文件系统初步

毛迪林

dlmao@fudan.edu.cn

大纲

- 文件系统：file、du和df
- 文件系统：Linux系统目录
- 通配符和扩展通配符扩展
- 重定向：cat/split和tee命令

查看文件类型: file

file [OPTION...] [FILE...]

查看各种文件的类型，注意file并不是根据文件扩展名来判断类型

- 文件系统内部保存的文件类型属性
- 文件内容里面包含的magic字符
- 文本文件内的字符所在的字符集等

~ $ file /etc/passwd /usr/bin/id /bin /dev/sda2 /dev/tty /usr/bin/viewer

/etc/passwd: ASCII text

/usr/bin/id: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 2.6.32, BuildID[sha1]=ca513ae4d630324b1eadcd78122490248a27b8b6, stripped

/bin: directory

/dev/sda2: block special (8/2)

/dev/tty: character special (5/0)

/usr/bin/viewer: a /usr/bin/python3 script, ASCII text executable

查看文件使用的空间：du

- **查看目录和文件使用的空间： du [OPTION]... [FILE]...**

没有FILE时，表示当前目录。对于目录，缺省是递归显示子目录使用的空间，如果为文件，则显示文件占用的空间

选项	含义	选项	含义
-c	最后显示一个总数	-B SIZE	显示的空间单位，缺省1K
-s	仅仅显示每个参数的总使用情况	-h	易读方式(1k 234M 2G)，注意采用二进制描述
-d N	递归显示目录时仅仅N级，N=0表示不递归	--si	与-h类似，只是以十进制而不是二进制来描述

```
dlmao@mars:~$ du /boot
172     /boot/grub/locale
2348    /boot/grub/fonts
2452    /boot/grub/i386-pc
7340    /boot/grub
115888  /boot
dlmao@mars:~$ cd /usr
dlmao@mars:/usr$
```

```
dlmao@mars:/usr$ du -s include games
21380   include
684     games
dlmao@mars:/usr$ du -sch include games
21M     include
684K    games
22M     total
```

查看磁盘使用情况： df

- 查看文件系统使用情况： df [OPTION]... [FILE]...

-T 显示文件系统类型 -t TYPE 仅仅显示相应类型(ext4/ext3…)的文件系统信息

-h/H 以易读方式（1k=1024）显示， -H表示1k=1000

-i 查看inode节点信息

FILE如果为绝对路径名且为一个设备文件，则显示该设备文件对应的文件系统的信息，否则显示该文件所在文件系统的信息

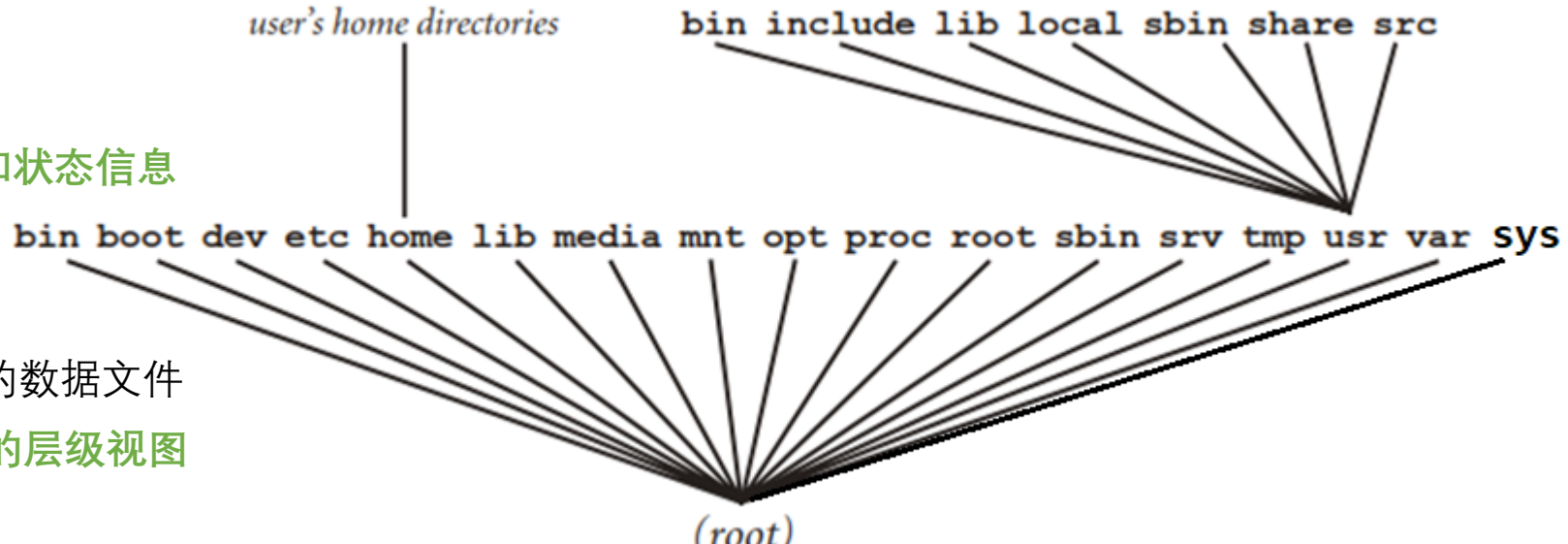
```
dlmao@mars:~$ df
Filesystem     1K-blocks    Used Available Use% Mounted on
udev             1991004       0   1991004   0% /dev
tmpfs             403072    1244    401828   1% /run
/dev/sda1       20509264 6486188  12958220  34% /
tmpfs            2015340       0   2015340   0% /dev/shm
/dev/loop0          3840    3840         0 100% /snap/gnome-system-monitor/127
...
dlmao@mars:~$ df -T -H /home
Filesystem     Type  Size  Used Avail Use% Mounted on
/dev/sda1      ext4   22G  6.7G   14G  34% /
```

Linux系统目录：Filesystem Hierarchy Standard

- /： 整个文件系统树的根，包含了许多子目录，还可能包含多个内核文件(vmlinuz)等（符号连接到boot目录下的文件）
- /bin: 基本执行程序，系统管理员单用户模式需要用到的程序，用户也可执行里面的大部分程序
- /boot: 系统启动需要用到的程序，包括内核以及启动配置程序grub等
- **/dev: 设备文件, 一般由udev管理**
- **/etc 配置文件**
- /home 用户主目录
- /lib 为/bin /sbin服务的基本的共享库和内核模块
- /lost+found: fsck恢复的文件
- /media 可移动存储挂载点
- /mnt 固定存储挂载点
- /opt 第三方应用
- **/proc : proc文件系统, 进程以及内核设置和状态信息**
- /root： 超级用户主目录
- **/sbin 系统管理员使用的程序**
- /srv : 本地提供的服务(比如Web服务)用到的数据文件
- **/sys 内核文件系统，提供内核和系统硬件的层级视图**

红颜色表示有时使用独立的文件系统

- /tmp 临时文件目录，每个用户可以创建，但只有创建者可以删除
- /usr 保存静态信息（程序、文档、库等）的辅助目录，大部分用户实用工具和应用
- /var 保存动态可变信息（日志、邮件、打印文件等）的辅助目录
- **/run 运行时可变的数据，也可通过/var/run访问**



Linux系统目录:

通用程序:	系统管理程序:	第三方程序:
/bin	/sbin	/opt/xxx
/usr/bin	/usr/sbin	/var/opt/xxx
/usr/local/bin	/usr/local/sbin	

为什么将执行程序放在/bin和/usr/bin?

- 历史的原因，早期UNIX 设计者有两个存储设备，一个小一些但是访问速度快，另外一个大一些，但是访问速度慢
- 小的存储保证系统能够启动执行，但空间不够，只好将一部分放到另外一个文件系统

目录	内容
/usr/bin	非基本的执行程序，给用户使用
/usr/include	C语言头文件
/usr/lib	非基本执行程序用到的共享库文件
/usr/sbin	非基本的系统管理员用到的程序
/usr/share	共享系统数据， /usr/share/{doc,man,info}等
/usr/src	源代码
/usr/local	系统管理员存放本地用户可能需要的应用

大纲

- 文件系统：file、du和df
- 文件系统：Linux系统目录
- **通配符和扩展通配符扩展**
- 重定向：cat/split和tee命令

文件通配符(wildcard)扩展

locale命令可查看本地化策略

命令行中，如果一个文件名中出现了'?' '*'或'['通配字符时，进行通配符扩展，匹配**相应目录中存在的**且通配模式所匹配的文件名，展开后，那些匹配的路径名之间以空格隔开，如果**没有匹配，原样出现**

- ?　　匹配单个字符，比如 ls **/bin/???**
- *　　匹配0个或多个任意的字符，比如ls **/bin/d***
- [characters]　匹配任意一个属于字符集characters的字符，比如 ls **/bin/[abc]?***
  - 范围表达式：如果一对字符以连字符隔开，则表示位于两个字符之间(包括那两个字符)的所有字符，比如ls **/bin/[a-c]?***
    - 以前[a-z]为小写英文字母，即字符按照ASCII码(0..9…A…Z…a…z) 进行排序，LC_COLLATE=C
    - 现在的本地化策略中，比如LC_COLLATE=en_US.UTF-8: 采用字典排序方式,将大小写字母成对组合按照aAbBcC…zZ的顺序排序。[a-c]相当于[aAbBc]
  - 否定：[!characters]或[^characters]，即如果第一个字符为!或者^，匹配任一不在字符集characters中的字符，建议采用!的形式，比如 ls **/bin/[!a-fp-z]?**
  - 如果!不出现在第一个，-出现在第一个或最后一个，]出现在第一个，则表示字符本身。比如**[]!-]** 可以匹配]或!或-
  - [:class:] 匹配单个预定义字符类中的字符，常用字符类包括digit/upper/lower/alpha/alnum/xdigit等。

  比如ls -dF **/etc/[[:upper:]]***
- 可以采用转义字符\, 表示后面字符不是通配符，比如d\? 相当于d后面跟着?

文件通配符(wildcard)扩展

命令行中，如果一个文件名中出现了'?' '*'或'['通配字符时，进行通配符扩展，匹配**相应目录中存在的**且通配模式所匹配的文件名，展开后，那些**匹配的路径名**之间以空格隔开，如果**没有匹配，原样出现**

- 通配符**无法匹配/**
- 匹配不是针对完整的路径，而是**路径中的各个部分**，如/home/dl*/t*.txt
- 匹配为**完全匹配**，而不是部分匹配，比如a*不匹配 cat
- 通配符**无法匹配隐藏文件名中最前面的点**

```
dlmao@mars:~$ set -x
dlmao@mars:~$ ls /bin/d??*e
+ ls --color=auto /bin/date /bin/dnsdomainname /bin/domainname
/bin/date  /bin/dnsdomainname  /bin/domainname
dlmao@mars:~$ ls /bin/d??*k
+ ls --color=auto '/bin/d??*k'
ls: cannot access '/bin/d??*k': No such file or directory
dlmao@mars:/$ ls /usr/*bin/k*[st]
+ ls --color=auto /usr/bin/kerneloops-submit /usr/sbin/kerneloops
/usr/bin/kerneloops-submit  /usr/sbin/kerneloops
dlmao@mars:~$ ls -d *
# ... 无法看到隐藏文件
dlmao@mars:~$ ls -d .[!.] .??*  2>/dev/null
# .[!.] 2个字符的隐藏文件(不包括..)    .??*  3个以上字符的隐藏文件
```

文件通配符(wildcard)扩展

- 缺省支持通配符扩展：set –o noglob 关闭通配符扩展， set +o noglob开启
- 控制文件通配符扩展的其他shell选项：可以通过shopt -s xxx开启，shopt -u xxx关闭
  - nullglob: 如果无法找到匹配的文件，该参数扩展为空字符串
  - nocaseglob: 匹配时大小写无关
  - dotglob: 通配符可以匹配文件名中最前面的点(dot)字符
  - globstar：支持通配符**，但只有模式中的路径部分正好是**才起作用。其出现在模式的最后时，表示匹配目录和各级子目录中的所有文件，而**/表示匹配目录和子目录

```
dlmao@mars:~$ shopt | grep glob
dotglob         off
extglob         on
failglob        off
globasciiranges off
globstar        off
nocaseglob      off
nullglob        off
dlmao@mars:~$ shopt -s globstar
```

```
dlmao@mars:/$ ls /etc/**/g*.conf
/etc/dbus-1/system.d/gdm.conf  /etc/sane.d/gphoto2.conf
/etc/gai.conf                  /etc/sane.d/gt68xx.conf
/etc/geoclue/geoclue.conf      /etc/security/group.conf
/etc/sane.d/genesys.conf
```

扩展通配符(Extended glob)

- 通配符扩展只能描述匹配任意多个字符或单个字符，但是无法匹配某些字符模式出现多次以上，也无法匹配多个模式中任意一个出现等
- 扩展通配符：bash缺省打开了extglob选项
  - 打开扩展通配符选项：shopt –s extglob，关闭时执行 shopt -u extglob

扩展	含义
*(pattern-list)	pattern-list里面的任一一个pattern出现0次以上
+(pattern-list)	pattern-list里面的任一一个pattern出现1次以上
?(pattern-list)	pattern-list里面的任一一个pattern出现0次或者1次
@(pattern-list)	pattern-list里面的任一一个pattern出现1次
!(pattern-list)	pattern-list里面的所有pattern都不出现
pattern-list中的pattern为采用glob或extglob定义的模式，多个模式之间通过**|隔开**	

- 后缀为.jpg或jpeg的文件
  - *.jp?(e)g
  - *.@(jpg|jpeg)
- 后缀为多个数字的文件
  - *.+([0-9])
- 除.py .c .o外的其他文件
  - *!(.py|.c|.o)

大纲

- 文件系统：file、du和df
- 文件系统：Linux系统目录
- 通配符和扩展通配符扩展
- **重定向：cat/split和tee命令**

cat 查看文件

cat [OPTION] [FILE]...

- 如果没有参数，表示将自于标准输入的内容输出到标准输出(缺省为屏幕)。否则将参数给出的1或多个文件的内容输出到标准输出，如果某个参数为-，表示标准输入
- 标准输入缺省指来自于用户通过键盘输入的内容
  - 在从标准输入读时，采用文本方式读写，即逐行读取并输出
  - 直到用户通过组合键<Ctrl-D>(用^D表示)输入EOF时结束

```
dlmao@mars:~$ cat /etc/lsb-release
DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=18.04
DISTRIB_CODENAME=bionic
DISTRIB_DESCRIPTION="Ubuntu 18.04.4 LTS"
dlmao@mars:~$ cat
1. hello world
1. hello world
2. bye
2. bye
^D
```

- 多个文件合并(concatenate)

cat FILE1 FILE2 > output

将FILE1和FILE2的内容合并重定向到文件output中

```
dlmao@mars:~$ echo -e 'id\tname\tdesc\n\n\n1\tJohn\t99\n\n2\tTony\t95' > tmp.txt
```

```
dlmao@mars:~$ cat tmp.txt
id      name    desc


1       John    99

2       Tony    95
dlmao@mars:~$ cat -n tmp.txt
     1  id      name    desc
     2
     3
     4  1       John    99
     5
     6  2       Tony    95
dlmao@mars:~$ cat -ns tmp.txt
     1  id      name    desc
     2
     3  1       John    99
     4
     5  2       Tony    95
```

```
dlmao@mars:~$ cat -A tmp.txt
id^Iname^Idesc$
$
$
1^IJohn^I99$
$
2^ITony^I95$
```

-n, --number	最前面添加行号
-b, --number-nonblank	添加行号，但是对于空行不添加
-s, --squeeze-blank	连续空行仅仅保留1个空行
-v, --show-nonprinting	非打印字符以^X等形式显示
-A, --show-all	显示非打印字符、Tab(^I)和换行($)

split拆分文件

split [OPTION]... [FILE [PREFIX]]

将(文本或二进制)文件分拆成多个文件。对于文本文件，缺省每1000行分割，保存在xaa/xab/...如果有第2个参数，表示不使用缺省的前缀x，而是采用指定的PREFIX。-d选项表示采用数字形式的后缀，即00/01等

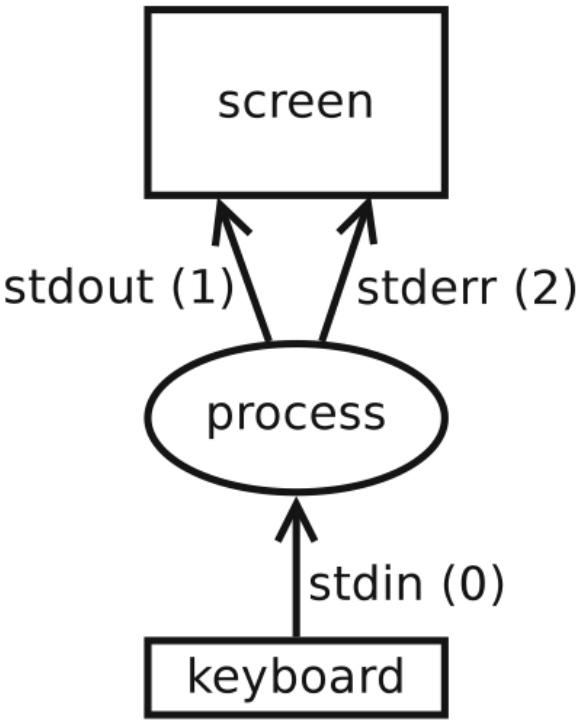
-l, --lines=N	拆分时每N行一个文件。缺省为1000行一个文件
-b, --bytes=SIZE	拆分时每SIZE大小一个文件，SIZE可包含单位（K,M,G等)
-d	拆分后文件名后缀为点加上2位数字，从0开始。缺省采用2位字符作为后缀(aa/ab…az/ba/…)
-a, --suffix-length=N	拆分的文件名后缀的位数为N，缺省为2

```
$ split -db 900k /bin/bash bash.
$ cat bash* > mybash
$ ls -l *bash*
-rw-rw-r-- 1 demo demo  921600 Nov 11 19:31 bash.00
-rw-rw-r-- 1 demo demo  115864 Nov 11 19:31 bash.01
-rw-rw-r-- 1 demo demo 1037464 Nov 11 19:31 mybash
```

- 将二进制文件以900k字节大小分割，保存在bash.00, bash.01…
- 使用cat命令将分割的文件合并

重定向：文件描述字

- 每个进程(执行的程序)会打开三个文件，每个打开的文件通过文件描述字(数字类型的ID）标识和访问
  - fd 0称为标准输入STDIN，缺省为键盘输入
  - fd 1称为标准输出STDOUT，缺省为屏幕输出
  - fd 2称为标准错误STDERR，缺省为屏幕输出
- 每个进程对于输入输出采用一种标准的方式，不用关心STDIN/STDOUT/STDERR具体是哪个文件，可是某个普通文件，也可是某个设备等




```
# 查看当前shell进程($$表示其进程ID)打开的文件描述字
$ ls -l /proc/$$/fd
总用量 0
lrwx------ 1 dlmao dlmao 64 Sep 25 16:36 0 -> /dev/pts/0
lrwx------ 1 dlmao dlmao 64 Sep 25 16:36 1 -> /dev/pts/0
lrwx------ 1 dlmao dlmao 64 Sep 25 16:36 2 -> /dev/pts/0
lrwx------ 1 dlmao dlmao 64 Sep 25 16:36 255 -> /dev/pts/0
```

重定向：文件描述字

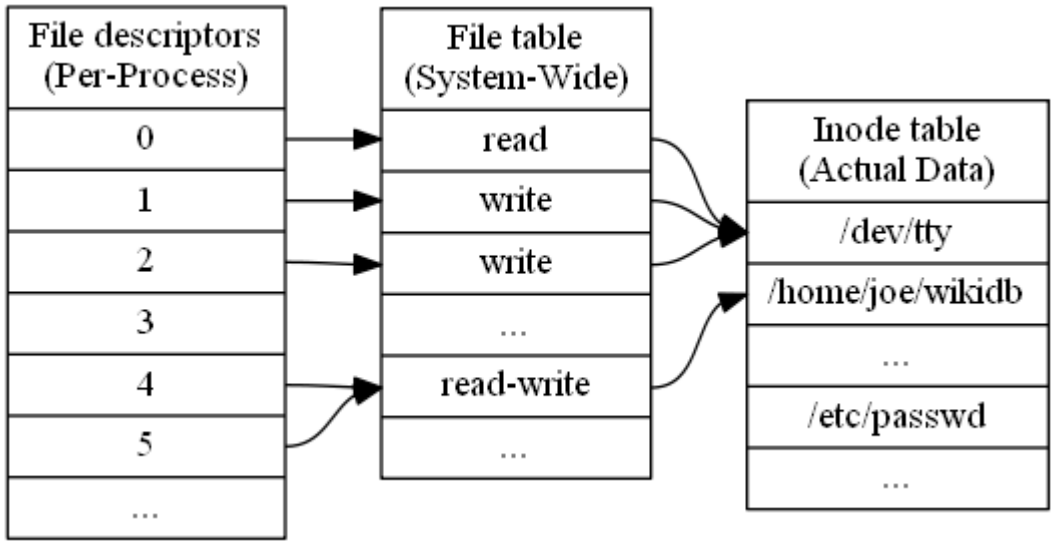
- 重定向：改变文件描述字与打开的文件之间的映射
  - 改变fd 0(从该文件读数据）称为输入重定向
  - 改变fd 1和fd 2（往该文件写数据) 称为输出和错误重定向
  - 除了fd 0/1/2外，也可对任何一个文件描述字进行重定向操作

从程序员的角度，输出重定向类似于：

```
fd = open(file,'w')
dup2(fd,1)  // 复制fd到文件描述字1
```

输入重定向类似于

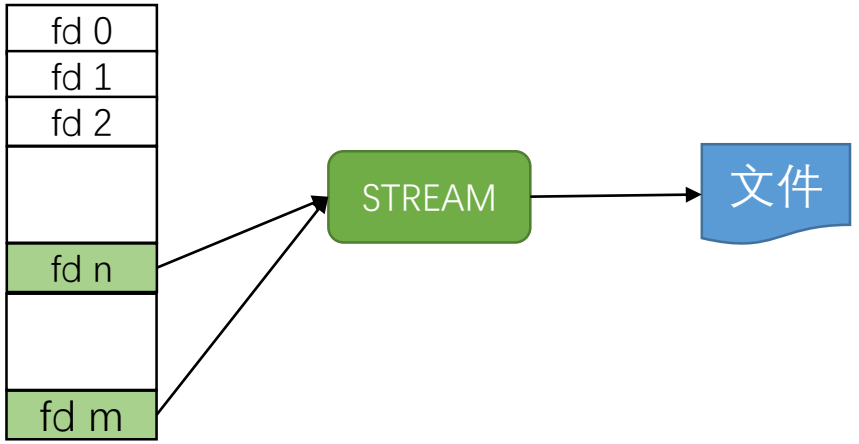
```
fd = open(file,'r')
dup2(fd,0)  // 复制fd到文件描述字0
```



重定向：文件描述字与打开的文件的映射

```
fd = open(file,'r')
dup2(fd,0)
```

如何改变文件描述字fd与实际文件的对应关系?



- 重定向元字符 < 和 >，除非特别说明，前后的空格可选
  - <表示输入重定向(打开文件读)
    - n< 表示重定向要改变的文件描述字为n
    - 注意**数字n与<之间不能有空格**
    - < 等价于 0<
  - > 表示输出重定向(打开文件写)
    - n> 表示重定向要改变的文件描述字为n
    - >等价于 1>

```
cmd <FILE1  >FILE2
cmd >FILE 2>&1
```

- 重定向到哪里？ 文件或 已经打开的文件描述字n
  - >FILE或<FILE 表示打开文件FILE，然后fd重定向(指向)该打开的文件
  - >&n或<&n 表示fd重定向当前文件描述字n对应的文件(该文件已经打开)
  - 注意元字符&也用于表示命令后台执行，因此这里**>&之间不能有空格**

输出重定向

输出重定向到文件时，比如 cmd >file

- 如果file不存在，首先创建文件file
- 如果file存在，则覆盖该文件
- **>> 附加到该文件**，文件不存在时与 >file等价
- 如何避免不小心覆盖原有文件？
  - 设置shell选项noclobber，覆盖时报错
    - set –o noclobber开启选项，set +o noclobber关闭选项
    - >| 强制覆盖，注意 >| 以及 >>的两个字符中间不能有空格

- bash首先进行重定向，然后执行相应的命令
- **>file** 可以创建一个新文件或删除文件的内容

```
$ rm -f hosts
$ set -o noclobber   # 重定向覆盖时报错
$ >hosts       # 创建一个空文件hosts
$ cat < /etc/hosts > hosts  # 从/etc/hosts读,输出到文件hosts, 覆盖时报错
-bash: hosts: cannot overwrite existing file
$ cat </etc/hosts >| hosts   # 强制覆盖
$ set +o noclobber     # 关闭noclobber选项
$ cat /etc/hosts >> hosts    # 标准输出附加到文件hosts中
```

标准错误重定向

- 命令执行过程中出现异常时一般会通过标准错误输出相应的错误信息
- 缺省时标准输出和标准错误都为屏幕
- 如果不需要来自于标准错误输出的错误信息，可以重定向到位桶(bit-bucket)文件中
- 标准错误输出重定向： 2> target

位桶(bit-bucket)文件：

- /dev/zero 读时产生一系列的NULL字符，写入时数据丢弃
- /dev/null 读时产生EOF，写入时丢弃
- /dev/urandom: 读时立刻不断返回随机数
- /dev/random: 读时可能要等待噪声足够的情况下才返回随机数，比urandom更加安全

```
dlmao@mars:~$ ls /usr/binn   # 下面信息来自于STDERR
ls: cannot access '/usr/binn': No such file or directory
dlmao@mars:~$ ls /usr/binn 2> /dev/null
dlmao@mars:~$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 18.04.4 LTS
Release:        18.04
Codename:       bionic
dlmao@mars:~$ lsb_release -a 2> /dev/null
```

重定向：标准输出和错误定向到同一个文件

- cmd >**FILE** 2>&1　将标准输出和标准错误都重定向到**FILE**中
  - 2>&1 表示将文件描述字2输出重定向到描述字为1所对应的打开文件中
  - 注意重定向的先后顺序，cmd 2>&1 > FILE 首先重定向标准错误到标准输出（此时为屏幕），然后标准输出重定向到FILE
- &> (&>>可附加到文件)或>&将标准输出和标准错误都重定向到文件

cmd &>FILE　或 cmd >&FILE 注意：&> 或>&之间不能有空格

建议采用第一种格式 cmd &>FILE

- **前一种格式可支持附加: cmd &>>FILE**
- 后一种格式不支持附加(>>)到文件，而且**文件名不能为数字**

- >& 之后为数字或其他字符表示不同的含义
  - >&后面为数字 (>&n)：重定向到文件描述字n
  - >& 后面为其他字符(>&FILE): 标准输出和错误输出重定向到文件FILE

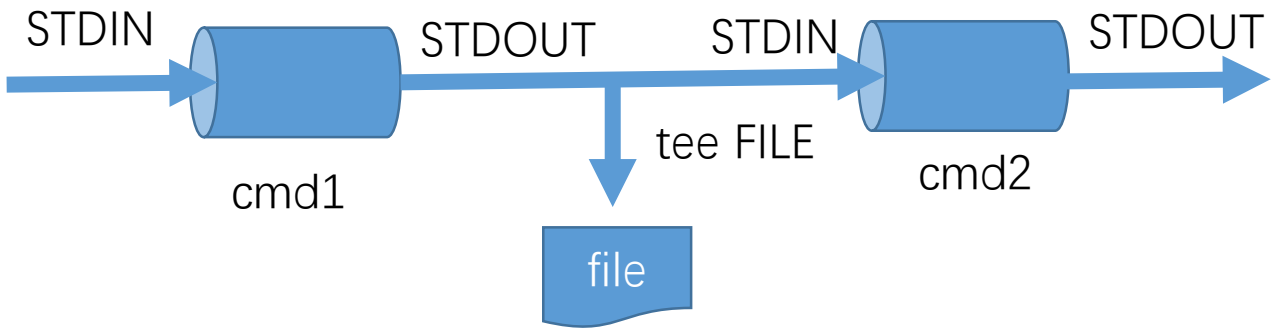
管道

- 多个命令之间可以通过管道字符(|)来隔开，把上一个命令的标准输出定向到下一个命令的标准输入中

  cmd1 | cmd2 | cmd3 ….

- 命令之间的连接称为**管道(pipe)**，采用管道连接的命令也称为**过滤器(filter)**。命令序列本身称为管道线(pipeline)
- |字符前后可以不加空格，但是建议在前后都增加空格
- 也可以采用|&连接过滤器，等价于 2>&1 | ，即STDERR和STDOUT的输出都连接到下一个命令的STDIN，比如 cmd1 |& cmd2 | cmd3
- 管道线分流命令tee：tee [-a] file … 表示除了标准输出外还将数据保存在文件（可以指定多个文件，每个文件保存一份拷贝），-a选项表示附加

* Write programs that do one thing and do it well.
* Write programs to work together.
* Write programs to handle text streams, because that is a universal interface.
* Interactive use instead of batch processing.




```
ls -al /usr/bin/ | grep zip  | tee ziplist.txt |  wc -l
```