# 背景

磁盘一旦格式化之后，空间大小就确定了，如果需要扩充磁盘空间，就需要更换一个更大的磁盘。这样还需要将原来磁盘的数据迁移出来，比较麻烦。如果我们不想进行数据迁移，则需要LVM。LVM逻辑卷，是个虚拟的磁盘。

还有一个原因，多个物理磁盘整合成一个大的虚拟磁盘，这是LVM的一个更加重要的作用。

**存储虚拟化技术**可以将不同存储设备进行格式化，屏蔽存储设备的能力、接口协议等差异性，将各种存储资源转化为统一管理的数据存储资源，但是性能较差（存储虚拟化需要先去找虚拟的存储节点，然后找逻辑卷，而LVM不需要去寻找虚拟存储节点，所以性能好一些）。

**非虚拟化存储**基于逻辑卷管理，较虚拟化存储有更高的性能，速度更快，效率更高，但是在快照、精简配置等支持程度没有存储虚拟化高。

**裸设备映射**：将物理裸设备直接映射给虚拟机，虚拟机磁盘能够处理SCSI命令，适用于关键业务场景，比如数据库业务。

# 卷管理

    卷管理软件对待RAID提交给上层操作系统的逻辑盘（操作系统视为物理磁盘）和真正的物理磁盘是一视同仁的，即不管底层的磁盘组织形式如何，只要OS认为它是一块物理磁盘，卷管理器就可以对它进行卷管理。稍微不同的是，对于RAID提交的逻辑卷，最终还是要通过RAID控制器来和底层的物理磁盘​交互。​

卷管理器包括：

LDM（逻辑磁盘管理）：Windows2000引入的动态磁盘

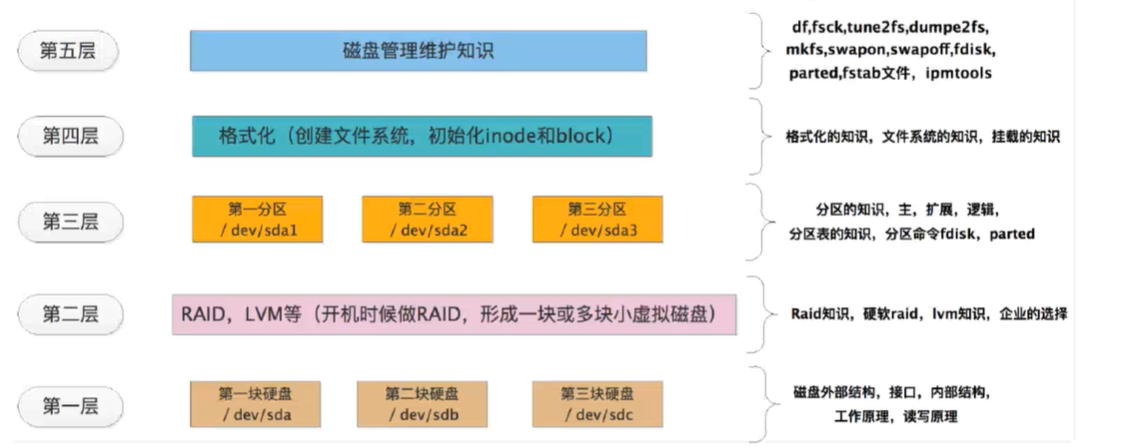
VxVM（Veritas Volume Manager）

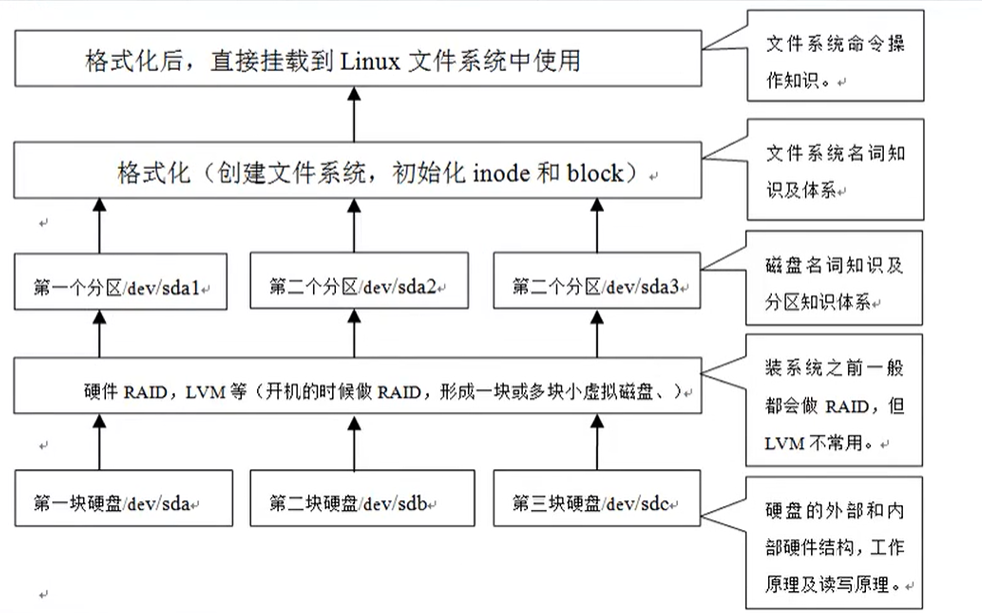
LVM（Logic Volume Manager）：Linux、AIX、HPUX

# LVM

## 概述

LVM是逻辑卷管理工具，它允许你使用逻辑卷和卷组的概念来管理磁盘空间。使用LVM相比传统分区最大的好处就是弹性地为用户和程序分配空间而不用考虑每个物理磁盘的大小。





在LVM中，哪些创建了逻辑分区的物理存储是传统的分区（比如/dev/sda1，/dev/sdb1）。**这些分区必须被初始化为“物理卷PV”并加上卷标（如“Linux LVM”）来使他们可以在LVM中使用**（通过fdisk T参数指定类型为LVM，编号8e）。一旦分上区被标记为LVM分区，就不能**直接**用mount命令挂载（必须经过PV->VG->LV之后才可以挂载，普通物理磁盘可以直接挂载）。

注：LVM是逻辑卷，它就是个虚拟的磁盘，将多个物理磁盘整合成一个大的虚拟磁盘。

/boot分区用于存放引导文件，不能基于LVM创建。

物理存储介质（The physical media）：LVM存储介质，可以是硬盘分区、整个硬盘、raid阵列或SAN硬盘。设备必须初始化为LVM物理卷，才能与LVM结合使用。

物理卷PV（physical volume）：物理卷就是LVM的基本存储逻辑块，但和基本的物理存储介质比较却包含与LVM相关的管理参数，创建物理卷可以用硬盘分区，也可以用硬盘本身。

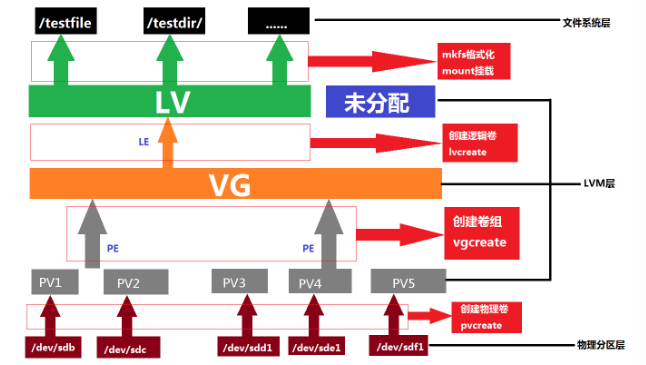
卷组VG（Volume Group）：LVM卷组类似于非LVM系统中的物理硬盘，一个卷组VG由一个或多个物理卷PV组成。可以在卷组VG上建立逻辑卷LV。

逻辑卷LV（logical volume）：类似于非LVM系统中的硬盘分区，逻辑卷LV建立在卷组VG之上。在逻辑卷LV之上建立文件系统。

物理块PE（physical Extent）：物理卷PV中可以分配的最小存储单元，PE的大小可以指定，默认为4MB

逻辑块LE（Logical Extent）：逻辑卷LV中可以分配的最小存储单元，在同一卷组VG中LE的大小和PE是相同的，并且一一相对。

## 原理



LVM使用基本思路：（分区->）PV->VG->LV->格式化分区->mount/fstab自动挂载->e2fsadm调整LV大小

大致步骤：

1. 先将每个小磁盘以固定大小切割成一块一块的小PE（默认是4M）；

注：PE是LVM内部进行寻找数据的一种方式，比如20M数据，每个PE是8M大小，则有可能存在8M位于磁盘1的n PE，后面的位于磁盘2的n+1 n+2PE位置上，在用户看来不知道是在多个磁盘上的，而是在多个磁盘上的。

1. 给每个小PE进行编号（磁盘1:1~100，磁盘2:101~200……）；
2. 创建PV（包含多个PE单元）；
3. 将多个PV组合成VG；
4. 在卷组VG基础上创建可挂载的逻辑卷LV；
5. 在LV上创建文件系统，格式化分区；
6. 直接mount挂载或者/etc/fstab挂载。

## 指令

|  |  |  |  |
| --- | --- | --- | --- |
| 功能 | 物理卷管理 | 卷组管理 | 逻辑卷管理 |
| 扫描scan | pvscan | vgscan | lvscan |
| 建立create | pvcreate | vgcreate | lvcreate |
| 显示display | pvdisplay | vgdisplay | lvdisplay |
| 删除remove | pvcreate | vgremove | lvremove |
| 扩展extend |  | vgextend | lvextend |
| 减少reduce |  | vgreduce | lvreduce |

## 特点

### 优点

卷组VG可以使多个硬盘空间看起来像是一个大硬盘。

逻辑卷LV可以创建跨多个硬盘空间的分区。

在使用逻辑卷LV时，可以在空间不足时动态调整大小，不需要考虑逻辑卷LV在硬盘上的位置，不用担心没有可用的连续的空间。

可以在线对卷组VG、逻辑卷LV进行创建、删除、调整大小等操作。但LVM上的文件系统也需要重新调整大小。

LVM允许创建快照，用来保存文件系统的备份。

注意：LVM是软件的卷管理方式，RAID是磁盘管理的方法。对于重要的数据使，用RAID保护物理硬盘不会因为故障而中断业务，再用LVM来实现对卷的良性管理，更好的利用硬盘资源。

LVM有两种写入机制：线性（写完一个PV再写下一个PV，默认）、条带（平均）

1、文件系统可以跨多个磁盘，因此文件系统大小不会受物理磁盘的限制。

2、可以在系统运行的状态下动态的扩展文件系统的大小。

3、可以增加新的磁盘到LVM的存储池中。

4、可以以镜像的方式冗余重要的数据到多个物理磁盘。

5、可以方便的导出整个卷组到另外一台机器。

### 缺点

1. 在从卷组中移除一个磁盘的时候必须使用reducevg命令（这个命令要求root权限，并且不允许在快照卷组中使用）。
2. 误删除数据恢复更加困难，LVM缩减分区大小风险较大。

2、当卷组中的一个磁盘损坏时，整个卷组都会受到影响。

3、因为加入了额外的操作，存贮性能受到影响。

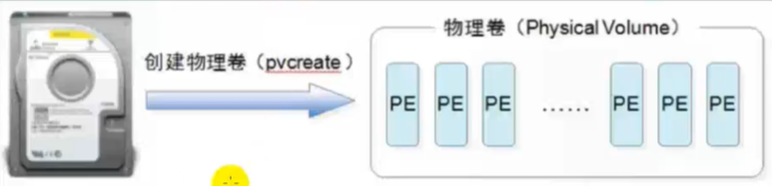
注：因为LVM的写入性能比较差，所以磁盘做RAID后直接使用裸LUN。

## 快照

# PV

## 概述

PV（Physical Volume），物理卷，整个磁盘或使用fdisk等工具创建的普通分区。包括许多默认4MB大小的PE（Physical Extent，基本单元/物理单元）。



注：LVM将操作系统识别到的物理磁盘（或者RAID控制器提交的逻辑磁盘）称为物理卷。

## 操作

### pvcreate

格式：pvcreate 设备1[设备2]

创建物理卷

注：这里的设备就是前面fdisk时指定type为Linux LVM时划分的磁盘（比如/dev/sda1，可以通过cat /proc/partitions查看）。

### pvchange

pvchange命令 管理员改变物理卷的分配许可，如果物理卷出现故障，可以使用pvchange命令禁止分配物理卷上的PE。

指令格式：

pvchange 选项 参数

-u 生成新的UUID

-x 是否允许分配PE

### pvremove

pvremove命令用于删除一个存在的物理卷，使用该指令删除物理卷时，它将LVM分区上的物理卷信息删除，使其不再被视为一个物理卷。

### povs

使用povs指令查看本地的PV，但是不一定是实际位置，因为我们都是经过链接实现关联的（例如/dev/vmc/serial-\*\*\*显然不是物理磁盘，是一个软链接），最后会链接到/dev/sd\*或者/dev/hd\*这样的磁盘上（注意是链接而不是挂载设备）。

### pvscan

pvscan命令会扫描系统中连接的所有磁盘，列出找到的物理卷列表。使用pvscan命令的-n选项可以显示硬盘中的不属于任何卷组的物理卷，这些物理卷是未被使用的。

### pvck

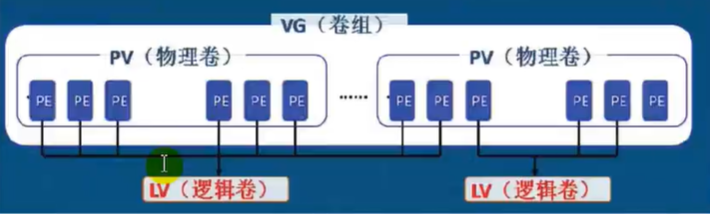
pvck命令用来检测物理卷的LVM元数据的一致性。默认情况下，物理卷中的前4个扇区保存着LVM卷标，可以使用—labelsector选项指定其他的位置（例如数据恢复时）。

# VG

## 概述

多个PV被放置在一个VG中，VG是一个虚拟的大存储空间，逻辑上是连续的，尽管有多个分散的PV组成，但是VG会将这些PV收尾链接组成逻辑上连续的大存储池。

注：VG相当于一个目录，不能直接挂载。



PP：物理区块，在逻辑上将VG分成连续的小块，注意是逻辑上的分割，而不是物理上的分割，也就是说LVM会记录PP的大小（由几个扇区组成）和PP序号的偏移。如果PV本身是已经经过RAID控制器虚拟化而成的一个LUN，那么这个扇区很有可能位于多个条带中；

LP：PP可以再次组成LP，即逻辑区块。

注：在实际的应用存储模型中，PV与存储设备device和访问单元unit相对应，VG与存储库repos相对应，LV与存储文件file相对应。

## 操作

### vgcreate

格式：vgcreate 卷组名 物理卷名1 物理卷2 选项 –s 指定PE大小（单位M）

### vgremove

### vgextend

VG扩容

### vgreduce

VG缩容

### vgscan

vgs会显示两个VG信息，其中一个VolGroup是系统盘。

### vgdisplay

vgdisplay命令用于显示LVM卷组的信息，如果不指定“卷组”参数，则分别显示所有卷组的属性。

# LV

## 概述

LV才是实际可以挂载的盘。

## 操作

### lvcreate

格式：lvextend –L+大小或直接写增加后的最终大小 /dev/卷组名/逻辑卷名

快照：lvcreate -L 20M -n lv0backup -s /dev/vg0/lv0（创建20M快照）

注：读取数据大小不能超过快照的大小，否则快照撑爆失效（比如dd if=/dev/zero of=/dev/\* bs=1 count=100）。

### lvremove

删除LV

### lvextend

格式：lvextend –L+大小或直接写增加后的最终大小 /dev/卷组名/逻辑卷名

拉伸逻辑卷基本步骤：

1. 保证VG中有足够的空闲空间：vgdisplay
2. 扩充逻辑卷：lvextend -L 1G /dev/\*
3. 查看扩充后LV的大小：lvdisplay
4. 更新文件系统大小：resize2fs /dev/\*\*\*
5. 查看更新后文件系统大小：df -h

注：扩充大小是自下而上，先扩充LV，然后扩充文件系统。

### resize2fs

格式：resize2fs /dev/卷组名/逻辑卷名

功能：调整逻辑卷分区大小

### lvreduce

缩减LV

基本操作步骤：

1. 卸载已挂载的逻辑卷：umount /dev/\*
2. 缩小文件系统：resize2fs /dev/\* 1G
3. 缩小LV：lvreduce -L -1G /dev/\*\*\*
4. 查看缩小后的LV：lvdisplay
5. 挂载：mount /dev/\*\* /mnt

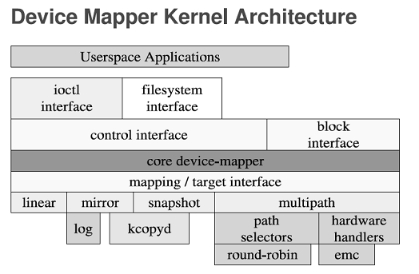
注：缩小是先从上层文件系统开始缩小，然后再去缩小LV。

### lvscan

扫描LV

### lvdispaly

# Device Mapper



在内核中Device mapper通过一个个模块化的target driver插件实现对IO请求的过滤或者重新定向等工作，包括软raid、软加密、逻辑卷条带、多路径、镜像、快照等。

Device mapper遵循内核设计中策略和机制分离的原则，将所有与策略相关的工作放到用户空间中，比如逻辑设备和哪些物理设备建立映射，怎么建立这些映射关系等等，内核主要提供支持这些策略所需要的机制，比如具体过滤和重定向IO请求。

在linux系统中你使用一些命令时（如nmon、iostat），有可能会看到一些名字为dm-xx的设备，那么这些设备到底是什么设备呢，跟磁盘有什么关系呢？

其实dm是Device Mapper的缩写，**Device Mapper 是 Linux 2.6 内核中提供的一种从逻辑设备到物理设备的映射框架机制**，在该机制下，用户可以很方便的根据自己的需要制定实现存储资源的管理策略，当前比较流行的 Linux下的逻辑卷管理器如 LVM2（Linux Volume Manager 2 version)、EVMS(Enterprise Volume Management System)、dmraid(Device Mapper Raid Tool)等都是基于该机制实现的。

首先我们来看看dm-0,dm-1,dm-2这个三个文件设备:

[root@mylnx01 ~]# ls /dev/dm\*

/dev/dm-0 /dev/dm-1 /dev/dm-2

[root@mylnx01 ~]# ll /dev/dm\*

brw-rw---- 1 root root 253, 0 Dec 7 16:45 /dev/dm-0

brw-rw---- 1 root root 253, 1 Dec 7 16:45 /dev/dm-1

brw-rw---- 1 root root 253, 2 Dec 7 16:45 /dev/dm-2

[root@mylnx01 ~]#

或者你使用fdisk -l 命令也能看到

[root@mylnx01 ~]# fdisk -l

Disk /dev/sda: 85.8 GB, 85899345920 bytes

255 heads, 63 sectors/track, 10443 cylinders

Units = cylinders of 16065 \* 512 = 8225280 bytes

Device Boot Start End Blocks Id System

/dev/sda1 \* 1 13 104391 83 Linux

/dev/sda2 14 6788 54420187+ 8e Linux LVM

/dev/sda3 6789 10443 29358787+ 83 Linux

Disk /dev/dm-0: 107.2 GB, 107206410240 bytes

255 heads, 63 sectors/track, 13033 cylinders

Units = cylinders of 16065 \* 512 = 8225280 bytes

Disk /dev/dm-0 doesn't contain a valid partition table

Disk /dev/dm-1: 12.8 GB, 12884901888 bytes

255 heads, 63 sectors/track, 1566 cylinders

Units = cylinders of 16065 \* 512 = 8225280 bytes

Disk /dev/dm-1 doesn't contain a valid partition table

Disk /dev/dm-2: 30.0 GB, 30031216640 bytes

255 heads, 63 sectors/track, 3651 cylinders

Units = cylinders of 16065 \* 512 = 8225280 bytes

Disk /dev/dm-2 doesn't contain a valid partition table

虽然/dev/下面没有所谓的dm-3、dm-4....，一般你用nmon、iostat 之类的命令就能看到，其实也可以查看这些dm对应的那些设备，一般都位于/dev/mapper下面。

[root@mylnx01 VolGroup01]# ls -l /dev/mapper

total 0

crw------- 1 root root 10, 62 Dec 7 16:45 control

brw-rw---- 1 root disk 253, 0 Dec 7 16:45 VolGroup00-LogVol00

brw-rw---- 1 root disk 253, 2 Dec 7 16:45 VolGroup00-LogVol00--PS--user--snapshot

brw-rw---- 1 root disk 253, 1 Dec 7 16:45 VolGroup00-LogVol01

brw-rw---- 1 root disk 253, 11 Dec 7 16:45 VolGroup01-LogVol00

brw-rw---- 1 root disk 253, 12 Dec 7 16:45 VolGroup01-LogVol00--PS--user--snapshot

brw-rw---- 1 root disk 253, 9 Dec 7 16:45 VolGroup02-LogVol00

brw-rw---- 1 root disk 253, 10 Dec 7 16:45 VolGroup02-LogVol00--PS--user--snapshot

brw-rw---- 1 root disk 253, 7 Dec 7 16:45 VolGroup03-LogVol00

brw-rw---- 1 root disk 253, 8 Dec 7 16:45 VolGroup03-LogVol00--PS--user--snapshot

brw-rw---- 1 root disk 253, 5 Dec 7 16:45 VolGroup04-LogVol00

brw-rw---- 1 root disk 253, 6 Dec 7 16:45 VolGroup04-LogVol00--PS--user--snapshot

brw-rw---- 1 root disk 253, 3 Dec 7 16:45 VolGroup05-LogVol00

brw-rw---- 1 root disk 253, 4 Dec 7 16:45 VolGroup05-LogVol00--PS--user--snapshot

[root@mylnx01 ~]# ls /dev/VolGroup\*

/dev/VolGroup00:

LogVol00 LogVol00-PS-user-snapshot LogVol01

/dev/VolGroup01:

LogVol00 LogVol00-PS-user-snapshot

/dev/VolGroup02:

LogVol00 LogVol00-PS-user-snapshot

/dev/VolGroup03:

LogVol00 LogVol00-PS-user-snapshot

/dev/VolGroup04:

LogVol00 LogVol00-PS-user-snapshot

/dev/VolGroup05:

LogVol00 LogVol00-PS-user-snapshot

[root@mylnx01 ~]# cd /dev/VolGroup01

[root@mylnx01 VolGroup01]# ll

total 0

lrwxrwxrwx 1 root root 31 Dec 7 16:45 LogVol00 -> /dev/mapper/VolGroup01-LogVol00

lrwxrwxrwx 1 root root 51 Dec 7 16:45 LogVol00-PS-user-snapshot ->

/dev/mapper/VolGroup01-LogVol00--PS--user--snapshot

[root@mylnx01 VolGroup01]#

其实我们可以使用命令dmsetup ls查看dm-0、dm-1、dm-x对应的设备，其中dm后面的数字对应（253，xx）后面的数字。如下所示

[root@mylnx01 ~]# dmsetup ls

VolGroup03-LogVol00--PS--user--snapshot (253, 8)

VolGroup00-LogVol00--PS--user--snapshot (253, 2)

VolGroup05-LogVol00 (253, 3)

VolGroup04-LogVol00 (253, 5)

VolGroup05-LogVol00--PS--user--snapshot (253, 4)

VolGroup03-LogVol00 (253, 7)

VolGroup02-LogVol00--PS--user--snapshot (253, 10)

VolGroup02-LogVol00 (253, 9)

VolGroup01-LogVol00 (253, 11)

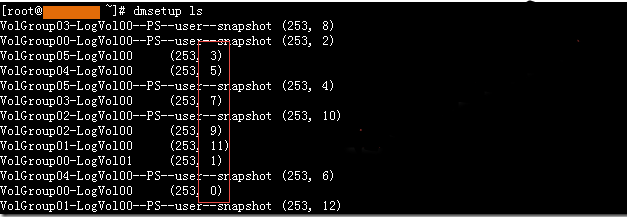
VolGroup00-LogVol01 (253, 1)

VolGroup04-LogVol00--PS--user--snapshot (253, 6)

VolGroup00-LogVol00 (253, 0)

VolGroup01-LogVol00--PS--user--snapshot (253, 12)

[root@mylnx01 ~]#



253后面的数字就对应dm后面的数字，如果你要查看具体的信息就使用命令 dmsetup info 就能看到下面详细信息。

[root@mylnx01 ~]# more /etc/fstab

/dev/VolGroup00/LogVol00 / ext3 defaults 1 1

/dev/VolGroup01/LogVol00 /u01 ext3 defaults 1 1

/dev/VolGroup02/LogVol00 /u02 ext3 defaults 1 1

/dev/VolGroup03/LogVol00 /u03 ext3 defaults 1 1

/dev/VolGroup04/LogVol00 /u04 ext3 defaults 1 1

/dev/VolGroup05/LogVol00 /u05 ext3 defaults 1 1

LABEL=/boot /boot ext3 defaults 1 2

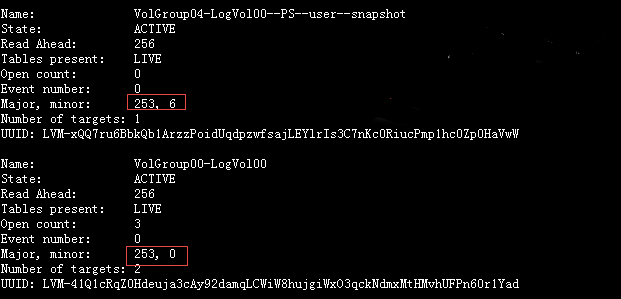
tmpfs /dev/shm tmpfs defaults 0 0

devpts /dev/pts devpts gid=5,mode=620 0 0

sysfs /sys sysfs defaults 0 0

proc /proc proc defaults 0 0

/dev/VolGroup00/LogVol01 swap swap defaults 0 0



有了上面信息我们就能知道dm-0、dm-1、dm-2分别对应下面的一些设备

dm-0对应LVM的 VolGroup00-LogVol00 对应根目录/

dm-1对应LVM的 VolGroup00-LogVol01 对应swap

dm-2对应LMV的 VolGroup00-LogVol00--PS--user--snapshot