存储结构与磁盘划分

1. Linux的目录结构：一切从 / 开始

|  |  |
| --- | --- |
| 目录名称 | 应放置文件的内容 |
| / | 根目录 |
| /boot | 开机所需文件-内核、开机菜单及所需配置文件 |
| /dev | 任何设备与接口都以文件形式存放在此目录 |
| /etc | 配置文件 |
| /home | 用户主目录 |
| /bin | 单用户维护模式下还能够被操作的命令 |
| /lib | 开机时用到的函数库及/bin与/sbin下面命令调用的函数 |
| /sbin | 开机过程中需要的 |
| /media | 一般挂载或删除的设备 |
| /opt | 放置第三方软件 |
| /root | 系统管理员的主文件夹 |
| /srv | 一些网络服务的数据目录 |
| /tmp | 任何人均可使用的”共享”临时目录 |
| /proc | 虚拟文件系统，例如系统内核，进程，外部设备及网络状态等 |
| /usr/local | 用户自行安装的软件 |
| /usr/sbin | 非系统开机时需要的软件/命令/脚本 |
| /usr/share | 帮助与说明文档，也可放置共享文件 |
| /var | 主要存放经常变化的文件，如日志 |
| /lost+found | 当文件系统发生错误时，将一些丢失的文件片段放在这里 |

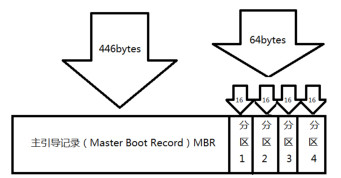
相对路径与绝对路径：

相对路径：相对于当前路径的写法

绝对路径：由根目录(/)开始写起的目录或文件名

1. 物理设备命名规则

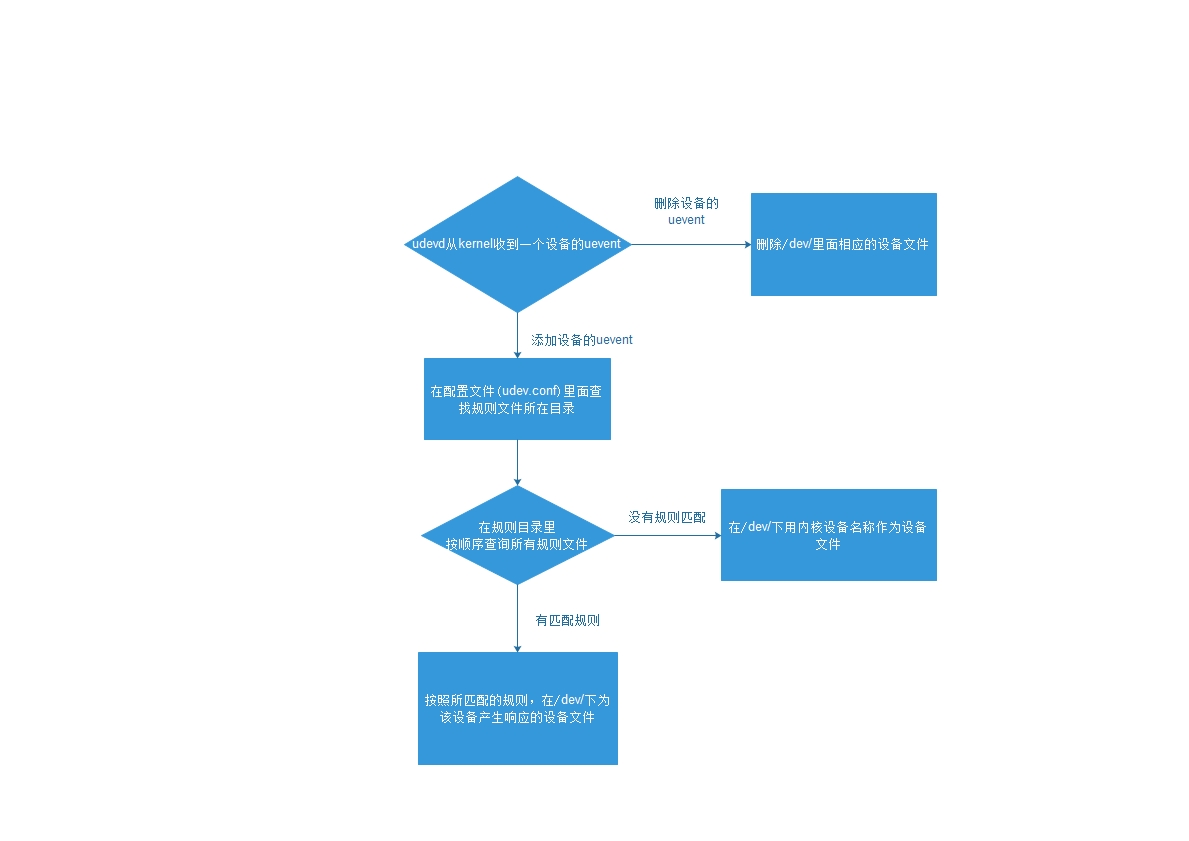
硬盘设备由大量的“扇区”组成的，其中第一个扇区保存着主引导记录与分区表信息。主引导记录需要占用446bytes，分区表的为64bytes，每记录一个分区信息需要16bytes。如图所示：



所以一块硬盘中主分区不能超过四个，一般设置为3个主分区加一个扩展分区，扩展分区中能够创建无限个逻辑分区。

Linux系统中一切都是文件，包括硬件。系统内核的设备管理器(Udev)会自动将硬件名称规范起来，我们可以通过设备名称猜出设备大致的属性以及分区信息等。Udev会一直以守护进程的形式运行并监听来自内核发出的uevent来管理/dev目录下的设备文件。

Udev会根据内核发出的uevent来动态添加或删除/dev目录中的设备文件，命名流程如图所示：



常见的命名规则：

|  |  |
| --- | --- |
| 硬盘设备 | 文件名称 |
| IDE设备 | /dev/hd[a-d] |
| SCSI/SATA/U盘 | /dev/sd[a-p] |
| 软驱 | /dev/fd[0-1] |
| 打印机 | /dev/lp[0-15] |
| 光驱 | /dev/cdrom |
| 鼠标 | /dev/mouse |
| 磁带机 | /dev/st0或/dev/ht0(IDE设备) |

注意：

目前IDE设备已经很少见了，所以一般硬盘设备都是以/dev/sd开头，一台主机上可以有多块硬盘，系统便会用a-p来代表16块不同的硬盘（默认从a开始分配）且分区编号也很讲究。

主分区编号从1开始到4结束，按顺序（也可以指定分配数字）。

逻辑分区从编号5开始按顺序（也可以指定分配数字）。

示例：/dev/sda5

1. /dev 目录下都是硬件设备
2. sd 存储设备
3. a 第一个被识别的设备
4. 5 代表第一个逻辑分区
5. 文件系统与数据资料

文件管理系统的作用是将硬盘合理规划，使得用户能够在上面正常建立文件、写入、读取、修改、转存文件与控制文件。在linux系统中支持超过数十种文件管理系统可供选择，常见的如下：

1. Ext3 是一款日志文件系统能够在异常停机中避免文件系统资料不一致的情况，自动修复数据的不一致与错误，然而一般重整文件系统相当耗费时间（尤其容量大的硬盘），但是也不能保证100%资料不流失。它将会将整个磁盘的写入动作预先记录下来（每个细节），所以在异常停机后可以回溯追踪到被中断的部分。
2. Ext4 Ext3的后续版本，作为RHEL6系统的默认文件管理系统，其支持更大的文件系统到1EB，另外Ext4文件系统能够批量分配block块并作Extends极大的提高读写效率。

当我们拿到一块新的硬盘，进行的操作依次为分区、格式化、最后写入内容。硬盘记录的数据非常多，有一个叫super block 的“硬盘地图”并在上面记录着整个文件系统的信息，但是我们不会把整个数据都直接写在这个大地图中，每个文件的权限和属性都会记录在inode table中（每个文件都会占用一个独立的inode表格，默认为128bytes）上面记录着：

1. 、文件的访问权限（read、write、execute）
2. 、文件的所属主与组
3. 、文件的大小（size）
4. 、文件的创建或状态修改时间（ctime）
5. 、文件的最后一次访问时间（atime）
6. 、文件的修改时间（mtime）
7. 、文件的特殊权限
8. 、文件的真实数据地址（point）

ext4系统的extent

# 术语简介

磁盘块：块设备对磁盘的一个抽象，对于文件系统而言，磁盘就是一个一个连续的块，每个块通常大小为4KB，并且磁盘块有序编码，每个块对应有一个磁盘块号

文件逻辑块号：从逻辑上讲，一个文件可以看做一系列连续的数据块，每个数据块的大小和磁盘块大小相同；

从物理上讲，一个文件可以对应磁盘上若干个磁盘块，这些磁盘块可以在物理上不连续。

逻辑块号和磁盘块号的对应关系，很像虚拟地址和物理地址的关系：文件给你的感觉就是连续的数据，也就是连续的逻辑块，但实际上文件对应的磁盘块是可以不连续的，也就是不连续的物理块

inode：保存文件的元数据，元数据可以描述一个文件

很可惜inode里没有一下子就能想到的filename

一个inode最基本也最重要的信息就两个

用来标识inode的inode号，每个inode都不一样

用来指明这个文件对应着哪些磁盘块的信息——即逻辑块号和磁盘块号的对应关系

* 对于文件系统，每个文件会对应一系列**磁盘块**，通过在**inode**中有序的存放**磁盘块号**，也就保存下了<文件逻辑块号, 磁盘块号>的映射关系
* 一个文件的**逻辑块号**必然是连续的，而磁盘块号则不必连续
* 通常一个block大小为4KB，所以一个比较大的文件，就需要存相当多的块号——而这是一个十分笨拙的办法
* 对于很大的文件，有一种解决办法就是**间接存放块号**，也就是说inode中有部分块号指向的block不是存放这个file的数据，而是存放块号——即一种间接寻址的逻辑。通常会包括三级，一部分直接指向文件的数据块，一部分指向存有块号的块，一部分指向存有“存有块号的块的块号”的块
* 而ext4采用的办法是使用**extent**来保存<文件逻辑块号, 磁盘块号>的映射关系：一个extent对应一系列连续的块号，故可以想到，一个extent最基本的几个域有——文件逻辑块号，起始磁盘块号，块数量
* ext4中一个inode可以直接存放4个extent
* 对于很大的文件，ext4采用extent\_tree的方式，其本质同样是一种间接寻址的关系

# 从文件名到磁盘块

* inode存放文件元数据，但是并没有存放filename——那么ext4是如何把一个filename和一个inode绑定在一起呢？
* 也就是说存在一个filename和inode号之间的对应关系，而这个关系也是存放在一个文件里——目录文件。如根目录/就是一个文件，这个文件也对应一个inode，文件的数据就是根目录下的文件名和对应的inode号

### 简略的过程

基于文件系统，我们看到的就是一个个文件，比如我现在有一个文件/home/xxoo/xxoo.txt

1. 当我们输入cat xxoo.txt时，cat命令接收到xxoo.txt参数，进而根据当前工作目录计算出这个文件的绝对路径为/home/xxoo/xxoo.txt
2. 解析这个路径，首先是/即根目录，根目录这个文件对应的inode号固定为2，所以可以直接找到根目录的inode
3. 根据根目录的inode中存放的磁盘块号信息，可以知道数据存放在哪些磁盘块中，于是从这些磁盘块里读出数据
4. 目录文件的数据，简单的看来就是一个表，有两列，一列是文件名，一列是对应的inode号。对于根目录，文件名就是常见的dev、usr、home等等，于是找到了home对应的inode号
5. 于是读出了/home这个文件的inode，发现这也是一个目录文件，继续读出数据，找到xxoo对应的inode号
6. 于是读出了/home/xxoo这个文件的inode，发现这还是一个目录文件，继续读出数据，找到xxoo.txt对应的inode号
7. 于是读出了/home/xxoo/xxoo.txt这个文件的inode，发现这是一个普通文件，可以使用cat命令，于是读出数据并打印在屏幕上
8. 挂载硬件设备

挂载操作是指当用户需要使用硬盘设备或分区数据时，需要先将其与已存在的目录文件做关联，这个动作就叫挂载。

1. 添加硬件设备

当全新安装了一块新的硬盘设备后，为了更充分、安全的利用硬盘空间首先要进行磁盘的分区，然后格式化，最后挂载。

实验：对新添加的硬盘设备进行分区、格式化并挂载到/xxoo目录

1. 添加硬盘
2. 对硬盘分区

fdisk命令用于管理磁盘分区，格式为 fdisk [磁盘名称]

|  |  |
| --- | --- |
| 参数 | 作用 |
| m | 查看全部可用参数 |
| n | 添加新的分区 |
| d | 删除某个分区信息 |
| l | 列出所有可用的分区类型 |
| t | 改变某个分区的类型 |
| p | 查看分区表信息 |
| w | 保存并退出 |
| q | 不保存直接退出 |

1. 使用fdisk 命令对sdb硬盘进行分区：fdisk /dev/sdb
2. 查看分区信息：p
3. 创建新的分区信息：n
4. 选择主分区：p （p代表主分区 e代表扩展分区）
5. 选择分区编号：1
6. 选择磁盘的起始扇区：直接回车即可
7. 设置分区大小：+5G（代表分区大小为5G）
8. 查看分区表信息：p
9. 保存分区信息：w
10. 格式化为ext4文件系统： mkfs.ext4 /dev/sdb1
11. 将硬盘设备挂在到/xxoo: mkdir /xxoo;mount /dev/sdb1 /xxoo
12. 查看文件系统的使用情况：df -h /xxoo
13. 磁盘冗余阵列（RAID）
14. 什么是RAID

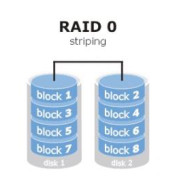
为了避免硬盘的突然损坏导致数据丢失还加入了冗余备份机制。Raid最早的设计理念为不贵的硬盘组，现在的含义为独立的硬盘组。作用是防止硬盘物理损坏以及增加存储设备的吞吐量。

RAID可以根据磁盘的组合方式不同分成不同的级别。

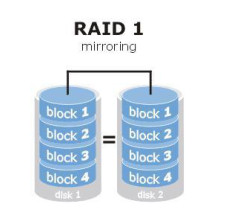
常见的组合有0、1、5和10

1. RAID0需要至少两块（含）硬盘，可以有效的提高硬盘的性能和吞吐量，但没有数据的冗余和错误修复能力。

将多块硬盘通过硬件或软件的方式串联在一起，成为一个大的卷集。将数据依次写入各个硬盘中，这样性能会得到极大的提升，但若任意一块硬盘故障则整个系统的数据都会受到破坏。

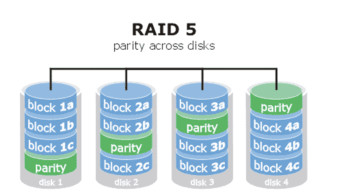


1. RAID1 需要至少两块（含）硬盘，可以有效的提高数据资料的安全性和可修复性，但成本提高了。



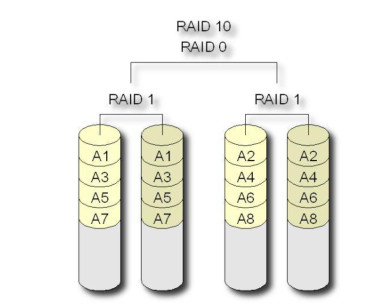
实现原理是在数据写入硬盘的时候也会在另一块闲置的硬盘上生成镜像文件，在不影响性能的情况下最大限度保证数据资料的可靠性，只要在一对镜像盘中还有一块可以使用，那么数据就不会丢失，具有很好的硬盘冗余能力，虽然对数据来讲绝对的安全，但成本却明显增加，磁盘利用率仅为50%。

1. RAID5 需要至少三块（含）硬盘，兼顾存储性能、数据安全和存储成本。



图中的parity块中保存的是其他硬盘数据的奇偶校验信息（并非其他硬盘的数据），以数据的奇偶校验信息来保证数据的安全，RAID5不以单独的硬盘来存放数据的奇偶校验信息，而是保存在各个磁盘上。

当任何一个硬盘数据损坏都可以根据其他硬盘上的奇偶校验信息来尝试重建损坏的 数据，性能也很高，兼顾了存储的性能、数据安全和存储成本，可以看作是RAID0和RAID1的折中方案。

1. RAID10 需要至少四块（含）硬盘，兼顾速度和安全性，但成本高。

继承了RAID0的快速与RAID1的安全，RAID1在这里提供了冗余备份的阵列，而RAID0则负责数据的读写阵列。因为折中结构成本高，一般用于存放要求速度与差别控制的数据。

硬件磁盘阵列与软件磁盘阵列

硬件磁盘阵列是通过磁盘阵列卡来实现的。磁盘阵列卡上面有一块专门的芯片处理RAID的任务。硬件磁盘阵列的效率偏高但是硬件磁盘阵列很贵，所以有了软件磁盘阵列。

软件磁盘阵列主要是通过软件来仿真硬件磁盘阵列，因为是软件所以会损耗较多的系统资源。

mdadm用于管理系统软件RAID磁盘阵列，格式为：

mdadm [模式] <RAID设备名称> [选项] [成员设备名称]

|  |  |
| --- | --- |
| 名称 | 作用 |
| Assemble | 将设备加入到以前定义的阵列 |
| Build | 创建一个没有超级块的阵列 |
| Create | 创建一个新的阵列，每个设备具有超级块 |
| Manage | 管理阵列（如添加和删除） |
| Misc | 允许单独对阵列中的某个设备进行操作（如停止阵列） |
| Follow or Monitor | 监控状态 |
| Grow | 改变阵列的容量和设备数目 |

Mdadm管理RAID阵列的参数有：

|  |  |
| --- | --- |
| 参数 | 作用 |
| -n | 指定设备数量 |
| -l | 指定raid级别 |
| -C | 创建 |
| -v | 现实过程 |
| -f | 模拟设备损坏 |
| -r | 移除设备 |
| -a | 添加设备/检查设备名称 |
| -Q | 查看摘要信息 |
| -D | 查看详细信息 |
| -S | 停止阵列 |

RAID10 配置

1. 在虚拟硬盘中再添加4块硬盘
2. 使用mdadm命令创建RAID10，名称为 /dev/md0

mdadm -Cv /dev/md0 -a yes -n 4 -l 10 /dev/sdb /dev/sdc /dev/sdd /dev/sde

-C 创建

-v 显示创建过程

-a yes 检查RAID名称

-n是用到的硬盘数量

-l 是定义RAID的级别而后面写上要加入阵列的硬盘名称

1. 格式化并挂载使用

1）将RAID磁盘阵列格式化为ext4

mkfs.ext4 /dev/md0

1. 创建挂载目录

mkdir /RAID

1. 进行文件系统的挂载

mount /dev/md0 /RAID

1. 查看磁盘挂载信息

df -h

1. 将此磁盘阵列挂载信息设置为重启后也依然生效

echo “/dev/md0 /RAID ext4 default 0 0” >> /etc/fstab

1. 查看/dev/md0设备信息

mdadm -D /dev/md0

1. 模拟有一块硬盘损坏的情况

将阵列中的/dev/mdb移出

mdadm /dev/md0 -f /dev/sdb

查看阵列此时的状态

mdadm -D /dev/md0

1. 损坏后依然正常使用

因为RAID10 级别能够允许一组RAID1硬盘中存在一个故障而不影响使用，所以依然可以正常的创建或删除文件。现在可以添加新的硬盘，也可将硬盘sdb回复使用

mdadm /dev/md0 -a /dev/sdb

-------------------------------------------------------------------------------------------------------------

1. 逻辑卷管理器

当用户根据实际情况需要对分区增加、减小等调整时，经常会受到硬盘“灵活性”的限制，很不方便。

逻辑卷管理器则是在磁盘分区与文件系统之间添加的逻辑层，提供一个抽象的卷组，使得管理者可以忽略底层磁盘布局，从而实现对分区的灵活动态调整，这毫不夸张，linux系统已经默认启用了LVM（logical volume manager）机制。

物理卷(PV)：整个硬盘设备或使用fdisk命令建立的硬盘分区

卷组(VG)：由一个或多个物理卷组成的整体

逻辑卷(LV)：从卷组切割出的空间，用于创建文件系统，大小由PE的个数决定。

PE(基本单元)：默认为4MB的基本快

|  |  |  |  |
| --- | --- | --- | --- |
| 功能/命令 | 物理卷管理 | 卷组管理 | 逻辑卷管理 |
| 扫描 | pvscan | vgscan | lvscan |
| 建立 | pvcreate | vgcreate | lvcreate |
| 显示 | pvdisplay | vgdisplay | lvdisplay |
| 删除 | pvremove | vgremove | lvremove |
| 扩展 |  | vgextend | lvextend |

实验一：创建一个容量为150M的逻辑卷vo，格式化为ext4并挂载到/mnt/ext4

1. 在虚拟机中添加一块用来做逻辑卷实验的硬盘
2. 创建一个大小为300M的分区（sdb1）标签为lvm
   1. 对硬盘sdb进行分区

fdisk /dev/sdb

* 1. 创建新的分区

n

* 1. 类型为主分区

p

* 1. 分区号为1

1

* 1. 大小为300M

+300M

* 1. 查看分区信息

P

* 1. 修改分区类型

t

* 1. 修改分区类型为lvm(代码为8e)

8e

* 1. 再次查看分区信息

p

* 1. 保存分区信息

w

* 1. 让内核同步分区信息（此步骤仅在没有找到分区设备的情况下才需要执行，非必要动作）

partprobe

1. 启用LVM并创建vo逻辑卷并格式化为ext4格式
2. 将新建的分区设置为物理卷

pvcreate /dev/sdb1

1. 将物理卷加入到卷组（rhcsa）

vgcreate rhcsa /dev/sdb1

1. 查看卷组信息

vgdisplay

1. 生成大小为37个PE的逻辑卷(37\*4MiB 为148M)

lvcreate -n vo -l 37 rhcsa

1. 格式化为ext4

mkfs.ext4 /dev/rhcsa/vo

1. 创建一个名为/rhcsa 的目录用于挂载该逻辑卷

mkdir /rhcsa

1. 挂载硬盘

mount /dev/rhcsa/vo /rhcsa

1. 查看挂载信息

df -h

实验二：将上个实验中的逻辑卷vo容量扩展到290M

1. 对LVM进行调整，一定要先卸载

umont /rhcsa

1. 将逻辑卷扩展到290M

lvextend -L 290M /dev/rhcsa/vo

1. 检查磁盘的完整性，重置硬盘容量

e2fsck -f /dev/rhcsa/vo

resize2fs /dev/rhcsa/vo

1. 重新挂载硬盘设备

mount /dev/rhcsa/vo /rhcsa

1. 查看挂在信息

df -h

实验三：将上个实验中的逻辑卷vo容量减小到120M

1. 卸载文件系统

umount /rhcsa

1. 检查文件系统的完整性

e2fsck -f /dev/rhcsa/vo

1. 将逻辑卷减小到120M

resize2fs /dev/rhcsa/vo 120M

1. 使用lvreduce命令将文件系统调整为120M

lvreduce -L 120M /dev/rhcsa/vo

1. 重新挂载文件系统

mount /dev/rhcsa/vo /rhcsa

1. 查看挂载信息

df -h

实验四：使用逻辑卷快照功能

LVM的逻辑卷快照功能可以将逻辑卷的数据保存为备份、以及快速的数据恢复。

1. 查看逻辑卷详细信息  
    vgdisplay
2. 创建原始文件，写入一行文字

echo “welcome to class one” > /rhcsa/readme.txt

1. 查看/rhcsa下文件列表

ls /rhcsa

1. 对rncsa卷组的vo逻辑卷做一个名称为SNAP而大小为150M的逻辑卷快照

lvcreate -L 150M -s -n SNAP /dev/rhcsa/vo

1. 查看逻辑卷和快照信息

lvs

1. 在逻辑卷中创建一个100M的文件

dd if=/dev/zero of=/rhcsa/files count=1 bs=100M

1. 再次查看逻辑卷快照的使用量

lvs

1. 将文件系统卸载

umount /rhcsa

1. 恢复SNAP逻辑卷快照内容

lvconvert --merge /dev/rhcsa/SNAP

1. 再次查看逻辑卷信息（快照恢复后会被自动删除）

lvs

1. 重新挂载文件系统

mount /dev/rhcsa/vo /rhcsa

1. 查看文件是否被恢复（100M的文件不见了）

ls /rhcsa

1. 磁盘容量配额

磁盘配额（quota）就是限制某个用户或者用户组使用磁盘空间，一旦超出预算就不在允许他们使用磁盘空间。

Quota的磁盘限制可以限制用户的磁盘可用空间或最大文件创建数量。并且还有软/硬限制的区别：

软限制：当达到软限制时会提示用户，但允许用户在规定期限继续使用。

硬限制：当达到硬限制时会提示用户，且强制终止用户的操作。

查看内核是否支持quota功能:

dmesg | grep quota

查看quota程序包是否已经安装:

rpm -q quota

Quota文件系统的支持

1. 磁盘配额需要一个独立文件系统
2. 文件系统为ext2、ext3、ext4
3. 挂载时文件系统必须支持usrquota、grpquota

mount -0 usrquota,grpquota 设备 挂载目录

1. 如果想重启后能够自动挂载，需要加入到/etc/fstab

设备 挂载目录 ext4 rw,usrquota,grpquota 0 0

1. 设置挂载目录权限

Chmod -Rf o+w /boot

实验：限制用户tom创建文件数量

1. 添加用户tom

useradd tom

1. 启用磁盘配额

Quotaon -ucvg 目录

1. 针对用户tom进行磁盘配额

edquota -u tom

4、切换用户tom进行验证