



Linux的文件管理

信息工程学院

- ◆ 在Linux中一切皆是文件。
 - 每个硬件也都看作一个文件即设备文件，读/写设备文件实现对硬件的访问。
- ◆ Linux没有盘符的概念。
 - 文件的绝对路径从根“/”开始，如： `/home/wlw/test.txt`
(windows则如 `c:\users\wlw\test.txt` 表示)
- ◆ Linux文件没有扩展名。
 - Linux文件的扩展名与文件类型没有任何关系。
如： `test.txt` 可以是可执行文件， `test.exe` 可以是文本文件。

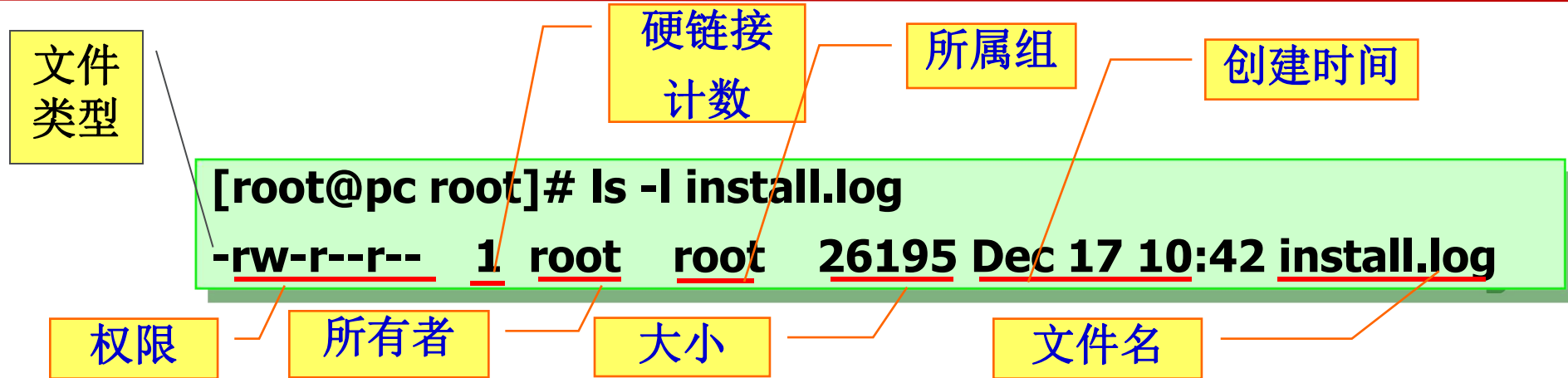
文件颜色的区别

- 白色： 普通文件（或黑色）
- 蓝色： 目录文件
- 青绿色： 可执行文件(**//bin; /sbin**)
- 红色： 压缩文件
- 黄色： 设备文件盘 (**/dev**)
- 粉红色： 图片文件
- 浅兰色： 链接文件（软）

用 **ls** 命令显示当前目录下的文件

用 **file** 命令显示指定文件的类型

文件的详细信息



文件类型:

-或f	表示普通文件;	c	表示字符型设备文件;
d	表示目录;	b	表示块设备文件;
l	表示软链接;	p	表示管道文件
		s	表示socket文件。

查看Linux文件类型还可使用: **file 文件名[文件名.....]**

如: **# file home**

home:directory

普通文件

◆数据没有结构化，只是作为有序的字符序列把它提交给应用程序，应用程序自己组织解释为下列类型：

- 文本文件：由**ASCII**字符构成的信件、报告、**SHELL**解释的**Script**脚本文件
- 数据文件：由数字和文本数据构成的电子表格、数据库、字处理文档
- 可执行的二进制程序文件：由机器指令和数据构成的命令

目录文件

◆数据进行了结构化处理，由成对的“|节点号/文件名”构成列表。它是**Linux**储存文件名的唯一地方。

设备文件

- ◆ 是一种特别文件，文件节点中存放属性信息外，不包含任何数据，只有设备名称，常放在/**dev**目录内。
- ◆ 分为字符设备和块设备
 - 字符设备：允许以字符为单位线性传送任意大小的数据，如键盘、打印机及鼠标。
 - 块设备：利用核心缓冲区以块为单位随机进行传送，以**KB**为单位，如硬盘。

软链接文件(符号链接文件)

类似**win**下的快捷方式，是将一个路径名链接到一个文件。

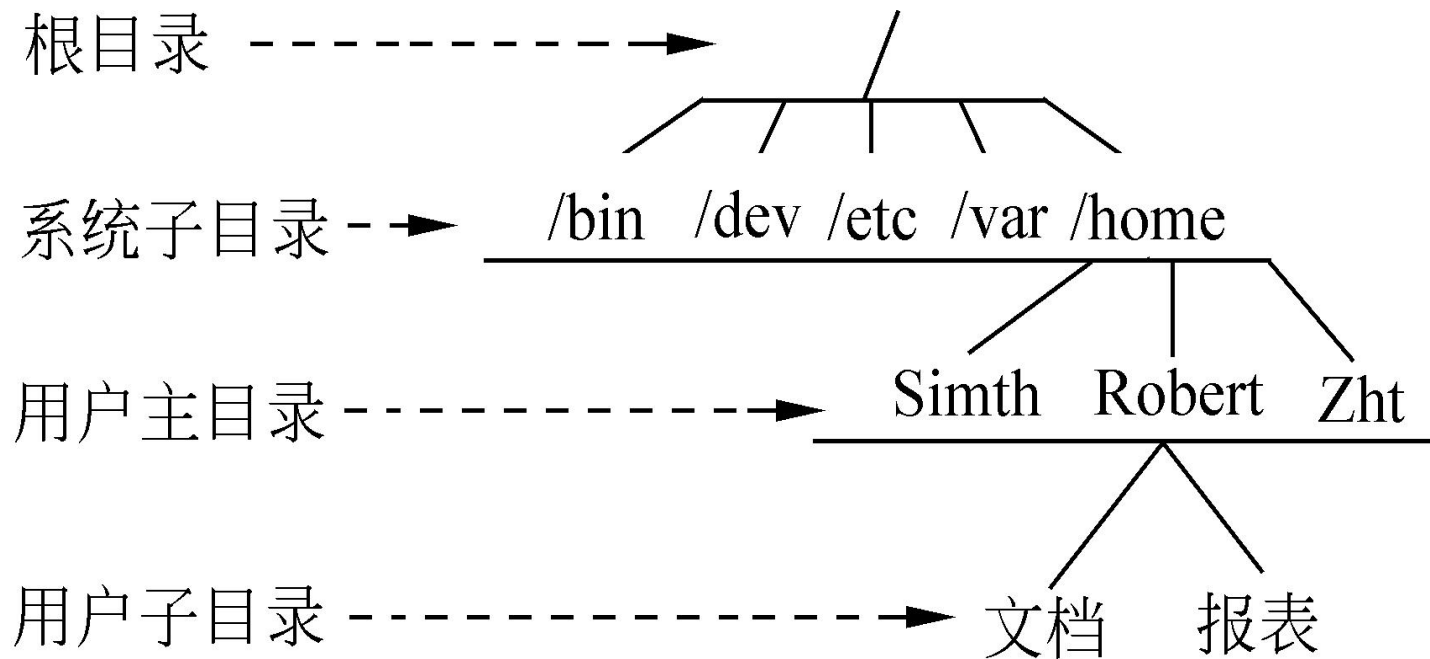
- ◆ **Linux**具有为一个文件起多个名字的功能，称为链接。
- ◆ 被链接的文件，可以实现同步修改更新，相当于同时备份。

文件的命名

- ◆ **Linux**文件名几乎可以由**ASCII**字符的任意组合构成，文件名最长可达**255**个字符。
 - 不要使用“/”和空字符，这两个字符被核心当作表示路径名的特殊字符来解释。
 - 避免使用“； | < > ‘ ” ’ \$! % & * ? \ () []”，这些字符对**Shell**来说有特殊含义。
 - 习惯上使用“_”下划线和“.”点来区别文件类型。

以“.”圆点开头的文件名是隐含文件，在默认方式下，使用**ls**命令显示不出来，要用**ls -a**显示。

◆ Linux文件系统采用带链接的树形结构



一些常用目录

- ◆/ ——根目录
- ◆/proc、/srv、/sys ——存放与内核相关的文件，一般不动
- ◆/boot ——存放引导系统的文件
- ◆/etc ——存放软件的配置文件
- ◆/dev ——存放设备文件如cpu等，类似windows设备管理器
- ◆/media ——存放自动识别的设备如cdrom、U盘等
- ◆/mnt ——专门给挂载的文件系统使用，如windows的共享目录
- ◆/bin,/sbin ——存放常使用的命令/存放系统管理使用的命令

- ◆/opt ——存放安装前的安装文件
- ◆/usr ——存放安装后的应用文件,类似windows的program files
- ◆/lib ——存放库文件,类似windows的dll库
- ◆/var ——存放不断变化的文件，如日志文件
- ◆/selinux ——安全子系统，类似360
- ◆/root ——超级用户的的家目录。
- ◆/home ——存放普通用户的家目录，每个用户对应一个家目录。



关于Linux目录



- ◆ **Linux**系统一切皆文件，通过文件管理设备。
- ◆ **Linux**目录有且仅有一个根目录。
- ◆ 使用**Linux**过程中脑海中始终要有一颗目录树。

Linux各目录存放的内容是规划好的，要记住各目录存放什么内容，不要乱放！

目录和路径的概念

用户主目录

- 当注册进入系统时，当前工作目录即为用户主目录。
- 主目录往往位于 **/home** 目录之下，并且与注册名相同。

如：

- ◆ 在 **may** 用户下进入系统时显示 **[may@localhost ~]\$**
~ 表示当前在 **/home/may** 用户主目录下。

- ◆ 用 **pwd** 命令可以显示当前工作目录的绝对路径名

[may@localhost ~]\$ pwd

/home/may

抢答:

用户主目录下显示的文件名中有一项和@前显示的字符串是相同的，分别代表什么？为什么颜色不同？

```
SP@liujiali2_1:/home
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
[ SP@liujiali2_1 桌面]$ cd \
> ^C
[ SP@liujiali2_1 桌面]$ cd /home
[ SP@liujiali2_1 home]$ ls
SP
[ SP@liujiali2_1 home]$
```

新建文件&显示文件

- ◆ 建立空文件或修改已有文件的时间标签:

touch <文件名1>,

- ◆ 只读显示文本文件内容: **cat** <文件名>

参数: **-n** 带行号显示

实例: # **cat -n /etc/passwd**

说明: 常用**cat** <文件名> | **more** 可分页显示

- ◆ 文件信息统计:

wc [选项] 文件名

参数: **-c** 统计字符数; **-w** 统计单词数; **-l** 统计行数

- ◆ 显示文本文件内容 **more <文件名>**
说明：用回车键向下换行，或空格向下翻页，按 Q 键退出；
Ctrl+B上翻一页，**Ctrl+F**下翻一页。
 - ◆ 显示大文本文件内容 **less <文件名>**
说明：用回车键向下换行，或空格向下翻页，按 Q 键退出；
Page up /Page Down键上下翻页。
- 说明：**less**比**more**更强大，可以支持各种显示终端，不是一次加载整个文件，是根据需要加载，速度更快，适合大型文件的显示。
- ◆ 显示文本文件开始n行：**head -n <文件名>** 不写n默认为10行
 - ◆ 显示文本文件最后n行：**tail -n <文件名>** 不写n默认为10行
- 实例：
- ```
head -20 /a.txt
tail -20 /a.txt
head -c 5 /etc/services //打印最前面5个字符
tail -f /var/log/messages //显示动态文件的末尾，
 CTRL+C退出，一般用于日志文件
```

# 新建目录&显示目录

- ◆ 新建目录: **mkdir dir**
  - 参数 **-p** 可一次建立多级目录, 即当新建目录中有些子目录尚不存在, 此选项可以自动建立它们, 如 **mkdir -p /home/class/stud/**, 当**class**不存在, 则会自动建立。
- ◆ 删除**空**目录: **rmdir dir**
  - 参数 **-p** 递归删除目录, 当子目录删除后其父目录为空时, 也一同被删除。如果有非空的目录, 则该目录保留下来。
- ◆ 删除**非空**目录及数据: **rm -rf dir**
- ◆ 列出目录内容: **ls dir**

# 关于目录操作命令

- ◆ 显示当前目录的绝对路径命令 `pwd`
- ◆ 改变当前路径命令 `cd <相对路径名/绝对路径名>`
  - 例: `cd ~` //切换到用户家目录
  - `cd -` //切换到上次进入的目录
- ◆ 显示目录中的文件: `ls [参数] [目录名]`
  - `-a:` //显示目录下所有文件
  - `-l:` //以长格式显示目录下的内容
  - `-t:` // 按照修改时间排列显示
  - `-F:` //显示文件名同时显示类型
  - `-d:` //显示本目录信息

(\*: 表示可执行的普通文件; /: 表示目录; @: 链接文件; |: 管道文件)



# 抢答



抢答：

当执行 `[root@localhost ~]# pwd` 时，运行结果是什么？

当切换到普通用户后 `[lh@localhost ~]# pwd`，执行结果又会是什么？

有的说是： `/root`

有的说是： `/home/lh`

为什么？

# 绝对路径和相对路径

- ◆ 绝对路径名，如： **/home/stud**
  - 每个文件都有**唯一的全路径名**，以"/"字符开头。
- ◆ 相对路径名，如： **home/stud**
  - 不能以"/"字符开头，要以当前工作目录名开始。
- ◆ 点(.)表示当前目录； 点点(..)表示上一级目录
  - 每个目录中都有点点目录文件
  - 点点(..)可以连续使用，直至根目录

建议：

- 都尽量使用绝对路径，虽然长，但不容易误解和混淆。
- 当路径很长时，可以使用**TAB**键进行补全操作。

# 举例

- ◆ 设当前目录是/home/may，执行下面的命令：

```
[may@localhost ~]$ ls -l test/test1.txt
```

作用是：列出/home/may/test目录中test1.txt文件。

- ◆ 设当前目录是/home/may/test1，执行下面的命令：

```
[may@localhost test1]$ ls ../test2/
```

作用是：列出/home/may/test2目录中的内容。

- ◆ 设当前目录/home/may/test1/test11,执行下面的命令：

```
[may@localhost test11]$ ls ../../test2/
```

作用是：列出/home/may/test2目录中的内容

抢答:

当前工作目录是/root, 想要进入/home目录,  
采用绝对路径方式和相对路径方式, 分别要执行的完整指令是什么?



# 同学出现的操作现象



```
root@liweijia225:/home
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
[alan@liweijia225 ~]$ su - root
Password:
Last login: Thu Mar 24 04:42:38 EDT 2022 on pts/0
[root@liweijia225 ~]# pwd
/root
[root@liweijia225 ~]# cd /root/home
-bash: cd: /root/home: No such file or directory
[root@liweijia225 ~]# cd root/home
-bash: cd: root/home: No such file or directory
[root@liweijia225 ~]# cd ../home
[root@liweijia225 home]#
```

# 文件操作常用命令

## ◆复制文件：cp [参数] <源文件/原目录> <目标路径>

- -f //若文件在目标路径中存在则强制覆盖
- -r //递归复制（包含子目录一起复制）
- -b //生成覆盖文件的备份
- -v //显示命令执行过程

注意：cp前加“\”是强制覆盖。lcp 是强制复制

## ◆移动或重命名文件：mv [参数] <源路径> <目标路径>

- 参数（同CP命令）

说明：目录不变实现重命名；目录不同，实现移动。

## ◆删除文件：rm [参数] <文件名/目录名>

- -f //强制删除
- -r //递归删除
- -i //提示是否删除
- -v //显示命令执行过程

- ◆ **find** 可查文件，也可按字符串查找，但是实时相对**locate**慢
- ◆ **find [路径] [参数] [文件名]**（支持通配符）  
常用
  - 按文件名查找: **find 路径 -name “文件名”**  
例: **# find ./ -name “named\*”** //查找当前目录下**named**开始的文件
  - 按用户名查找: **find 路径 -user “用户名”**  
例: **# find /home -user root** //查找**home**目录下**root**用户的文件  
**# find / -nogroup** //查出系统无属组的文件
  - 按类型查找: **find 路径 -type "类型名"**  
例: **# find ./ -type d** //查看当前目录下所有的目录
  - 按大小查找: **find 路径 -size +nM/-nM/nM(大于/小于/等于)**  
例: **# find / -size +20M** //查找系统中大于**20M**的文件
  - 按时间查找: **-mtime**

# 查找文件

- ◆ **locate**: 可查文件，按字符串非完全匹配查找，非实时速度快
  - 系统中所有文件的路径会自动生成在 **/var/lib/mlocate/mlocate.db** 数据库文件中，通过数据库文件非实时扫描查找，速度快。

```
[may@CM00 ~]$ locate sudoers //会将带有该字符串的所有文件的目录列出
/etc/sudoers.d
/usr/share/augeas/lenses/dist/sudoers.aug
```

.....

说明：每次运行前 **updatedb** 命令更新系统中的所有文件路径，每晚会自动更新

- ◆ **which** : 只用于查找命令，因为在各 **bin** 和 **sbin** 目录中查找 (**whereis** 也用于查找命令及相关的 **man** 文档，很少用)  

```
[may@CM00 ~]$ which cat
/bin/cat
```

◆ **grep** [参数] 查找内容 源文件 //用于查找指定文件的内容。

- **-C[数字]** 列出符合匹配模式行的前后各**n**行;
- **-n** 列出匹配模式所在行的行号 (非常有用)
- **-i** 忽略字母大小写 (非常有用)

常用一: **grep** “字符串” <文件名> (不支持通配符)

例: # **grep print hello.c** //查找**hello.c**文件中**print**字符串  
# **grep -C 2 print hello.c** //查找**hello.c**文件中**print**字符串所在行及前后**2**行的内容  
# **grep -in Print hello.c** //查找**hello.c**文件中**Print**字符串,忽略大小写同时显示行号

常用二: 显示命令 | **grep** “字符串”

例: # **cat /etc/passwd | grep -ni “stud”**  
//显示**/etc/passwd**文件中含**stud**字符的行及行号, 不区分**stud**大小写

抢答：

查找显示用户信息文件passwd文件中，当前普通用户名所在行的信息，执行的完整指令是什么？

- ◆ **sort** 对文本文件的各行进行排序
- ◆ **uniq** 从排好序的文件中去除重复行
- ◆ **comm** 对两个已排序的文件逐行比较
- ◆ **diff** 对两个未排序文件逐行比较
- ◆ **cmp** 对两个文件逐字节比较

# 常用命令小结

---

- ◆ **ls -l** //详细信息显示; **ls -d** //本级目录信息显示
- ◆ **cd ~** //切换到家目录; **cd -** //切换到上次的目录
- ◆ **cp -r <dir> <newdir>** //递归拷贝目录
- ◆ **mkdir -p <dir>** //建立多级目录
- ◆ **rmdir <dir>** //只删空目录
- ◆ **rm -r <dir/file>** //交互提问是否删除
- ◆ **rm -rf <dir/file>** //不提问直接删除, 小心使用
- ◆ **stat <file>** //查看文件更详细信息
- ◆ **less <file>** //可向前向后翻看文件

# 输入/输出重定向

- ◆ 执行一个**shell**命令行时，会自动打开三个标准文件：
  - 标准输入（**stdin**），指键盘输入（使用数字**0**表示）；
  - 标准输出（**stdout**），指输出正确，默认输出到屏幕，也可以指定到其它文件（使用数字**1**表示）
  - 错误输出（**stderr**），指输出错误结果，默认到屏幕，也可以指定到其它文件（使用数字**2**表示）。

```
[may@CM00 ~]$ touch file1
```

```
[may@CM00 ~]$ ls file1
```

**file1**

//标准输出

```
[may@CM00 ~]$ ls file2
```

**ls: 无法访问file2: 没有那个文件或目录**

//错误输出

```
[may@CM00 ~]$ ls file1 file2
```

**ls: 无法访问file2: 没有那个文件或目录**

//错误输出

**file1**

//标准输出

# 输入/输出重定向&管道

## ◆ 直接使用标准输入/输出文件存在以下问题：

- 输入数据从终端输入时，用户费了半天劲输入的数据只能用一次。下次再用这些数据时就得重新输入。而且在终端上输入时，若输入有误修改起来不是很方便。
- 输出到终端屏幕上的信息只能看不能动。我们无法对此输出作更多处理，比如将输出作为另一命令的输入进行进一步的处理等。

为了解决上述问题，**Linux**系统为输入、输出的传送引入了两种机制：

- ◆ 输入/输出重定向
- ◆ 管道

◆ **输入重定向**：输入可以不来自键盘，而来自指定的文件，主要用于改变命令的输入源，特别是改变那些需要大量输入的输入源。

➤ **一般形式**：命令 < 文件名

◆ **输出重定向**：把命令的执行结果显示输出重定向到指定文件中，不用在标准输出终端屏幕上显示，以供以后分析。

➤ 命令 1>或> 文件名 //标准输出重定向，覆盖原文件

➤ 命令 2> 文件名 //错误输出重定向，覆盖原文件

➤ 命令 &> 文件名 //错误输出和标准输出重定向，结果覆盖原文件

➤ 命令 1>>或>> 文件名 //标准输出重定向，追加原文件之后

➤ 命令 2>> 文件名 //错误输出重定向，追加原文件之后

➤ 命令 &>> 文件名 //错误输出和标准输出重定向，结果追加原文件

```
[may@CM00 ~]$ ls file1 file2
ls: 无法访问file2: 没有那个文件或目录
file1
```

```
[may@CM00 ~]$ ls file1 file2 1> file3
ls: 无法访问file2: 没有那个文件或目录
[may@CM00 ~]$ cat file3
file1
```

```
[may@CM00 ~]$ ls file1 file2 2> file3
file1
[may@CM00 ~]$ cat file3
ls: 无法访问file2: 没有那个文件或目录
```

```
[may@CM00 ~]$ ls file1 file2 &> file3
[may@CM00 ~]$ cat file3
ls: 无法访问file2: 没有那个文件或目录
file1
```

**说明:**

**/dev/null 特殊文件，类似于垃圾桶**

**常用于将错误输出丢入垃圾桶**

```
[may@CM00 ~]$ls file1 file2 2> /dev/null
file1
```

```
[may@CM00 ~]$ls file1 file2 &> /dev/null
file1
```

- ◆ 管道：连接一个进程的输出到另一个进程的输入。

语法： **command1 | command2 | .....**

示例

- [may@CM00 ~]\$ **cat /etc/passwd | tail -3** //查看文件后3行
- [may@CM00 ~]\$ **tail -3 /etc/passwd** //同上，查看文件后3行
- [may@CM00 ~]\$ **tail -5 /etc/passwd | head -1** //查看文件倒数第5行
- [may@CM00 ~]\$ **head -5 /etc/passwd | head -1** //查看文件正数第5行
- [may@CM00 ~]\$ **cat /etc/passwd | more** //翻页查看文件

# 文件打包&压缩操作

◆ **tar [参数] [tar包名] [源文件名] [-C 指定存放目录]**

- **-c:** 创建tar包
- **-x:** 解tar包 (\*.tar)
- **-v:** 显示操作信息
- **-f:** 指定文件名
- **-z:** 使用gzip压缩/解压缩文件

◆ **常用压缩: tar -zcvf 目标包名 源文件**

例: 压缩多个文件 **tar -zcvf a.tar.gz a1.txt a2.txt**

压缩整个目录 **tar -zcvf myhome.tar.gz /home**

◆ **常用解压: tar -zxvf 包名 [-C 指定存放目录]**

例: 解压到当前目录 **tar -zxvf a.tar.gz**

解压到目标目录 **tar -zxvf a.tar.gz -C /home/stud**

注意目标目录必须存在。

## 常见还有**gzip,bzip,xz**工具压缩

- **-z**: 使用**gzip**压缩/解压缩文件
- **-j**: 使用**bzip2**压缩/解压缩文件
- **-J**: 使用**xz**压缩/解压缩文件

例: [may@CM00 ~]\$ tar -cvzf testtar.tar.gz ./testtar

例: [may@CM00 ~]\$ tar -cvjf testtar.tar.bz2 ./testtar

例: [may@CM00 ~]\$ tar -cvJf testtar.tar.xz ./testtar

## **gzip,bzip2,xz**文件解压都可以统一使用 **tar xvf**

例: [may@CM00 ~]\$ tar xvf testtar.tar.bz2 //解压到当前目录

例: [may@CM00 ~]\$ tar xvf testtar.tar.xz -C ./test //解压到指定目录

## 另两组压缩和解压命令

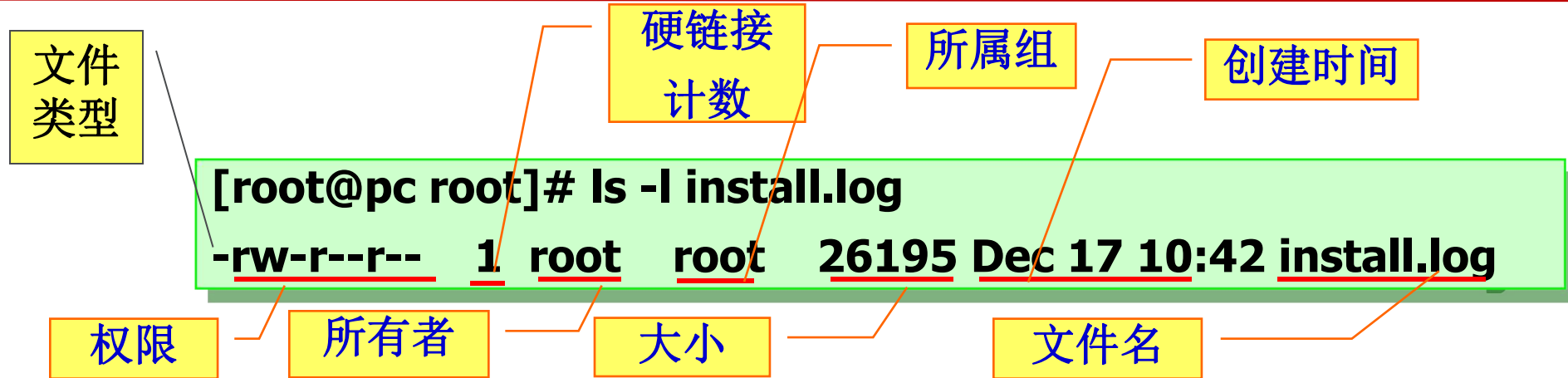
- ◆ **gzip [文件名] / gunzip [文件名.gz]**

说明: **gzip**压缩后的文件只能是**.gz**,且压缩后不会保留原文件

- ◆ **zip [选项] 文件名.zip 要压缩的文件或目录 / unzip [选项] 文件名.zip**

说明: **zip** 加 **-r** 是递归压缩 / **unzip** 加 **-d <目录>** 是解压后的存放目录

# 文件的详细信息



文件类型:

- 或f 表示普通文件;
- d 表示目录;
- l 表示软链接;
- c 表示字符型设备文件;
- b 表示块设备文件;
- p 表示管道文件;
- s 表示socket文件。

## 链接文件

- ◆ **Linux**具有为一个文件起多个名字的功能，称为链接。
- ◆ 被链接的文件：可以实现同步修改更新，相当于同时备份。
  - 可以存放相同目录下，但不能同名，可以不用在硬盘上为同样的数据重复备份。
  - 可以有相名，但不能存放相同目录下，可以实现同名链接文件的修改。
- ◆ 某文件的各链接文件，可以给不同用户赋予不同的存取权限，以控制对信息的共享和增强安全性。

- ◆ 文件链接分为硬链接和软链接(符号链接)两种形式。
  - **软链接**: 类似win下的快捷方式, 是将一个路径名链接到一个文件。
  - **硬连接**: 其实是一个指针, 系统并不为它重新分配inode号, 多个文件名指向同一索引节点。

链接文件的建立命令: **ln [参数] <源文件/源目录> <链接文件名>**

- 没有参数, 则默认为建立硬链接文件
- **-s**: 建立软链接文件
- **-i**: 提示是否覆盖目标文件
- **-f**: 直接覆盖已存在的目标文件

说明: 常用“**ln -s**”命令建立软链接时, **最好源文件用绝对路径名**, 这样可以在任何工作目录下进行软链接。当源文件用相对路径时, 如果当前的工作路径和要创建的软链接文件所在路径不同时, 就不能进行链接。



# 建立软链接应用



例：对/root目录建立软链接

```
[root@localhost home]# ln -s /root linktoroot
```

```
[root@localhost home]# ls -l
```

...

```
lrwxrwxrwx. 1 root root 5 03-20 13:46 linktoroot -> /root
```

...

```
[root@localhost home]# cd linktoroot
```

..... //显示的内容与cd /root结果相同

但是，

```
[root@localhost linktoroot]# pwd
```

```
/home/linktoroot //仍然是软链接本身所在目录home下
```

# 软链接与硬链接文件的区别

例：为现目录下**file1**建立硬链接，**file2**建立软链接。

```
[wlw@CM00 ~]$ ls -il
总用量 32
401890 -rw-rw-r--. 1 wlw wlw 0 3月 20 17:56 file1.txt
401891 -rw-rw-r--. 1 wlw wlw 0 3月 20 17:56 file2.txt

[wlw@CM00 ~]$ ln file1.txt file1hard
[wlw@CM00 ~]$ ln -s file2.txt file2soft
```

```
[wlw@CM00 ~]$ ls -il
总用量 32
401890 -rw-rw-r--. 2 wlw wlw 0 3月 20 17:56 file1hard
401890 -rw-rw-r--. 2 wlw wlw 0 3月 20 17:56 file1.txt
401894 lrwxrwxrwx. 1 wlw wlw 9 3月 22 16:42 file2soft -> file2.txt
401891 -rw-rw-r--. 1 wlw wlw 0 3月 20 17:56 file2.txt
```

## ◆在inode号上

- 硬链接的原文件/链接文件**inode**号相同，说明是同一个文件
- 软链接的原文件/链接文件**inode**号不同，说明是不同的文件

## ◆在文件属性上，

- 软链接明确显示是链接文件
- 硬链接没有体现出来，因为本质上硬链接文件和原文件是完全相同

# 软链接与硬链接文件的区别

例：为现目录下**file1**建立硬链接，**file2**建立软链接。

```
[wlw@CM00 ~]$ ls -il
总用量 32
401890 -rw-rw-r--. 1 wlw wlw 0 3月 20 17:56 file1.txt
401891 -rw-rw-r--. 1 wlw wlw 0 3月 20 17:56 file2.txt
```

```
[wlw@CM00 ~]$ ln file1.txt file1hard
[wlw@CM00 ~]$ ln -s file2.txt file2soft
```

```
[wlw@CM00 ~]$ ls -il
总用量 32
401890 -rw-rw-r--. 2 wlw wlw 0 3月 20 17:56 file1hard
401890 -rw-rw-r--. 2 wlw wlw 0 3月 20 17:56 file1.txt
401894 lrwxrwxrwx. 1 wlw wlw 9 3月 22 16:42 file2soft -> file2.txt
401891 -rw-rw-r--. 1 wlw wlw 0 3月 20 17:56 file2.txt
```

## ◆在链接数目上，

- 硬链接数目会增加
- 软链接不会增加

## ◆在文件大小上，

- 硬链接文件显示的大小是跟原文件是一样的
- 软链接显示的大小与原文件不同

- ◆ 例：用软链接vi来实现使用vim。

```
[root@CM00 ~]# which vi
/bin/vi
[root@CM00 ~]# which vim
/usr/bin/vim
[root@CM00 ~]# mv /bin/vi /bin/vi.bak
[root@CM00 ~]# vi
-bash: /bin/vi: 没有那个文件或目录
[root@CM00 ~]# ln -s /usr/bin/vim /bin/vi
[root@CM00 ~]# ll /bin/vi
lrwxrwxrwx. 1 root root 12 3月 30 20:29 /bin/vi -> /usr/bin/vim
```

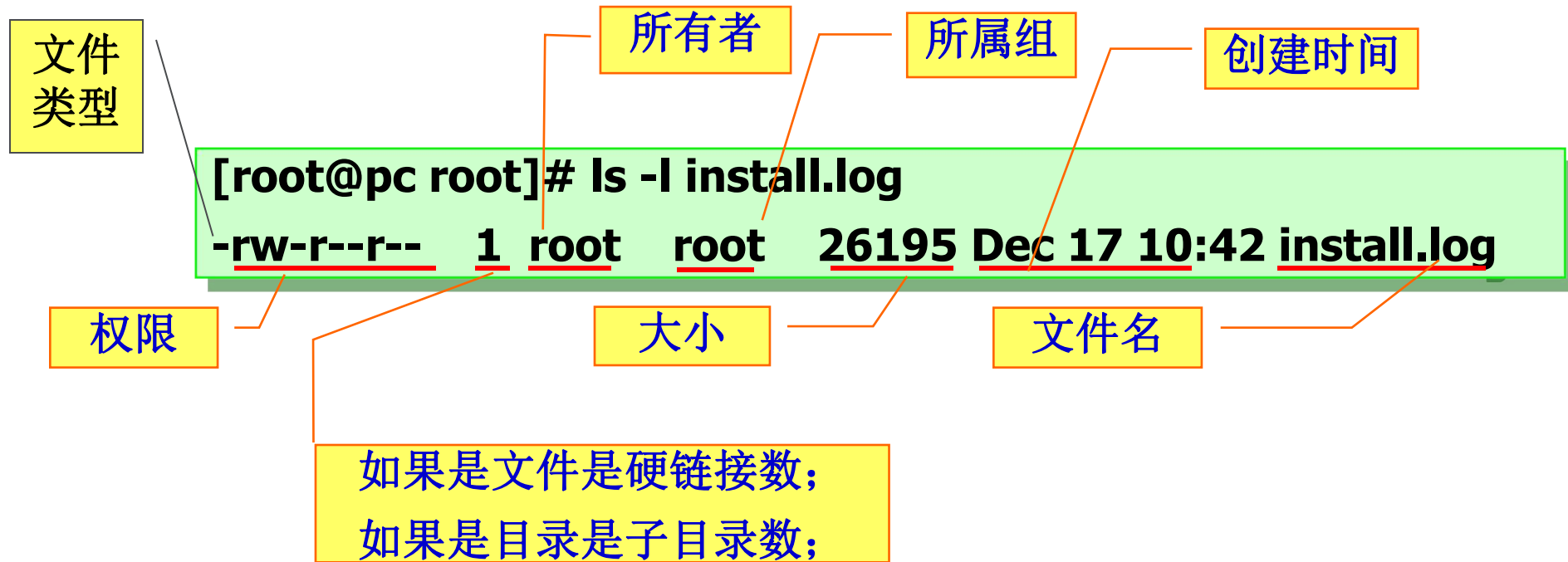
## ◆ 链接文件使用特点：

- 可以对目录和文件进行软链接，但对目录不能进行硬链接；可以在不同文件系统之间做软链接，但不能做硬链接。因此，软链接更灵活。
- 当源文件移动到其他目录后再访问软链接文件，系统就找不到了，但再访问硬链接可以找到。因为软链接里包含源文件路径，硬链接没有，因此，硬链接的源文件可以随意移动。

## ◆ 链接文件的删除用rm命令：

- 对于软链接删除源文件时，只删除了数据，不会删除链接。一旦以同名创建了源文件，链接将继续指向该文件的新数据。
- 对于硬链接可以删除其中任何一个文件，每次只会删除一个指针，链接数同时减一，只有将所有指向文件内容的指针，也即链接数减为0时，内核才会把文件内容从磁盘上删除。对于重要文件建立硬链接可以防止误删。

# 文件的详细信息



权限:

- **r** 表示读取权限, **w**表示修改权限, **x**表示执行权限。
- 每三个为一组, 按顺序分别是:

所有者(**user**), 所属组(**group**), 其他人(**other**)。

# 权限的表示

## 1、Linux系统中文件访问权限通常分为三类：

| 权限名 | 对文件的含义                     | 对目录的含义                       |
|-----|----------------------------|------------------------------|
| 读r  | 读取查看文件内容<br>如cat类          | 查看目录内容<br>如ls类               |
| 写w  | 修改文件内容，但不可删除<br>如vi,>,>>,等 | 修改创建删除目录内容<br>如touch ,mkdir等 |
| 执行x | 执行命令或文件                    | 可以进入目录<br>如cd命令              |

## 2、权限值的表示方法：八进制数字和字符表示方法

- **r只读:4**                      **w写:2**                      **x执行:1**
- **rw 读写:6**    **rx读和执行:5**    **wx 写和执行:3**
- **rw x 读写执行:7**        **--- 无权限:0**

# 文件权限设置操作——字符&数字设置法

字符设定法: **chmod** [操作对象] [操作符] [权限] 文件名

## (1) 操作对象

- **a** 表示“所有用户 (**all**)”，系统默认值。
- **u** 表示“属主用户 (**user**)”。
- **g** 表示“同组用户 (**group**)”。
- **o** 表示“其他用户 (**others**)”。

## (2) 操作符号

- **=** 赋予给定权限（如果有其他所有权限同时取消）。
- **+** 添加某个权限。
- **-** 取消某个权限。

## (3) 权限组合

- **r** 读, **w** 写, **x** 执行

实例:

```
➤# chmod u+x /home/abc.txt
➤# chmod g=rx /home/abc.txt
➤# chmod o=rx /home/abc.txt
```

# 文件权限设置操作

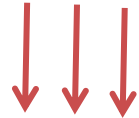
数字设定法：chmod [权限值] 文件名

权限值：

八进制表示法，数字格式是三个八进制数，其顺序是（u所有者）（g所属组）（o其他人）

实例：

**ugo**



➤ **# chmod 755 /home/abc.txt**

- ◆问题： 查看某用户对某文件的权限
- ◆操作步骤： 先用id命令查看某用户的属性是U还是G还是O，再根据属性查看9位权限值中的哪一组权限。

```
[root@CM00 ~]# touch /tmp/111
```

```
[root@CM00 ~]# useradd stud
```

```
[root@CM00 ~]# su stud
```

```
[stud@CM00 root]$ ll /tmp/111
```

```
-rw-r--r--. 1 root root 0 3月 30 21:44 /tmp/111
```

```
[stud@CM00 root]$ id stud
```

```
uid=502(stud) gid=502(stud) 组=502(stud)
```

```
#查看stud对于/tmp/111属于O组，只能r
```

```
[stud@CM00 root]$ cat /tmp/111
```

```
#可执行r
```

```
[stud@CM00 root]$ echo hello>/tmp/111
```

```
#不可执行w
```

```
bash: /tmp/111: 权限不够
```

- ◆ 改变拥有者: **chown** <所有者>[:<所有组>] <文件/目录>

实例: **# chown user f**

**# chown user:stud f**

**# chown -R user /home/abc/**

- ◆ 更改所属组: **chgrp** <组名称> <文件/目录>

实例: **# chgrp stud /home/a.txt**

**# chgrp -R stud /home/abc/**

**注意: chmod和chown要修改一个目录及其子目录下的所有文件, 则需要加-R参数, 实现递归修改。**

# 补充：特殊权限S位

- ◆ [root@CM00 ~]# ll /etc/shadow  
-----. 1 root root 1196 3月 30 21:44 /etc/shadow
- ◆ [root@CM00 ~]# ll /usr/bin/passwd  
-rwsr-xr-x. 1 root root 30768 11月 24 2015 /usr/bin/passwd

使用场景： **shadow**文件显示无任何权限是不能被修改的，但文件内包含所有用户的密码，当普通用户改密码时，即操作**shadow**文件，应该只能改自己的密码，由于**passwd**命令文件的属主权限有**s**权限，故可操作**shadow**文件。

**S位：**一个可执行文件拥有**s**位时，当别的用户来执行这个可执行文件时，使用的权限是此可执行文件属主或属组的权限

- **s**位在前三位（**suid**）或中间三位（**sgid**）
- 只对命令文件有效

# 补充：特殊权限t位

◆ [root@CM00 ~]# ll -d /tmp

drwxrwxrwt. 29 root root 4096 3月 30 22:06 /tmp

使用场景：用户对目录有w权限时，就可以在内部创建文件或子目录，同时也可以删除目录内任意文件。比如当多个用户上传文件到服务器同一目录，当某用户登录后就可以内该目录内的所有文件进行删除，如果目录加t位后，多个用户就不能任意删除内部文件，只能删除ower为自己的文件。

t位：当多个用户对某个共享目录操作时，互相之间就不能任意删除内部文件，只能删除ower为自己的文件。

- t 位在后三位
- 只对目录有效

- ◆ **chmod**可以用字符法修改**s**和**t**位，也可用数字修改
- ◆ 前三位上加**s 4**；中间三位加**s 2**；后三位加**t 1**
- ◆ 小写 表示 有**x**执行权限
- ◆ 大写 表示 没有**x**执行权限

例： **/tmp**目录的权限为**1777**

**passwd**命令文件的权限为**4755**

## 补充：提示

当用**root**权限都不能修改某个文件，大部分原因是曾经用**chattr**命令锁定该文件了。

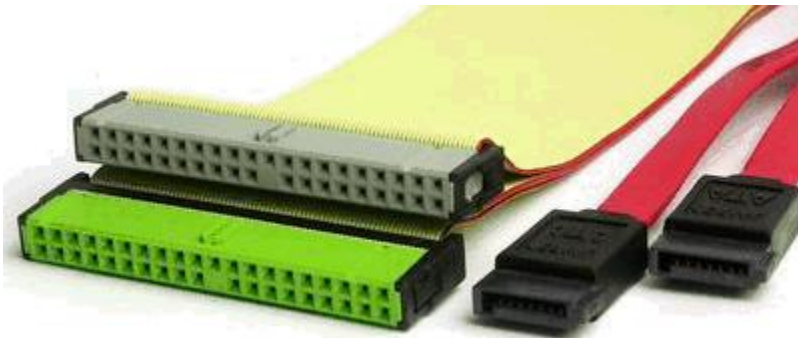
- ◆ **chattr**命令可以修改更底层的属性，能够提高系统的安全性，但**chattr**命令不能保护`/`、`/dev`、`/tmp`、`/var`目录。
- ◆ **lsattr**命令是显示**chattr**命令设置的文件属性。
- ◆ 与**chmod**这个命令相比，上述两个命令是用来查看和改变文件、目录属性的，**chmod**只是改变文件的读写、执行权限，更底层的属性控制是由**chattr**来改变的。

**IDE(Integrated Drive Electronics)**

属于**Parallel-ATA**（并行）

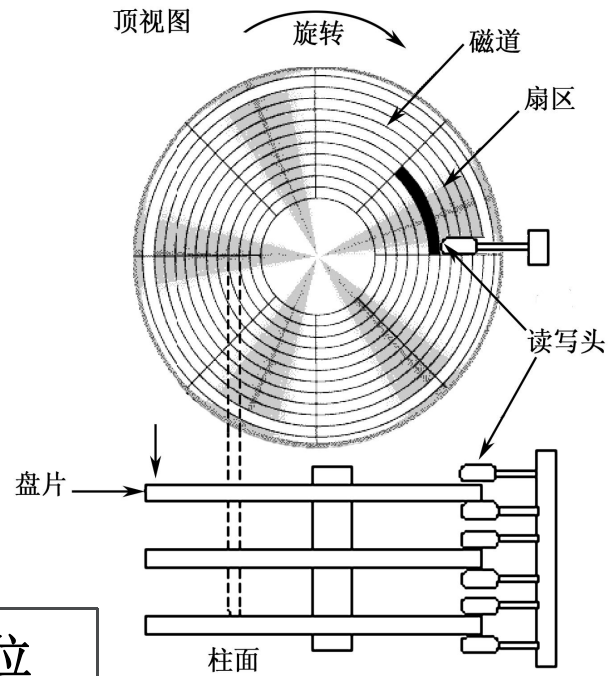
**SATA(Serial-ATA)**

属于**Serial-ATA**（串行）



# 分区基本知识

PC的BIOS的工作就是做完机器自检之后，如果是从硬盘启动则只会认定硬盘的**0磁面0磁道1扇区**，将之读入内存，并将控制权交给它。这个**0磁面0磁道1扇区**被称为**硬盘主引导扇区的主引导记录MBR (Master Boot Record)**，共**512字节**。



| 446位     | 64位分区表 |     |     |     | 2位  |
|----------|--------|-----|-----|-----|-----|
| 引导系统启动相关 | 16位    | 16位 | 16位 | 16位 | 校验位 |

分区表共**64位**，每**16位**表示**1个分区**，共**4个主分区**。

# 分区的基本知识

- ◆ 一块硬盘最多可以同时创建四个**主分区**，其中一个是**活动分区**启动系统。
- ◆ 当四个主分区不能满足需求时，就要用到**扩展分区**，一种特殊的分区，只是逻辑分区的“容器”，不能直接存储数据。一块硬盘可以划分三个主分区加一个扩展分区，在扩展分区上划分出多个逻辑分区。
- ◆ **逻辑分区**是在扩展分区中创建的，没有个数限制。用户可以在主分区和逻辑分区中存储数据。

每一个操作系统的启动或者称作是引导程序，都存放在主分区上。  
这就是主分区和扩展分区及逻辑分区的最大区别。



# windows分区和linux分区

**Windows**中，一个逻辑分区对应一个逻辑驱动器，如**D盘**、**E盘**等。

## windows下的磁盘分区



**Linux**中，分区以文件形式显示，如/dev/sda1、/deb/sda2等。

```
[root@CM00 ~]# lsblk
```

| NAME                        | MAJ: MIN | RM | SIZE  | RO | TYPE | MOUNTPPOINT |
|-----------------------------|----------|----|-------|----|------|-------------|
| sr0                         | 11: 0    | 1  | 1024M | 0  | rom  |             |
| sda                         | 8: 0     | 0  | 20G   | 0  | disk |             |
| └─sda1                      | 8: 1     | 0  | 500M  | 0  | part | /boot       |
| └─sda2                      | 8: 2     | 0  | 19.5G | 0  | part |             |
| └─vg_centos6-lv_root (dm-0) | 253: 0   | 0  | 17.6G | 0  | lvm  | /           |
| └─vg_centos6-lv_swap (dm-1) | 253: 1   | 0  | 2G    | 0  | lvm  | [SWAP]      |
| sdb                         | 8: 16    | 0  | 4T    | 0  | disk |             |

# Linux磁盘分区的表示

**/dev/sdx~**

hd代表IDE设备  
sd代表SATA设备

a代表基本盘，b代表基本从属盘，  
c代表辅助主盘，d代表辅助从属盘

1~4为主分区或扩展分区  
从5开始是逻辑分区

/dev/sda1 表示第一块硬盘第1个分区

/dev/sdb3 表示第二块硬盘第3个分区

.....

**/dev/sda** 如果硬盘上没有分区，不加数字，表示整个硬盘。

## Linux分区

- ◆ 在 Linux 中规定，每一个硬盘设备最多能有 4 个主分区构成，任何一个扩展分区都要占用一个主分区号码，即在一个硬盘中，主分区和扩展分区一共最多是 4 个号码，因此，从5开始的是逻辑分区的编号。



```
[root@CM00 ~]# lsblk
```

| NAME                        | MAJ: MIN | RM | SIZE  | RO | TYPE | MOUNTPOINT |
|-----------------------------|----------|----|-------|----|------|------------|
| sr0                         | 11:0     | 1  | 1024M | 0  | rom  |            |
| sda                         | 8:0      | 0  | 20G   | 0  | disk |            |
| └─sda1                      | 8:1      | 0  | 500M  | 0  | part | /boot      |
| └─sda2                      | 8:2      | 0  | 19.5G | 0  | part |            |
| └─vg_centos6-lv_root (dm-0) | 253:0    | 0  | 17.6G | 0  | lvm  | /          |
| └─vg_centos6-lv_swap (dm-1) | 253:1    | 0  | 2G    | 0  | lvm  | [SWAP]     |
| sdb                         | 8:16     | 0  | 4T    | 0  | disk |            |

```
[root@CM00 wlw]# lsblk -f
```

| NAME                        | FSTYPE      | LABEL | UUID                                   | MOUNTPOINT |
|-----------------------------|-------------|-------|----------------------------------------|------------|
| sr0                         |             |       |                                        |            |
| sda                         |             |       |                                        |            |
| ├─sda1                      | ext4        |       | 9584aee5-487f-4646-97a3-01ec8cc5bd24   | /boot      |
| └─sda2                      | LVM2_member |       | 79Uonk-D3FQ-Z0e2-fs2P-1UU5-Ry6l-lhfIxb |            |
| ├─vg_centos6-lv_root (dm-0) | ext4        |       | f6defa28-7e98-4dc2-91e7-cdc701f73251   | /          |
| └─vg_centos6-lv_swap (dm-1) | swap        |       | 61aa98bc-0d81-4fdf-a684-dd0c25c198c7   | [SWAP]     |
| sdb                         |             |       |                                        |            |

分区情况

分区类型

分区的40位唯一标识码

挂载点

- ◆ 系统上的文件物理上是存放在硬盘的分区中，逻辑上是存放在目录中。
- ◆ **Linux**不能对分区直接操作，要“**挂载**”来使用分区的数据。
- ◆ 挂载就是将存储介质的内容映射到指定的目录中，即一个分区和一个目录联系，此目录即为该设备的挂载点(**mount point**)。

◆安装Linux时，根分区和交换分区（**swap**）必不可少。

一般需要三个分区：

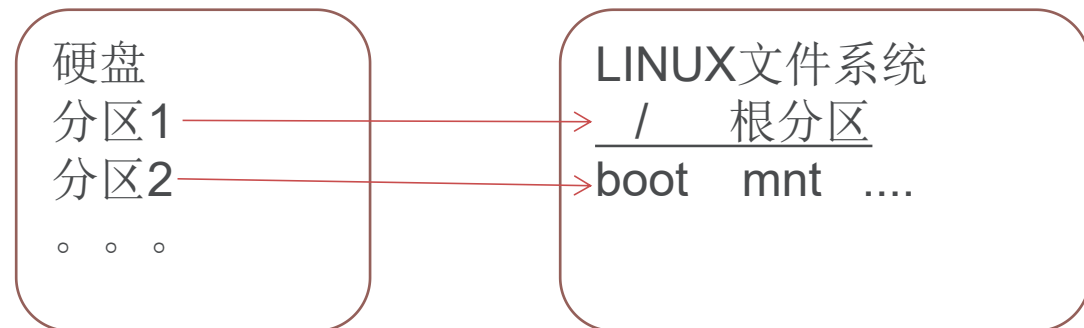
- 一个根分区 **/**：用于存放系统的所有系统文件
- 一个**boot**引导分区 **/boot**：存放系统内核和引导文件
- 一个交换分区（大小为物理内存的两倍）没有装载点

```
[root@CM00 ~]# lsblk
```

| NAME                        | MAJ: MIN | RM | SIZE  | RO | TYPE | MOUNTPOINT |
|-----------------------------|----------|----|-------|----|------|------------|
| sr0                         | 11:0     | 1  | 1024M | 0  | rom  |            |
| sda                         | 8:0      | 0  | 20G   | 0  | disk |            |
| ├─sda1                      | 8:1      | 0  | 500M  | 0  | part | /boot      |
| └─sda2                      | 8:2      | 0  | 19.5G | 0  | part |            |
| ├─vg_centos6-lv_root (dm-0) | 253:0    | 0  | 17.6G | 0  | lvm  | /          |
| └─vg_centos6-lv_swap (dm-1) | 253:1    | 0  | 2G    | 0  | lvm  | [SWAP]     |
| sdb                         | 8:16     | 0  | 4T    | 0  | disk |            |

●交换文件和交换分区等同于**WINDOS**中的虚拟内存，虚拟内存的文件名是**win386.swp**。

- **Linux**系统中的硬盘上的各个分区都会在启动过程中自动挂载到指定的目录，并在关机时自动卸载。
- 移**Linux**系统中的动存储介质既可以在启动时自动挂载，也可以在需要时手动挂载/卸载。
- **Linux**系统的挂载有一定先后顺序的，必须先挂载根目录/，才能挂载其它的文件系统到/的某个目录下。根目录/是一个独立且唯一的文件结构。其他每个分区都是用来组成整个文件系统的一部分。
- 不要认为都在/根目录之下就认为在一个分区之内，可能/**usr**和/**boot**是在两个不同的分区（甚至硬盘）之下，因为可能挂载在不同的分区。



◆ 查看当前分区和目录的挂载情况

命令： **df** [参数] [分区号/装载点]

参数： **-h/-H** 以KB、MB、GB等单位输出（以1000B为1KB，而非1024B）

```
[root@CM00 wlw]# df -h
```

| Filesystem                     | Size | Used | Avail | Use% | Mounted on |
|--------------------------------|------|------|-------|------|------------|
| /dev/mapper/vg_centos6-lv_root | 18G  | 4.4G | 12G   | 27%  | /          |
| tmpfs                          | 931M | 228K | 931M  | 1%   | /dev/shm   |
| /dev/sda1                      | 477M | 35M  | 417M  | 8%   | /boot      |
| . host: /                      | 301G | 143G | 158G  | 48%  | /mnt/hgfs  |

Linux的根分区

Linux的交换分区

Linux的启动分区

共享windows目录的文件系统

# 挂载文件系统 步骤

- ◆ 首先，挂载文件系统之前要对分区进行**格式化**。

## 建立文件系统即格式化

- 命令格式: **# mkfs [参数] <分区名称>**
    - 功能: 建立文件系统格式化分区
    - 参数: **-t** 文件系统类型 // 设定文件类型
    - c** // 检查分区有无坏道
    - v** // 显示详细信息
- 例: **mkfs -t ext4 /dev/sdb1**

- ◆ 然后，挂载文件系统，可以手动也可自动挂载。

## (1) 手动挂载

格式: **mount** [参数] <设备名> <装载点>

功能: 装载文件系统到指定的目录

参数: 例: **[root@CM00 ~]#mount /dev/sdb /maysdb**

- t** 文件系统类型 //指定文件类型
- l**: 显示已加载的文件系统列表;
- n**: 加载没有写入文件“/etc/mtab”中的文件系统;
- r**: 将文件系统加载为只读模式;
- a**: 加载文件“/etc/fstab”中描述的所有文件系统。

## (2) 自动装载

格式: **# vi /etc/fstab**

功能: 系统启动时自动装载

说明: **fstab**文件结构

| 卷标       | 装载点     | 类型   | 装载选项     | 备份选项 | 检查顺序 |
|----------|---------|------|----------|------|------|
| /dev/sdb | /maysdb | ext4 | defaults | 0    | 1    |

- ◆ **/etc/mtab**用于记录系统已经装载的文件系统
- ◆ **/etc/fstab**文件中列出了在系统**初启时需要自动安装**的所有分区和需要安装的每个文件系统。

### **/etc/fstab**文件结构（共6列）

| 设备名(分区/卷标)      | 挂载点            | 类型         | 装载选项            | 备份选项     | 检查顺序     |
|-----------------|----------------|------------|-----------------|----------|----------|
| <b>/dev/sdb</b> | <b>/maysdb</b> | <b>xfs</b> | <b>defaults</b> | <b>0</b> | <b>0</b> |

说明：

- 设备名(分区或卷标)列有时是用设备的**UUID**(用**blkid**命令可查看到已格式化了的设备的**UUID**)
- 装载选项 默认启动时自动装载就写**defaults**
- 最后两列就写**0，0**，旧版本时的作用，忽略。

## 卸载文件系统

(1) 通常在`/etc/fstab`文件中定义的文件系统都会自动卸载。

(2) 手工卸载文件系统

格式: `# umount [参数] <装载点>`

参数: `-t` 文件系统类型 //指定文件系统类型

实例:

`# umount /mnt/cdrom`

`# umount /dev/sdb` 或 `# umount /mnt/usb`

说明: 通常不能使用`mount`命令来安装`swap`文件系统, 而需要使用`swapon`命令来完成添加和使用。

# 在虚拟机的Linux系统使用U盘

## ◆ 在虚拟机中挂载/卸载U盘

1. 在虚拟机挂载U盘之前，要在Windows中启动VMware USB Arbitration Service和Virtual Disk 服务。
2. 为虚拟机增加U盘为硬盘设备。
3. 以root用户登录linux系统，开始挂载U盘。  
先**#mkdir /mnt/usb**  
再**#mount -t vfat /dev/sdb /mnt/usb**
4. 卸载U盘  
**# umount /dev/sdb** 或 **# umount /mnt/usb**

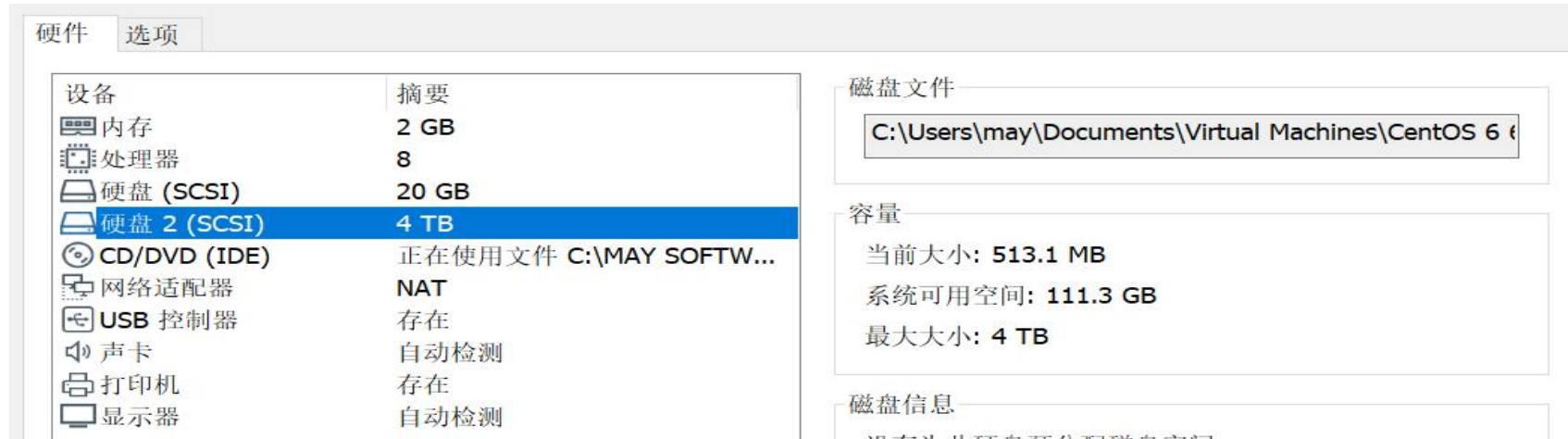
## ◆ 在虚拟机中自动挂载/卸载U盘

1. 同上；
2. 同上；
3. 再执行：**# vi /etc/fstab**
4. 再添加：**/dev/sdb /mnt/usb vfat defaults 0 1**

注意：

- ◆所有的接挂目录都必须事先存在的；
- ◆一个设备可以同时被装载到不同的目录中，一个目录也可以同时装载到不同的设备；
- ◆一个目录一旦被装载，或者说某个目录被挂接成另外的文件系统，那么该目录下原有的内容将被全部隐藏不可访问，直到取消装载，文件又会重现；
- ◆除了硬盘分区之外，也可以mount网络文件系统如nfs，smb等，以及**CDROM**和软盘等等。

# 在虚拟机的Linux系统增加一块硬盘sdb



```
[root@CM00 ~]# parted /dev/sdb
GNU Parted 2.1
使用 /dev/sdb
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted) print
错误: /dev/sdb: unrecognised disk label
(parted) █
```

- ◆ 当新加一个硬盘后，显示不出分区类型，要做格式化和挂载文件系统才能使用。

# 在虚拟机的Linux系统增加一块硬盘sdb

## ◆ 手动挂载硬盘

1) 硬盘添加：在虚拟机配置里

要重启虚拟机用**lsblk -f**才能看到**sdb**，但没有分区类型和**UUID**码。

2) 硬盘分区：执行**fdisk/parted /dev/sdb**

用**lsblk -f**此时能看到**sdb1**

3) 硬盘格式化：执行**mkfs -t ext4 /dev/sdb1**

4) 创建挂载目录：执行**mkdir /home/newdisk**

5) 硬盘手动挂载：执行**mount /dev/sdb1 /home/newdisk**

用**lsblk -f**此时能看到**sdb1**的挂载点**/home/newdisk**

6) 硬盘自动挂载：先执行**vi /etc/fstab**,

再添加**/dev/sdb1 /home/newdisk ext4 defaults 0 0**

# 补充：分区表类型

## 分区表类型

### ◆ MBR分区(MSDOS分区)

- 最多支持四个分区，有主分区、逻辑分区、扩展分区的概念
- 系统只能安装在主分区，扩展分区要占一个主分区
- 单个分区不能超过**2TB**，传统分区兼容性好

### ◆ GTP分区

- 支持无限个主分区，但操作系统可能限制，比如windows最多**128**个
- 最大支持**18EB**（**1EB=1024PB, 1PB=1024TB**）
- windows7 64位以后支持**GTP**

## 分区工具

### ◆ fdisk

- 只能对**MBR**类型分区

### ◆ parted

- 可以对**MBR**类型和**GTP**类型分区

# 补充：分区表类型

```
[root@CM00 ~]# parted /dev/sda
GNU Parted 2.1
Using /dev/sda
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted) help
 align-check TYPE N check partition N for TYPE(minlopt) alignment
 check NUMBER do a simple check on the file system
 cp [FROM-DEVICE] FROM-NUMBER TO-NUMBER copy file system to another partition
 help [COMMAND] print general help, or help on COMMAND
 mklabel, mktable LABEL-TYPE create a new disklabel (partition table)
 mkfs NUMBER FS-TYPE make a FS-TYPE file system on partition NUMBER
 mkpart PART-TYPE [FS-TYPE] START END make a partition
 mkpartfs PART-TYPE [FS-TYPE] START END make a partition with a file system
 move NUMBER START END move partition NUMBER
 name NUMBER NAME name partition NUMBER as NAME
 print [devices!free!list,all!NUMBER] display the partition table, available devices, free
 space, all a particular partition
 quit exit program
 rescue START END rescue a lost partition near START and END
 resize NUMBER START END resize partition NUMBER and its file system
 rm NUMBER delete partition NUMBER
 select DEVICE choose the device to edit
 set NUMBER FLAG STATE change the FLAG on partition NUMBER
 toggle [NUMBER [FLAG]] toggle the state of FLAG on partition NUMBER
 unit UNIT set the default unit to UNIT
 version display the version number and copyright information of
 GNU Parted
(parted)
```

指明创建分区的类型

创建分区

显示分区类型等信息

给出分区号，删除分区

# 补充：分区表类型

```
[root@CM00 wlw]# fdisk -l /dev/sda
```

```
Disk /dev/sda: 21.5 GB, 21474836480 bytes
255 heads, 63 sectors/track, 2610 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: 0x0001b74a
```

| Device                                         | Boot | Start | End  | Blocks   | Id | System    |
|------------------------------------------------|------|-------|------|----------|----|-----------|
| /dev/sda1                                      | *    | 1     | 64   | 512000   | 83 | Linux     |
| Partition 1 does not end on cylinder boundary. |      |       |      |          |    |           |
| /dev/sda2                                      |      | 64    | 2611 | 20458496 | 8e | Linux LVM |

```
[root@CM00 wlw]# parted /dev/sda
```

```
GNU Parted 2.1
```

```
使用 /dev/sda
```

```
Welcome to GNU Parted! Type 'help' to view a list of commands.
```

```
(parted) print
```

```
Model: VMware, VMware Virtual S (scsi)
```

```
Disk /dev/sda: 21.5GB
```

```
Sector size (logical/physical): 512B/512B
```

```
Partition Table: msdos
```

| Number | Start  | End    | Size   | Type    | File system | 标志  |
|--------|--------|--------|--------|---------|-------------|-----|
| 1      | 1049kB | 525MB  | 524MB  | primary | ext4        | 启动  |
| 2      | 525MB  | 21.5GB | 20.9GB | primary |             | lvm |

◆ 磁盘分区使用和挂载情况: **df -lh**

```
[root@CM00 wlw]# df -h
Filesystem Size Used Avail Use% Mounted on
/dev/mapper/vg_centos6-lv_root
 18G 4.4G 12G 27% /
tmpfs 931M 228K 931M 1% /dev/shm
/dev/sda1 477M 35M 417M 8% /boot
. host: / 301G 143G 158G 48% /mnt/hgfs
```

◆ 指定目录的占用情况: **du -h /目录名**

例: 查询**opt**目录深度为1的磁盘占用情况

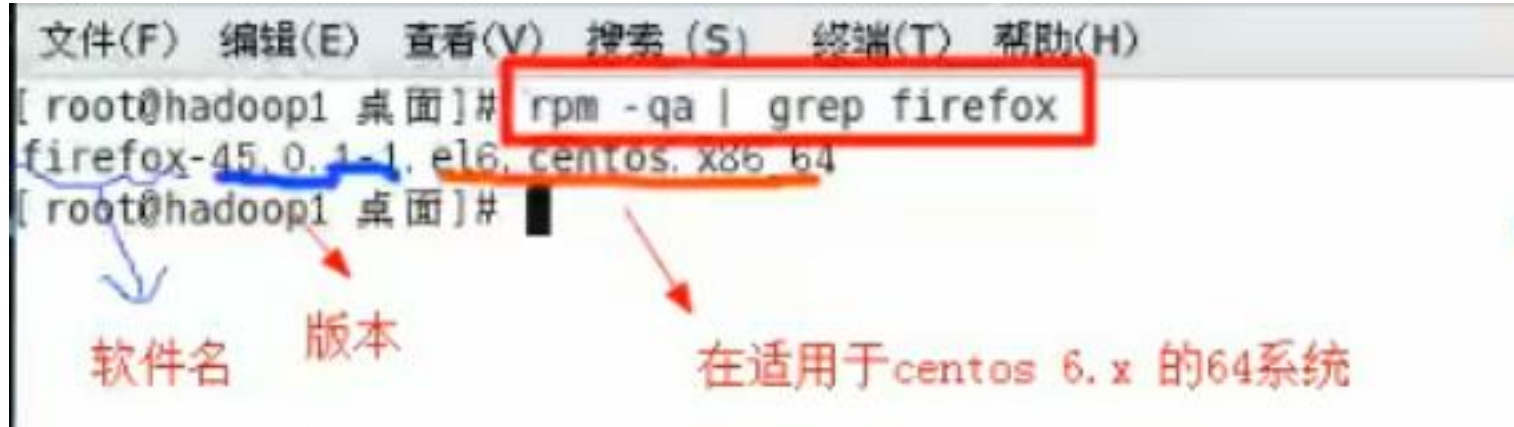
**du -ach --max-depth=1 /opt**

# 其他实用指令举例

---

- ◆ 统计/home文件夹下文件的个数
- ◆ 统计/home文件夹下目录的个数
- ◆ 统计/home文件夹下含子文件夹下文件的个数
- ◆ 统计/home文件夹下含子文件夹下目录的个数
- ◆ 以树状显示目录结构

## 查询已安装的rpm包指令及包格式



```
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
[root@hadoop1 桌面]# rpm -qa | grep firefox
firefox-45.0.1-1.el6.centos.x86_64
[root@hadoop1 桌面]#
```

Annotations:

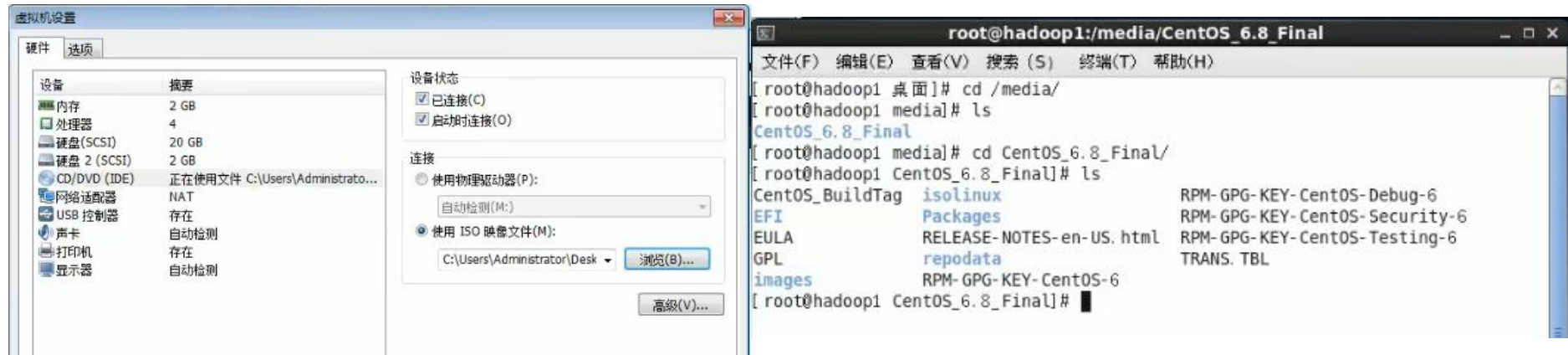
- 软件名 (points to `firefox`)
- 版本 (points to `45.0.1-1`)
- 在适用于centos 6.x 的64系统 (points to `el6.centos.x86_64`)

常用——

- `rpm -qa | grep ***` //查询是否安装了rpm包
- `rpm -qi 软件包名` //查询安装的软件包信息
- `rpm -ql 软件包名` //查询rpm包中的文件，文件安装在哪里
- `rpm -qf 文件全路径名` //查询文件属于哪个软件包

## ◆ rpm -ivh 软件包名全路径名 //安装软件包

参数说明: **i**-安装; **v**-提示; **h**-进度条



1)演示安装**firefox**浏览器  
步骤先找到 **firefox** 的安装**rpm**包你需要挂载上我们安装**centos**的**ios**文件, 然后到 **media** 下去找**rpm** 找。

```
[root@hadoop1 Packages]# ls -l firefox-45.0.1-1.el6.centos.x86_64.rpm
-r--r--r--. 2 root root 77493120 5月 12 2016 firefox-45.0.1-1.el6.centos.x86_64.rpm
[root@hadoop1 Packages]# cp firefox-45.0.1-1.el6.centos.x86_64.rpm /opt/
[root@hadoop1 Packages]# cd /opt/
[root@hadoop1 opt]# ls
firefox-45.0.1-1.el6.centos.x86_64.rpm VMwareTools-10.0.5-3228253.tar.gz
home vmware-tools-distrib
rh 金庸-射雕英雄传txt精校版.txt
tmp
[root@hadoop1 opt]# rpm -ivh firefox-45.0.1-1.el6.centos.x86_64.rpm
Preparing... ##### [100%]
1: firefox ##### (33%)
```

## ◆ rpm -e 软件包名 //卸载软件包

注意:

1)如果其它软件包依赖于您要卸载的软件包，卸载时则会产生错误信息。如:**\$rpm -e foo**

**removing these packages would break dependencies:foo is needed by bar-1.0-1**

2)如果就是要删除foo这个rpm包，可以增加参数**-nodeps**就可以强制删除，但是一般不推荐这样做，因为依赖于该软件包的程序可能无法运行。如:**\$rpm-e-nodeps foo**

# Yum-前端软件包管理

**Yum**是一个**Shell**前端软件包管理器。基于**RPM**包管理，能够从指定的**yum**服务器自动下载**RPM**包并且安装，可以自动处理依赖性关系，并且一次安装所有依赖的软件包。

说明：使用前提是能联网。

**yum**的基本指令

- ◆ 查询**yum**服务器是否有需要安装的软件

**yum list | grep xx**软件列表

安装指定的**yum**包

**yum install xxx** 下载安装

- ◆ **yum**举例:安装**firefox**