杨亮

文件系统、系统管理

Topics

- 文件系统
- 磁盘配置管理
- 链接
- 进程与资源管理、性能监控
- 例行性命令
- 账号和用户组
- 软件安装

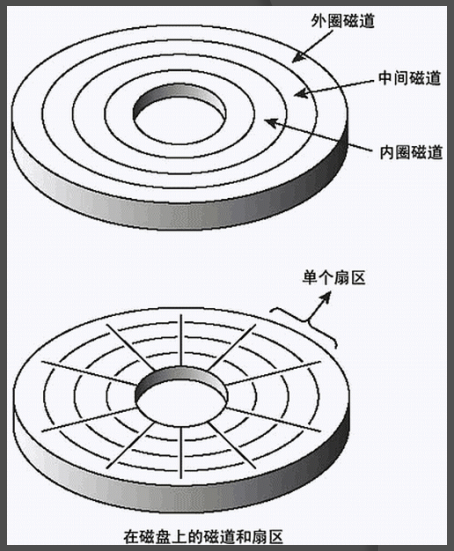
文件系统

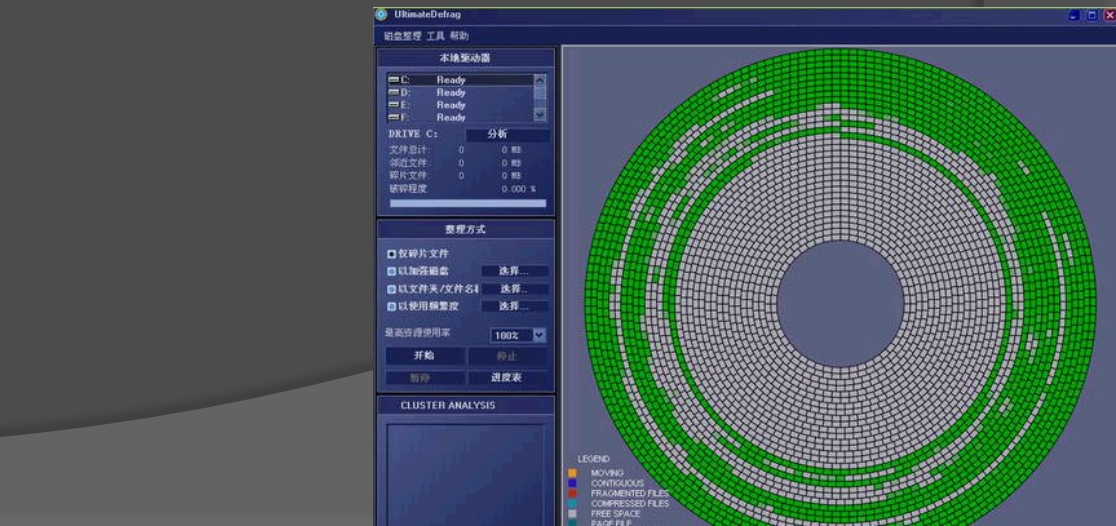
- 操作是同是如何把文件存到硬盘中的扇区？
- 这些文件时如何管理的？
- 目录和文件有什么不同么？
- 如果根据路径查找文件？

这不是操作系统原理课程
所以不会考虑很多高级的方法
只是告诉你Linux是如何实现的

关于磁盘你应该知道的

- 硬盘的最小存储单位是扇区
- 一个扇区512B（15-20行代码）
- 磁头是一个扇区一个扇区读取的
- 一个10M的文件要读取2048次





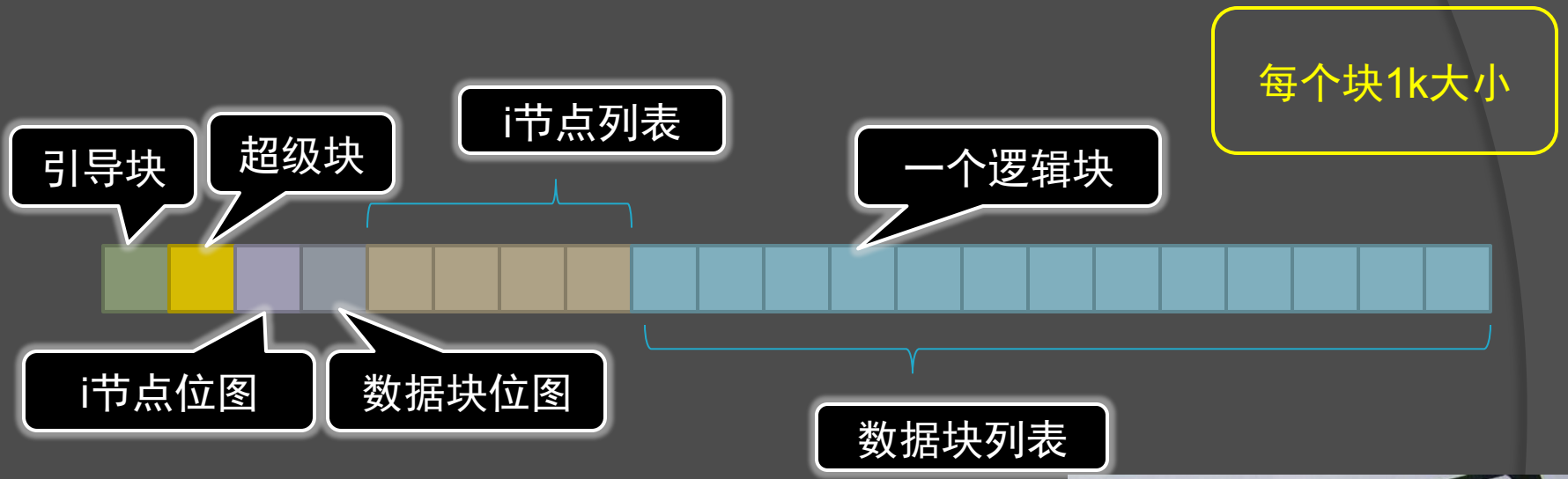
逻辑块（数据块）

4KB

- 为了提高读取效率可以将多个扇区同时读取
- 在系统分区格式化的时候指定逻辑块的大小，逻辑块成为最小的存储单元
- 以扇区为基础，逻辑块大小应该是2^n个扇区
- 一个逻辑块最多可容纳一个文件（ext2），产生磁盘的浪费
  - 如果作代码服务器（CVS），逻辑块可以小一些
  - 如果作为视频服务器，逻辑块应该稍微大一些

- 一个磁盘可以分为多个分区
- 每个分区含有且只含有一个文件系统

Minix文件系统（360k 软盘）



引导块：如果此文件系统包括操作系统，则存放操作系统的引导信息或系统启动代码
超级块：文件系统结构信息，说明各部分大小
i节点列表：类型、访问权限等信息（不包括文件名）
数据块列表：存放文件的真实数据
i节点位图：指明哪些i节点可用
数据块位图：指明哪些数据区的逻辑块可用

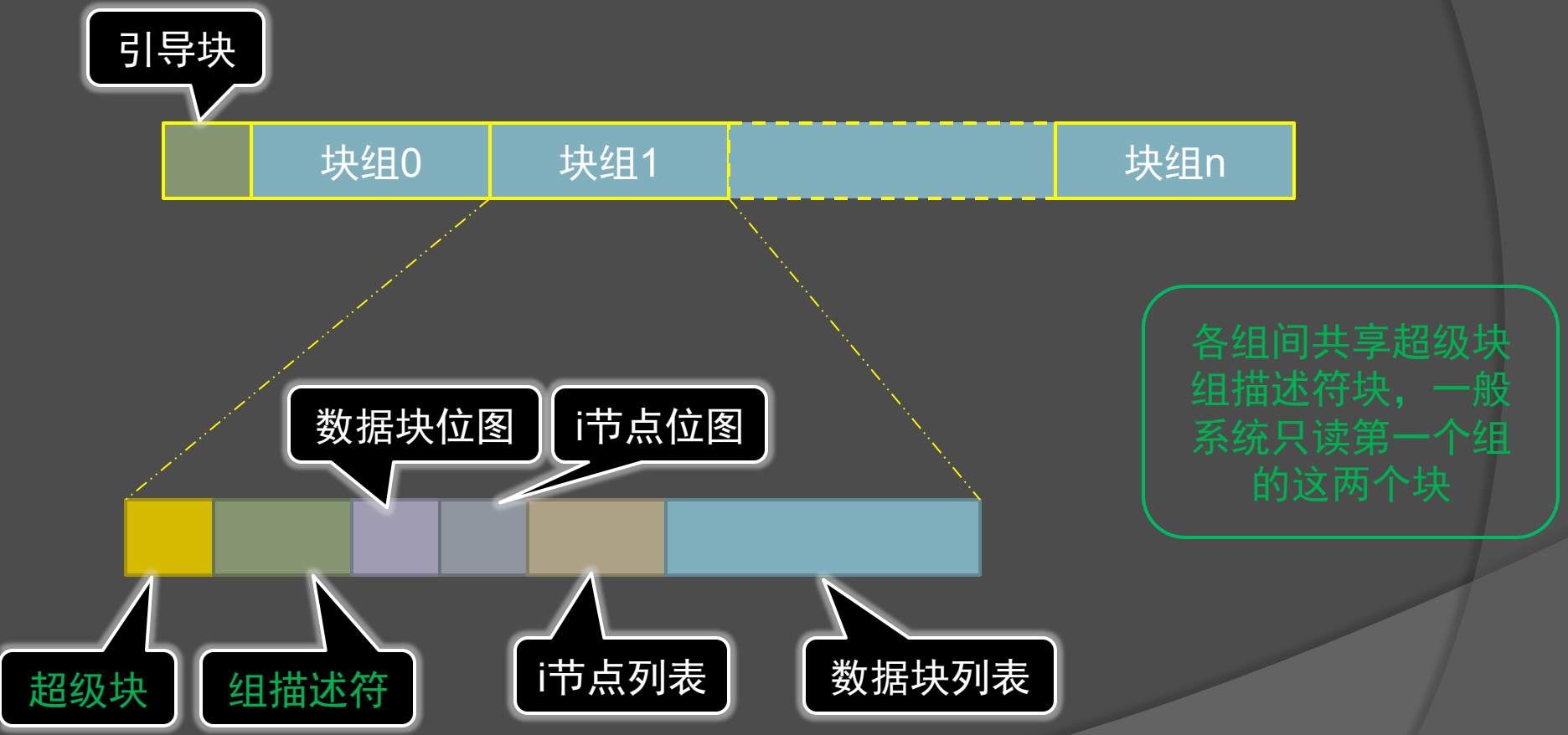
超级块结构

- 整个文件系统相关信息
- block与inode的总量、已使用量、未使用量
- block与inode的大小
- 文件系统创建时间、写入数据时间、最近一次检索磁盘信息

i节点（inode）结构

- 文件的权限、及特殊标志位
- 所有者、与组信息
- 文件大小
- 创建时间、最近读取时间、修改时间
- 文件内容block指针
- 128byte
- 一个文件一个inode

ext2文件系统


引导块
块组0
块组1
块组n
数据块位图
i节点位图
超级块
组描述符
i节点列表
数据块列表
各组间共享超级块组描述符块，一般系统只读第一个组的这两个块

- 组描述符
  - 每个group的开始和结束的block号
- inode bitmap
  - 确定哪些inode被占用，哪些是空的
  - 删除文件删除相应的 bitmap项而不是文件内容
- block bitmap
  - 与inode bitmap类似

查看文件系统结构

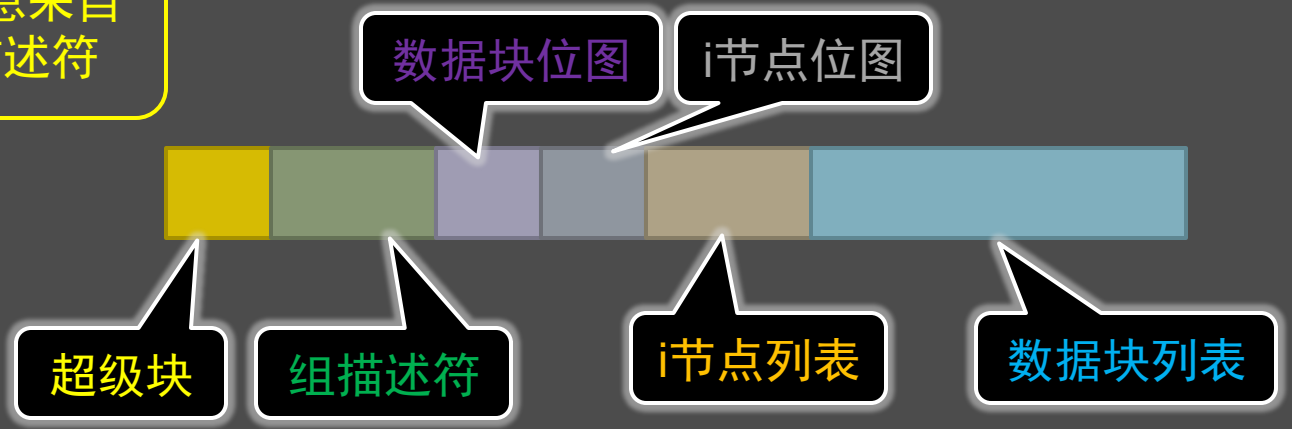
- dumpe2fs

这些信息来自于超级块

```
[root@fedora dev]# dumpe2fs sda1
dumpe2fs 1.41.10 (10-Feb-2009)
Filesystem volume name:   <none>
Last mounted on:          /boot
Filesystem UUID:          e2d5aae8-27d2-4158-a089-03f5f92db695
Filesystem magic number:  0xEF53
Filesystem revision #:    1 (dynamic)
Filesystem features:      has_journal ext_attr resize_inode dir_
_isize
Filesystem flags:         signed_directory_hash
Default mount options:    user_xattr acl
Filesystem state:         clean
Errors behavior:          Continue
Filesystem OS type:       Linux
Inode count:              128016
Block count:              512000
Reserved block count:     25600
Free blocks:              451895
Free inodes:              127974
First block:              1
Block size:               1024
Fragment size:            1024
Reserved GDT blocks:      256
Blocks per group:         8192
Fragments per group:      8192
Inodes per group:         2032
Inode blocks per group:   254
Flex block group size:    16
Filesystem created:       Thu Aug  5 06:51:01 2010
Last mount time:          Wed Aug 25 05:59:09 2010
Last write time:          Wed Aug 25 05:59:09 2010
```

```
Group 0: (Blocks 1-8192) [ITABLE_ZEROED]
  校验和 0x06ea,2010个未使用的inode
  主 superblock at 1, Group descriptors at 2-3
  保留的GDT块位于 4-259
  Block bitmap at 260 (+259), Inode bitmap at 276 (+275)
  Inode表位于 292-545 (+291)
  5814 free blocks, 2011 free inodes, 2 directories, 2010个未使用的inodes
  可用块数: 4379-8192
  可用inode数: 16, 23-2032
```

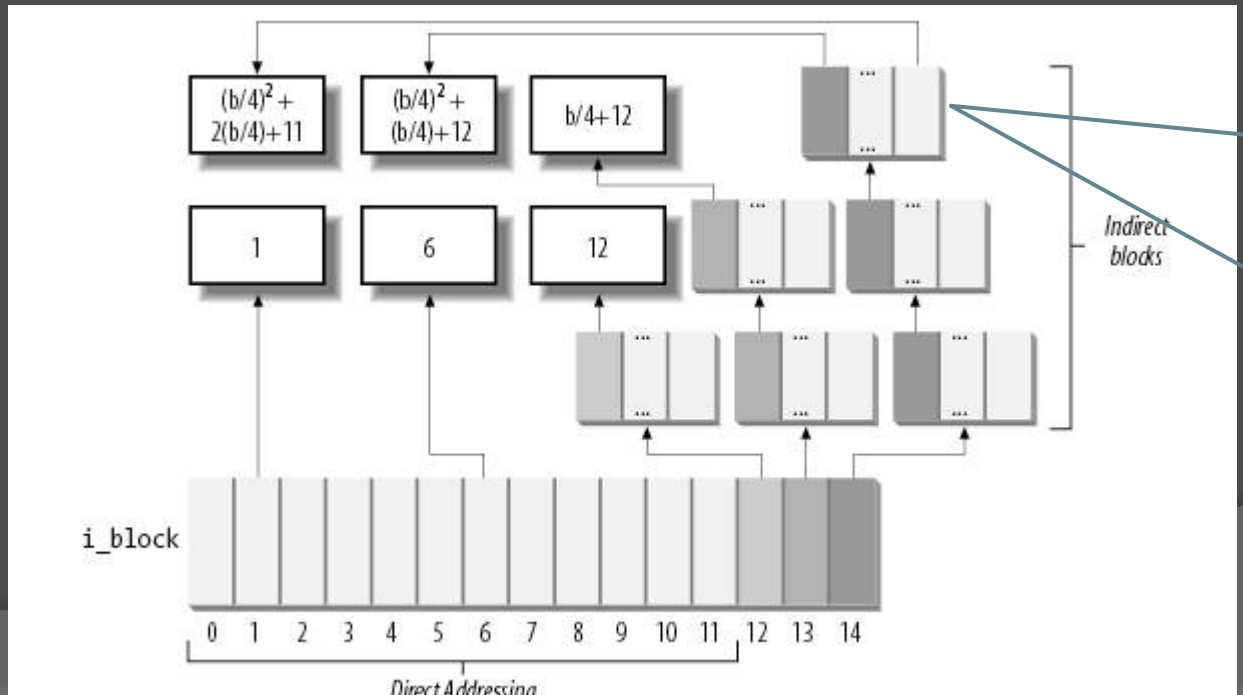
这些信息来自于组描述符




```
Group 1: (Blocks 8193-16384) [INODE_UNINIT, ITABLE_ZEROED]
  校验和 0x9ece,2032个未使用的inode
  备份 superblock at 8193, Group descriptors at 8194-8195
  保留的GDT块位于 8196-8451
  Block bitmap at 261, Inode bitmap at 277
  Inode表位于 546-799
  443 free blocks, 2032 free inodes, 0 directories, 2032个未使用的inodes
  可用块数: 11916-12288, 16315-16384
  可用inode数: 2033-4064
```

文件的多重索引存储结构

- inode中包含文件和磁盘的关联管理
- _u32 iblock[EXT2_NBLOCKS]
- 随着系统的不断升级，EXT2_NBLOCKS也在不断的变大，9 -> 12 -> 15



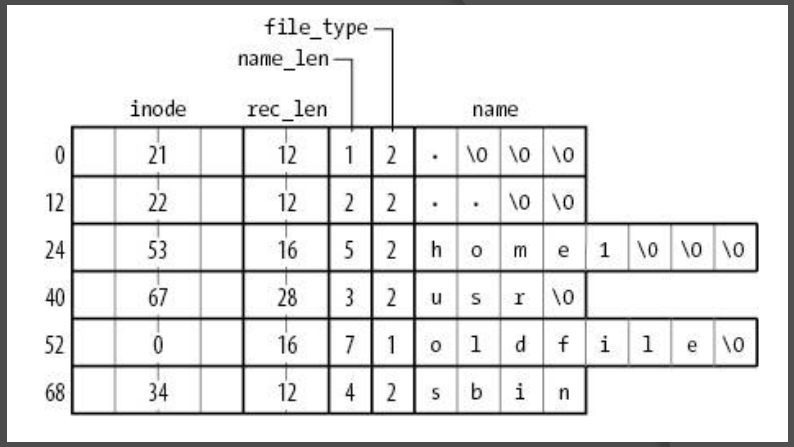
每个间接块也占用一个逻辑块，由于每个逻辑块的编号需要4个字节，所以假设块的大小是b，那么每二个中转块可以索引b/4个节点

不同的文件块大小

Block size	Direct	1-Indirect	2-Indirect	3-Indirect
1,024	12 KB	268 KB	64.26 MB	16.06 GB
2,048	24 KB	1.02 MB	513.02 MB	256.5 GB
4,096	48 KB	4.04 MB	4 GB	~ 4 TB

到现在都一直在说文件的存储
那目录是如何存储的呢
如果没有目录关系
整个系统就是一个平的系统，而不是树

目录



- 目录也是一种文件
- 他的文件内容就是他之内的文件的文件名和文件的inode（i节点）
- 每个目录中默认有两个文件
  - . 默认是所在的目录本身 （cd .）
  - .. 默认是所在目录的上一层目录 （cd ..）

Linux中一切物体都是文件，目录是文件，管道是文件，设备也是文件

```
[yangliang@fedora conn]$ ls -idl /home/yangliang/
131100 drwx------. 31 yangliang yangliang 4096  8月  14 23:19 /home/yangliang/
[yangliang@fedora conn]$ cd /home/yangliang/conn/
[yangliang@fedora conn]$ ls -ial /home/yangliang/conn/
总用量 16
131671 drwxrwxr-x.  4 yangliang yangliang 4096  8月  14 23:23 .
131100 drwx------. 31 yangliang yangliang 4096  8月  14 23:19 ..
131679 drwxrwxr-x.  2 yangliang yangliang 4096  8月  14 23:20 subconn
131681 drwxrwxr-x.  2 yangliang yangliang 4096  8月  14 23:23 subconn2
```

文件名只保存在相应的目录文件中

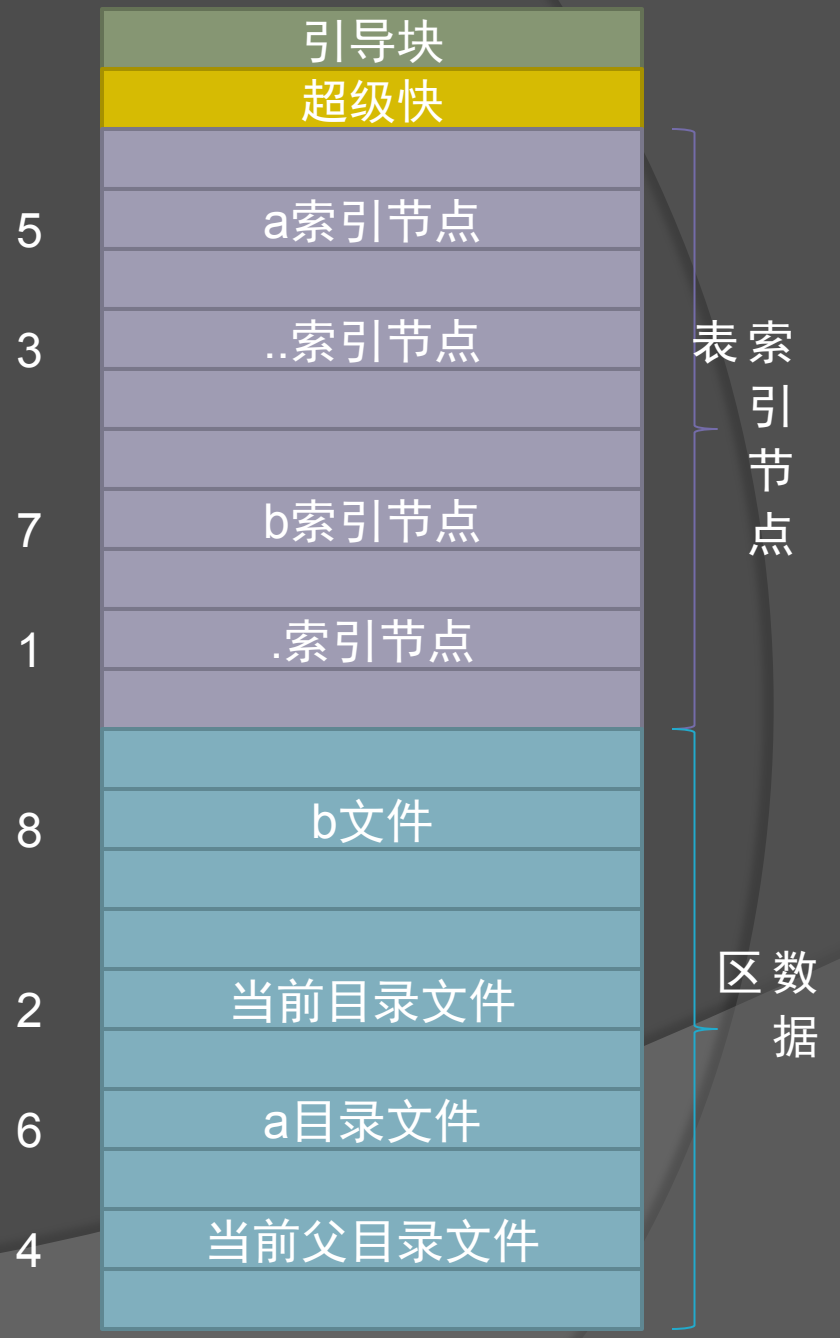
文件的创建过程

- 申请inode，设置相应的inode位图
- 将文件名和inode写入相应的目录文件中
- 申请block，设置相应的inode位图
- inode指向block
- 将文件内容写入block中

文件的检索过程

- ../a/b

当前所在目录的inode节点号



磁盘配置管理

- df 列出磁盘系统的整体使用量
  - -a 列出所有的文件系统
  - -k -m 以KB和 MB的大小显示
  - -h 以人们较为易读的G M K 显示文件大小
  - -T 显示该分区的文件系统的类型
  - -i 不用硬盘容量，而是用i节点的数量表示

```
[yangliang@fedora ~]$ df
文件系统                1K-块      已用      可用 已用% 挂载点
/dev/mapper/vg_fedora-lv_root
                     14965776   2633712  12180080  18% /
tmpfs                  254968       272    254696   1% /dev/shm
/dev/sda1              495844     43949    426295  10% /boot
[yangliang@fedora ~]$ df -i
文件系统              Inode  已用(I)  可用(I) 已用(I)%% 挂载点
/dev/mapper/vg_fedora-lv_root
                     950272   99286  850986   11% /
tmpfs                 63742       6   63736    1% /dev/shm
/dev/sda1            128016      42  127974    1% /boot
```

Linux中很多东西都是文件，目录也是文件，设备也是文件，内核数据（进程信息）也是文件，管道也是文件，socket也是文件

```
[yangliang@fedora ~]$ df -aTh
文件系统     类型      容量  已用  可用 已用% 挂载点
/dev/mapper/vg_fedora-lv_root
               ext4     15G  2.6G   12G  18% /
proc           proc        0     0     0    -  /proc
sysfs          sysfs       0     0     0    -  /sys
devpts         devpts      0     0     0    -  /dev/pts
tmpfs          tmpfs    249M  272K  249M    1% /dev/shm
/dev/sda1      ext4     485M   43M  417M  10% /boot
none    binfmt_misc        0     0     0    -  /proc/sys/fs/binfmt_misc
gvfs-fuse-daemon
fuse.gvfs-fuse-daemon        0     0     0    -  /home/yangliang/.gvfs
```

我们后面会解释proc tmpfs /dev/sda1用途

- du
  - -a -h -k -m的含义同df命令的一致
  - -s 只列出此路径的内容大小的合计
  - -S 只计算文件夹下的文件大小，不累计计算其内的子目录的大小，

```
[yangliang@fedora ~]$ du -s ./
93776   ./
[yangliang@fedora ~]$ du ./
84      ./.cache/ibus/bus
68      ./.cache/ibus/pinyin
156     ./.cache/ibus
172     ./.cache
4       ./testmask/dir1
4       ./testmask/dir2
12      ./testmask
```

```
[yangliang@fedora ~]$ du -S  ./
84      ./.cache/ibus/bus
68      ./.cache/ibus/pinyin
4       ./.cache/ibus
16      ./.cache
4       ./testmask/dir1
4       ./testmask/dir2
4       ./testmask
```

为什么文件夹的大小是4k？

156 = 84 + 68 + 4

12 = 4 + 4 + 4

du、df区别

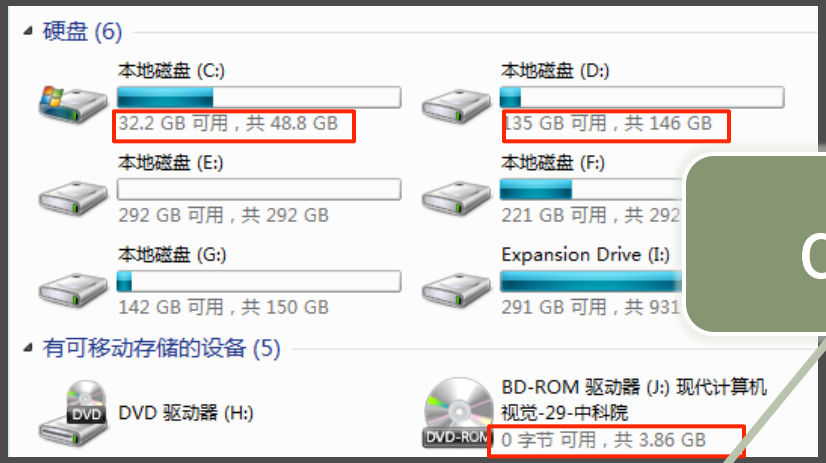
- df查询整个系统的每个文件系统使用情况
- du查询某些特定目录占用硬盘的情况

time cmd --- 用来测试脚本、命令指示时间

```
/dev/mapper/vg_fedora-lv_root
                      14965776   2633716  12180076  18% /
tmpfs                   254968       432    254536   1% /dev/shm
/dev/sda1               495844     43949    426295  10% /boot

real    0m0.004s
user    0m0.001s
sys     0m0.003s
```

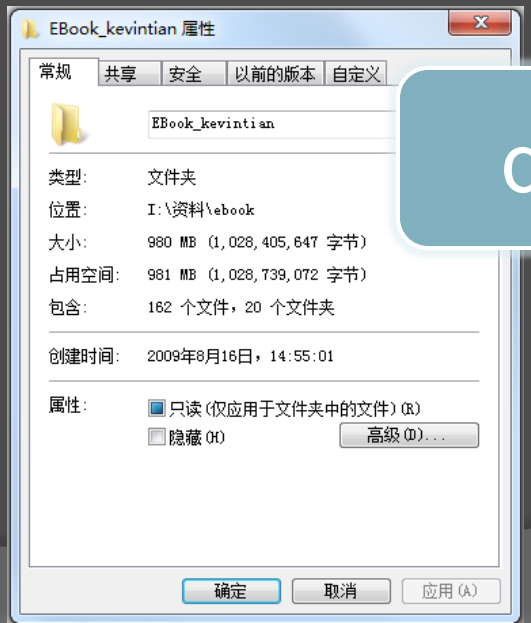
```
[yangliang@fedora ~]$ time du -s /tmp
du: 无法读取目录"/tmp/pulse-hIcPUYkTy8G2": 权限不够
du: 无法读取目录"/tmp/orbit-gdm": 权限不够
68      /tmp

real    0m0.035s
user    0m0.001s
sys     0m0.013s
```

为什么查询df的速度要快于查询某些特定文件夹的速度



df

```
[yangliang@fedora ~]$ df
文件系统                  1K-块      已用      可用 已用% 挂载点
/dev/mapper/vg_fedora-lv_root
                      14965776   2633712  12180080   18% /
                        254968       272    254696    1% /dev/shm
a1                      495844     43949    426295   10% /boot
ang@fedora ~]$ df -i
                      Inode  已用(I)  可用(I) 已用(I)%% 挂载点
apper/vg_fedora-lv_root
                      950272   99286  850986   11% /
tmpfs                  63742       6   63736    1% /dev/shm
/dev/sda1             128016      42  127974    1% /boot
```



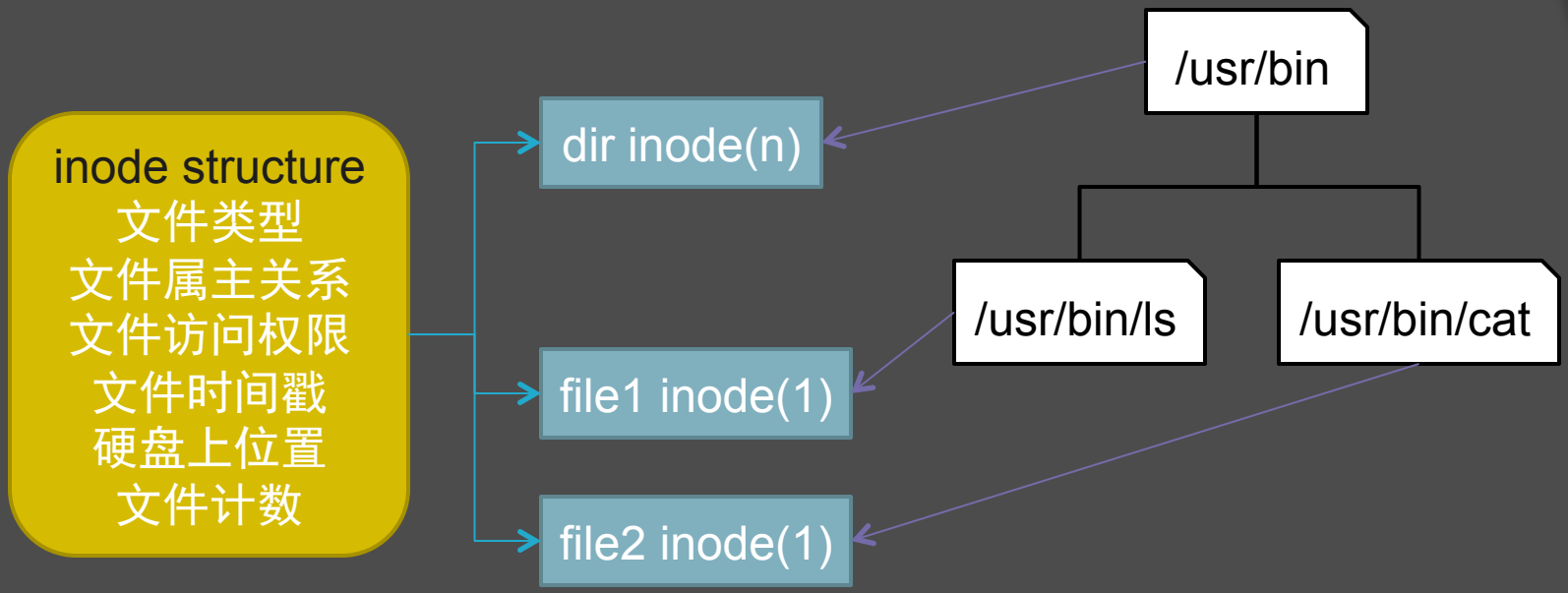
du

```
[yangliang@fedora ~]$ du /home/yangliang/
84      /home/yangliang/.cache/ibus/bus
        /home/yangliang/.cache/ibus/pinyin
        /home/yangliang/.cache/ibus
        /home/yangliang/.cache
        /home/yangliang/testmask/dir1
        /home/yangliang/testmask/dir2
12      /home/yangliang/testmask
```

df只查询超级块就可以得到所有信息
du就像查询其内的所有文件一样遍历相应的inode和数据块

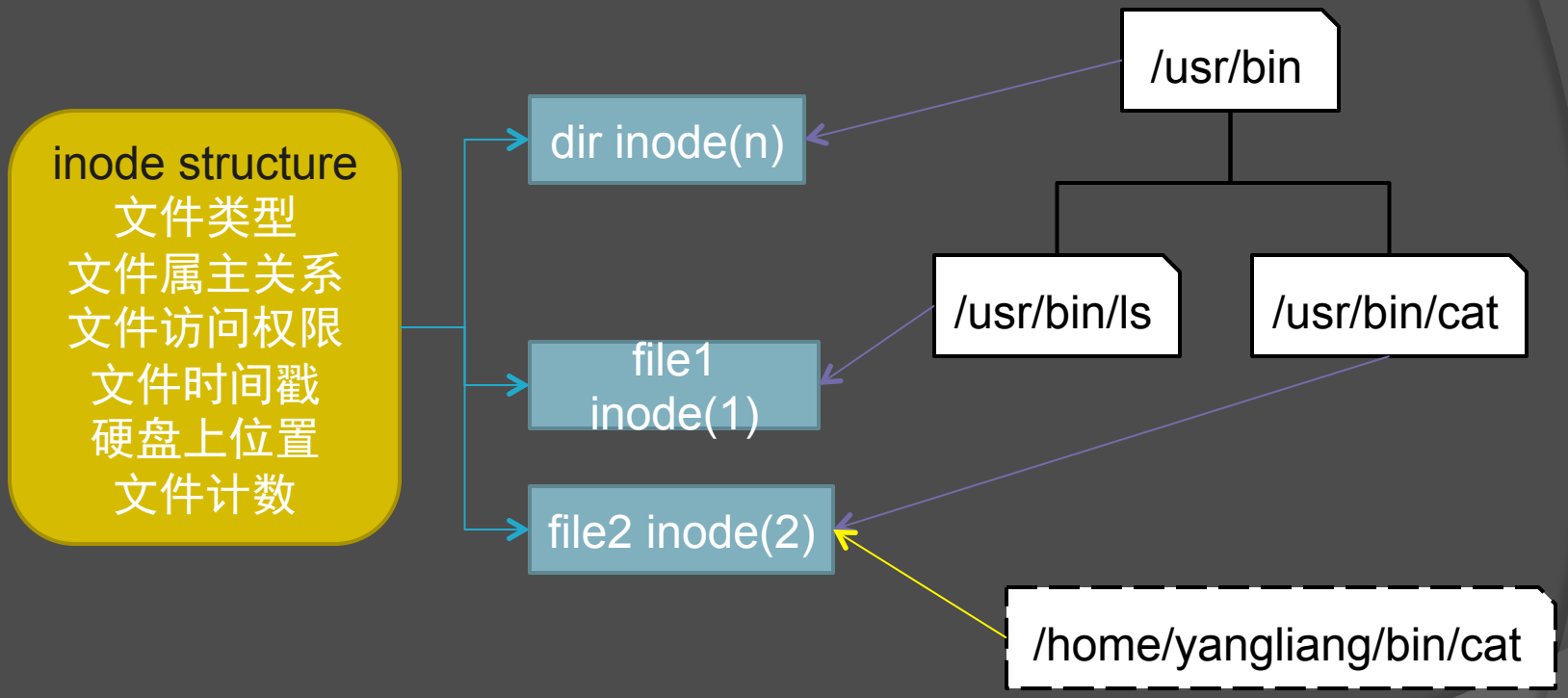
- dd
  - 拷贝一个文件并进行相应的转换
  - if=输入文件或者设备名称
  - of=输出文件或者设备名称
  - ibs obs bs 一次读取、写入的字节数
  - skip=n 跳过前n块
  - count=n 只拷贝n块
  - 主要用于备份文件、磁盘（光盘）复制等
  - dd if= /dev/cdrom of=cdrom.img

文件结构




```
[yangliang@fedora ~]$ ls -il
总用量 248
131647 lrwxrwxrwx. 1 yangliang yangliang      7  8月  11 15:42 alink -> iamroot
131657 -rw-rw-r--. 1 yangliang yangliang      0  8月  11 21:45 file2
131448 -rw-r--r--. 1 yangliang yangliang    781  8月   6 05:53 fstab
131640 -rw-r--r--. 1 root      root           0  8月  11 15:29 iamroot
131662 -rw-rw-r--. 1 yangliang yangliang  10088  8月  11 22:55 lsman
131413 drwxrwxr-x. 2 yangliang yangliang   4096  8月   6 21:35 pdf
131631 -rw-rw-r--. 1 yangliang yangliang      9  8月   6 22:59 test
131641 drwxrwxr-x. 3 yangliang yangliang   4096  8月  11 16:15 test1
131658 -rw-rw-r--. 1 yangliang yangliang      0  8月  11 21:36 test2
131198 drwx------. 2 yangliang yangliang  12288  8月   6 21:12 Zhi-Hua Zhou's Publications
131645 -rwxr-xr-x. 1 yangliang yangliang 177362  8月  11 15:12 (ZhouZhihua)-ecml03.pdf
```

硬链接



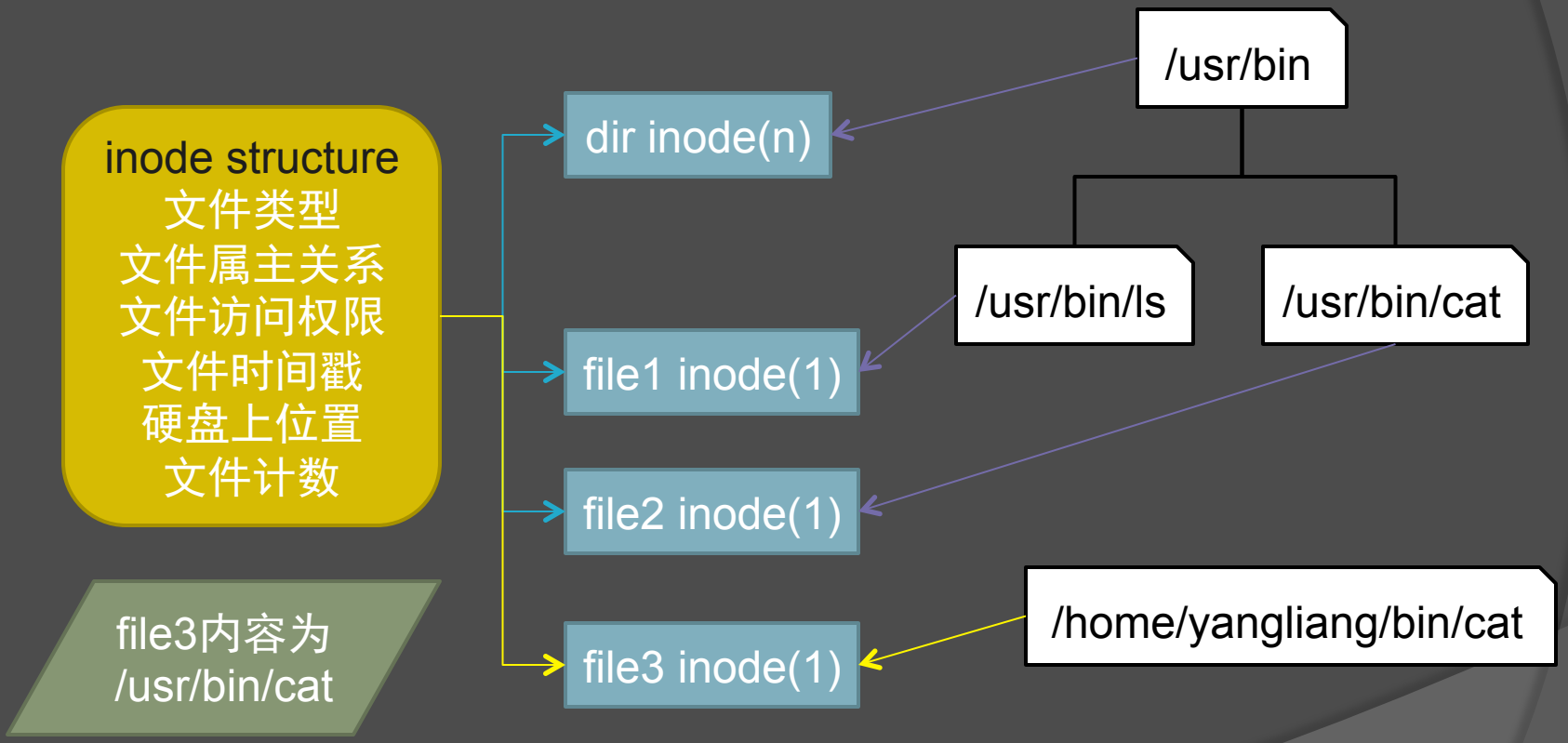
ln –d /usr/bincat /home/yangliang/bin/cat

硬链接性质

- 只能在同一个文件系统中
- 硬链接指向同一个inode，每次连接增加一个计数，每次删除减少一次计数
- 当inode的计数为0，则删除此文件
- 修改一个文件另一个文件随之改变
- cp –l file1 linker

```
[yangliang@fedora test2]$ ls -il
总用量 0
131644 -rw-rw-r--. 1 yangliang yangliang 0  8月  11 15:04 file2
131659 -rw-rw-r--. 1 yangliang yangliang 0  8月  11 22:20 file3
[yangliang@fedora test2]$ cp file2 file4
[yangliang@fedora test2]$ cp -l file2 file5
[yangliang@fedora test2]$ ls -il
总用量 0
131644 -rw-rw-r--. 2 yangliang yangliang 0  8月  11 15:04 file2
131659 -rw-rw-r--. 1 yangliang yangliang 0  8月  11 22:20 file3
131663 -rw-rw-r--. 1 yangliang yangliang 0  8月  11 22:20 file4
131644 -rw-rw-r--. 2 yangliang yangliang 0  8月  11 15:04 file5
```

软链接（符号链接）



ln –s /usr/bincat /home/yangliang/bin/cat

符号链接性质

- 可以跨文件系统
- 可以对目录和文件建立符号链接
- 可以对不存在的文件创建符号链接
- 删除原文件，链接失效
- 类似于windows的快捷方式
- cp –s file1 linker

```
[yangliang@fedora test2]$ cp -s file2 file6
[yangliang@fedora test2]$ ls -il
总用量 0
131644 -rw-rw-r--. 2 yangliang yangliang 0  8月  11 15:04 file2
131659 -rw-rw-r--. 1 yangliang yangliang 0  8月  11 22:20 file3
131663 -rw-rw-r--. 1 yangliang yangliang 0  8月  11 22:20 file4
131644 -rw-rw-r--. 2 yangliang yangliang 0  8月  11 15:04 file5
131664 lrwxrwxrwx. 1 yangliang yangliang 5  8月  11 22:28 file6 -> file2
```

复制、软连接、硬链接区别

- 复制
  - 新建一些数据块，再建一个inode指向这些数据块，再把这个新文件的inode号和文件名添加到所在目录的目录文件
- 硬连接
  - 将原有文件的inode中的计数+1，在相应的目录的目录文件中加入这个inode号和新的文件的文件名
- 软链接
  - 新建一个数据块，数据块的内容是源文件的地址，新建一个inode指向这个数据块，并把新的inode号和新的文件名写入相应目录的目录文件

目录的连接数

普通文件的连接数是1

为什么是30？

```
[yangliang@fedora ~]$ ls -ial file2
131657 -rw-rw-r--. 1 yangliang yangliang 0  8月 11 21:45 file2
[yangliang@fedora ~]$ ls -iald /home/yangliang/
131100 drwx------. 30 yangliang yangliang 4096  8月 14 23:18 /home/yangliang/
[yangliang@fedora ~]$ mkdir conn
[yangliang@fedora ~]$ ls -iald /home/yangliang/conn/
131671 drwxrwxr-x. 2 yangliang yangliang 4096  8月 14 23:19 /home/yangliang/conn/
[yangliang@fedora ~]$ cd /home/yangliang/conn/
[yangliang@fedora conn]$ ls -ial
总用量 8
131671 drwxrwxr-x.  2 yangliang yangliang 4096  8月 14 23:19 .
131100 drwx------. 31 yangliang yangliang 4096  8月 14 23:19 ..
[yangliang@fedora conn]$ mkdir subconn
[yangliang@fedora conn]$ ls -ial
总用量 12
131671 drwxrwxr-x.  3 yangliang yangliang 4096  8月 14 23:20 .
131100 drwx------. 31 yangliang yangliang 4096  8月 14 23:19 ..
131679 drwxrwxr-x.  2 yangliang yangliang 4096  8月 14 23:20 subconn
[yangliang@fedora conn]$ cd subconn/
[yangliang@fedora subconn]$ ls -ial
总用量 8
131679 drwxrwxr-x. 2 yangliang yangliang 4096  8月 14 23:20 .
131671 drwxrwxr-x. 3 yangliang yangliang 4096  8月 14 23:20 ..
[yangliang@fedora subconn]$ cd ..
[yangliang@fedora conn]$ ls
subconn
[yangliang@fedora conn]$ mkdir subconn2
[yangliang@fedora conn]$ ls -ial
总用量 16
131671 drwxrwxr-x. 4 yangliang yangliang 4096  8月 14 23:23 .
131100 drwx------. 31 yangliang yangliang 4096  8月 14 23:19 ..
131679 drwxrwxr-x.  2 yangliang yangliang 4096  8月 14 23:20 subconn
131681 drwxrwxr-x.  2 yangliang yangliang 4096  8月 14 23:23 subconn2
```

同一个文件件有一个别名叫做 。

同一个文件件如果他有一个子文件夹那他还有一个别名叫做 。。

conn的inode的连接数为什么是4？conn，.，subconn下的..，subconn2下的..

磁盘分区

- 加入新的硬盘后，重启
- fdisk /dev/sdb

```
Command (m for help): p

Disk sdb: 1073 MB, 1073741824 bytes
255 heads, 63 sectors/track, 130 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x1d637f05

Device Boot      Start         End      Blocks   Id  System
  sdb1               1          26      208813+  83  Linux
```

```
Command action
   a   toggle a bootable flag
   b   edit bsd disklabel
   c   toggle the dos compatibility flag
   d   delete a partition
   l   list known partition types
   m   print this menu
   n   add a new partition
   o   create a new empty DOS partition table
   p   print the partition table
   q   quit without saving changes
   s   create a new empty Sun disklabel
   t   change a partition's system id
   u   change display/entry units
   v   verify the partition table
   w   write table to disk and exit
   x   extra functionality (experts only)
```

设备名　起始柱面　结束柱面　总块数

```
Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 2
First cylinder (27-130, default 27): 27
Last cylinder, +cylinders or +size{K,M,G} (27-130, default 130): +300M
```

开始柱面范围

结束柱面范围，可以是分区的大小

/dev

```
sda
sda1
sda2
sdb
sdb1
sdb2
sdb3
```

格式化

- mkfs （make file system）

```
[root@fedora sbin]# ls mkfs*
mkfs  mkfs.cramfs  mkfs.ext2  mkfs.ext3  mkfs.ext4  mkfs.ext4dev  mkfs.msdos  mkfs.ntfs  mkfs.vfat  mkfs.xfs
```

- mkfs -t ext4 /dev/sdb1

```
[root@fedora dev]# mkfs -t ext4 sdb1
mke2fs 1.41.10 (10-Feb-2009)
文件系统标签=
操作系统:Linux
块大小=1024 (log=0)
分块大小=1024 (log=0)
Stride=0 blocks, Stripe width=0 blocks
52208 inodes, 208812 blocks
10440 blocks (5.00%) reserved for the super user
第一个数据块=1
Maximum filesystem blocks=67371008
26 block groups
8192 blocks per group, 8192 fragments per group
2008 inodes per group
Superblock backups stored on blocks:
        8193, 24577, 40961, 57345, 73729, 204801

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

This filesystem will be automatically checked every 28 mounts or
```

块大小

inode数，块数

分组情况

超级快备份

这些参数都是mkfs默认设置的

蒸腾mke2fs

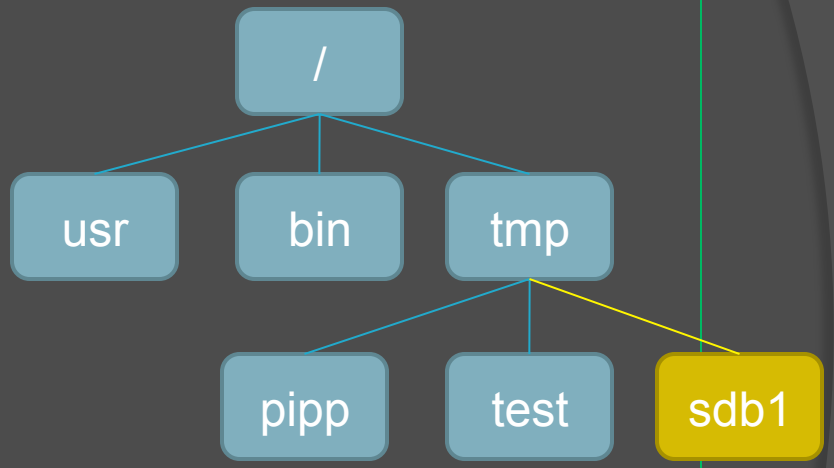
磁盘挂载

- windows

C: D: E:

cdrom U盘 G:

- Linux



同一个文件系统只能挂载一次
同一个目录只能挂载一个文件系统
作为挂载目录，应该是空目录

- mount dev dir

```
[root@fedora boot]# df
文件系统                 1K-块      已用      可用 已用% 挂载点
/dev/mapper/vg_fedora-lv_root
                      14965776   2631844  12181948  18% /
tmpfs                   254968       420    254548   1% /dev/shm
/dev/sda1               495844     43949    426295  10% /boot
[root@fedora boot]# mkdir /mnt/sdb1
[root@fedora boot]# mount  /dev/sdb1 /mnt/sdb1/
[root@fedora boot]# df
文件系统                 1K-块      已用      可用 已用% 挂载点
/dev/mapper/vg_fedora-lv_root
                      14965776   2631844  12181948  18% /
tmpfs                   254968       420    254548   1% /dev/shm
/dev/sda1               495844     43949    426295  10% /boot
/dev/sdb1               202219      5902    185877   4% /mnt/sdb1
[root@fedora boot]# cd /mnt/sdb1/
[root@fedora sdb1]# ls
lost+found
```

- mount 已挂载

```
[root@fedora sdb1]# mount
/dev/mapper/vg_fedora-lv_root on / type ext4 (rw)
proc on /proc type proc (rw)
sysfs on /sys type sysfs (rw)
devpts on /dev/pts type devpts (rw,gid=5,mode=620)
tmpfs on /dev/shm type tmpfs (rw,rootcontext="system_u:object_r:tmpfs_t:s0")
/dev/sda1 on /boot type ext4 (rw)
none on /proc/sys/fs/binfmt_misc type binfmt_misc (rw)
gvfs-fuse-daemon on /home/yangliang/.gvfs type fuse.gvfs-fuse-daemon (rw,nosuid,nodev,user=yangliang)
/dev/sdb1 on /mnt/sdb1 type ext4 (rw)
```

我们来看一个综合的例子

进程与资源管理

- ps
  - 每个正在运行的程序都会是一个或者多个进程
  - ps有很多参数，但是只需要记住这一条就可以了

ps aux | grep

```
[yangliang@fedora ~]$ ps aux | grep  -v "\["
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.2   2828  1352 ?        Ss   05:58   0:02 /sbin/init
root       513  0.0  0.2   2852  1136 ?        S<s  05:59   0:00 /sbin/udevd -d
root       928  0.0  0.0   2848   268 ?        Ss   05:59   0:00 /usr/bin/system-setup-keyboard
root      1051  0.0  0.1  12876   824 ?        S<sl 05:59   0:00 auditd
root      1053  0.0  0.1  13408   772 ?        S<sl 05:59   0:00 /sbin/audispd
root      1057  0.0  0.4  15300  2108 ?        S<   05:59   0:00 /usr/sbin/sedispatch
root      1078  0.0  0.2  35592  1272 ?        Sl   05:59   0:00 /sbin/rsyslogd -c 4
root      1119  0.0  0.0   2388   224 ?        Ss   05:59   0:00 mdadm --monitor --scan -f --pid-file=/var/run/mdad
dbus      1128  0.0  0.3  13808  1800 ?        Ssl  05:59   0:01 dbus-daemon --system
root      1139  0.0  0.8  20892  4568 ?        Ssl  05:59   0:00 NetworkManager --pid-file=/var/run/NetworkManager/
root      1146  0.0  0.4   4660  2228 ?        S    05:59   0:00 /usr/sbin/modem-manager
avahi     1151  0.0  0.0   3064   352 ?        Ss   05:59   0:00 avahi-daemon: chroot helper
```

pstree

```
[yangliang@fedora ~]$ pstree
init─┬─NetworkManager───{NetworkManager}
     ├─abrtd
     ├─acpid
     ├─atd
     ├─auditd─┬─audispd─┬─sedispatch
     │        │         └─{audispd}
     │        └─{auditd}
     ├─avahi-daemon───avahi-daemon
     ├─bonobo-activati───{bonobo-activat}
     ├─clock-applet
     ├─console-kit-dae───63*[{console-kit-da}]
     ├─crond
     ├─cupsd
     ├─2*[dbus-daemon───{dbus-daemon}]
     ├─2*[dbus-launch]
     ├─fprintd
     ├─gconf-im-settin
     ├─gconfd-2
     ├─gdm-binary─┬─gdm-simple-slav─┬─Xorg
     │            │                 ├─gdm-session-wor─┬─gnome-session─┬─abrt-applet
     │            │                 │                 │               ├─bluetooth-apple
     │            │                 │                 │               ├─deja-dup-monito
     │            │                 │                 │               ├─evolution-alarm
     │            │                 │                 │               ├─gdu-notificatio
     │            │                 │                 │               ├─gnome-panel
     │            │                 │                 │               ├─gnome-power-man
     │            │                 │                 │               ├─gnome-volume-co
     │            │                 │                 │               ├─gpk-update-icon
     │            │                 │                 │               ├─metacity───{metacity}
     │            │                 │                 │               ├─nautilus
     │            │                 │                 │               ├─nm-applet
     │            │                 │                 │               ├─polkit-gnome-au
     │            │                 │                 │               ├─python
     │            │                 │                 │               ├─restorecond
```

top 查看实时进程情况

TOP

```
top - 10:38:21 up  4:39,  3 users,  load average: 0.03, 0.03, 0.00
Tasks: 154 total,   1 running, 153 sleeping,   0 stopped,   0 zombie
Cpu(s):  1.0%us,  0.3%sy,  0.0%ni, 98.7%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:    509936k total,   385976k used,   123960k free,    26056k buffers
Swap:  1048568k total,        0k used,  1048568k free,   193176k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 1347 root      20   0  167m  29m 7520 S  0.7  5.8   1:00.15 Xorg
 2084 yanglian  20   0  131m  18m  10m S  0.3  3.6   0:20.91 gnome-terminal
    1 root      20   0  2828 1352 1152 S  0.0  0.3   0:02.11 init
    2 root      20   0     0    0    0 S  0.0  0.0   0:00.01 kthreadd
    3 root      RT   0     0    0    0 S  0.0  0.0   0:00.00 migration/0
    4 root      20   0     0    0    0 S  0.0  0.0   0:00.02 ksoftirqd/0
    5 root      RT   0     0    0    0 S  0.0  0.0   0:00.00 watchdog/0
    6 root      20   0     0    0    0 S  0.0  0.0   0:00.54 events/0
    7 root      20   0     0    0    0 S  0.0  0.0   0:00.00 cpuset
    8 root      20   0     0    0    0 S  0.0  0.0   0:00.00 khelper
    9 root      20   0     0    0    0 S  0.0  0.0   0:00.00 netns
   10 root      20   0     0    0    0 S  0.0  0.0   0:00.00 async/mgr
   11 root      20   0     0    0    0 S  0.0  0.0   0:00.00 pm
   12 root      20   0     0    0    0 S  0.0  0.0   0:00.03 sync_supers
   13 root      20   0     0    0    0 S  0.0  0.0   0:00.03 bdi-default
   14 root      20   0     0    0    0 S  0.0  0.0   0:00.00 kintegrityd/0
   15 root      20   0     0    0    0 S  0.0  0.0   0:00.31 kblockd/0
   16 root      20   0     0    0    0 S  0.0  0.0   0:00.00 kacpid
   17 root      20   0     0    0    0 S  0.0  0.0   0:00.00 kacpi_notify
   18 root      20   0     0    0    0 S  0.0  0.0   0:00.00 kacpi_hotplug
```

q离开

统计信息

详细信息

top 查看实时进程

特定时间间隔内运行队列中的平均进程数，如果一个进程没有等待IO，也没有主动进入等待状态，那么进程就进入运行队列

系统平均负载（1分钟、5分钟、15分钟）

系统时间

系统运行时间

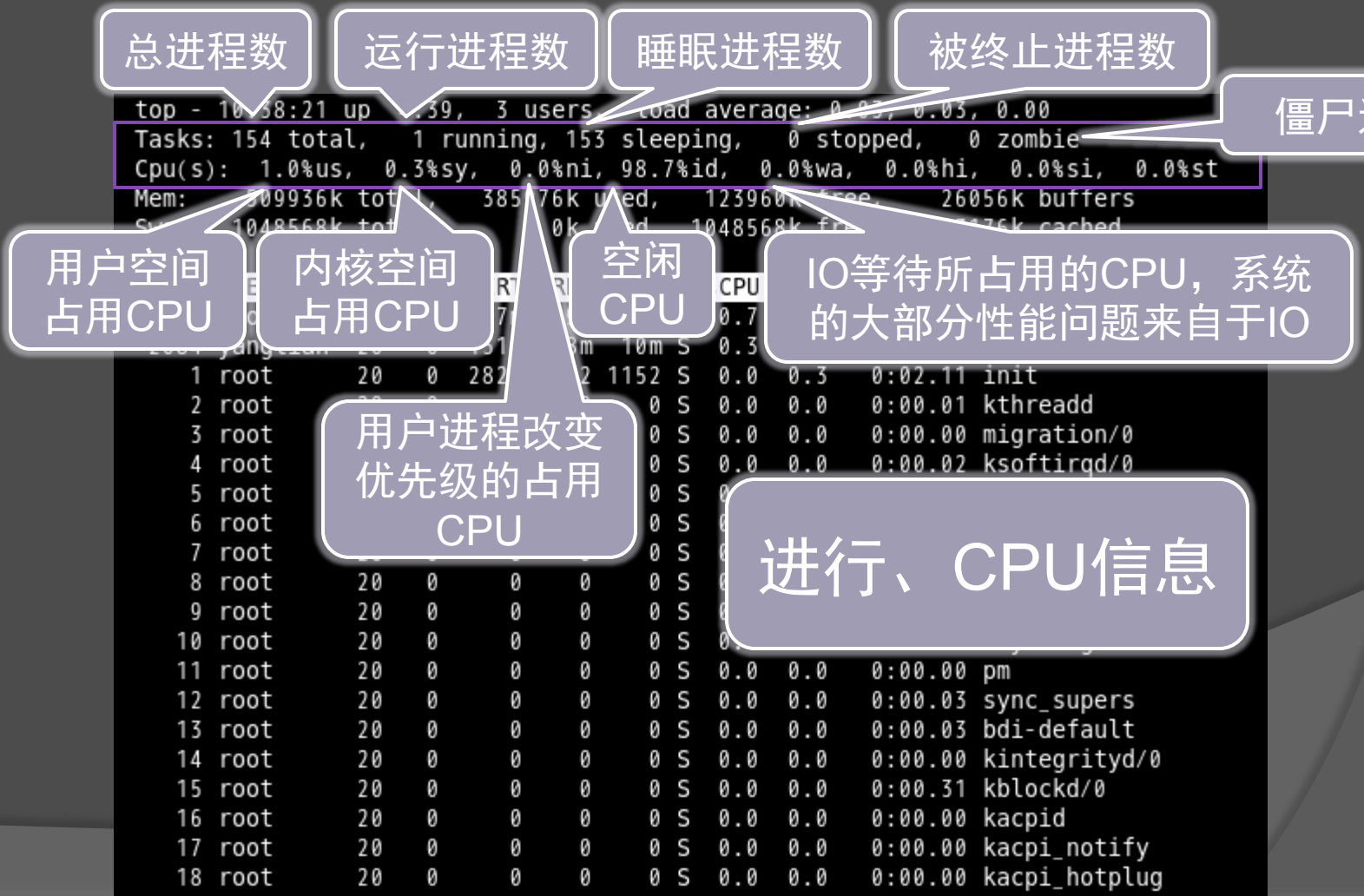
在线用户数

uptime

系统总体信息

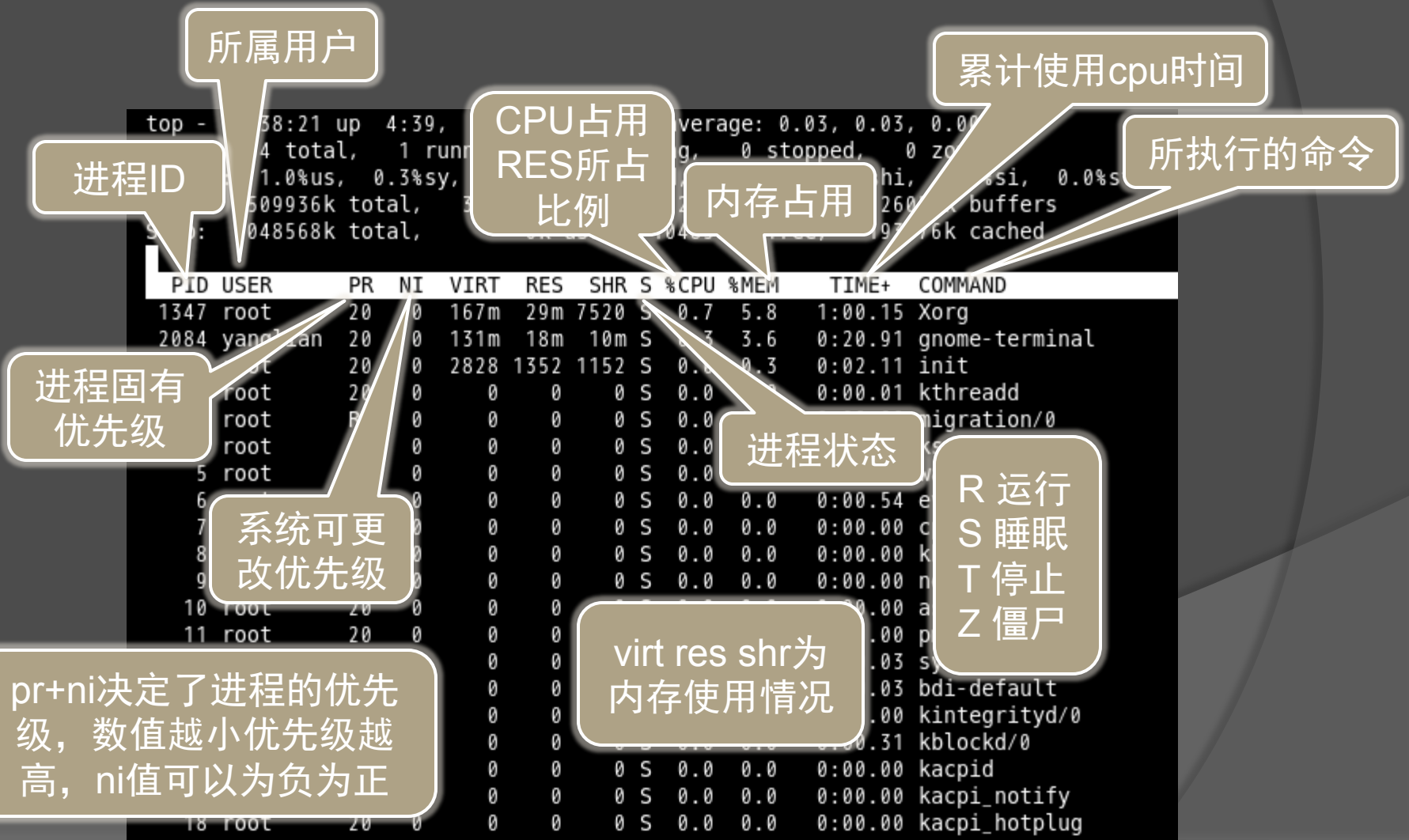
```
top - 10:38:21 up  4:39,  3 users,  load average: 0.03, 0.03, 0.00
Tasks: 154 total,   1 running, 153 sleeping,   0 stopped,   0 zombie
Cpu(s):  1.0%us,  0.3%sy,  0.0%ni, 98.7%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:    509936k total,   483880k used,    26056k buffers
Swap:  1048568k total,        0k used,   193176k cached

  PID USER      PR  NI  VIRT  RES  SHR S  %CPU %MEM    TIME+  COMMAND
 1347 root      20   0  131m  18m  10m S   0.3  3.6   0:00.15 Xorg
 2084 yanglian  20   0  131m  18m  10m S   0.3  3.6   0:20.91 gnome-terminal
    1 root      20   0  2828 1352 1152 S   0.0  0.3   0:02.11 init
    2 root      20   0     0    0    0 S   0.0  0.0   0:00.01 kthreadd
    3 root      RT   0     0    0    0 S   0.0  0.0   0:00.00 migration/0
    4 root      20   0     0    0    0 S   0.0  0.0   0:00.02 ksoftirqd/0
    5 root      RT   0     0    0    0 S   0.0  0.0   0:00.00 watchdog/0
    6 root      20   0     0    0    0 S   0.0  0.0   0:00.54 events/0
    7 root      20   0     0    0    0 S   0.0  0.0   0:00.00 cpuset
    8 root      20   0     0    0    0 S   0.0  0.0   0:00.00 khelper
    9 root      20   0     0    0    0 S   0.0  0.0   0:00.00 netns
   10 root      20   0     0    0    0 S   0.0  0.0   0:00.00 async/mgr
   11 root      20   0     0    0    0 S   0.0  0.0   0:00.00 pm
   12 root      20   0     0    0    0 S   0.0  0.0   0:00.03 sync_supers
   13 root      20   0     0    0    0 S   0.0  0.0   0:00.03 bdi-default
   14 root      20   0     0    0    0 S   0.0  0.0   0:00.00 kintegrityd/0
   15 root      20   0     0    0    0 S   0.0  0.0   0:00.31 kblockd/0
   16 root      20   0     0    0    0 S   0.0  0.0   0:00.00 kacpid
   17 root      20   0     0    0    0 S   0.0  0.0   0:00.00 kacpi_notify
   18 root      20   0     0    0    0 S   0.0  0.0   0:00.00 kacpi_hotplug
```

top 查看实时进程情况



top 查看实时进程情况

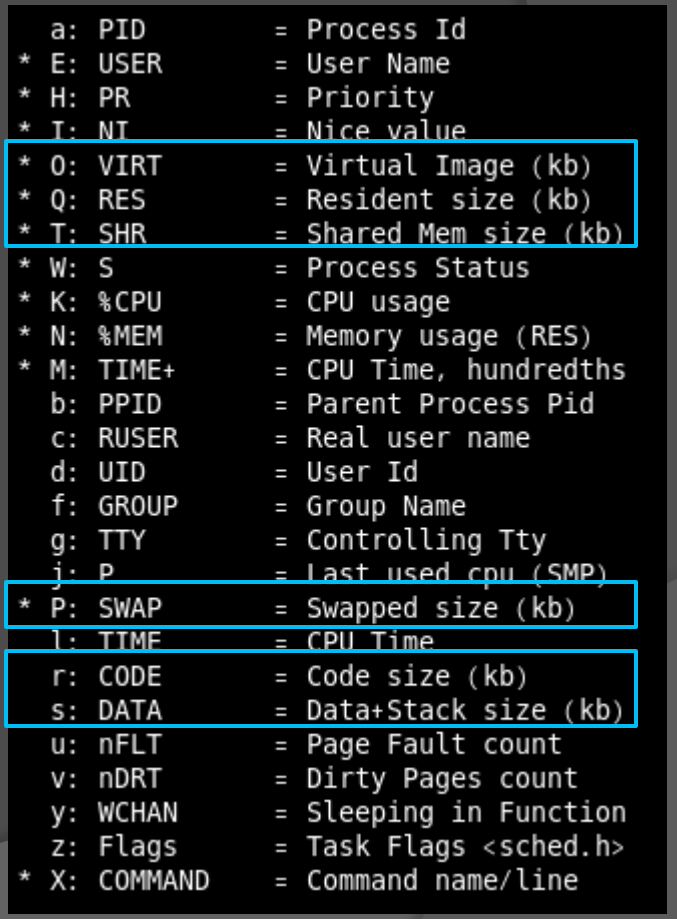
top 查看实时进程情况



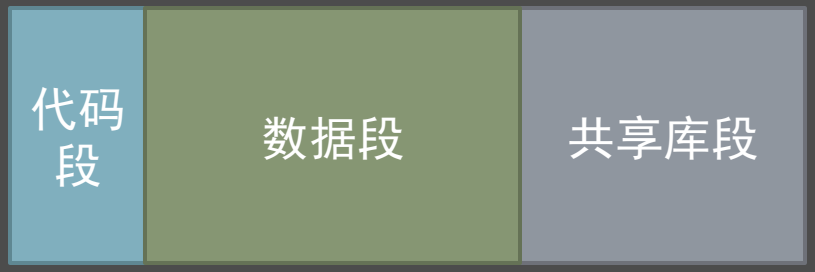
top的其他列

- VIRT： virtual memory
  - 虚拟内存使用量，虚拟地址空间占用量
- RES： resident memory
  - 驻留在内存中的大小
- SWAP：swapped memory
  - 被丢到交换区上的大小
- SHR： shared memory
  - 共享内存数据大小（共享库）
- CODE
  - 代码的大小
- DATA
  - 数据的大小，包括堆栈的内容

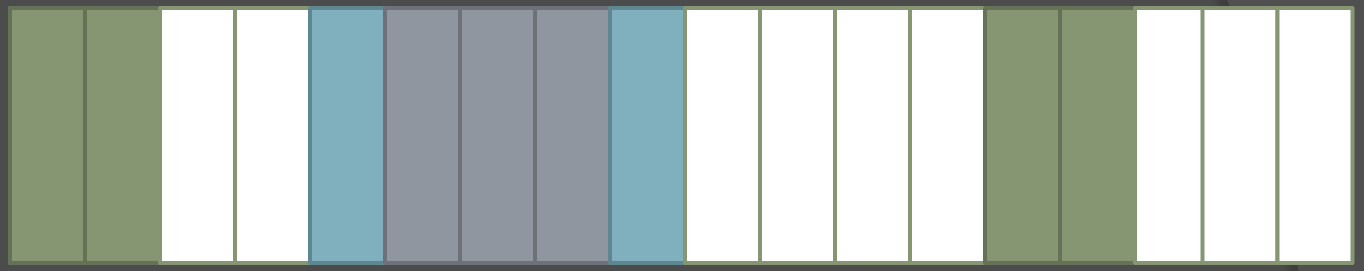
按f修改要显示的列



虚拟内存
（分段）



内存（分页）



交换空间
（硬盘）

内存不足时

USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	SWAP	CODE	DATA	COMMAND
root	20	0	168m	29m	7744	S	3.3	5.9	1:47.48	138m	1860	21m	Xorg
yanglian	20	0	132m	19m	10m	S	2.0	3.9	0:36.31	113m	284	18m	gnome-terminal
yanglian	20	0	118m	12m	9928	S	0.3	2.4	0:03.04	106m	536	12m	metacity
yanglian	20	0	104m	12m	9936	S	0.3	2.4	0:00.99	92m	48	1972	wnck-applet
yanglian	20	0	106m	18m	10m	S	0.3	3.7	0:05.03	88m	4	7200	python
yanglian	20	0	2696	1136	868	R	0.3	0.2	0:09.96	1560	60	468	top
root	20	0	2828	1352	1152	S	0.0	0.3	0:02.20	1476	128	256	init
root	20	0	0	0	0	S	0.0	0.0	0:00.03	0	0	0	kthreadd
root	RT	0	0	0	0	S	0.0	0.0	0:00.00	0	0	0	migration/0
root	20	0	0	0	0	S	0.0	0.0	0:00.08	0	0	0	ksoftirqd/0

VIRT = RES + SWAP
RES = CODE + DATA + SHR
内存真正使用量 = RES - SHR

free 查看内存使用情况

buffers 和 cached都占用内存，但是也都可以释放，他们只是为了提高系统IO性能

对文件系统中的inode，block等块缓存

buffer cache

对文件内容进行缓存

page cache

```
[yangliang@fedora ~]$ free
             total       used       free     shared    buffers     cached
Mem:        509936     394892     115044          0      27120     194796
-/+ buffers/cache:     172976     336960
Swap:      1048568          0    1048568
```

-buffers/cached used 系统真正已经使用的内存 Mem-used - buffers - cached
+buffers/cached free 系统真正可以使用的内存 Mem-free + bufers + cached

http://blog.chinaunix.net/u1/33412/showart_327801.html
http://club.cqvip.com/showtopic-634087.aspx

没有什么能否阻挡，我对自由的向往
---- 许巍 《蓝莲花》

- ctrl-C
  - 如果想终止一个在前段已经运行的命令
- kill -signal pid
  - signal=15 (SIGTERM)正常终止一个进程
  - signal=9 (SIGKILL) 强制中断一个进程
  - ps aux | grep ‘syslog’ | grep -v ‘grep’
  - kill -9 pid
- killall -signal command
  - 终止某项服务
  - 系统中所有以command启动的进程全部删除

Linux性能分析工具

- **CPU性能分析工具：**
  - vmstat
  - ps
  - sar
  - time
  - strace
  - pstree
  - top
- **Memory性能分析工具：**
  - vmstat
  - strace
  - top
  - ipcs
  - ipcrm
  - cat /proc/meminfo
  - cat /proc/slabinfo
  - cat /proc/ /maps
- **I/O性能分析工具：**
  - vmstat
  - ipstat
  - repquota
  - quotacheck
- **Network性能分析工具：**
  - ifconfig
  - ethereal
  - tethereal
  - iptraf
  - iwconfig
  - nfsstat
  - mrtg
  - ntop
  - netstat
  - cat /proc/sys/net

crontab

- 在某些特定时间多次执行某一命令
- 文件备份、系统更新、磁盘清理。。。
- crontab -l　　列出所有的定时任务
- crontab -e　　编辑定时任务，就像使用vi

```
[root@fedora etc]# crontab -l
#what you want to do
0 12 * * * ls -l > /dev/null
```

分钟　小时　日　月　星期几　命令

在每天的12点整执行 ls -l > /dev/null 命令

分钟	小时	日	月	星期几
0-59	0-23	1-31	1-12	0-7 (0,7周日)

*	任意时间
,	并列的几个时间
-	代表一个范围内的每个值
/n	代表每n个值发生一次

表示	含义
0 12 * * 1	每周一12：00
0 7 10,20,30 * *	每月的10，20，30号早7点
0 9-17 * * 1-5	上班时间的每个小时整点
*/10 * * * *	每十分钟进行一次
0 0-6/2 * * *	每天0-6点没两小时一次
0 0 24 12 *	每年平安夜
0 0 1 1 *	每年元月1日凌晨0点

账号和用户组

软件安装

Reference

- Linux内核完全剖析
- 深入理解Linux内核
- 鸟哥的Linux私房菜（基础篇）
- Unix操作系统教程（张红光）
- Professional Linux® Kernel Architecture