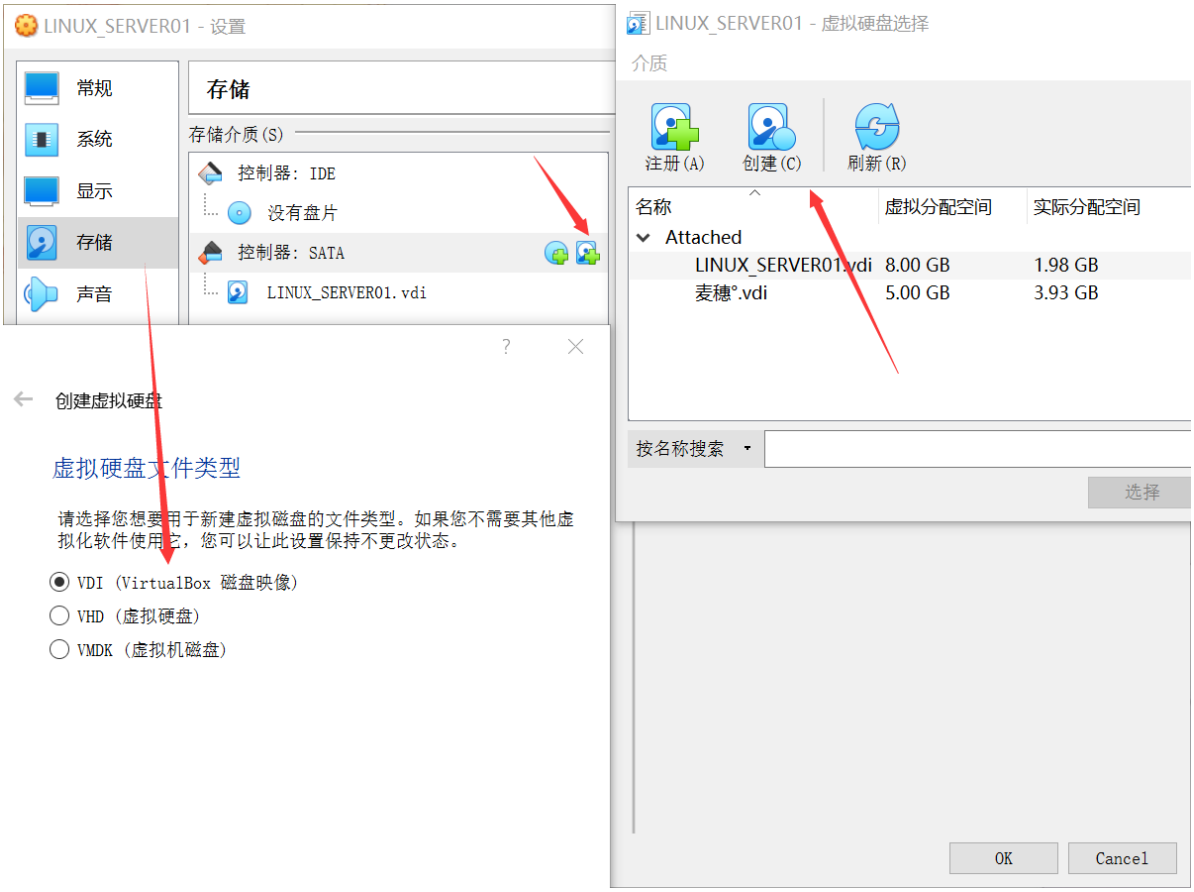
- 前言：

这一节内容多且难，但是这些关于硬盘的知识，必须掌握住。

- 正文：

1、首先要理解一个概念，不像windows，Linux上不能直接操作一个硬盘的，它是把硬盘分区和文件夹捆绑在一起的，后面我们也会说这个叫挂载，只有通过文件夹才能访问一个硬盘。如果我们的硬盘是一整块的没有多分区，在这个情况下，因为Linux是从根目录开始的，所以 \ 就对应着这唯一一个分区；如果这块盘a分了很多个分区，那么根目录还是对应sda1（永远不变），其他的对应别的文件夹，例如/home对应sda2，/user/local对应sda3等。

2、学会添加硬盘：



如图，点开设置，选择存储，点击控制器旁的添加虚拟硬盘，依次选择VDI、动态分配，自行取名和分配空间，然后创建完后点击控制器盘的添加把它加到虚拟机上。

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

Disk /dev/sda: 8589 MB, 8589934592 bytes, 16777216 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Disk identifier: 0x000cfea9

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *        2048     2099199     1048576   83  Linux
/dev/sda2         2099200    16777215     7339008   8e  Linux LVM

Disk /dev/sdb: 3221 MB, 3221225472 bytes, 6291456 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes


Disk /dev/mapper/centos-root: 6652 MB, 6652166144 bytes, 12992512 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
```

登录虚拟机，fdisk看一下，还真有了，我设置的3GB。

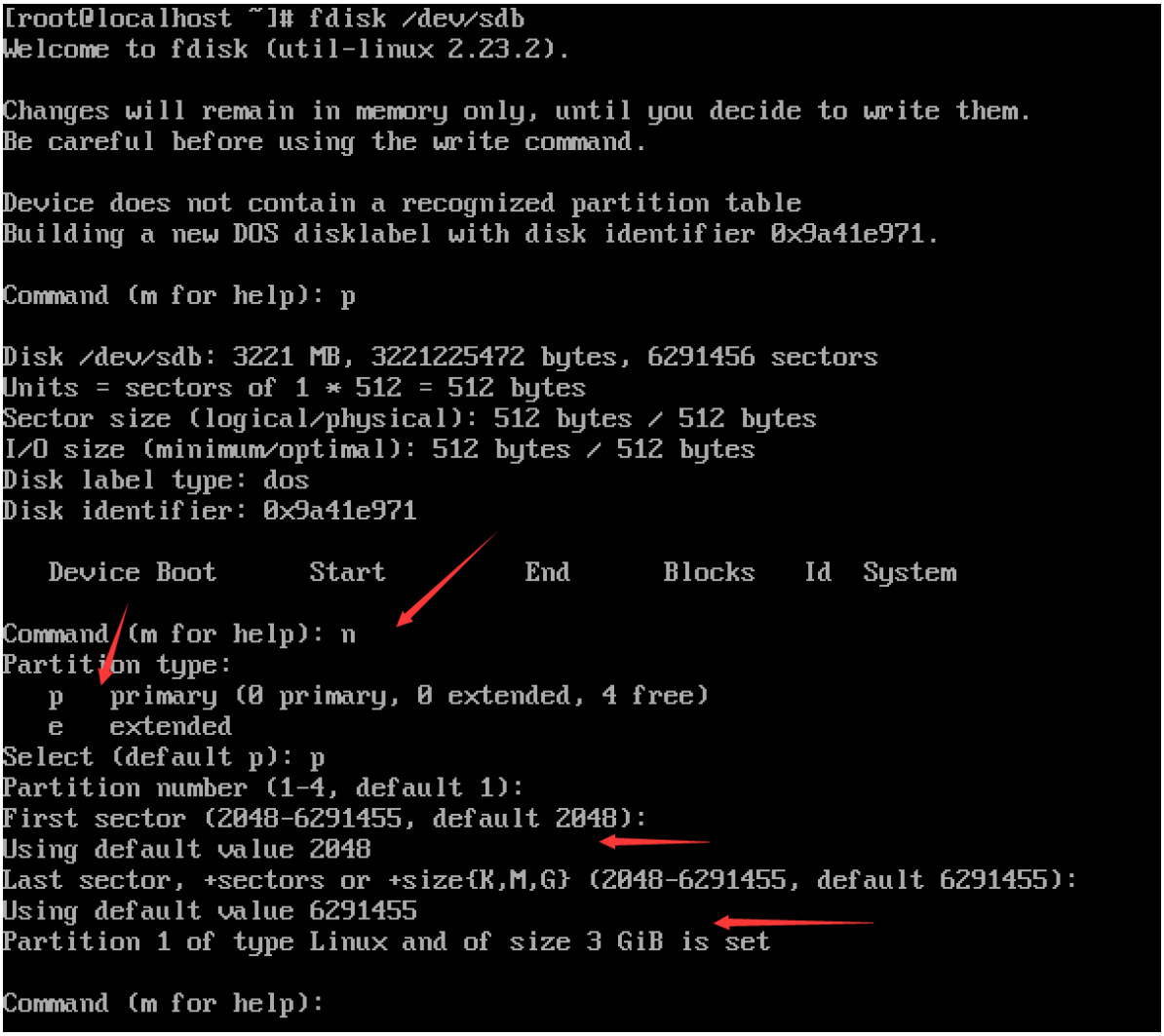
3、接下来不急着建立分区，我们先学一些基本概念。

```
[root@localhost ~]# fdisk /dev/sdb
Welcome to fdisk (util-linux 2.23.2).

Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Device does not contain a recognized partition table
Building a new DOS disklabel with disk identifier 0xfad642d4.

Command (m for help): p

Disk /dev/sdb: 3221 MB, 3221225472 bytes, 6291456 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Disk identifier: 0xfad642d4

   Device Boot      Start         End      Blocks   Id  System

Command (m for help): q_
```

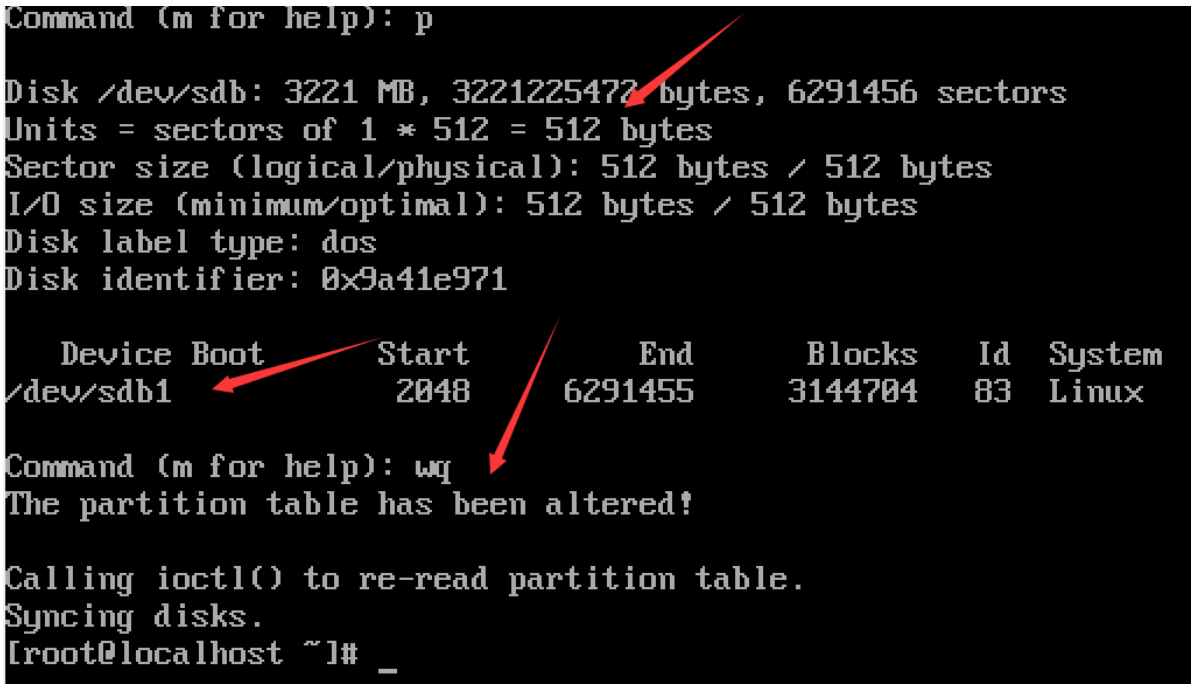
如图，fdisk /dev/sdb表示进入对sdb盘的编辑模式，后面的分区工作在这里进行，我们先看一看，读读英文，然后q不保存退出。

Linux下硬盘的分区分为三种——主分区，扩展分区和逻辑分区，一块硬盘最多能有四个主分区。但是我们硬是要分4个以上怎么办？好说，这时候把第四个分区当作扩展分区，然后扩展分区下全是逻辑分区（可以无限个，标号从05开始）。一句话，Linux可以把主分区当作扩展分区，且主分区+扩展分区不超过4个，扩展分区可以分无数个逻辑分区，逻辑分区的标号永远从05开始，不管它所属的扩展分区是第几个主分区。

4、多说无益，我们用n指令实操分区试试：



如图，用了n指令后，出现两个选项，按下p创建主分区，按下e创建扩展分区（然后再在里面创建逻辑分区），我选择了p，随即提示将要创建的主分区的编号，由于没有，默认为1。接下来，它询问first sector从哪里开始，和last sector从哪里结束，至于这个sector什么意思，我后面再解释一下。其实我们都知道，像光盘啊硬盘一类的东西，它不是严格从头开始存放数据一直到结尾的，特别像是硬盘，它把数据这里放一块，那里又放一块，那么这里的first和last也是这个道理，Linux的分区像是划分蛋糕一样，你只需割两刀，至于割哪里，自行选择。sector是扇区的意思，Linux把这块硬盘划分成6291455块，我们取其中默认的开头和结尾，即是把整个sdb当成一个分区sdb1。



提示创建成功后，p命令查看硬盘信息，发现确实多了一个分区sdb1。顺便讲讲扇区，如图箭头sdb信息里说明了一个sector是512个字节，512*6291455 = 3221224960 个字节，换算成MB是3071MB，和我们开始分配的3GB差1MB。这里要说明一下，Linux分硬盘空间不是那么精确的，做不到是一个整数的数量，要么多要么少，不过这里还好，没什么差别。准确来说，计算机都这样的，这是一个特性，之前我买的16GBU盘，缩水到14GB，好气啊。最后别忘了wq退出，直接q退出的话一切操作都是无效的。

注意，我们现在虽然创建好了分区1，但其实是用不了的，要给这个分区加上一个文件系统才能用，我们后面接着说。

5、创建文件系统并且挂载到目录上：

我们先来简单了解一下文件系统这个概念。如果说硬盘分区是个新建成的空教室，即将到来的要存放于此的文件是学生，那么学生们能直接进来这个空教室上课吗？显然不行，且不说没有老师没有黑板没有粉笔等等等等教师物资，学生进来没桌椅可以坐，以后上课就干站着听课？所以文件系统就好比这个空教室里的一切辅助道具，它的目标就是服务这个分区进行正常的存储等工作。

```
[root@localhost ~]# mkfs.ext4 /dev/sdb1
mke2fs 1.42.9 (28-Dec-2013)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
196608 inodes, 786176 blocks
39308 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=805306368
24 block groups
32768 blocks per group, 32768 fragments per group
8192 inodes per group
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912

Allocating group tables: done
Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done

[root@localhost ~]#
```

mkfs.ext4 + 目标分区，即可快速为其创建文件系统（说白了就是格式化）。ext4 (extended file system 4) 就是文件系统的后缀，而且ext4是最新一代的Linux文件系统。最早的ext是个不成熟的雏形，等到第二代时才宣布成熟成为里程碑，后面的版本都是其功能的强化而已。那么文件系统到底是什么，它是怎么管理分区的，我们得好好讲讲。

首先认识两个英文单词及其概念：inode和block。Linux文件系统的核心就是把文件分成inode和block两部分来存放。现在有一个txt文件，Linux会把它分成两部分来对待，一部分是文件属性，一部分是文件的真实内容。

①inode：我们之前学过文件和文件夹的各种属性，类似属主属组这些属性，会单独存放在inode里，每一个分区的inode是有限个的，没在里面存放一个文件就消耗一个inode，当inode数量为0时，纵使分区还有空间，系统也不允许再存放文件了。如上图，我们刚才创建的文件系统又196608个indoe。

②block：逻辑块，这个没啥好说的，就是存储空间，一个indoe可以联合多个block来存储文件。

其实说到这里，我们都能在脑海里浮现出一幅画面了：这文件系统就是个储物柜嘛，inode就是柜号，block就是格子，不同的是，Linux里不一定是一个柜号对应一个格子，可能是联合几个格子算作xx号柜子。

inode和block先了解到这里，后面再补充一些相关内容。至于挂载到文件上，就是给分区这个空教室装上门牌和门，好让我们能找到它。

```
Command (m for help): p

Disk /dev/sdb: 3221 MB, 3221225472 bytes, 6291456 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Disk identifier: 0x9a41e971

   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1            2048     2222222     1110087+  83  Linux
/dev/sdb2         2224128     4444444     1110158+  83  Linux
/dev/sdb3         4446208     6291455      922624   83  Linux

Command (m for help): _
```

我们删掉(d命令)原来的分区，创建(n命令)三个分区试试挂载。别忘了挂载前一定要mkfs创建文件系统，不然会提示出错的。

```
[root@localhost /]# ls
bin   dev  home  lib64  mnt  opt1  opt3  root  sbin  sys  usr
boot  etc  lib   media  opt  opt2  proc  run   srv   tmp  var
[root@localhost /]# mount /dev/sdb1 /opt1
[root@localhost /]# mount /dev/sdb2 /opt2
[root@localhost /]# mount /dev/sdb3 /opt3
[root@localhost /]# df -h | grep sdb
/dev/sdb1               1.1G  2.7M  963M   1% /opt1
/dev/sdb2               1.1G  2.7M  963M   1% /opt2
/dev/sdb3               871M  2.3M  808M   1% /opt3
[root@localhost /]# cd /opt1/
[root@localhost opt1]# dd if=/dev/zero of=tmpfile bs=1M count=800
800+0 records in
800+0 records out
838860800 bytes (839 MB) copied, 3.33524 s, 252 MB/s
[root@localhost opt1]# ls -ltrh tmpfile
-rw-r--r--. 1 root root 800M Sep 12 05:05 tmpfile
[root@localhost opt1]# ls
lost+found  tmpfile
[root@localhost opt1]# df -h
Filesystem              Size  Used Avail Use% Mounted on
devtmpfs                484M     0  484M   0% /dev
tmpfs                   496M     0  496M   0% /dev/shm
tmpfs                   496M  6.8M  489M   2% /run
tmpfs                   496M     0  496M   0% /sys/fs/cgroup
/dev/mapper/centos-root  6.2G  1.5G  4.8G  24% /
/dev/sda1               1014M  141M  874M  14% /boot
tmpfs                   100M     0  100M   0% /run/user/0
/dev/sdb1               1.1G  803M  163M  84% /opt1
/dev/sdb2               1.1G  2.7M  963M   1% /opt2
/dev/sdb3               871M  2.3M  808M   1% /opt3
[root@localhost opt1]# _
```

如图，先在根目录下创建opt1、opt2、opt3三个目录，确保三个分区都执行了文件系统命令，然后用mount 分区名 挂载目录，来执行挂载操作。可尝试在sdb1里创建一个800MB文件看看是不是挂载成功。但是，这种mount命令是暂时的，在重启机器后全部失效，不信自行reboot重启后df看看是不是这样。

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

#
# /etc/fstab
# Created by anaconda on Sun Aug 29 04:06:05 2021
#
# 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 /                       xfs     defaults        0 0
UUID=4e064f45-e615-433e-91b8-72293f1523d9 /boot                   xfs     defau
/dev/mapper/centos-swap swap                    swap    defaults        0 0
/dev/sdb1               /opt1                   ext4    defaults        0 0
/dev/sdb2               /opt2                   ext4    defaults        0 0
/dev/sdb3               /opt3                   ext4    defaults        0 0
[root@localhost /]#
```

我的选择是，用Vim打开/etc/fstab，在末尾加上这三行，让它开机自己执行挂载。

另外，若想取消挂载，直接umount 目标文件夹即可（如 umount /opt1/）。顺带一提，如果打开着文件夹或者打开着文件夹的内容时（分区忙碌中），是取消不了挂载的，这时候用lsof | grep 文件名 或者 fuser 命令来通过文件找到打开文件的进程，记下PID再用kill指令杀死进程即可，不过如果你能手动退出文件夹或者关闭文件就没那么多屁事了😂。可能有的小伙伴会有疑问，那我取消挂载后是不是里面的文件就没了呢？并不，取消挂载相当于把教室门拆了封起教室，原本存在教室里的东西当然还在教室里，再挂载后就可以看到原来的内容了。

6、如何给分区扩容：

有时候空间不够用啦，想给分区扩容，那怎么操作呢？

分两种方式：①umount分区，备份原区文件到另一个区，然后删掉分区，重新建一个，并且调容量到理想值，挂载好新分区再把原来的文件搬进来。②直接扩容，但是也要先卸掉挂载，然后在fdisk编辑模式里删掉原区，立刻建立新区。并且左起点要和原来一样，但是右终点要比原来大，这样可分配的block就多了，多了才能将这部分应用。随后执行硬盘分区检验命令 e2fsck -f 分区名，通过了才可以进行文件系统的填充命令 resize2fs 分区名。注意不能直接上来就mkfs.ext4，这样会把原来的数据清空的。最后挂载到文件夹上，就可以看到原文件的存在了，df命令可以查看更新后的存储量。

```
[root@localhost ~]# e2fsck -f /dev/sdb3
e2fsck 1.42.9 (28-Dec-2013)
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Checking reference counts
Pass 5: Checking group summary information
/dev/sdb3: 11/69360 files (0.0% non-contiguous), 19025/276896 blocks
[root@localhost ~]# resize2fs /dev/sdb3
resize2fs 1.42.9 (28-Dec-2013)
Resizing the filesystem on /dev/sdb3 to 922624 (1k) blocks.
The filesystem on /dev/sdb3 is now 922624 blocks long.

[root@localhost ~]#
```

我这里操作太快了，没有截全图，所以就放这一张吧，一切都很像C语言里的realloc函数，不是吗？

7、再讲inode和block的分组：

如何查看一个分区的详细信息而不是像前面那样一直查看硬盘信息呢？dumpe2fs 分区名 即可实现这个功能，当然其中的内容特别特别多，所以可以加上 | less 进行可翻页查看。当然如果只是查看inode数量，直接 df -i 就可以了。

其实有一个问题，一个分区里面那么多个inode和block杂乱无章的，会不会很影响访问效率呢？之前我们在dumpe2fs下查看的详细信息，会发现有很多以Group开头的信息组，这是我们接下来要说的block group，Linux通过将所有的inode和block平均分组分在一起的方式提高访问效率。

```
Group 0: (Blocks 1-8192) [ITABLE_ZEROED]
  Checksum 0xd907, unused inodes 2029
  Primary superblock at 1, Group descriptors at 2-9
  Reserved GDT blocks at 10-260
  Block bitmap at 261 (+260), Inode bitmap at 277 (+276)
  Inode table at 293-547 (+292)
  3806 free blocks, 2029 free inodes, 2 directories, 2029 unused inodes
  Free blocks: 4387-8192
  Free inodes: 12-2040
Group 1: (Blocks 8193-16384) [INODE_UNINIT, ITABLE_ZEROED]
  Checksum 0xa109, unused inodes 2040
  Backup superblock at 8193, Group descriptors at 8194-8201
  Reserved GDT blocks at 8202-8452
  Block bitmap at 262 (bg #0 + 261), Inode bitmap at 278 (bg #0 + 277)
  Inode table at 548-802 (bg #0 + 547)
  7932 free blocks, 2040 free inodes, 0 directories, 2040 unused inodes
  Free blocks: 8453-16384
  Free inodes: 2041-4080
Group 2: (Blocks 16385-24576) [INODE_UNINIT, BLOCK_UNINIT, ITABLE_ZEROED]
  Checksum 0x9910, unused inodes 2040
  Block bitmap at 263 (bg #0 + 262), Inode bitmap at 279 (bg #0 + 278)
  Inode table at 803-1057 (bg #0 + 802)
  8192 free blocks, 2040 free inodes, 0 directories, 2040 unused inodes
  Free blocks: 16385-24576
  Free inodes: 4081-6120
Group 3: (Blocks 24577-32768) [INODE_UNINIT, ITABLE_ZEROED]
  Checksum 0x0536, unused inodes 2040
  Backup superblock at 24577, Group descriptors at 24578-24585
  Reserved GDT blocks at 24586-24836
  Block bitmap at 264 (bg #0 + 263), Inode bitmap at 280 (bg #0 + 279)
  Inode table at 1058-1312 (bg #0 + 1057)
  7932 free blocks, 2040 free inodes, 0 directories, 2040 unused inodes
  Free blocks: 24837-32768
  Free inodes: 6121-8160
Group 4: (Blocks 32769-40960) [INODE_UNINIT, BLOCK_UNINIT, ITABLE_ZEROED]
:
```

如图，是不是很多组？结合前面说过的，一个inode被一个文件占用时，它会迅速联合相应的blocks一起给这个文件存储空间，这种分组方式可以让他们快速配对。那么总得有个记总账的地方，那就是superblock——它虽然不大，但是包含了整个文件系统的信息，例如block和inode的使用量、挂载时间、对照表等等。一般第一个block组里有superblock，其他的block组里可能含有它的备份。

8、Linux下的硬链接和软链接：

```
[root@localhost ~]# ls
anaconda-ks.cfg  test.txt
[root@localhost ~]# ln test.txt test
[root@localhost ~]# ls
anaconda-ks.cfg  test  test.txt
[root@localhost ~]# pwd
/root
[root@localhost ~]# ln -s /etc/ssh/sshd_config ssh_config_sl
[root@localhost ~]# ls
anaconda-ks.cfg  ssh_config_sl  test  test.txt
[root@localhost ~]# ls -l
total 12
-rw-------. 1 root root 1306 Aug 29 04:12 anaconda-ks.cfg
lrwxrwxrwx. 1 root root   20 Sep 12 09:41 ssh_config_sl -> /etc/ssh/sshd_config
-rw-r--r--. 2 root root   48 Sep 12 09:39 test
-rw-r--r--. 2 root root   48 Sep 12 09:39 test.txt
[root@localhost ~]#
```

①硬链接：ln 源文件 新文件名。是不是点像cp指令？其实这个新出的test完全不占空间，因为它和test.txt是一体的。所谓的copy，是新建了一个inode，然后又新建了blocks，再把原来的内容复制到blocks里，这是占空间的；而硬链接等于是一个新的文件名，指向到原本已有的inode，再指向到原本的blocks，说白了就是用一个文件名指向同样的inode和block，障眼法而已。当然，硬链接由于是使用同

一个inode，所以不能跨区执法。

②软链接：ln -s 源文件 新文件名。一句话让你理解软链接原理——windows的快捷方式。软链接创造一个新的文件，这个文件有新的inode和block，但是block里的内容存着的是源文件的完整的路径（查看内容时看到的是源文件内容），说白了就是发送快捷方式，所以可以跨区执法

另外，硬链接中，如果删除了源文件，照样可以访问，毕竟源文件也只是一个文件名，删了一个文件名还有一个文件名呢；至于软链接，那就不行了，源文件删了，快捷方式自然找不到路了。

9、逻辑卷LVM：

①这玩意是什么：

一个硬盘分好区后，大小已经固定下来了，除非卸载重分，要是遇到里面的数据被使用，那就要停下它们，否则umount命令都不能执行。另外，如果一个分区分完了硬盘，但是还是想扩容，插了个新硬盘后，由于是不同硬盘，这个区自然不能延申到新盘上，那么只能在新盘上建新区了。LVM的诞生解决了这些问题，它像是一个大海，把给的几块硬盘融在逻辑卷LVM里成一个仓库，然后随便用，跨盘执法也是可以的。逻辑卷LVM里不再叫区，而是卷，但本质上还是区，像之前的分区操作一样该干嘛干嘛，当不够用的时候，直接从仓库里拿空间补充在卷上却不影响别的事情。记住啊，就是多块硬盘或区合成一块用，所以要保证盘够多容量够足。

②实践操作：

```
[root@localhost ~]# umount /opt1 /opt2 /opt3
[root@localhost ~]# umount /opt1 /opt2 /opt3
umount: /opt1: not mounted
umount: /opt2: not mounted
umount: /opt3: not mounted
[root@localhost ~]# pvcreate /dev/sdb1 /dev/sdb2 /dev/sdb3
WARNING: ext4 signature detected on /dev/sdb1 at offset 1080. Wipe it? [y/n]: y
  Wiping ext4 signature on /dev/sdb1.
WARNING: ext4 signature detected on /dev/sdb2 at offset 1080. Wipe it? [y/n]: y
  Wiping ext4 signature on /dev/sdb2.
WARNING: ext4 signature detected on /dev/sdb3 at offset 1080. Wipe it? [y/n]: y
  Wiping ext4 signature on /dev/sdb3.
  Physical volume "/dev/sdb1" successfully created.
  Physical volume "/dev/sdb2" successfully created.
  Physical volume "/dev/sdb3" successfully created.
[root@localhost ~]# vgcreate myvg01 /dev/sdb1 /dev/sdb2/ dev/sdb3
  Device /dev/sdb2/ not found.
  Device dev/sdb3 not found.
[root@localhost ~]# vgcreate myvg01 /dev/sdb1 /dev/sdb2 /dev/sdb3
  Volume group "myvg01" successfully created
[root@localhost ~]#
```

首先，pvcreate命令标记那些高级融合素材，这里我选了一个盘的三个区。然后它问我要不要签名，那我勉为其难签个名。最后，发动高级融合魔法vgcreate，将它们全融在一起，成了myvg01。这样，一个仓库完成了。

```
[root@localhost ~]# vgdisplay myvg01
  --- Volume group ---
  VG Name               myvg01
  System ID
  Format                lvm2
  Metadata Areas        3
  Metadata Sequence No  1
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                0
  Open LV               0
  Max PV                0
  Cur PV                3
  Act PV                3
  VG Size               <2.99 GiB
  PE Size               4.00 MiB
  Total PE              765
  Alloc PE / Size       0 / 0
  Free  PE / Size       765 / <2.99 GiB
  VG UUID               XmBDyZ-DcRp-nVLw-eCFG-ajq0-syO6-3moG6H

[root@localhost ~]# _
```

可以vgdisplay查看已有的LVM仓库，发现就一个。

```
[root@localhost ~]# lvcreate -L 1G -n mylv01 myvg01
  Logical volume "mylv01" created.
[root@localhost ~]# lvdisplay
  --- Logical volume ---
  LV Path                /dev/centos/swap
  LV Name                swap
  VG Name                centos
  LV UUID                vm55Sj-6Lpj-TVmK-tKp1-WtRC-aaky-3A5HR0
  LV Write Access        read/write
  LV Creation host, time localhost, 2021-08-29 04:06:04 -0400
  LV Status              available
  # open                 2
  LV Size                820.00 MiB
  Current LE             205
  Segments               1
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     8192
  Block device           253:1

  --- Logical volume ---
  LV Path                /dev/centos/root
  LV Name                root
  VG Name                centos
  LV UUID                ehjaOY-N7BK-a7dg-3LJc-04AO-b780-Akjt3I
  LV Write Access        read/write
  LV Creation host, time localhost, 2021-08-29 04:06:05 -0400
  LV Status              available
  # open                 1
  LV Size                <6.20 GiB
  Current LE             1586
  Segments               1
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     8192
  Block device           253:0
```

仓库建好了，可以开始分蛋糕了。lvcreate -L 大小 -n 新卷名字 要分的仓库 ，这一条命令启用，mylv01便宜嗯造出来，lvdisplay一下，会显示已有的逻辑卷。不看不知道，一看吓一跳，首先显示的是建系统时用的sda盘，其中两个分区都是以逻辑卷形式创造，再往下翻页才能看到mylv01。可见这个LVM的技术是多常用。

```
[root@localhost ~]# mkfs.ext4 /dev/myvg01/mylv01
mke2fs 1.42.9 (28-Dec-2013)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
65536 inodes, 262144 blocks
13107 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=268435456
8 block groups
32768 blocks per group, 32768 fragments per group
8192 inodes per group
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376

Allocating group tables: done
Writing inode tables: done
Creating journal (8192 blocks): done
Writing superblocks and filesystem accounting information: done

[root@localhost ~]# mount /dev/myvg01/mylv01 /opt1
[root@localhost ~]# _
```

mkfs.ext4一下，mount一下，df -h 查看一下，这个逻辑卷就能用了。

③想扩容了：

```
[root@localhost opt1]# lvextend -L +1G /dev/myvg01/mylv01
  Size of logical volume myvg01/mylv01 changed from 1.00 GiB (256 extents) to 2.00 GiB (512 extents).
  Logical volume myvg01/mylv01 successfully resized.
[root@localhost opt1]# df -h | mylv01
-bash: mylv01: command not found
[root@localhost opt1]# df -h | grep mylv01
/dev/mapper/myvg01-mylv01  976M  2.6M  907M   1% /opt1
[root@localhost opt1]# resize2fs /dev/myvg01/mylv01
resize2fs 1.42.9 (28-Dec-2013)
Filesystem at /dev/myvg01/mylv01 is mounted on /opt1; on-line resizing required
old_desc_blocks = 1, new_desc_blocks = 1
The filesystem on /dev/myvg01/mylv01 is now 524288 blocks long.

[root@localhost opt1]# df -h | grep mylv01
/dev/mapper/myvg01-mylv01  2.0G  3.0M  1.9G   1% /opt1
[root@localhost opt1]# _
```

卷就比较简单了，lvextend快速扩容，但是就像前面的分区扩容一样，要resize2fs填充文件系统，不然系统不认的，多出的空间也不能用。至于LVM仓库扩容，例如要加入一块硬盘sdc作为融合材料，此时先pvcreate /dev/sdc标记了，再vgextend myvg01 /dev/sdc实现合并。

- 后记：

我超，这一节怎么这么多内容啊，学麻了。