

尹正杰

博客园

首页

联系

订阅

管理

随笔 - 666 文章 - 42 评论 - 41

昵称：尹正杰
园龄：3年7个月
粉丝：238
关注：7
[+加关注](#)

< 2019年3月 >						
日	一	二	三	四	五	六
24	25	26	27	28	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31	1	2	3	4	5	6

- 常用链接
- 我的随笔

我的评论

我的参与

最新评论

我的标签

- 我的标签
- Java基础(106)

Hadoop生态圈(88)

Hadoop进阶之路(50)

python自动化运维之路(49)

系统运维(47)

数据库从入门到精通(43)

每天一个linux命令(37)

GO语言的进阶之路(32)

zabbix(26)

Scala进阶之路(23)

更多

- 阅读排行榜
1. find常用参数详解(61284)

高级Linux运维工程师必备技能（扫盲篇）

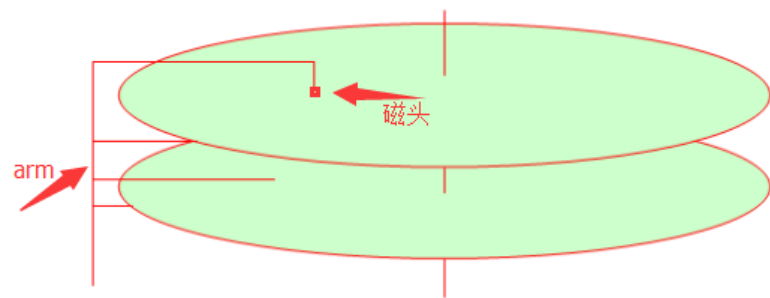
高级

Linux运维工程师必备技能（扫盲篇）

作者：尹正杰

版权声明：原创作品，谢绝转载！否则将追究法律责任。

在了解文件系统之前，我们要学习一下磁盘存储数据的方式，大家都知道文件从内存若要持久化存储的话就得把它存到硬盘上，想毕都知道文件存入磁盘都是二进制存取的。那么硬盘是如何存储的呢？我们现在标配基本上都是1T呢，现在都是2017了，500G的硬盘都已经遭嫌弃了。



1.生活小知识。

上面是一幅图可以看到，有磁头，而且这个磁头是悬浮在盘面上的，如果这个盘面有震动，很可能直接滑到盘面，导致上面写的的数据被划掉，造成数据丢失，因此，我们在购买硬盘的时候都写着不能强剧烈震动。硬盘在工作的时候，这个盘面一直在转，如果访问里面的数据，需要一定时间，我们称之为“平均寻道时间”。把不同盘面上的相同磁道(可以理解就是盘面上的一个圈，整个盘面有好多个这个样的打圈小圈形成的)划分成同一个分区内部。那么这些维护不同盘面的相同编号的磁道我们成为柱面(cylinder).磁盘划分实际上是按照柱面划分的，那么很显然在最外层的柱面划分出来的分区的性能是最好的，你可能会问为什么？原因很简单，就是在相同的角速度中，最外层的周长是最大(也就意味着它存储的数据将越多)。因此我们在划分分区的时候依次划分C,D,E硬盘，最先划分的是C盘(分区软件默认把C盘划分到最外层)，因此大家都用C盘做系统盘是有原因的哟！

2.MBR分区

在整块硬盘的最外侧磁道上，在第零("0")个扇区上,这个扇区是不能用来划分区的，因为这个扇区上存放着整块磁盘的分区信息。这个分区通常被称作为MBR(Master Boot Record,主引导记录)分区(现在都流行GPT分区了)，这个扇区仅仅占用了512个字节(bytes),你可别小看这512bytes，a>.它包函了引导加载器(bootloader,其占用了446bytes)；

2. curl命令的基本使用(23919)
3. OSPF基础介绍(19806)
4. zabbix3.x添加H3C网络设备详解(18572)
5. golang格式化输出-fmt包用法详解(17902)
6. MySQL数据类型以及基本使用详解(17094)
7. HTTP协议和SOCKS5协议(15014)
8. GO语言的进阶之路-网络编程之socket(14227)
9. ISO七层模型详解(12470)
10. zabbix监控企业esxi虚拟机(12255)
11. GoLang基础数据类型--->字典(map) 详解(12214)
12. zip命令的常用选项(9279)
13. Telnet的三种登录方式(8034)
14. Python中包(package) 的调用方式(7729)
15. GO语言的进阶之路-Golang高级数据结构定义(7281)
16. Golang的交互模式进阶-读取用户的输入(6824)
17. parted分区工具用法(6422)
18. zabbix利用自带的模板监控mysql数据库(6361)
19. H3C常用命令详解(6332)
20. iftop命令命令详解(5807)
21. GO语言的进阶之路-面向对象编程(5770)
22. iptables参数详解(5517)
23. GO语言的进阶之路-Golang字符串处理以及文件操作(5061)
24. SVN的Windows和Linux客户端操作详解(4827)
25. Elasticsearch日志分析系统(4777)
26. zabbix监控路由器所有接口信息(4586)

b>.fat,即分区表，其占用了64bytes,每16bytes一个分区，总共只能分区4个[据说比尔盖茨层发表过言论：“硬盘这么小，有谁能用到多余4个分区呢？）】【当时的硬盘都很小而且特别贵，只有几百兆(M)大小】”；

c>.5A,其为十六进制的“5A”，占用最后2个字节，用来标记这个MBR分区是否是有效数据的(2个字节被填充了2个5A，MBR有效性标记)。

硬盘内部都是真空的，为什么呢？因为它要旋转，有的硬盘的转速高达1.5w/m,如果不坐车真空的这么高的转速很容易和真空中的微粒发生碰撞，导致温度过高！所以不要轻易拆开硬盘，基本上你拆开就不是真空了，即使拆开后能用，也用不了多长时间就会坏掉的。

3.扩展分区与逻辑分区

随着硬盘的存储数据进一步的增长，我们分区4个是远远够用的，因此需要从4个主分区中拿出一个分区单独，用来存放其他的分区信息我们叫它为“扩展分区”，如果这个扩展分区足够大，就可以对其进行划分多个分区让不同用户使用。因此这个扩展分区(引用额外的分区表)是不能被格式化的，这样它就不能被使用，需要额外划分出一个或多个逻辑分区才能被使用。

4.MBR分区方法

a>.4个主分区

b>.3个主分区和一个扩展分区。

注意：MBR最大支持2T的硬盘。大于2TB就得使用GPT分区格式！

5.硬盘接口

DMA:Direct Memory Access（直接内存访问机制）

磁盘设备存放于/dev/文件夹下，

IDE接口的磁盘：/dev/hda、/dev/hdb、/dev/hdc、/dev/hdd

SCSI接口的磁盘：/dev/sda、/dev/sdb、/dev/sdc、/dev/sdd

/dev/XdYZ

/dev/ 表示的是一个设备目录

X h IDE硬盘

s SATA、SISC、U盘

Y a 第一块硬盘

b 第二块硬盘

c 第三块硬盘

。 。 。 。 。 。 。 。 。 。

Z 1-3表示主分区，4一般为扩展分区

5是逻辑分区第一个分区

6是逻辑分区第二个分区

。 。 。 。 。 。 。 。 。 。

a>.IDE（ATA）：并口，每个控制器可接两个硬盘，master/slave,133MB/S（这个速率就是被淘汰的根部原因）

/dev/hd[a-z]（注意：在Centos6.x版本以后，所有的硬盘即便是IDE接口的都被识别为sd,早起的设备被设置为hd）

/dev//hda[1-4](标识4个主分区)

/dev/hda[5+]（逻辑分区5开始）

b>.SCSI:Small Computer System Interface（小型计算机接口，在读取数据上效率很高，因为它有单独的SCSI控制器，容错能力强且抗衰老【但是价格贵啊,相同存储空间是机械硬盘的8倍价格呢！】）速率：320mb/s 也是并口的（有的人为了省钱用IDE做raid阵列）

c>.SATA(Serial):300Mbps,600Mbps,6Gbps

27. Linux操作系统原理(4389)
28. Java基础-原码反码补码(4349)
29. zabbix监控windows主机网卡流量(4068)
30. zabbix报警媒介----->微信报警(3994)
31. zabbix通过第三方插件percona监控mysql数据库(3631)
32. brctl创建虚拟网卡详解(3458)
33. CentOS Linux release 7.3破解密码详解(3309)
34. zabbix监控linux文件的一个目录大小(3113)
35. 通过zabbix自带模板监控window sPC机器(3004)
36. GO语言的进阶之路-初探GO语言(2833)
37. Golang的文件处理方式-常见的读写姿势(2788)
38. IP基本原理(2565)
39. Ubuntu16.4的安装过程以及基本配置(2419)
40. GO语言的进阶之路-go的程序结构以及包简介(2391)

d>.SAS:6Gbps
e>.USB：2.0接口： 3.0接口：

6.查看系统是如何识别磁盘分区的




```
1 [root@yinzhengjie ~]# cat /proc/partitions #查看系统识别的分区
2 major minor #blocks name
3
4      8          0 292968750 sda
5      8          16 292968750 sdb
6      9         127 278290432 md127
7    259          0    512000 md127p1
8    259          1 277777408 md127p2
9    253          0 524288000 dm-0
10   253          1 33038336 dm-1
11   253          2 192307200 dm-2
12 [root@yinzhengjie ~]#
```

注意：如果你对你的硬盘剩余的空间进行分区后，新加的分区信息内核是不能识别的，需要内核去重读硬件的分区表，重启是不顶事的！我在生产环境中就遇到过这么一个坑。

7.根在内核

根做为访问文件的入口，那么这个根到底在硬盘上呢？还是在操作系统上呢？根实际上是在内核中，我们访问数据都是操作系统将这个硬盘挂在到了根下，然后我们去访问它而已。那你又会问了，那内核在哪啊？答案是在磁盘上。实际上，在装载内核之前，先启动的是bootloader(内核未启动就还没有文件系统存在),bootloader区磁盘上找到内核并启动，内核启动后会自动生成一个"/"(根)，并将磁盘的文件都挂在到"/"下，这就形成了文件系统。

8.linux目录




```
1 /bin,/sbin #存放系统自身完成自己的启动和基本运行机制所要提供的程序；。
2 /usr/bin,/usr/sbin #存放完成操作系统基本功能的所提供的的二进制程序；
3 /usr/local/bin,/usr/local/sbin #存放第三方案序；
4 /lib,/lib64,/usr/lib,/usr/lib64 #存放库文件的；
5 /etc/ #存放配置文件的；
6 /media,/mnt #用于挂在的目录，当然你也可以自定义的；
7 /dev #存放各种设备文件；
8 /proc,/sys #存放运行中的内存映射数据；
9 /home,/root #存放各普通用户的家目录，比如 /home/yinzhengjie；
10 /var #存放日志的目录；
11 /opt,/misc #触发挂在目录；
12 /srv #存放服务相关数据的；
13 /tmp #存放各种临时文件，每次关机时自动清理；
14 /boot #存放内核、引导菜单等启动文件；
```

 [Linux目录详细版本，猛戳这里！！](#)

9.用来对设备进行分区的命令

用于分区的管理工具：fdisk,sfdisk,parted

a>.查看分区信息



```
1 [root@yinzhengjie ~]# fdisk -l /dev/[sh]d[a-z] #使用文件名通配过滤掉没有用的信息（不是正则表达式哟），可以看出下面只有2个硬盘
2
3 Disk /dev/sda: 21.5 GB, 21474836480 bytes
4 255 heads, 63 sectors/track, 2610 cylinders
```

```
5 Units = cylinders of 16065 * 512 = 8225280 bytes
6 Sector size (logical/physical): 512 bytes / 512 bytes
7 I/O size (minimum/optimal): 512 bytes / 512 bytes
8 Disk identifier: 0x00059922
9
10 Device Boot (是否可引导)      Start      End      Blocks  Id (对应文件系统的ID)
System
11 /dev/sda1  * (*表示可以引导)      1          39      307200   83  Linux
12 Partition 1 does not end on cylinder boundary.
13 /dev/sda2      39      2358    18631680   83  Linux
14 /dev/sda3      2358     2611    2031616   82  Linux swap / Solaris
15
16 Disk /dev/sdb: 10.7 GB, 10737418240 bytes
17 255 heads, 63 sectors/track, 1305 cylinders
18 Units = cylinders of 16065 * 512 = 8225280 bytes
19 Sector size (logical/physical): 512 bytes / 512 bytes
20 I/O size (minimum/optimal): 512 bytes / 512 bytes
21 Disk identifier: 0x00000000
22
23 [root@yinzhengjie ~]#
```

b>虚拟文件系统

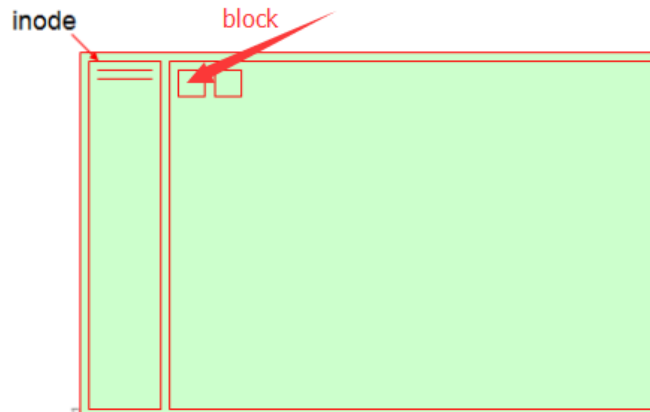
```
VFS: (Virtual File System) #虚拟文件系统
基本文件系统: Ext3, Ext3, Ext4, Reiserfs (早期的suse用的就是该文件系统哟), xfs (支持单个巨大的文件), JFS (日志文件系统, IBM开发的), vfat, NTFS
交换分区: swap
集群文件系统: GFS2 (红帽系统研发, 谷歌都再用呢), OCFS2 (甲骨文公司研发, 用的人不多)
网络文件系统: NFS, smbfs (window是CIFS)
光盘: iso9660
```

c>.对磁盘进行分区

```
1 #!/usr/bin/env python
2 # -*- coding:utf-8 -*-
3 #@author :yinzhengjie
4
#blog:http://www.cnblogs.com/yinzhengjie/tag/python%E8%87%AA%E5%8A%A8%E5%8C%96%E8%BF%90%E7%BB%B4%E4%B9%8B%E8%B7%AF/
5 #EMAIL:y1053419035@qq.com
6
7 '''
8 fdisk:
9     d 删除分区
10     n:新建一个分区
11     p:列出已有分区
12     t:调至分区ID
13     l:列出内核支持的分区id
14     w:保存退出
15     q:不保存退出
16     m:帮助
17 '''
```

- ☐ 用fdisk 工具对磁盘进行分区过程
- ☐ 用kpartx 让系统重读分区表
- ☐ 用fdisk删除分区信息
- ☐ 用fdisk修改分区类型

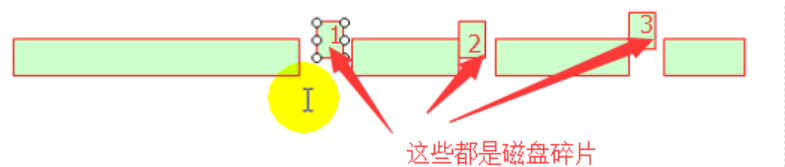
10.创建文件系统



你可以使用2个扇区（512字节）为一个块（block），那么这个一个快的大小就是1kb,4个扇区就是2kb,8个扇区就是4kb.那么问题来了，这个每个单位的块导师是1kb好呢？还是2kb或是4kb好呢？这就要看你存储的数据的大小了，如果你存储的数据是大文件的话，当然block越大越好，这样block存储相同的数据block越大，用的块数就越少，如果你存储的是小文件的话，当然block越小越好，因为block过大，存进去的数据却很小就造成了浪费！（因为一个block只能存储一个数据源）。

注意，在存储数据的时候，一个block只能属于一个文件，不能同时属于2个文件。硬连接除外，其实硬链接指的还是一同一个文件。也就是说，不同的文件不能使用相同的磁盘块。这些磁盘块都有其编号的，是为了方便数据源（主要是inode）只想存储数据的块(block)

磁盘碎片，就是不是连续的块（block）存储着属于同一个文件的数据，这样就导致了在存取的时候特别麻烦，大大的降低了磁盘的工作效率。如下图：



日志文件系统是可以将源数据（indoe）和块数据都写入日志区，等都写入成功了在把数据分别写入到源数据区和块数据区，假如你才写文档的时候，如果突然断电，恰巧你刚刚好写完数据，那么他回将数据分别写入indoe和block区域中，如果你没有写完，他就会对比在日志区的inode和block对应的是否完整，如果不完整，日志区会自动将Inode删除，清除掉不完整的block,就完成了了一次自检模式。



```
#!/usr/bin/env python
# *_coding:utf-8_*
#@author :yinzhenjie
#blog:http://www.cnblogs.com/yinzhenjie/tag/python%E8%87%AA%E5%8A%A8%E5%8C%96%E8%BF%90%E7%BB%B4%E4%B9%8B%E8%B7%AF/
#EMAIL:y1053419035@qq.com

...

mke2fs:
```

```
配置文件：/etc/mke2fs.conf
-t：指定文件类型{ext2|ext3|ext4}
-j：用于创建Ext3文件系统，相当于-t ext3
-L label：指定卷标，
-b{1024|2028|4096}：指定块大小
-i #：#个字节给指定一个indone
-N #：直接指定预留多少个indone
-I #：指定Inode大小
-m #：预留给管理员的空间百分比，默认为5
-O：指定分区特性

e2label /dev/SOMEDEVICE 查看卷标，

e2label /dev/SOMEDEVICE Label 直接更改卷标

e2label /dev/SOMEDEVICE "" 删除卷标

blkid 查看UUID和TYPE

dumpe2fs 查看超级块和是否有碎片
-h：仅显示超级块中保存的信息

tune2fs：调整mke2fs的信息
-l：查看超级块中的信息
-L：设定卷标
-m：预留管理员的空间百分比
-j：如果原来的文件系统为ext2，-j能够将其提升为ext3
-o：[^]mount-options[,...] 指定默认挂载选项
-O：[^]feature[,...] 调整分区特性
tune2fs -o 挂载选项 设备
tune2fs -o ^设备选项 取消

fsck：文件系统检测
-t 文件类型 设备
-f 强行检测
-a 自动修复错误
-r 交互式修复错误

e2fsck
-t 指定时间
-y 自动回答为yes
-f 强行检测
```



用mke2fs的-j参数格式化分区

用mke2fs的-L用法展示

用e2label查看或修改卷标名

mke2fs的-b与-m参数的用法展示

blkid 查看UUID和TYPE

用dumpe2fs 查看超级块和是否有碎片

dumpe2fs的-h参数用法展示

用tune2fs查看超级块中的信息。

tune2fs的对特性的开启与关闭用法展示

用fsck进行文件类型检查

11.创建交换分区



```
1 #!/usr/bin/env python
2 # -*- coding:utf-8 -*-
3 #@author :yinzhengjie
4
#blog:http://www.cnblogs.com/yinzhengjie/tag/python%E8%87%AA%E5%8A%A8%E5%8C%96%E8%BF%9
```

```
0%E7%BB%B4%E4%B9%8B%E8%B7%AF/
5 #EMAIL:y1053419035@qq.com
6
7 '''
8     如果物理内存不够用时，可以将那些最近很少使用的页面数据（Page）置换出去，即切换到硬盘上，但是要注意的是内存文件的格式和硬盘中文件的格式是不一样的，所以这个分区必须格式化成跟内存兼容的模式不能转换成文件的格式。以便
9     把内存的page直接存入这个分区，方便内存直接调用。而这个页面（page）数据对于32位的操作系统一个page大概是4K左右，
10    对于64位操作系统这个page大小是可变的，4k-2M的大小都是比较常见的。事实上到底能使用多大的页面（page）取决于CPU
11    而不取决于内存哟！这就是虚拟内存的概念。在linux上我们称之为交换分区。记住，虚拟内存必须是一个单独的分
12    区。
13
14    那么问题来了：虚拟内存能代替物理内存运行程序吗？
15    答案是否定的，只是使用虚拟内存暂时保存数据，而不是代替物理内存运行程序。
16    虚拟内存的作用是这样的：
17    当运行某个大程序、大游戏，需要的内存超过空闲内存但小于物理内存总量时，会暂时把内存里这些数据放到磁盘
18    上的虚拟内存里，空出物理内存运行游戏。等退出游戏后，又会把虚拟内存里的东西读出来，放回物理内存。所以，虚拟
19    内存，并不是用来虚拟物理内存的，而是暂存数据的。如果对内存的需求大于物理内存总量，那虚拟内存设多大都不管用。
20    电脑内存太低，根本的方法还是增加物理内存，才能流畅。虚拟内存机制上就不管用，即使管用，比物理内存低100倍的速度，也管不上什么实际的作用。所以，虚拟内存大了是没用的，反而白白占用磁盘空间。
21
22    '''
23
24    '''
25    交换分区：
26        mkswap 格式化为虚拟内存
27        -L label 指定卷标
28        swapon 启动虚拟内存
29        -a 启动所有的虚拟分区
30        -p：指定优先级
31        swapoff 关闭虚拟内存
32        更多参数请参考man mkswap
33    '''
```



调整分区为交换分区（swap）格式

用partx 重读一下分区表，避免系统未识别最新分区信息。

用mkswap定义卷标名称

swapon和swapoff的用法展示

12.获取IDE磁盘的相关信息

```
1 #!/usr/bin/env python
2 #_coding:utf-8_
3 #@author :yinzhengjie
4
5 #blog:http://www.cnblogs.com/yinzhengjie/tag/python%E8%87%AA%E5%8A%A8%E5%8C%96%E8%BF%90%E7%BB%B4%E4%B9%8B%E8%B7%AF/
6 #EMAIL:y1053419035@qq.com
7 '''
8 hdparm
9     -i 从操作系统读取
10    -I 直接从硬盘读取
11    -g 显示硬盘的布局信息
12    -t 测试硬盘的性能
13    -T 测试硬盘的性能
14
15    '''
```



hdparm用法展示


13.挂载



```
1 #!/usr/bin/env python
2 #_coding:utf-8_
3 #@author :yinzhengjie
4
5 #blog:http://www.cnblogs.com/yinzhengjie/tag/python%E8%87%AA%E5%8A%A8%E5%8C%96%E8%BF%9
6 #E7%BB%B4%E4%B9%8B%E8%B7%AF/
7 #EMAIL:y1053419035@qq.com
8
9 '''
10 挂载分类：
11     手动挂载
12     按需挂载（autofs，这种效率比较低，当用的时候才去挂载，需要等待时间。）
13     开机自动挂载
14
15 mount[options] -t 文件类型 -o option 设备 挂载点
16     [options]：命令的选项
17         -n:不更新/etc/mtab文件
18         --bind:dir1 dir2 将目录挂载到目录上，使得dir2也能访问dir1的文件
19         -t fstype
20         -r 只读挂载
21         -w：读写挂载
22         -L lable 以卷标指定，也可以使用LABEL="lable"
23         -U UUID：使用UUID挂载，也可以使用 UUID="uuid"
24         -o options:挂载时启动分区特性
25             async：异步I/O
26             sync：同步I/O
27             noatime/atime 是否更新文件时间戳，不是特别重要的文件，建议noatime
28             auto：是否能够被mount -a 自动挂载所有（/etc/fstab中）的文件自动
29             挂载
30             dev/nodev：是否能创建设备文件
31             diratime/nodirtime：是否更新目录的时间戳
32             exec/noexec：是否允许执行二进制程序
33             _netdev:网络设备
34             remount：重新挂载
35             relatime/norelatime 是否实时更新
36             acl 文件访问控制列表
37
38 挂载点：挂载以后原始数据将被隐藏
39     1、选择空闲目录
40     2、必须事先存在
41 卸载：
42     1、空闲时可以卸载
43     2、其实也可以强行卸载
44 umount 设备 | 挂载点
45
46 直接使用mount可以显示当前系统的挂载信息，也可以查看/proc/mounts或者/etc/mtab
47
48 光盘：
49     /dev/cdrom /dev/dvdrom /dev/sr0
50     [-t iso9660]
51 '''
```



	查看内核能识别的文件系统类型
	mount的基本使用展示
	mount命令基于卷标挂载用法展示
	mount命令基于UUID进行挂载用法展示
	mount命令的remount重新挂载用法展示
	mount命令启动acl功能用法展示
	利用tune2fs给mount添加默认acl功能用法展示
	mount命令用只读的方式挂载光盘
	umount卸载报错解决方案展示

 扩展：mount的前世今生

14.查看磁盘信息命令

a>.df 磁盘空间使用状态报告



```
1 #!/usr/bin/env python
2 # -*- coding:utf-8 -*-
3 #@author :yinzhengjie
4
#blog:http://www.cnblogs.com/yinzhengjie/tag/python%E8%87%AA%E5%8A%A8%E5%8C%96%E8%BF%9
0%E7%BB%B4%E4%B9%8B%E8%B7%AF/
5 #EMAIL:y1053419035@qq.com
6 '''
7 [root@yinzhengjie ~]# df #默认是以KB为单位的
8 Filesystem      1K-blocks      Used Available Use% Mounted on
9 /dev/sda2        18208184 2504904  14771696  15% /
10 tmpfs            502172      72    502100    1% /dev/shm
11 /dev/sda1        289293      28473   245460   11% /boot
12 [root@yinzhengjie ~]# df -m #以MB为空间
13 Filesystem      1M-blocks      Used Available Use% Mounted on
14 /dev/sda2        17782      2447   14426    15% /
15 tmpfs            491         1      491     1% /dev/shm
16 /dev/sda1        283         28      240    11% /boot
17 [root@yinzhengjie ~]# df -h #将文件大小显示易读格式
18 Filesystem      Size  Used Avail Use% Mounted on
19 /dev/sda2       18G   2.4G   15G   15% /
20 tmpfs           491M   72K   491M   1% /dev/shm
21 /dev/sda1       283M   28M   240M  11% /boot
22 [root@yinzhengjie ~]# df -i #显示inode数量
23 Filesystem      Inodes IUsed   IFree IUse% Mounted on
24 /dev/sda2       1164592 98347 1066245    9% /
25 tmpfs           125543    3  125540    1% /dev/shm
26 /dev/sda1       76912    38   76874    1% /boot
27 [root@yinzhengjie ~]# df -ih #划算成一度单位，注意下面的M或者K表示的不是文件的大小，而是表示
    的数字K=1000,M=1000000
28 Filesystem      Inodes IUsed IFree IUse% Mounted on
29 /dev/sda2       1.2M   97K   1.1M    9% /
30 tmpfs           123K    3  123K    1% /dev/shm
31 /dev/sda1       76K    38   76K    1% /boot
32 [root@yinzhengjie ~]# df -P #全部显示，就是当Filesystem名称过长的时候，不会换行显示。
33 Filesystem      1024-blocks      Used Available Capacity Mounted on
34 /dev/sda2        18208184 2504908  14771692    15% /
35 tmpfs            502172      72    502100     1% /dev/shm
36 /dev/sda1        289293      28473   245460    11% /boot
37 [root@yinzhengjie ~]# df -Ph #将文件大小显示易读格式且每行信息不换行显示。
38 Filesystem      Size  Used Avail Use% Mounted on
39 /dev/sda2       18G   2.4G   15G   15% /
40 tmpfs           491M   72K   491M   1% /dev/shm
41 /dev/sda1       283M   28M   240M  11% /boot
42 [root@yinzhengjie ~]# df -Ph /dev/sda1 #可以单独查看一个分区情况。
43 Filesystem      Size  Used Avail Use% Mounted on
44 /dev/sda1       283M   28M   240M  11% /boot
45 [root@yinzhengjie ~]#
46 [root@yinzhengjie ~]# df -T #显示时显示文件类型
47 Filesystem      Type  1K-blocks      Used Available Use% Mounted on
48 /dev/sda2       ext4   18208184 2504920  14771680  15% /
49 tmpfs           tmpfs   502172      72    502100    1% /dev/shm
50 /dev/sda1       ext4   289293      28473   245460   11% /boot
51 [root@yinzhengjie ~]#
52 '''
```



b>.du 显示文件占用磁盘的情况



```
1 #!/usr/bin/env python
2 # -*- coding:utf-8 -*-
3 #@author :yinzhengjie
4
#blog:http://www.cnblogs.com/yinzhengjie/tag/python%E8%87%AA%E5%8A%A8%E5%8C%96%E8%BF%9
```

```
0%E7%BB%B4%E4%B9%8B%E8%B7%AF/
5 #EMAIL:y1053419035@qq.com
6 '''
7 [root@yinzhengjie ~]# ls -ldh /etc/ #查看目录大小，很明显不准确！
8 drwxr-xr-x. 102 root root 4.0K May 12 05:48 /etc/
9 [root@yinzhengjie ~]# du -s /etc/ #用du查看文件大小，默认为K.
10 39788 /etc/
11 [root@yinzhengjie ~]# du -sh /etc/ #用du查看目录大小，默认为M
12 39M /etc/
13 [root@yinzhengjie ~]# du -sh /etc/sysconfig/network-scripts/ifcfg-eth0 #我们也可以查看文件的大小
14 4.0K /etc/sysconfig/network-scripts/ifcfg-eth0
15 [root@yinzhengjie ~]#
16
17 '''
```

15.开机自动挂载

```
#blog:http://www.cnblogs.com/yinzhengjie/tag/python%E8%87%AA%E5%8A%A8%E5%8C%96%E8%BF%9
0%E7%BB%B4%E4%B9%8B%E8%B7%AF/
5 #EMAIL:y1053419035@qq.com
6 '''
7 开机自动挂载
8 /etc/rc.d/rc.sysinit:系统初始化脚本
9 其中一个功能：挂载/etc/fstab文件中定义的文件系统挂载点
10 [root@yinzhengjie ~]# cat /etc/fstab
11
12 #
13 # /etc/fstab
14 # Created by anaconda on Tue Apr 11 05:43:15 2017
15 #
16 # Accessible filesystems, by reference, are maintained under '/dev/disk'
17 # See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info
18 #
19 UUID=421dd611-48a1-4a71-9f06-60605f68ef0d / ext4 defaults
20 1 1
21 UUID=003bdf87-a068-4553-b506-f174b1f82ed6 /boot ext4 defaults
22 1 2
23 UUID=1020e7eb-2f17-4a4d-b8bf-36776dcbf547 swap swap defaults
24 0 0
25 tmpfs /dev/shm tmpfs defaults 0 0
26 devpts /dev/pts devpts gid=5,mode=620 0 0
27 sysfs /sys sysfs defaults 0 0
28 proc /proc proc defaults 0 0
29 [root@yinzhengjie ~]#
30 我们看以下挂载的格式，从左往右一次又6列参数，那么这6列参数是啥意思呢？请看以下分析：
31 1>.要挂载的设备：设备文件、LABEL="label" UUID
32 2>.挂载点：有的文件系统没有挂载点 swap没有挂载点，挂载点为swap
33 3>.文件系统类型：
34 4>.挂载选项：多个选项间使用逗号分隔
35 5>.转储频率：
36 0：从不备份
37 1：每日备份
38 2：每隔一天备份
39 6>.自检次序
40 1：首先自检，通常只能被/使用
41 2-9：顺序
42 0：从不自检
43 '''
```

 修改 /etc/fstab配置文件进行开机自动挂载展示

16.查看内存空间使用状态



```

1 #!/usr/bin/env python
2 #_coding:utf-8_
3 #@author :yinzhengjie
4
#blog:http://www.cnblogs.com/yinzhengjie/tag/python%E8%87%AA%E5%8A%A8%E5%8C%96%E8%BF%9
0%E7%BB%B4%E4%B9%8B%E8%B7%AF/
5 #EMAIL:y1053419035@qq.com
6 '''
7 [root@yinzhengjie ~]# cat /proc/meminfo #查看内存信息
8 MemTotal:          1004348 kB
9 MemFree:            671056 kB
10 Buffers:            18764 kB
11 Cached:             126240 kB
12 SwapCached:         0 kB
13 Active:             108700 kB
14 Inactive:           110672 kB
15 Active(anon):        74616 kB
16 Inactive(anon):      1184 kB
17 Active(file):        34084 kB
18 Inactive(file):      109488 kB
19 Unevictable:         0 kB
20 Mlocked:            0 kB
21 SwapTotal:          4136088 kB
22 SwapFree:           4136088 kB
23 Dirty:              164 kB
24 Writeback:          0 kB
25 AnonPages:          74376 kB
26 Mapped:             42396 kB
27 Shmem:              1440 kB
28 Slab:               75132 kB
29 SReclaimable:        15608 kB
30 SUnreclaim:         59524 kB
31 KernelStack:        1560 kB
32 PageTables:         10916 kB
33 NFS_Unstable:        0 kB
34 Bounce:             0 kB
35 WritebackTmp:        0 kB
36 CommitLimit:        4638260 kB
37 Committed_AS:       336672 kB
38 VmallocTotal:       34359738367 kB
39 VmallocUsed:         156112 kB
40 VmallocChunk:       34359567284 kB
41 HardwareCorrupted:  0 kB
42 AnonHugePages:      12288 kB
43 HugePages_Total:    0
44 HugePages_Free:     0
45 HugePages_Rsvd:     0
46 HugePages_Surp:     0
47 Hugepagesize:       2048 kB
48 DirectMap4k:        6144 kB
49 DirectMap2M:        1042432 kB
50 DirectMap1G:        0 kB
51 [root@yinzhengjie ~]# free -m #其实free命令的数据时取自"/proc/meminfo"文件的，知识为了易
    读性可以用free -m参数查看内存使用情况
52              total        used          free      shared    buffers     cached
53 Mem:           980          325           655           1          18          123
54 -/+ buffers/cache:      183           797 #这一行才是正点，其他的都是虚的，实际使用内存为
    183，实际空余内存为797
55 Swap:         4039           0          4039
56 [root@yinzhengjie ~]# free -g #这个参数是以GB为单位，列出磁盘中内存信息，空余查看内存比较大
    的时候用，如果你的内存不深就很少的话就会跟我一样出现下面的尴尬参数了。
57              total        used          free      shared    buffers     cached
58 Mem:             0           0           0           0           0           0
59 -/+ buffers/cache:           0           0
60 Swap:            3           0           3
61 [root@yinzhengjie ~]#
62
63 '''

```



17.dd命令常用参数展示：



```

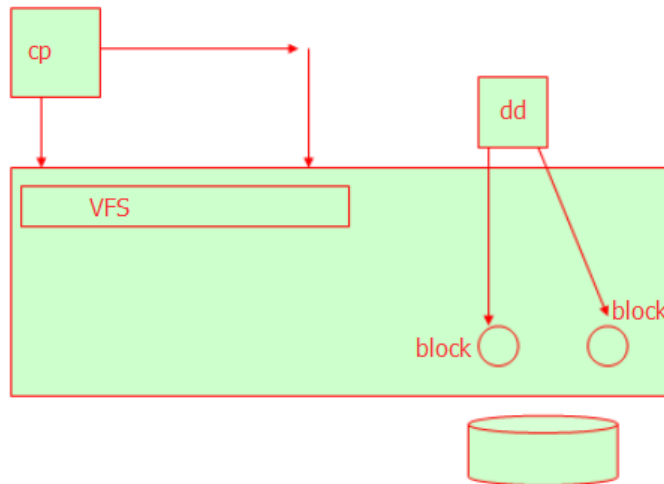
1 #!/usr/bin/env python
2 #_coding:utf-8_
3 #@author :yinzhengjie
4
5 #blog:http://www.cnblogs.com/yinzhengjie/tag/python%E8%87%AA%E5%8A%A8%E5%8C%96%E8%BF%90%E7%BB%B4%E4%B9%8B%E8%B7%AF/
6
7 #EMAIL:y1053419035@qq.com
8
9 '''
10 dd命令:
11     bs:一次读多大的数据量
12     count:读取次数
13     if:输入文件
14     of:输出文件
15     dd if=input_file of=output_file
16     dd if=input_file of=output_file bs=[b|k|m|g] count=#
17     cat /dev/cdrom > /tmp/linux.iso #制作光盘
18 /dev/zero: 吐01的
19 '''

```



利用dd拷贝文件效率要高与cp命令

其实dd相比cp是一个更低级的命令，跟语言一样，语言越高级越接近用户，处理速度就越慢，语言越低级就越接近机器，处理速度就越快，那么为什么dd命令处理速度要相比cp更快呢？其实很简单，cp命令才拷贝文件的时候，先通过VFS系统（即虚拟文件系统）将源数据通通的读取出来，然后在将这些数据传给VFS，并通过文件系统在重新生成一份一模一样的数据才完成了拷贝功能，但是dd命令就不一样了，他绕过了VFS系统，直接找到源数据的inode所对应的blocks,直接将这些blocks内存中拷贝一份，这就是两者的差距。



利用dd命令创建一个swap分区文件，但是不到玩不得不要用！

以下是dd生产环节中常用的几个案例：



```


1 #!/usr/bin/env python
2 #_coding:utf-8_
3 #@author :yinzhengjie
4
5 #blog:http://www.cnblogs.com/yinzhengjie/tag/python%E8%87%AA%E5%8A%A8%E5%8C%96%E8%BF%90%E7%BB%B4%E4%B9%8B%E8%B7%AF/
6
7 #EMAIL:y1053419035@qq.com
8
9 '''
10 linux制作光盘的2种方式:
11 [root@yinzhengjie ~]# dd if=/dev/cdrom of=/yinzhengjie/linux.iso #利用dd命令制作光盘,方法一
12 [root@yinzhengjie ~]# cat /dev/cdrom > /yinzhengjie/linux.iso #利用cat命令制作光盘,方法二
13 linux中的磁盘对拷贝:

```


```

12 [root@yinzhengjie ~]# dd if=/dev/sda of=/dev/sdb                                #即
将/dev/sda设备的信息拷贝一份到/dev/sdb上去。
13 [root@yinzhengjie ~]# dd if=/dev/sda of=/tmp/myfile bs=512 count=1            #
讲/dev/sda的MBR分区进行备份，当分区表破坏时，可以用/tmp/myfile进行还原。
14 [root@yinzhengjie ~]# dd if=/dev/zero of=/dev/sdb bs=512 count=1             #删
除/dev/sdb的分区信息，当然里面的数据还是可以找回来的，可以通过专业的数据恢复工具。
15 [root@yinzhengjie ~]# dd if=/dev/zero of=/yinzhengjie.swapfile bs=1M count=100 #创
建一个100M的本地回环设备，可以用来做分区格式化的（当然得需要一些额外的操作）
16 '''
17
18 #这里只是dd命令的冰山一角，想要了解dd命令更多的用法，请参考man帮助，在哪里或许你能找到你感兴趣的参
数。

```



18.巧记归档命令



```

1  #!/usr/bin/env python
2  # *_coding:utf-8_*_
3  #@author :yinzhengjie
4
#blog:http://www.cnblogs.com/yinzhengjie/tag/python%E8%87%AA%E5%8A%A8%E5%8C%96%E8%BF%9
0%E7%BB%B4%E4%B9%8B%E8%B7%AF/
5  #EMAIL:y1053419035@qq.com
6
7
8  '''
9  文件链接
10     ln 源文件 目标文件
11     硬链接：
12         不能跨分区，
13         指向同一个inode的两个位置
14         不能对目录创建硬链接
15         硬链接会改变文件被链接的次数
16     符号链接：
17         ln -s
18         符号链接可以跨分区
19         符号链接文件跟源文件不同一个inode
20         可以对目录创建符号链接
21         符号链接不会改变源文件被链接的次数
22
23
24 dev 第一个表示主设备号 第二个表示次设备号
25
26 压缩工具
27     zip:
28     gzip:gunzip= gzip -d, zcat
29     后缀:.gz
30     -c 指定要压缩的文件
31     bzip2,bunzip2
32     -k 保留源文件
33     后缀 .bz
34     xz
35     后缀.xz
36     -#：指定压缩比 1-9，默认的为6
37
38
39 归档工具：
40     tar [options] file.tar file1....
41     -c:创建归档
42     -x:展开归档
43     -t:不展开而直接查看被归档的文件
44     -z:使用gzip压缩
45     -j:使用bz2压缩
46     -J:使用xz压缩
47
48
49  '''

```



当你的才华还撑不起你的野心的时候，你就应该静下心来学习。当你的能力还驾驭不了你的目标的时候，你就应该沉下心来历练。问问自己，想要怎样的人生。！！！ 欢迎加入 基础架构自动化运维：598432640，大数据SRE进阶之路：959042252

标签: [系统运维](#)

好文要顶

关注我

收藏该文

尹正杰

关注 - 7

粉丝 - 238

+加关注

00

« 上一篇：[H3C配置FTP服务器](#)
» 下一篇：[parted分区工具用法](#)

posted @ 2017-05-11 13:42 尹正杰 阅读(960) 评论(0) 编辑 收藏

[刷新评论](#) [刷新页面](#) [返回顶部](#)

注册用户登录后才能发表评论，请 [登录](#) 或 [注册](#)，[访问网站首页](#)。

- 【幸运】99%的人不知道我们有可以帮你薪资翻倍的秘笈！
- 【推荐】超50万C++/C#源码：大型实时仿真组态图形源码
- 【推荐】百度云“猪”你开年行大运，红包疯狂拿
- 【推荐】55K刚面完Java架构师岗，这些技术你必须掌握

相关博文：

- [Linux Shel高级技巧\(目录\)](#)
- [linux运维工程师必备技能](#)
- [linux扫盲之CPU模式](#)
- [看Linux内核源码 练内力必备技能](#)
- [Linux系统（三）系统基础扫盲大全](#)

最新新闻：

- [为什么说亚马逊是所有科技公司的终极理想型？](#)
- [三星商城因黑客攻击被薅羊毛 客服：不予发货 补偿699元耳机](#)
- [越来越像传统企业，第一届传统互联网公司诞生了](#)
- [头号玩家马晓轶](#)
- [李笑来登GitHub趋势榜第一：币圈大佬的鸡汤编程指南](#)
- » [更多新闻...](#)