



第 2 章

文件管理与常用命令

文件管理与常用命令

- 文件和目录的层次结构，命名规则，文件通配符
- 文件管理，目录管理
- 文件的归档与压缩处理
- 硬连接和符号连接
- 系统调用
- 文件和目录的访问
- 文件和目录的权限



2.1 文件和目录的层次结构

文件和目录的布局 (1)

- / 根目录
 - 所有目录或文件的起点
- /boot 目录
 - 存放系统引导文件
- /etc 目录
 - 供系统维护管理用的命令和配置文件
 - **passwd** 文件: 用户相关的配置信息
 - **issue** 文件: 登录前在 **login** 之上的提示信息

文件和目录的布局 (2)

- /bin
 - 系统常用命令，是二进制可执行程序，如 `ls` , `ln` , `cp` , `cat` 等
- /home
 - 用户目录，用于存放除了超级用户以外的其他个人用户数据。个人用户的主目录在此目录下。
- /root
 - 超级用户的主目录。 **root** 主目录位置保证了即使其他分区出问题， **root** 也能正常工作

文件和目录的布局 (3)

- /tmp
 - 存放临时文件
- /dev
 - 存放设备文件，如终端设备，磁带机，打印机等
- /usr
- 二级目录结构， **UNIX** 系统资源，巨大多数应用软件都以这个目录为起点，这是 **Linux** 桌面系统中最大的目录
 - /usr/include
 - C 语言头文件存放目录
 - /usr/tmp
 - 存放临时文件
 - /usr/bin
 - 存放一些常用命令，如 **echo** ， **grep** ， **kill** 等

文件和目录的布局 (4)

- `/lib,/usr/lib`
 - 存放各种库文件，指 C 语言的链接库文件，以及 **terminfo** 终端库等等
 - 静态链接库文件有 **.a** 后缀 (archive, 存档)
 - 动态链接库文件后缀是 **.so**(shared objects)
 - **Linux** 缺省情况下首先连接动态链接库




2.2 文件和目录命名

文件和目录的命名

- 名字长度
 - 允许 1 – 256 字符
- 取名的合法字符
 - 文件名的字符包括：字母、数字、.（点）、_（下划线）和 -（连字符）。
 - 有些转意字符在 **Linux** 的命令解释器（**shell**）中有特殊的含义。这样的转意字符有：?（问号）、*（星号）、（空格）、\$（货币符）、&、扩号等等。在文件名中应尽量避免使用这些字符。文件名中可以有（空格），但建议用户用_（下划线）来替代。/ 既可代表目录树的根也可作为路径名中的分隔符（类似 **DOS** 下的 \），因此 / 不能出现在文件名中。

■ 大小写字母有区别



2.3 **shell** 的文件名通配符

文件通配符规则

(1)

- 星号 *
 - 匹配任意长度的文件名字符串（包括空字符串）
- 点字符 (.) ,
 - 当它作为文件名或路径名分量的第一个字符时，必须显式匹配
- \ (反斜线)
 - 也是一个特殊的字符。它屏蔽后继特殊字母的特殊含义（转意），使该字符仅取其符号所代表的字面意义。
- 例： ***file** 匹配 **file** , **makefile** , 不匹配 **.profile** 文件

try*c 匹配 **try1.c** **try.c** **try.basic**

文件通配符规则（2）

- 问号 **?**
 - 匹配任一单字符
- 方括号 **[]**
 - 匹配括号内任一字符，也可以用减号指定一个范围
 - 例：`[A-Z]*` `*.[ch]` `[Mm]akefile`
- 注意
 - 文件名通配符规则与正则表达式的规则不同，应用场合不同
 - 不同种类 shell 通配符规则会略有些差别

与 DOS 文件通配符的区别 (1)

- 例 1 : UNIX 文件通配符比 DOS 严谨, 无二义性
设当前目录下有 `xcom.exe` `xcom.c`
`xcom.obj`

- DOS 中

`DIR XCOM*` 会列出三个文件

`DEL XCOM*` 删不掉上述任何文件

- UNIX 不存在二义性解释

`ls xcom*` 会列出三个文件

`rm xcom*` 会删除三个文件

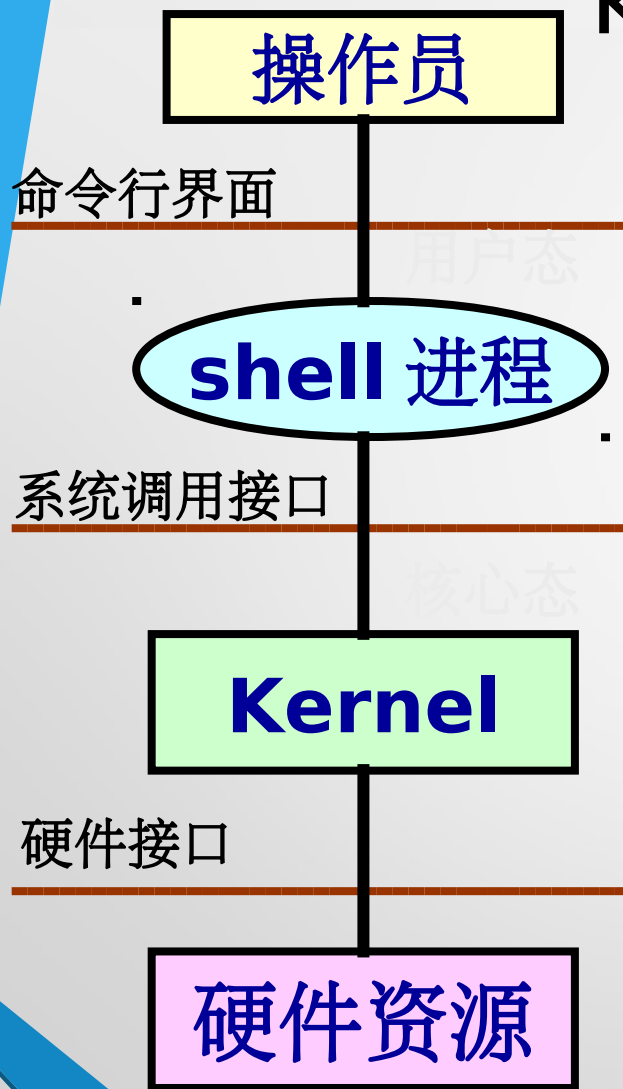
例 2: DOS 中 `*.*` 匹配所有文件

- Linux 中 `*.*` 要求文件名中必须含有圆点, 否则不匹配, 如: `*.*` 与 `makefile` 不匹配

与 DOS 文件通配符的区别 (2)

- 例 3 : 在 DOS 中, 无法使用通配串 `*temp*`
 - 在 Linux 中对 `*temp*` 严格按前述规则去理解, 而且 `*temp*list*` 也可用
- 例 4 : 子目录名的匹配
 - 在 UNIX 中可以使用 `*/*. [ch]` 通配符, DOS 中不许
- 例 5 : Linux 中文件通配符适用所有命令
 - Linux 中文件通配符允许用于任何命令, 而 DOS 中只能用于 `dir/del/copy` 等有限的几个命令中
 - Linux 中命令 `cat *.c` 可以列出所有的 `.c` 文件内容, DOS 中命令 `TYPE *.C` 不可

shell 与 kernel



- shell

- shell 是一个用户态进程
- 对用户提供了命令行界面
- 使用操作系统核心提供的功能

- kernel : 操作系统核心

- 管理系统资源（包括内存，磁盘等）运行在核心态
- 通过软中断方式对用户态进程提供系统调用接口

shell 文件名通配符处理

- 文件名通配符的处理由 **shell** 完成，分以下三步
 - 在 **shell** 提示符下，从键盘输入命令，被 **shell** 接受
 - **shell** 对所键入内容作若干加工处理，其中含有对文件通配符的展开工作（文件名生成），生成结果命令
 - 执行前面生成的结果命令

文件名通配符举例 (1)

- 设当前目录下只有 `try.c` , `zap.c` , `arc.c` 三文件
 - 键入内容 `cat *.c`
 - 实际执行 `cat arc.c try.c zap.c` (按字典序)
 - 对命令 `cat` 来说 , 指定了 3 个文件

文件名通配符举例 (2)

键入命令时的简化输入

手工键入

`vi m*e`



实际执行

`vi makefile`

手工键入

`cd *sna*`



实际执行

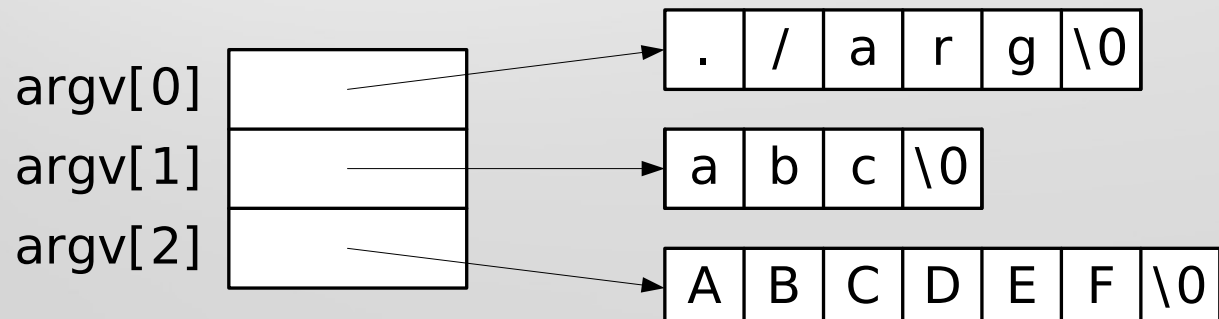
`cd configure-IBM-sna-network.d`

程序获取命令行参数的方式

- 从main的两个参数，可获得命令行参数的内容
- 演示程序arg.c

```
void main(int argc, char *argv[])  
{  
    int i;  
    for (i = 0; i < argc; i++)  
        printf("%d:[%s]\n", i, argv[i]);  
}
```

- 编译，链接： gcc arg.c -o arg
- 运行 ./arg abc ABCDEF



验证文件通配符处理方式

- 执行

- `./arg *`

- `./arg /usr/include/*`

- `./arg */* /usr/*`

- 执行结果与同样 `arg.c` 在 DOS 下执行结果比较

- Linux 由 shell 完成对文件通配符的展开


- DOS 由命令自身来解释文件通配符

```
[yli@gpul6 linuxtest]$ cc arg.c -o arg
[yli@gpul6 linuxtest]$ ./arg abc ABCDEF
0:[./arg]
1:[abc]
2:[ABCDEF]
```

```
[yli@gpul6 linuxtest]$ ./arg *
0:[./arg]
1:[arg]
2:[arg.c]
```

```
[yli@gpul6 linuxtest]$ ./arg */* /usr/*
0:[./arg]
1:[*/]
2:[/usr/bin]
3:[/usr/etc]
4:[/usr/games]
5:[/usr/include]
6:[/usr/lib]
7:[/usr/lib64]
8:[/usr/libexec]
9:[/usr/local]
10:[/usr/sbin]
11:[/usr/share]
12:[/usr/src]
13:[/usr/tmp]
```

```
[yli@gpul6 linuxtest]$ ./arg /usr/include/*
0:[./arg]
1:[/usr/include/aio.h]
2:[/usr/include/aliases.h]
3:[/usr/include/alloca.h]
4:[/usr/include/a.out.h]
5:[/usr/include/argp.h]
6:[/usr/include/argz.h]
7:[/usr/include/ar.h]
8:[/usr/include/arpa]
9:[/usr/include/asm]
10:[/usr/include/asm-generic]
11:[/usr/include/assert.h]
12:[/usr/include/autosprintf.h]
13:[/usr/include/bits]
14:[/usr/include/byteswap.h]
15:[/usr/include/bzlib.h]
16:[/usr/include/c++]
17:[/usr/include/com_err.h]
18:[/usr/include/complex.h]
19:[/usr/include/cpio.h]
20:[/usr/include/cpufreq.h]
21:[/usr/include/crypt.h]
22:[/usr/include/ctype.h]
23:[/usr/include/cublas_api.h]
```



2.4 文件管理

Linux 常用文件类型

- 普通文件

最常用的一类文件，其特点是不包含文件系统的结构信息。通常用户所接触到的文件，如图形文件、数据文件、文档文件等都属于这种文件。

- 目录文件

目录文件是用于存放文件名及其相关信息的文件。是内核组织文件系统的基本节点。

- 链接文件

是一种特殊文件，实际上是指向一个真实存在的文件链接，类似于 **Windows** 下的快捷方式。分为硬链接文件和软链接文件。

- 设备文件

- 管道文件：一种特殊文件，主要用于不同进程间的信息传递

Linux 目录常见概念

- 路径：用'/' 隔开 相对路径 绝对路径
 - `pwd` 命令：显示当前路径
- 根目录
 - “/”：所有目录的起点，操作系统本身的驻留程序存放在以根目录开始的专用目录中。
- 用户主目录
 - 用户主目录是系统管理员增加用户时
 - `sudo passwd root` //Ubuntu 初始时不建立 `root` 用户，需要设置密码后才建立好 `root` 用户

cat: 显示文件内容

cat[选项] [文件名]

- `cat try.c`

在终端显示出文本文件 `try.c` 的内容。

- `cat > try.txt`

从标准输入（键盘）获取数据，直到按 `ctrl+d` 键标志输入结束。键盘输入内容存入文件 `try.txt`

- `cat try1.c try2.c try.h`

将 3 个文件按顺序输出

- `cat try1.c try2.c try.h > trysrc`

- `cat makefile*.[ch] > src`

cat: 显示文件内容

表 2.2 cat 命令选项说明

选 项	说 明
-n 或--number	由 1 开始对所有输出行进行编号
-b 或--number-nonblank	和 -n 相似,只不过对于空白行不编号
-s 或--squeeze-blank	当遇到有连续两行以上的空白行,将其合并成一个空行
--help	显示该命令的用户,并退出,其返回码表示成功

ls: 文件名列表

- 基本功能

- 不给出实参时，列出当前目录下所有文件和目录
- 实参为文件时，列出文件项
- 实参为目录时，列出目录下的所有文件项
- 在同一命令行中可以指定多个实参

- ls 命令有几十个选项 **ls[选项][目录名或文件名]**

- 控制列表格式，有选择的为每个项目列出某些属性

- 选项 -F (Flag)

- 若列出的是目录，就在名字后面缀以斜线 **/**
- 若列出的是可执行文件，就在名字后面缀以星号 *****
- 若列出的是符号连接文件，就在名字后面缀以符号 **@**
- 若列出的是普通文件，则名字面后无任何标记

表 2.1 ls 命令选项说明

选 项	说 明
-a	列出目录下的所有文件,包括以.开头的隐含文件
-b	把文件名中不可输出的字符用反斜杠加字符编号的形式列出
-d	将目录像文件一样显示,而不是显示其中所包含的文件
-e	输出时间的全部信息,而不是输出简略信息
-k	以 k 字节的形式表示文件的大小
-l	列出文件的详细信息
-m	横向输出文件名,并以“,”作分格符
-n	用数字的 UID、GID 代替名称
-p 或 -F	在每个文件名后附上一个字符以说明该文件的类型,“*”表示可执行的普通文件;“/”表示目录;“@”表示符号链接;“ ”表示 FIFOs;“=”表示套接字
-r	对目录反向排序
-s	在每个文件名后输出该文件的大小
-t	以时间排序
-u	以文件上次被访问的时间排序
-x	按列输出,横向排序
-1	一行只输出一个文件
-color=no	不显示彩色文件名

ls 选项 -F 举例

- 命令 `ls -F` 的执行结果举例

bin/	pmd@
------	------

core	tmp/
------	------

dev/	unix@
------	-------

etc/	usr/
------	------

lost+found/	var/
-------------	------

mnt/	zap*
------	------

ls 选项 -l: 长格式列表 (1)

- 例: `ls -l arg`
 - `-rwxr-x--x 1 liang stud 519 Jul 5 15:02 arg`
- 第 1 列: 文件属性
 - 第 1 字符为文件类型
 - 普通文件
 - b** 块设备文件 (Block)
 - d** 目录文件 (Dir)
 - c** 字符设备文件 (Char)
 - l** 符号连接文件 (Link)
 - p** 命名管道文件 (Pipe)
- 文件的访问权限 (`rwX` 读权限, 写权限, 可执行权限)
 - 2-4 字符: 文件所有者对文件的访问权限
 - 5-7 字符: 同组用户对文件的访问权限
 - 8-10 字符: 其它用户对文件的访问权限

ls 选项 -l: 长格式列表 (2)

- 第 2 列: 文件 link 数或目录子目录数

```
-rwxr-x--x 1 liang stud 519 Jul 5 15:02 arg
```

- 第 3 列, 第 4 列: 文件主的名字和组名
- 第 5 列 文件大小
- 普通磁盘文件: 列出文件大小 (字节数)
- 目录: 列出目录表大小, 不是目录下文件长度和
- 符号连接文件: 列出符号连接文件自身的长度
- 字符设备和块设备文件: 列出主设备号和次设备号
- 管道文件: 列出管道内的数据长度
- 第 6 列: 文件最后一次被修改的日期和时间
- 第 7 列: 文件名

对于符号连接文件, 附带列出符号连接文件的内容

```

drwxr-xr-x    3 bin  bin    3584 Jul 11 11:55 bin
-rw-----    1 root root 164470 Oct  2 11:43 core
drwxr-xr-x   11 bin  bin    7168 Oct 18 09:55 dev
drwxrwxr-x   27 bin  auth    7680 Oct 18 09:55 etc
drwxr-xr-x    2 root root   1024 Jul 11 07:24 lost+found
drwxrwxrwx    2 root bin     512 Jul 28 1998 mnt
lrwxrwxrwx    1 root root     35 Jul 11 07:31 pmd ->
    /var/opt/K/SC0/Unix/5.0.5Eb/pmd/pmd
drwxrwxrwt    2 sys  sys    4096 Oct 18 10:48 tmp
lrwxrwxrwx    1 root sys     11 Jul 11 07:31 unix ->
    /stand/unix
drwxrwxr-x   25 root auth    512 Oct  2 17:18 usr
drwxr-xr-x    6 root sys     512 Jul 11 07:43 var
crw-r--r--    1 bin  ter      0,  9 Oct 18 09:56 /dev/tty10
prw-r--r--    1 root sys    2642 Oct 18 11:07 /tmp/pipe

```


ls 命令实例

- `cat > a` // 新建一个文件 a
- `$ls -l a`
- `ln -s a b` // 建立符号链接文件 b
- `$ ls -l b`
- 观察两个 `ls` 命令的输出结果。

分屏显示文件内容 **more** 命令

- 命令功能：分屏显示文件内容
- 命令格式： **more** [选项] 文件名

more 命令选项说明

选项	说明
-num	指定屏幕上一次所显示的行数；
-d	提示使用者，在画面下方显示 --More-- [Press space to continue, 'q' to quit.] ，如果使用者按错键，则会显示 [Press 'h' for instructions.] ；
-l	取消遇见特殊字元 ^L （送纸字元）时会暂停的功能；
-f	计算行数时，以实际上的行数，而非自动换行过后的行数（有些单行字数太长的会被扩展为两行或两行以上）；
-p	不以卷动的方式显示每一页，而是先清除屏幕后再显示内容；
-c	跟 -p 相似，不同的是先显示内容再清除其他旧资料；
-s	当遇到有连续两行以上的空白行，就代换为一行的空白行；
-u	不显示下引号（根据环境变数 TERM 指定的 terminal 而有所不同）；
+/	在每个文件显示前搜寻该字串（ pattern ），然后从该字串之后开始显示；
+num	从第 num 行开始显示。

在使用 **more** 命令时，可以用如下不同的方法对提示作出回答

- 按空格键，显示文本的下一屏内容
- 按 **Enter** 键，显示文本的下一行内容
- 按斜线符号 (/)，接着输入一个模式，可以在文本中寻找下一个相匹配的模式。
- 按 Q 键，退出 **more** 命令。

more 命令实例

- 显示 /etc/adduser.conf 文件内容

```
$more /etc/adduser.conf
```

- 逐页显示 **file1** 内容，如有连续两行以上空白行则以一行空白行显示。

```
$more -s file1
```

- 从第 20 行开始显示 **file1** 文件内容

```
$more +20 file1
```

- 建立一个由数字组成的文件 **f2** ， **more +/3 f2** ， 观察执行结果。

less 命令实例

表 2.4 less 命令选项说明

选 项	说 明
-a	在当前屏幕显示最后一页
-c	从顶部(从上到下)刷新屏幕,并显示文件内容,而不是通过底部滚动完成刷新
-f	强制打开文件,二进制文件显示时不提示警告
-i	搜索时忽略大小写,除非搜索串中包含大写字母
-I	搜索时忽略大小写,除非搜索串中包含小写字母
-m	显示读取文件的百分比
-M	显示读取文件的百分比、行号及总行数
-N	在每行前输出行号
-p pattern	搜索 pattern
-s	把连续多个空白行作为一个空白行显示
-Q	在终端下不响铃

head 命令

- 功能：显示文件的头几行
- 格式： `head -number filename`
- 显示前 `number` 行，若不指定则显示 **10** 行。
- 实例： `head -5 /etc/adduser.conf`

tail 命令

- 功能：显示文件的最后几行
- 格式： **tail -n filename**
- 显示文件的最后 n 行
- **tail -5 /etc/adduser.conf**

练习

自己创建一个文件 **file1**，行数大于 **50**，内容不限，要求里面包含“**fifth**”

- 1. 使用 **less** 命令显示 **file1** 内容的同时，每行加行号
- 2. 使用 **less** 命令显示 **file1** 中包含“**fifth**”的内容
- 3. 分别显示文件前 **20** 行、前 **10** 行
- 4. 显示文件最后 **20** 行、后 **10** 行

cp: 拷贝文件

- 命令功能: 复制文件或目录
- 命令格式: `cp [选项] 源文件或目录 目标文件或目录`

cp 命令选项说明

选项	说明
-a	该选项通常在复制目录时使用。它递归地将源目录下的所有子目录及其文件都复制到目标目录中，并且保留文件链接和文件属性不变。它等效于 -dpr
-d	复制时保留文件链接；
-f	若目标文件或目录已存在，则覆盖已存在的目标文件或目录并且不提示；默认
-i	和 f 选项相反，在覆盖目标文件之前将给出提示要求用户确认。回答 y 时目标文件将被覆盖，是交互式复制；
-p	此时 cp 除复制源文件的内容外，还将把其修改时间和访问权限也复制到新文件中；
-R , -r	若给出的源文件是一目录文件，此时 cp 将递归复制该目录下所有的子目录和文件至目的地。此时目标文件必须为一个目录名；
-l	不复制，而是创建指向源文件的硬链接文件，链接文件名由目标文件给出。
-s	对源文件建立符号链接，而非复制文件

cp: 拷贝文件

- 例 :cp a b // 将 a 复制到 b
 - cp -i c b // 询问是否覆盖, 选择不覆盖, b=a
 - cp -f c b // 强行复制, 此时 b=c 默认是不提示
 - cp a b c // 不执行, 因为 c 不是目录
 - cp -r dir1 dir2 // 建立 dir2 且将 dir1 及其子目录或文件复制到 dir2 中
 - cp -r a c dir2 // 将 a,c 复制到 dir2 中

cp: 拷贝文件（例 2）

- 例 1：将当前目录下所有的 C 语言程序复制到 /home/ 用户名 /linuxtest 目录中

```
cp *.c /home/ 用户名 /linuxtest
```

mv: 移动文件

- 基本语法

mv [选项] 源文件或目录 目标文件或目录

选项: **-i** 交互式操作; **-f**: 禁止交互操作。

- **mv** *file1 file2*
- **mv** *file1 file2 ... file_n dir*
- **mv** *dir1 dir2*

- 功能

- 使用 **mv** 命令可以将文件和目录改名
- 可以将文件和子目录从一个目录移动到另一个目录

mv 命令实例

- 将当前目录下的文件 `source.txt` 移动到 `/tmp` 下
- `mv source.txt /tmp`
- 将目录 `dir1` 、 `dir2` 移到目录 `dir_dist` 下
- `mv dir1 dir2 dir_dist`
- 将文件 `old.txt` 更名为 `new.txt`
- `mv old.txt new.txt`

rm: 删除文件

- 命令格式

- `rm file1 file2 ... fileN`

- 例

`rm core a.out`

`rm *.o *.tmp`

`rm *.bak`

- 选项

- **-r** 递归地 (Recursive) 删除实参表中的目录，也就是删除一整棵目录树。
 - **-i** 每删除一个文件前需要操作员确认 (Inform)
 - **-f** 强迫删除 (Force)。只读文件也被删除并且无提示

- 其它问题

- 正在运行的可执行程序文件不能被删除

显式地区分命令选项和处理对象

- 问题

设当前目录下只有 `a` , `b` , `c` 三个文件

`rm -i`

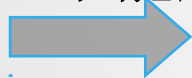
只提供选项，未指定任何文件，命令格式错
生成文件 `-i` (符合文件的命名规则)

`who>-i`

`rm -i`

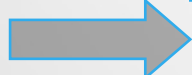
不能删除文件 `-i`

`rm *`



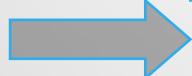
`rm -i a b c`

`cat *`



`cat -i a b c`

`ls *`



`ls -i a b c`

- 解决方法

- 许多 UNIX 命令 (如

`cp` , `ls` , `mv` , `rm` , `cat` , `grep` , `set` 等) 用 `--` 显式地
标志命令行选项的结束，识别以 `-` 开头的处理对象。如: `rm`

`-- -i` (删除文件 `-i`)

Sort 命令

- 功能说明：将文本文件内容加以排序 ,**sort** 可针对文本文件的内容，以行为单位来排序。
- 格式： **sort** [选项] filename

sort 命令选项

- **-m** 将已排序的输入文件，合并为一个排序后的输出数据流。
- **-n** 以整数类型比较字段
- **-o outfile** 将输入写到指定文件，而非标准输出。如果该文件为输入文件之一，则 **sort** 在进行排序写到输入文件之前，会先将它复制到一个临时文件
- **-r** 倒置排序的顺序为 由大至小（ **descending** ），而非默认的由小至大（ **ascending** ）
- **-t char** 使用单个字符 **char** 作为默认的字段分割字符，取代默认的空白字符。
- **-u** 只有唯一的记录，丢弃所有具有相同键值的记录，只留其中的第一条。只有键值字段是重要的，也就是说：被丢弃的记录其他部分可能是不同值。
- 行为模式： **sort** 会读取指定的文件，如果未给定文件，则读取标准输入，在将排序好的数据写至标准输出。

Sort 命令选项

- **-b** 忽略开头的空白
- **-c** 检查输入是否已正确排序，如输入未经排序，但退出码 (exit code) 为非零值，则不会有任何输出
- **-d** 字典顺序：仅文字数字与空白才有意义
- **-g** 一般数值：以浮点数字类型比较字段。这个选项的运作有点类似 **-n**，差别仅在于这个选项的数字可能有小数点及指数。（仅 **GNU** 版本提供此功能）
- **-f** 以不管字母大小写的方式排序
- **-i** 忽略无法打印的字符
- **-k** 定义排序键值字段（该选项后接一个字段编号，或则是一对数字。有时 **-k** 之后可用空白分隔。每个编号后都可以接一个点号的字符位置，及 / 或 修饰符 (modifier) 字母之一。且当出现多个 **-k** 选项时候，会先从第一个键值开始排序，找出匹配该键值的记录后，再进行第二个键值字段的排序，以此类推。）

sort 将文件的每一行作为一个单位，相互比较，比较原则是从首字符向后，依次按 **ASCII** 码值进行比较，最后将他们按升序输出

```
[root@zhoucentos log]# cat seq
```

```
banana
```

```
apple
```

```
pear
```

```
orange
```

```
[root@zhoucentos log]# sort seq
```

```
apple
```

```
banana
```

```
orange
```

```
pear
```

sort 的 -u 选项它的作用很简单，就是在输出行中去除重复行

```
[root@zhoucentos log]# cat seq
banana
apple
pear
orange
apple
pear
[root@zhoucentos log]# sort -u seq
apple
banana
orange
pear
```

sort 的 -n 选项，按数值排序

```
[root@zhoucentos log]# cat number
```

1

3

2

5

78

11

4

```
[root@zhoucentos log]# sort number
```

1

11

2

3

4

5

78

```
[root@zhoucentos log]# sort -n number
```

1

2

3

4

5

11

78

sort 的 -o 选项，想把排序结果输出到原文件中，用重定向可不行

```
[root@zhoucentos log]# cat number
```

```
78
```

```
5
```

```
4
```

```
3
```

```
2
```

```
11
```

```
1
```

```
[root@zhoucentos log]# sort -nr number -o number
```

```
[root@zhoucentos log]# cat number
```

```
78
```

```
11
```

```
5
```

```
4
```

```
3
```

```
2
```

```
1
```


sort 的 -t 选项和 -k 选项

```
[root@zhoucentos log]# cat date  
2017-12-02  
2017-01-09  
2017-10-23  
2017-04-24
```

这个文件有三列，列与列之间用“-”隔开了，第一列表示年，第二列表示月，第三列表示日。那么想以月来排序，也就是以第二列来排序，如何利用 **sort** 实现？

方法：**sort** 提供了 **-t** 选项，后面可以设定间隔符。指定了间隔符之后，就可以用 **-k** 来指定列数了。

```
[root@zhoucentos log]# sort -n -k 2 -t'-' date
```

```
2017-01-09
```

```
2017-04-24
```

```
2017-10-23
```

```
2017-12-02
```

```
jlgao:x:1116:100:~/users5/jlgao:/bin/bash
yuzhouzhang:x:1117:100:~/users8/yuzhouzhang:/bin/bash
yjtian:x:1118:100:~/users8/yjtian:/bin/bash
sychen:x:1119:100:~/users5/sychen:/bin/bash
```

- (1)-k2.4,5.6 指的是从第二个字段的第 4 个字符开始比较，一直比到第五个字段的第六个字符。
- (2) `sort -t: -k1,1 /etc/passwd` 以冒号隔开的第一个字段：用户名称 对 `/etc/passwd` 进行排序
- (3)`sort -t: -k3nr /etc/passwd` 以冒号隔开的第 3 个字段 `uid` 反向（由大到小）排序
- (4)`sort -t: -k4n -k3n /etc/passwd` 以冒号隔开的第 4 个字段 `GID`，以及第 3 个字段 `uid` 排序

作业

创建文件 testsort.txt 如下所示

```
[root@FDMdevBI opt]# cat testsort.txt
google 110 5000
baidu 100 5000
guge 50 3000
sohu 100 4500
```

第一个域是公司名称，
第二个域是公司人数，
第三个域是员工平均工资。

完成如下题目

- 1. 让这个文件按公司的字母顺序排序，也就是按第一个域进行排序
- 2. 从公司英文名称的第二个字母开始进行排序
- 3. 按照公司人数排序，人数相同的只保留一个
- 4. 让 **testsort.txt** 按照公司人数排序，人数相同的按照员工平均工资降序排序

- 2. 从公司英文名称的第二个字母开始进行排序

```
[root@FDMdevBI opt]# sort -t ' ' -k1.2 testsort.txt
```

```
baidu 100 5000
```

```
sohu 100 4500
```

```
google 110 5000
```

```
guge 50 3000
```

```
[root@FDMdevBI opt]# sort -t ' ' -k1.3 testsort.txt
```

```
guge 50 3000
```

```
sohu 100 4500
```

```
baidu 100 5000
```

```
google 110 5000
```

- 3. 按照公司人数排序，人数相同的只保留一个

```
[root@FDMdevBI opt]# cat testsort.txt
```

```
google 110 5000
```

```
baidu 100 5000
```

```
guge 50 3000
```

```
sohu 100 4500
```

```
[root@FDMdevBI opt]# sort -n -k2 testsort.txt
```

```
guge 50 3000
```

```
baidu 100 5000
```

```
sohu 100 4500
```

```
google 110 5000
```

```
[root@FDMdevBI opt]# sort -n -k2 -u testsort.txt
```

```
guge 50 3000
```

```
baidu 100 5000
```

```
google 110 5000
```

4. 让 testsort.txt 按照公司人数排序，人数相同的按照员工平均工资降序排序

```
[root@FDMdevBI opt]# sort -n -t ' ' -k3r -k2 testsort.txt  
baidu 100 5000  
google 110 5000  
sohu 100 4500  
guge 50 3000
```


uniq 命令

- 功能：删除相邻的重复行，只保留一行
- 格式： **uniq [选项] filename**
- c 在输出行前面加上每行在输入文件中出现的次数。
- d 仅显示重复行。
- u 仅显示不重复的行。

```
linux:$cat fruit
apples
apples
peaches
pears
bananas
cherries
cherries
```

```
linux:$uniq fruit
apples
peaches
pears
bananas
cherries
```

```
linux:$uniq -c fruit
      2 apples
      1 peaches
      1 pears
      1 bananas
      2 cherries
```

WC 命令

- 功能：统计给定文件中的行数、字节数和字数。
- 字是指由空格字符区分开的最大字符串。
- 格式： **wc [option] filename**
 - **-c**：统计字节数，使用的是 **UTF-8** 编码，每个汉字占 3 个字节，行尾回车占一个字节。
 - **-l**：统计行数
 - **-w**：统计字数
- 默认都统计，输出顺序总是行数，字数，字节数
- **wc -cl test** 与 **wc -lc test** 是一样的。
- 例 用 **cat** 命令建立一个文件，然后后 **wc** 命令统计文件信息。

comm 命令

- 功能：对两个已经排好需的文件进行比较，若没有排序好，则出错。
- 格式： `comm [option] file1 file2`
 - `option` : 1 , 2 , 3 控制相应的列是否显示
 - `-12` : 表示 1 , 2 列不显示，默认都显示
- 输出 3 列：
 - 第 1 列：仅在 `file1` 中出现的行
 - 第 2 列：仅在 `file2` 中出现的行
 - 第 3 列：两个文件都存在的行
- 演示： a 文件 `{1\n2\n3\n4\n5}` , b 文件 `{1\n3\n4\n5\n6}`
`$comm a b`

diff 命令

- 功能：逐行比较两个文本文件，列出其不同之处，不要求实现对文件进行排序。
- 格式： `diff [option] file1 file2`
- Option
 - `-b`：忽略行尾的空格，且字符串中的一个或多个空格都视为相等。
 - `-c`：采用上下文输出格式，提供三行上下文。
 - `-C n`：采用上下文输出格式，提供 `n` 行上下文。
 - `-e`：产生一个合法的 `ed` 脚本作为输出。

diff 命令

- 输出格式：如何将 **file1** 转变成 **file2**
 - **n1 a n3,n4** : a 表示增加
 - **n1,n2 d n3** : d 表示删除
 - **n1,n2 c n3,n4** : c 表示修改
 - **n1,n2**
 - **-12** : 表示 1 , 2 列不显示, 默认都显示
- **n1,n2** 是针对 **file1** 的, **n3,n4** 是针对 **file2** 的
- 每一行后面跟随受到影响的若干行, 以 "**<**" 开头的行属于 **file1**, "**>**" 开头的行属于 **file2** 。
- 演示实例:


diff 演示实例

```
file1.txt
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
void main()
{
    int count=0;
    printf("hello, world!\n");
}
```

```
file2
#include <stdio.h>
#include <string.h>
void main()
{
    int count;
    char *s="hello, world";
    printf("hello, world!\n");
}
```

\$diff file1.txt file2.txt

```
3d2
< #include <stdlib.h>
6c5,6
< int count=0;
---
> int count;
> char *s="hello, world";
```



2.5 目录管理

路径名

- 绝对路径名与相对路径名
 - 路径分量分隔符用斜线 / ，而不是反斜线 \
 - 例如 `/usr/stud/liu test/data1/cfg`
- 当前工作目录
 - 当前工作目录是进程属性的一部分，每进程一个
 - 没有逻辑盘的概念
- 文件 . 与 ..
 - 在目录表中确实有两个文件
 - 这两个目录项由系统创建和删除
 - `ls -a`
- 主目录 (Home Directory)
 - 每个用户都有自己独立的主目录 `/home/ 用户名`
用 `env` 命令查环境变量 `HOME` 的值

- “/”：所有目录的起点，操作系统本身的驻留程序存放在以根目录开始的专用目录中。
- 用户主目录：用～表示
- 用户主目录是系统管理员增加用户时创建起来的（以后也可以改变），每个用户都有自己的主目录，不同用户的主目录一般互不相同。
- 用户刚登录到系统中时，其工作目录便是该用户主目录，通常与用户的登录名相同。用户可以通过一个～字符来引用自己的主目录。

打印 / 改变当前目录

- **pwd 命令**：打印当前工作目录
print working directory
- **cd 命令**：改变当前工作目录 (Change Directory)
 - `cd /usr/include`
 - `cd /` 斜线前必须有空格，进入根目录
 - `cd ..`
 - `cd` 命令无实参
 - 回到用户的主目录 (Home Directory)
- **注意**
 - `cd` 是 shell 的一个内部命令

创建目录

- 创建目录 `mkdir`
 - 例: `mkdir sun/work.d`
 - `mkdir` 除创建目录外, 系统自动建立文件 `.` 与 `..`
- 删除目录 `rmdir`
 - 例: `rmdir sun/work.d`
 - 要求被删除的目录除 `.` 与 `..` 外无其它文件或目录
 - 其他命令: `rm -r sun/work.d`

cp: 复制目录

- **cp** 命令选项 **-r**，递归地复制一个目录

cp -r dir1 dir2

- 若 *dir2* 不存在，则新建子目录，并将 *dir1* 下内容拷入
- 若 *dir2* 已存在，则将所有文件拷入目录 *dir2*
- 选项 **-v**，冗长 (**verbose**) 方式
 - 复制时列出所拷贝的文件名
- 选项 **-u**，增量拷贝 (**update**)，便于备份目录
 - 根据文件的时戳，不拷贝相同的或者过时的版本的文件，以提高速度
 - *dir1* 和 *dir2* 不慎颠倒位置，不会出现灾难性后果
 - DOS 中类似功能的命令 **XCOPY**，选项 **/D** 可以用来实现增量拷贝 (Date)

cp: 复制目录（举例）

- 复制目录

将目录 `work.d` 复制为 `bak.d`

```
cp -r work.d bak.d
```

- 增量拷贝

将 `work.d` 中的内容增量拷贝到备份目录 `bak.d` 中

```
cp -ruv work.d bak.d
```

- 命令 `touch`

将文件的最后一次修改时间设置为当前时间，但不修改文件内容。

例如： `touch *.ch`

find: 在目录中查找文件

- 功能

find 命令从指定的查找范围开始，递归地查找子目录，凡满足条件的文件或目录，执行规定的动作

- 格式：**find** [搜索路径] [选项] [-print -exec -ok ...]

- 举例

- **find** ver1.d ver2.d -name '*.c' -print
- 查找范围：当前目录的子目录 **ver1.d** 和 **ver2.d**
- 条件：与名字 ***.c** 匹配。注 ***.c** 应当用引号括起
- 动作：把查找到的文件的路径名打印出来

- 命令的特点

- 功能强，选项较多

递归式查找，提供了一种遍历目录树的手段，其它命令经常借用 **find** 的“递归式查找”特性

find 关于条件的选项 (1)

表 2.9 find 命令选项说明

选 项	说 明
-name filename	查找与指定文件名相匹配的文件,支持通配符“*”和“[]”的使用
-perm	按照文件权限来查找文件,支持完全指定和“-”符号、“+”符号部分符合
-prune	使用这一选项可以使 find 命令不在当前指定的目录中查找,如果同时使用-depth 选项,那么-prune 将被 find 命令忽略
-user	按照文件属主来查找文件
-group	按照文件所属的组来查找文件
-mtime -n +n	按照文件的更改时间来查找文件
-amin -n +n	按照文件的访问时间来查找文件
-cmin -n +n	按照文件状态的更改时间来查找文件,其中“-n”表示文件更改时间距现在 n 天以内,“+n”表示文件更改时间距现在 n 天以前
-nogroup	查找无有效所属组的文件,即该文件所属组在/etc/groups 中不存在
-nouser	查找无有效属主的文件,即该文件的属主在/etc/passwd 中不存在

find 关于条件的选项 (2)

➤ 复合条件

可以用 **!(非)** , **\(\)(与)** , **-o(或)** 等表示多条件的与或非。

续表

选 项	说 明
<code>-newer file1 ! -newer file2</code>	查找更改时间比文件 file1 新但比文件 file2 旧的文件
<code>-type</code>	查找指定类型的文件,如 b—块设备文件、d—目录、c—字符设备文件、p—管道文件、l—符号链接文件、f—普通文件
<code>-size n [c]</code>	查找文件长度为 n 块的文件,带有 c 时表示文件长度以字节计
<code>-depth</code>	在查找文件时,首先查找当前目录中的文件,然后再在其子目录中查找
<code>-fstype</code>	查找位于某一类型文件系统中的文件,这些文件系统类型通常可以在配置文件 /etc/fstab 中找到,该配置文件中包含了本系统中有关文件系统的信息
<code>-mount</code>	在查找文件时不跨越文件系统 mount 点
<code>-follow</code>	如果 find 命令遇到符号链接文件,就跟踪至链接所指向的文件
<code>-regex pattern</code>	对搜索结果的[<code>color= # DC143C</code>]整个路径[/ <code>color</code>],按正规表达式进行过滤,必须对全路径考虑。例如,结果是./test,则正规表达式应该用“./t.*”,而不能“t.*”
<code>-cpio</code>	对匹配的文件使用 cpio 命令

find 关于条件的选项 (3)

需要特别注意的选项

-amin n	查找系统中最后 n 分钟访问的文件
-atime n	查找系统中最后 $n * 24$ 小时访问的文件
-cmin n	查找系统中最后 n 分钟被修改文件状态的文件
-ctime n	查找系统中最后 $n * 24$ 小时被修改文件状态的文件
-mmin n	查找系统中最后 n 分钟被修改文件数据的文件
-mtime n	查找系统中最后 $n * 24$ 小时被修改文件数据的文件
-empty	查找系统中空白的文件,或空白的文件目录,或目录中没有子目录的目录
-false	查找系统中总是错误的文件
-fstype type	查找系统中存在于指定文件系统的文件,如 ext2
-gid n	查找系统中文件组 ID 为 n 的文件
-group gname	查找系统中文件属于 gname 文件组,并指定组和 ID 的文件

find 关于动作的选项

- **-print**

打印查找的文件的路径名

- **-exec**

- 对查找到的目标执行某一命令
- 在 **-exec** 及随后的分号之间的内容作为一条命令，**{}** 代表所查到的路径名，最后以 **\;** 结尾。

- **-ok**

- 与 **-exec** 类似，只是对查找到符合条件的目标执行一个命令前需要经过确认

find 使用举例 (1)

- `find . -type d -print`
 - 从当前目录开始查找，寻找所有目录，打印路径名
 - 按层次列出当前的目录结构
- `find / -name 'stud*' -type d -print`
 - 指定了两个条件：名字与 `stud*` 匹配，类型为目录
 - 两个条件逻辑“与”，必须同时符合这两个条件
- `find / -type f -mtime -10 -print`
 - 从根目录开始检索最近 10 天之内曾经修改过的普通磁盘文件

find 使用举例 (2)

- `find . -atime +30 -mtime +30 -print`
 - 从当前目录开始检索最近 30 天之内既没有读过，也没有写过，而且也没有被当作命令执行过的文件
 - 筛选出一个时间周期内不活跃的文件
- `find . ! -type d -links +2 -print`
 - 从当前目录开始检索 link 数大于 2 的非目录文件
 - 条件“非”用 !
 - 注意：! 号与 `-type` 之间必须保留一空格

find 使用举例 (3)

- 找出 yli 用户主目录下所有的 .txt 文件并删除
 - `find /home/yli -name "*.txt" -ok rm {} \;`
 - 上例中, **-ok** 和 **-exec** 行为一样, 不过它会给出提示, 是否执行相应的操作。
 - 查找当前目录下所有 .txt 文件并把他们拼接起来写入到 all.txt 文件中
 - `find . -type f -name "*.txt" -exec cat {} \;> all.txt`
- 将 30 天前的 .log 文件移动到 old 目录中
- `find . -type f -mtime +30 -name "*.log" -exec cp {} old \;`

find 使用举例 (4)

- `find / -size +100000c \(-name core -o -name '*.tmp' \) -print`
 - 寻找大于 100K 的名叫 `core` 或有 `.tmp` 后缀
 - 使用了两条件“或” (`-o`) 及组合 (圆括号)
 - 不要遗漏了所必需的引号, 反斜线和空格, 尤其是圆括号前和圆括号后。圆括号是 `shell` 的特殊字符
 - 其他写法

```
find / -size +100000c '(' -name core -o -name '*.tmp ')' -  
print
```

```
find / -size +100000c \( -name core -o -name '*.tmp' \) -  
print
```

find 使用举例 (5)

- `find / -name make -print -exec ls -l {} \;`
 - `-exec` 及随后的分号之间的内容作为一条命令执行
 - `shell` 中分号有特殊含义，前面加反斜线 `\`
 - `{}` 代表所查到的符合条件的路径名。注意，两花括号间无空格，之后的空格不可省略
 - 对所有满足条件的文件或目录，依次执行 `print` 和 `exec` 命令
- `-ok` 选项在执行指定的命令前等待用户确认
 - `find / -size +100000c \(-name core -o name '*.tmp' \) -ok rm {} \;`

2.6 文件的归档与压缩处理

tar: 文件归档 (1)

- 功能
 - 将多个文件或目录打包在一个文件里，便于传输和保持。
- 命令用法
 - **tar** [options] 包名 *file-list* （待归档文件或目录列表）
- 选项第一字母指定要执行的操作，是必需的
 - **-c**: 创建新的文件。如果用户想备份一个目录或是一些文件，就要选择这个选项。
 - **-f** : **-f**< 备份文件 > 或 **--file**=< 备份文件 > 指定备份文件名。
 - **-t**: 列出备份文件的内容，查看已经备份的文件。
 - **-x**: 从备份文件中还原文件。
 - **-v**: 显示指令执行过程。

tar: 文件归档 (2)

- **-z** : 用 **gzip** 来压缩 / 解压缩文件, 加上该选项后可以将文件进行压缩, 但还原时也要用该选项进行解压缩。
- **-r** : 添加文件到归档包文件的尾部
- **-C** : 切换到指定的目录 **dir**

tar 命令使用举例 (1)

- 打包，常用参数 `-cf` 或 `-cvf`

- `tar -cf example.tar *`

对当前目录下的所有文件进行打包，生成 `example.tar`

- `tar -cvf example.tar *`

打包同时，列出包里的文件

■ 查看包中内容，参数 `-tf`

- 查看包 `example.tar` 内容

`tar -tf example.tar`

tar 命令使用举例 (2)

- 还原 tar 包，常用参数 **-xf** 或 **-xvf**

- 还原 example.tar 包内容

```
tar -xf example.tar
```

```
tar -xvf example.tar
```

■ 向包中添加新文件，参数 **-rf** 或 **rvf**

- 将文件 file5 添加到包 example.tar 中

```
tar -rf example.tar file5
```

tar 命令使用举例 (3)

- 用 `gzip` 压缩打包文件, 参数 `-czf` 或 `czvf`
 - 对当前目录下的所有文件或目录进行 `gzip` 压缩, 生成 `tmp.tar.gz`
`tar -zcvf tmp.tar.gz *`
 - 查看压缩包 `tmp.tar.gz` 的内容
`tar -ztf tmp.tar.gz`
- 解压缩 `gzip` 压缩的包, 参数 `-xzf` 或 `xzvf`
 - 解压缩 `tmp.tar.gz` 包
`tar -xzvf tmp.tar.gz`
 - 解压缩 `tmp.tar.gz` 包到当前目录下的目录 `b`
`tar -zxvf tmp.tar.gz -C ./b`

tar 命令使用举例 (4)

- 目录打包

设 **work1** 是一个复杂的有多个层次
子目录

```
tar cvf work1.tar work1
```

从归档文件中恢复数据的命令:

```
tar xvf work1.tar
```

文件压缩和解压缩

- 命令 `compress` 和 `uncompress`
 - 采用 LZW 算法对文件压缩，是一种字典压缩算法
 - 压缩算法对文件中有规律的数据内容压缩效率很高
 - 普通文本文件可压掉 50-80%
 - 有许多空白字段的数据库文件甚至可压掉 90% 以上
 - 压缩完的文件名后缀是 `.Z`
- 举例
 - `compress ch5` 压缩，生成新文件 `ch5.Z`
 - `zcat ch5.Z` 读取压缩格式的文件
 - `uncompress ch.Z` 解压缩，还原文件 `ch5`
- Linux 的文件压缩 / 解压缩工具
 - `gzip/gunzip`

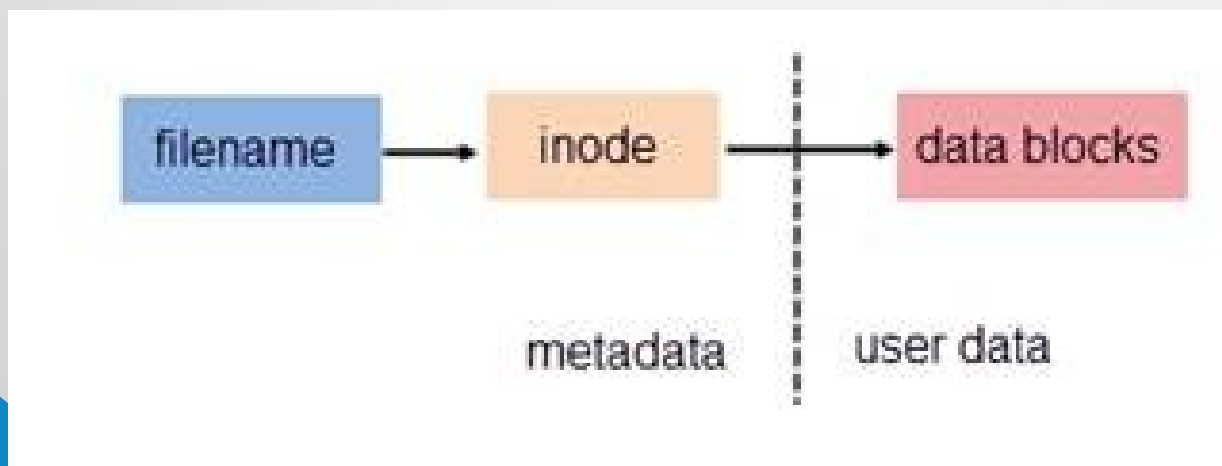
文件归档与压缩的应用

- 通用性
 - 文件归档 **tar** 与压缩处理 **compress** 在不同 UNIX 系统间有通用性。可以用来在不同 UNIX 中交换程序或数据
- 举例：在两台主机之间拷贝一棵目录树
 - 在主机 A
 - **tar cvf abc.tar abc** 将目录 **abc** 存到文件 **abc.tar**
 - **compress abc.tar** 压缩生成文件 **abc.tar.Z**
(惯例：后缀为 **.tar.Z** 的文件)
 - 将文件 **abc.tar.Z** 通过网络传到主机 B
 - 主机 B
 - **uncompress abc.tar.Z**
 - **tar xvf abc.tar**



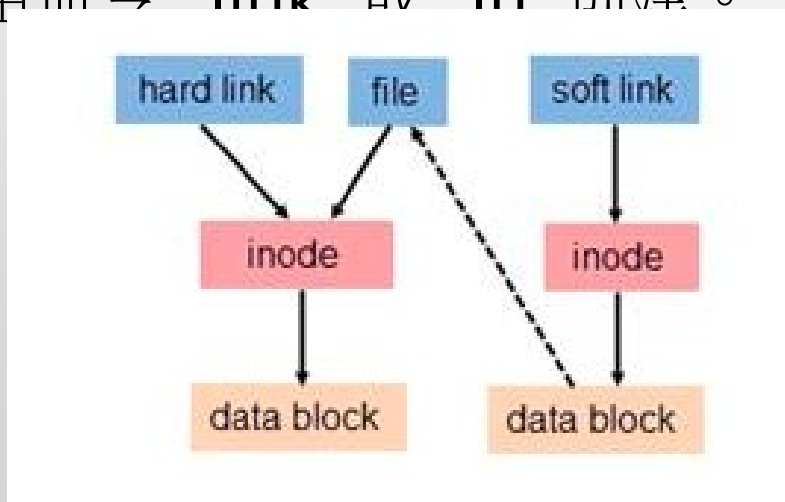
2.7 硬链接和软链接

文件都有文件名与数据，这在 **Linux** 上被分成两个部分：用户数据（**user data**）与元数据（**metadata**）。用户数据，即文件数据块（**data block**），数据块是记录文件真实内容的地方；而元数据则是文件的附加属性，如文件大小、创建时间、所有者等信息。在 **Linux** 中，元数据中的 **inode** 号（**inode** 是文件元数据的一部分但其并不包含文件名，**inode** 号即索引节点号）才是文件的唯一标识而非文件名。文件名仅是为了方便人们的记忆和使用，系统或程序通过 **inode** 号寻找正确的文件数据块。



硬链接

- 为解决文件的共享使用，Linux 系统引入了两种链接：硬链接（hard link）与软链接（又称符号链接，即 soft link 或 symbolic link）。链接为 Linux 系统解决了文件的共享使用，还带来了隐藏文件路径、增加权限安全及节省存储等好处。
- 若一个 inode 号对应多个文件名，则称这些文件为硬链接。换言之，硬链接就是同一个文件使用了多个别名（见图 hard link 就是 file 的一个别名，他们有共同的 inode）。
- 硬链接可由命令 `link` 或 `ln` 创建。



硬链接

由于硬链接是有着相同 **inode** 号仅文件名不同的文件，因此硬链接存在以下几点特性：

- 文件有相同的 **inode** 及 **data block** ；
- 只能对已存在的文件进行创建；
- 不能交叉文件系统进行硬链接的创建；
- 不能对目录进行创建，只可对文件创建；
- 删除一个硬链接文件并不影响其他有相同 **inode** 号的文件。

ln: 普通文件的硬连接

格式: **ln [option] file link**

无选项情况下, 建立硬连接。

```
$ ln chapt0 intro
```

```
$ ls -l chapt0 intro
```

```
-rw-rw-rw- 2  kc kermi 17935 Dec 12 18:07 chapt0
```

```
-rw-rw-rw- 2  kc kermi 17935 Dec 12 18:07 intro
```

(前面的几项必相同)

```
$ ls -i chapt0 intro
```

```
13210  chapt0
```

```
13210  intro
```

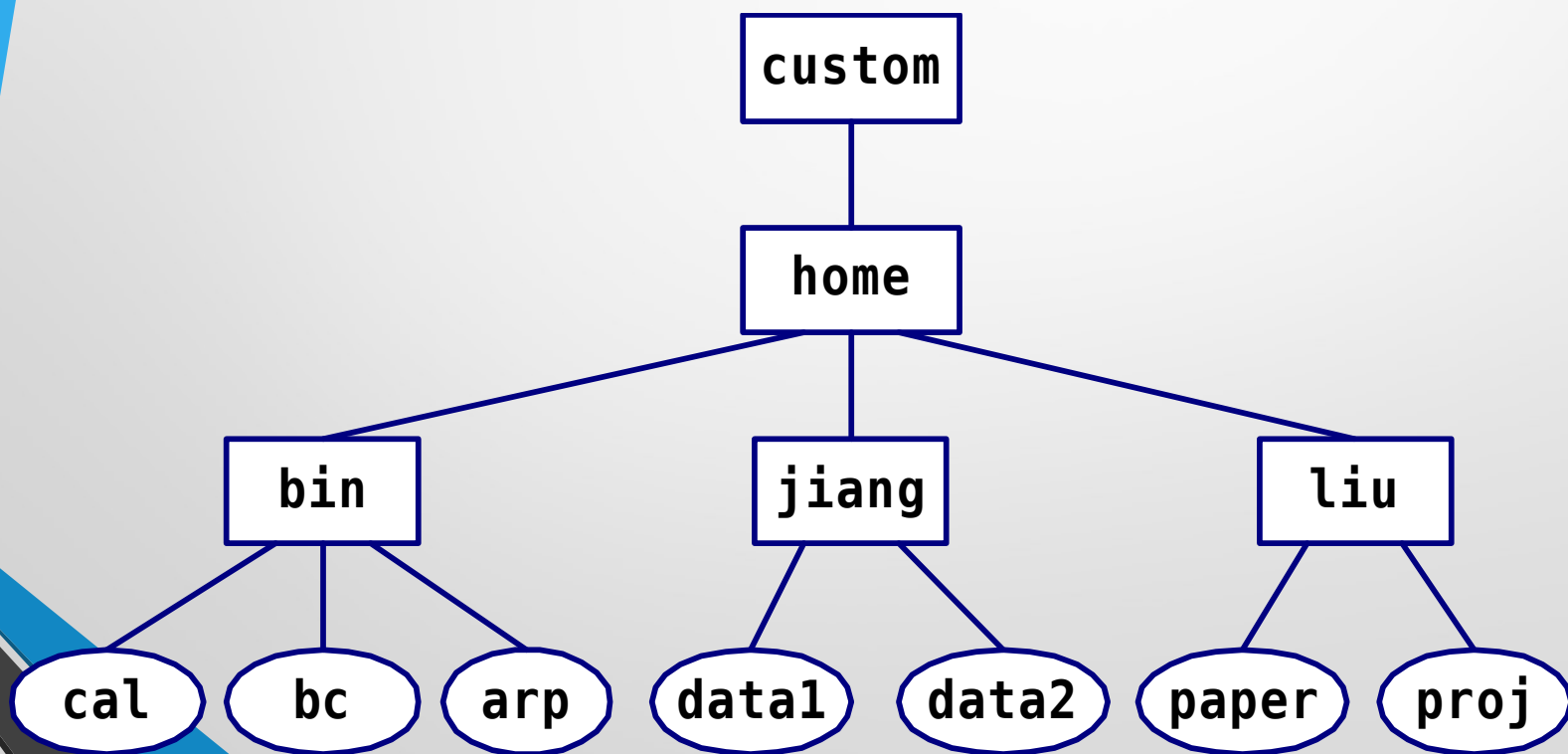
- chapt0 与 intro 同时存在时, 地位完全平等
 - 删 chapt0 文件, 则 intro 仍存在但 link 数减 1
- 硬连接, 只限于同一文件系统中的普通文件

硬链接举例

```
linux:$cat a
aaa
aaa
linux:$cat d
ddd
ddd
linux:$ln d e
linux:$cat e
ddd
ddd
linux:$cp a e
linux:$cat d
aaa
aaa
```

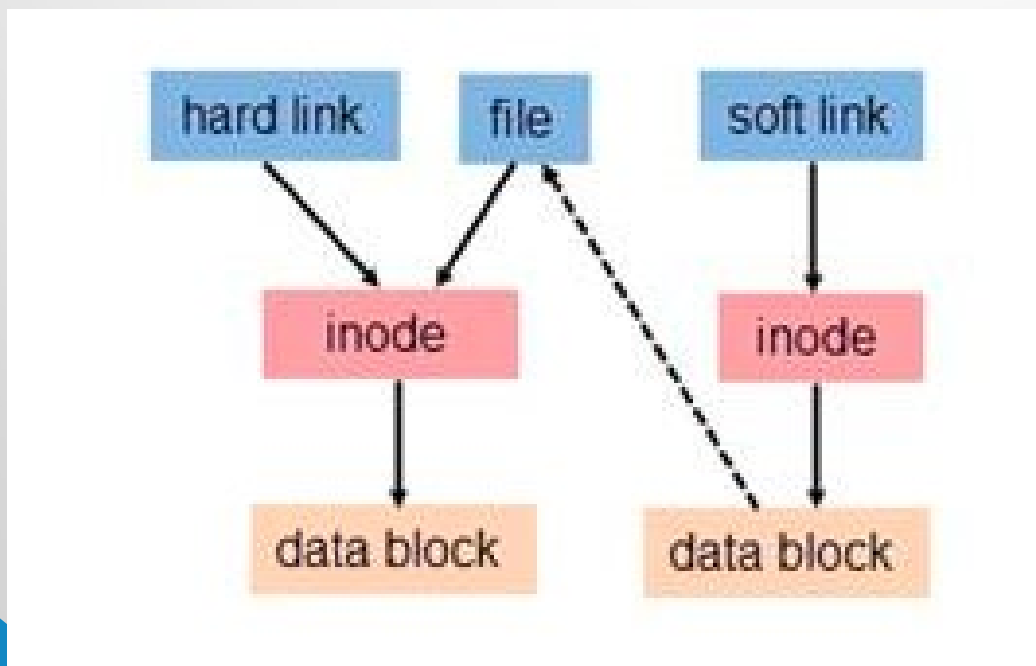
目录表的硬连接

- 不允许对目录用 `ln` 命令建立硬连接
- 一般来说，目录的 `link` 数 = 直属子目录数 + 2



软链接

- 软链接与硬链接不同，若文件用户数据块中存放的内容是另一文件的路径名的指向，则该文件就是软连接。
- 软链接就是一个普通文件，只是数据块内容有点特殊。软链接有着自己的 **inode** 号以及用户数据块（见图）。



软链接

- 软链接有自己的文件属性及权限等；
- 可对不存在的文件或目录创建软链接；
- 软链接可交叉文件系统；
- 软链接可对文件或目录创建；
- 创建软链接时，链接计数不会增加；
- 删除软链接并不影响被指向的文件，但若被指向的原文件被删除，则相关软连接被称为死链接（即 **dangling link**，若被指向路径文件被重新创建，死链接可恢复为正常的软链接）。

符号链接

- 符号链接也叫软链接
- 用特殊文件“符号链接文件”来实现

- 文件中仅包括了一个路径名
- 命令 `ln -s file link`

```
ln -s users_on sym.link
```

```
ls -l sym.link
```

```
lrwxrwxrwx 1 guest other 8 Jul 26 16:57 sym.link->users_on
```

- 类型为 1，大小为 8 字节，文件中只存放 `users_on` 字符串
- 文件的最后一次写时间以后不再变化
- 一旦建立了符号连接，删除操作删除的是符号连接文件，其它所有操作都将访问符号连接所引用的文件

符号链接的规则

系统处理路径名分量时发现符号链接，就把符号链接内容加到路径名的剩余部分的后面产生结果路径名

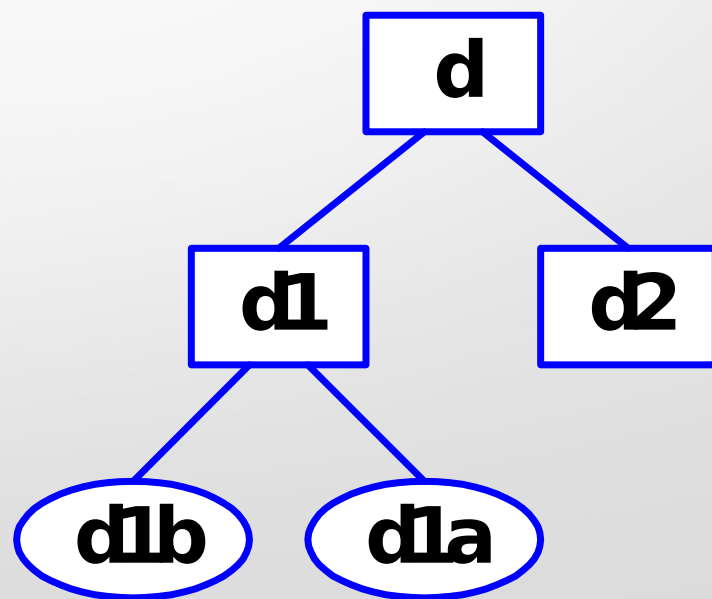
- 若符号链接包含绝对路径名，使用绝对路径名
- 否则，根据在文件层次中该链接的相对位置，计算符号链接内容（不是根据调用进程的当前的工作目录）

设当前目录为 **d**

`ln -s d1/d1b d1/dx`

在 **d1** 下新建文件 **dx**

访问 **d1/dx** 实际访问 **d1/d1b**





2.8 文件和目录的权限

文件的权限

用于控制进程对系统中文件和目录的访问

- 权限的三个级别
 - 文件主，同组用户，其他用户
 - 每个文件有唯一的属主
- 普通文件的权限
 - 读、写、可执行

chmod 命令

- 功能
 - 改变文件或目录的访问权限
- 两种方法
 - 文字设定法，数字设定法

修改权限

- 字母形式

`chmod [ugoa][+ -=][rwx] 文件名表`

`u - - user` 文件主的权限

`g - - group` 同组用户的权限

`o - - other` 其他用户权限

`a - - all` 所有上述三级权限

r(Read, 读取)：对文件而言，具有读取文件内容的权限；对目录来说，具有**浏览**目录的权限。

w(Write, 写入)：对文件而言，具有新增，修改，删除文件内容的权限；对目录来说，具有新建，删除，修改，移动目录内文件的权限。

x(eXecute, 执行)：对文件而言，具有执行文件的权限；对目录来说该用户具有进入目录的权限。

+ 表示增加权限、**-** 表示取消权限、**=** 表示唯一设定权限。

修改权限

例：

```
chmod u+rw *  
chmod go-rwx *.[ch]  
chmod a+x batch  
chmod u=rx try2
```

修改权限（续）

- 数字形式（八进制数字）
- `chmod abc file`
其中 `a,b,c` 各为一个数字，分别表示 User、Group、及 Other 的权限。

例： `chmod 674 xyz1 xyz2`

八进制：	6	7	4
------	---	---	---

二进制：	110	111	100
------	-----	-----	-----

权限：	rw-	rwX	r--
-----	-----	-----	-----

注：只允许文件主和超级用户修改文件权限

\$ who am i

jiang pts/2 Jun 06 08:34

\$ who > mydata

\$ ls -l mydata

-rw-r--r-- 1 jiang usr 58 Jun 06 09:04 mydata

\$ chmod u-w mydata

\$ who >> mydata (只读文件不许写)

mydata: The file access permissions do not allow
the specified action.

\$ rm mydata (只读文件可以被删除)

rm: Remove mydata? y

\$ ls -l mydata

ls: 0653-341 The file mydata does not exist.

演示：文件的读权限

```
$ who > mydata
```

```
$ chmod u-rw mydata
```

```
$ cat mydata ( 无法读取不允许读的文件中内容 )
```

```
cat: 0652-050 Cannot open mydata.
```

\$ chmod u-w . (当前目录不许写)

\$ who > mydata2 (不能创建新文件)

mydata2: The file access permissions do not allow the specified action.

\$ who >> mydata (但是可以修改已有的文件 , 追加一部分数据)

\$ rm mydata (不能删除文件)

rm: 0653-609 Cannot remove mydata.

The file access permissions do not allow the specified action.

\$ cp /etc/passwd mydata (可以覆盖旧文件)

\$ cp /etc/passwd mydata2 (不能创建新文件))

cp: mydata2: The file access permissions do not allow the specified action.

演示： 目录的 w 权限

\$ mv mydata MyData （文件不许改名）

mv: 0653-401 Cannot rename mydata to MyData:
The file access permissions do not allow the
specified action.

\$ mkdir Test （不可创建子目录）

mkdir: 0653-357 Cannot access directory ..
.: The file access permissions do not allow the
specified action.

\$ pwd

/usr/jiang

\$ chmod u-r .

\$ ls (不可读的目录无法列表出其中文件)

ls: .: The file access permissions do not allow the specified action.

\$ chmod 000 . (取消当前目录所有权限)

\$ ls

ls: 0653-345 .: Permission denied.

\$ chmod 755 . (试图恢复当前目录权限失败，因为试图访问当前目录下的 . 文件)

chmod: .: The file access permissions do not allow the specified action.

\$ chmod 755 /usr/jiang (这种访问不需要当前目录权限，可恢复当前目录权限)

- 子目录没有读写权限，但是保留了 x 权限

```
$ chmod u=x ttt
```

```
$ cat ttt/ccp.c // 可以执行
```

```
main(int argc, char **argv)
```

```
{ ...
```

```
}
```

```
$ rm ttt/arg.c ( 子目录没有写权限，不能删除其中的文件 )
```

```
rm: 0653-609 Cannot remove ttt/arg.c.
```

```
The file access permissions do not allow the  
specified action.
```

```
$ ls ttt ( 子目录没有读权限，不能列出其中的文件 )
```

```
ls: ttt: The file access permissions do not allow  
the specified action.
```


- 子目录有读写权限，但没有 x 权限

```
$ chmod u=rw ttt
```

```
$ ls ttt
```

```
BUGS.report  arg.c      ccp.c      chap.h  
mydata
```

```
arg      auth.c    chap.c    disk.img
```

```
$ cat ttt/arg.c
```

```
cat: 0652-050 Cannot open ttt/arg.c.
```

（试图设置其他用户的文件或目录的权限）

```
$ chmod 777 /
```

```
chmod: /: Operation not permitted.
```

umask 命令 (1)

- 功能：决定文件 / 目录的初始权限，而 **umask** 值则表明了需要从默认权限中去掉哪些权限来成为最终的默认权限值。
 - 用 **vi** 新建文件
 - 用输出重定向创建文件
 - 创建新目录
- **umask** 是进程属性的一部分
 - **umask** 是 shell 内部命令
 - **umask** 是进程属性的一部分
- 命令
 - **umask** 打印当前的 **umask** 值
 - **umask 022** 将 **umask** 值设置为八进制的 022

umask 命令 (2)

- 掩码值的含义

- 普通文件: `rw` 权限受影响, 初始权限中无 `x` 权限
- 目录文件: `rw``x` 权限均受掩码值影响
- 例: 掩码值: `022`

二进制: `000010010`

普通文件: `rw-r--r--`

目录: `rwxr-xr-x`

若用户创建一个文件, 则文件的默认访问权限为 `rw-rw-rw-`, 创建目录的默认权限 `rwxrwxrwx`

- 实例

- `$ umask 077`
- `$ vi c`
- `$ ls -l c`
`-rw- --- ---`
- `$ mkdir d`
- `$ ls -l d`
`drwx --- ---`



SUID 权限和 SGID 权限

三级权限存在的问题

- 问题

- 系统中任一个用户，要么对文件的全部内容具有访问权，要么不可访问文件。有的情况下，很不方便

- 举例

- 用户 liu 的文件 list.txt：希望用户 liang，只能读取行首为 # 的行和与用户 liang 有关的行

```
#=====
# 登录名 工作证号 姓名 月份 工资 奖金 补助 扣除 总额
#-----
tian 2076 田晓星 03 1782 1500 200 175 3307
liang 2074 梁振宇 03 1560 1400 180 90 3050
sun 3087 孙东旭 03 1804 1218 106 213 2915
tian 2076 田晓星 04 1832 1450 230 245 3267
liang 2074 梁振宇 04 1660 1450 230 70 3270
sun 3087 孙东旭 04 1700 1310 283 270 3023
#=====
# 注：煤气费不再从工资中扣除，由煤气公司自行收缴。
```

```
1 #include <stdio.h>
2 #include <string.h>
3 void main(void)
4 {
5     FILE *f;
6     char line[512], login_name[16];
7     f = fopen("list.txt", "r");
8     if (f == NULL) {
9         perror("*** ERROR: Open file \"list.txt\" ");
10        exit(1);
11    }
12    while (fgets(line, sizeof(line), f)) {
13        if (line[0] == '#') {
14            printf("%s", line);
15        } else {
16            if (sscanf(line, "%s", login_name) > 0) {
17                if (strcmp(login_name, getlogin()) == 0)
18                    printf("%s", line);
19            }
20        }
21    }
22    exit(0);
23 }
```

简单的三级权限的问题

- 用户 liu

源程序经过编译后，生成可执行文件 query 。为了保密用户 liu 将文件 list.txt 的权限设为 rw-----，文件 query 的权限为 rwx--x--x

```
$ chmod 600 list.txt
```

```
$ chmod 711 query
```

```
$ ls -l list.txt query
```

```
-rw----- 1 liu leader 722 Dec 10 23:04 list.txt
```

```
-rwx--x--x 1 liu leader 56134 Dec 10 23:07 query
```

```
$
```

- 用户 liang 执行命令 query

```
$ query
```

```
*** ERROR: Open file "list.txt" : Permission denied
```

```
$ cat list.txt
```

```
cannot open list.txt: Permission denied
```

- 文件 `query` 的文件主 `liu` 给文件 `query` 增加 `SUID` 权限

```
$ chmod u+s query
```

```
$ ls -l query
```

```
-rws--x--x  1 liu  leader  56134 Dec 10 23:07 query
```

- 用户 `liang` 通过 `query` 命令查询只许 `liu` 可读的文件 `list.txt`

```
$ ls -l list.txt query
```

```
-rw-----  1 liu  leader    722 Dec 10 23:04 list.txt
```

```
-rws--x--x  1 liu  leader  56134 Dec 10 23:07 query
```

```
$ query
```

```
#=====
```

```
# 登录名 工作证号 姓名 月份 工资 奖金 补助 扣除 总额
```

```
#-----
```

```
liang 2074 梁振宇 03 1560 1400 180 90 3050
```

```
liang 2074 梁振宇 04 1660 1450 230 70 3270
```

```
#=====
```

```
# 注：煤气费不再从工资中扣除，由煤气公司自行收缴
```

```
$ cat list.txt
```

```
cannot open list.txt: Permission denied
```


参考信息

- <https://www.cnblogs.com/fhefh/archive/2011/09/20/2182155.html>

SGID 权限

- **SGID** (设置组 **ID**) 权限和 **SUID** 权限类似，用于设置程序运行时所属组。
- 增加 **SGID** 权限时，文件必须事先有组执行权限
`chmod g+s file-list`
- 删除 **SGID** 权限
`chmod g-s file-list`
- **SGID** 权限的文件在 `ls -l` 列表时，组执行权限处显示小写字母 `s`，而不是 `x`

chown 命令

- 功能：更改某个文件或目录的属主和属组
- 说明：**root** 用户把自己的一个文件复制给用户 **li**，为了让用户 **li** 能够存取这个文件，**root** 用户可以把这个文件的属主设为 **li**。
- 格式：**chown [option] [user][:group] filename**
- option:
 - ◆ **-R**: 递归地改变指定目录下及其所有子目录和文件的拥有者；
 - ◆ **-v**: 显示 **chown** 命令所做的工作

chown 实例

- `$ sudo passwd root` // 输入 root 密码建立 root 用户
- `$ su root` // 进入 root 用户
- `# cat > a.txt` // 输入文件内容
- `# ls -l a.txt` // `-rw-r--r-- 1 root root ...lyj` 用户没有修改权限
- `# chown lyj a`
- `# ls -l a.txt` // `-rw-r--r-- 1 lyj root...`
- `# su lyj`
- `$ vi a.txt` // 此时 lyj 具有修改权限了
- `$ chown :lyj a.txt` // 将 a 的属组也改为了 lyj

本章小结

- cat ls head tail wc
- cp mv rm
- sort diff comm
- pwd cd
- mkdir rmdir
- touch find
- tar ln
- chmod
- SUID SGID

习题

- 1、用户显示当前目录中内容的命令是（ ）
- 2、同 **MSDOS** 下的 **copy** 命令一样，**Linux** 系统中的（ ）命令是将给出的文件或目录复制到另一文件或目录中，功能十分强大。
- 3、（ ）命令用于创建指定的目录名，要求创建目录的用户在当前目录中具有写权限。
- 4、能删除一目录中的一个或多个文件或目录，也可以将整个目录及其下的所有文件及目录均删除的命令是（ ）

A. mv B. rm C. mkdir D. rmdir

- 5、可以显示目录下的所有文件，包括以“.”开头的隐含文件的命令是（ ）

A. ls -a B. ls -b C. ls -c D. ls -d

- 6、（ ）命令可以用来压缩或解压缩扩展名维 .z 格式的文件。

A. zip B. gunzip C. tar D. compress

- 7、**Linux** 下主要有哪些不同类型的文件？

- 8、用什么命令可以将两个文件合并成一文件？

- 9、试说明以下两个文件的拥有者与其相关的权限是什么？

(1) -rw-r--r-- 1 root root 238 Jun 18 17:22 test1.txt

(2) -rwxr-xr-- 1 user1 usergroup 5238 Jun 19 10:25 test2.txt