作业管理、文件查找

主要内容

- 作业管理2
- 文件查找命令find和locate
- 动态构造并执行命令xargs

设置进程优先级nice

- 调度器基于进程优先级为各个进程动态分配资源

nice [-n niceness] command

renice priority [-g|-p|-u] identifier...

```
dlmao@mars:~$ sleep 300&
[1] 4221
dlmao@mars:~$ nice sleep 400&
[2] 4222
dlmao@mars:~$ nice -n 4 sleep 500&
[3] 4223
```

进程的niceness值(NI)为[-20,19]，值越低，优先级越高。普通命令的NI为0

nice表示以**低优先级(-n选项指定NI值，缺省为10）执行外部命令**，这些命令是那些需要大量CPU且可在后台运行的命令，一般会在后面添加&以后台运行

- renice改变**已经运行的进程的优先级**，优先级可以使用 -n pri或者pri，后面为进程ID，也可通过-g -u选项根据进程组或用户选择，普通用户只能**设置更大(即更低)的优先级**

普通用户设置的NI值只能[0,19]，只能为自己的进程设置，只能设置得更高

ps l可以查看NI值

```
dlmao@mars:~$ ps axl
F   UID   PID  PPID PRI  NI    VSZ   RSS WCHAN  STAT TTY        TIME COMMAND
1     0   315     2   0 -20      0     0 -      S<   ?          0:00 [loop0]
0  1000  3859  3858  20   0  26204  5532 wait   Ss   pts/4      0:00 -bash
0  1000  4221  3859  20   0  10736   880 hrtime S    pts/4      0:00 sleep 300
0  1000  4222  3859  30  10  10736   788 hrtime SN   pts/4      0:00 sleep 400
0  1000  4223  3859  24   4  10736   764 hrtime SN   pts/4      0:00 sleep 500
0  1000  4225  3859  20   0  32152  1524 -      R+   pts/4      0:00 ps axl
dlmao@mars:~$ renice -n 12 4222
4222 (process ID) old priority 10, new priority 12
```

显示进程树pstree

pstree [options] [PID | USER]

查看进程ID为PID开始的进程树，如果不传递参数，缺省为1（即整个进程树）。如果传递为名字，表示查看该用户的进程对应的进程树

-p, --show-pids 显示进程ID
-g, --show-pgids 显示进程组ID
-n, --numeric-sort 子节点排序为按照PID而不是名字排序
-a, --arguments 显示进程的命令行参数
-l, --long 显示时不要截取内容
-s, --show-parents 显示PID的先辈进程

pstree -pgn 查看整个进程树，包含进程ID和进程组ID，按PID排序

```
dlmao@mars:~$ pstree -pgn | less
systemd(1,1)-+-systemd-journal(220,220)
             |-systemd-udevd(242,242)
             |-ModemManager(568,568)-+-{gmain}(598,568)
             |                       `-{gdbus}(615,568)
             |-rsyslogd(585,585)-+-{in:imuxsock}(603,585)
             |                   |-{in:imklog}(604,585)
             |                   `-{rs:main Q:Reg}(605,585)
```

pstree -s $$ 查看当前shell开始的进程树，但也包括到进程树根的那一段

```
dlmao@mars:~$ pstree -pgs $$
systemd(1,1)---sshd(894,894)---sshd(3793,3793)---sshd(3858,3793)---bash(3859,3859)---pstree(4972,4972)
```

监视系统进程top

也可考虑使用htop
sudo apt install htop

top [-b] [-d delay] [-u user] [-n count] [-p pid[,pid]…]

实时查看系统中当前运行的进程的所有信息，每隔delay(-d secs）刷新状态信息，-n count指定刷新几次后退出，-p pid指出只查看哪些进程的状态，-b 表示批处理执行(可重定向到文件)，缺省屏幕状态执行，和less/vi类似，以原始模式运行，完全接管屏幕

- h或者?查看帮助
- q退出
- ESC取消
- b:打开/关闭加亮效果(状态为运行R的行y或者某排序列x)
- x和y: 打开/关闭排序列/运行态进程高亮效果
- >或< 改变排序列
- f: 更改显示列
- 箭头和翻页键确定显示的内容
- u 查看哪些用户，s设置刷新间隔，n设置屏幕显示的行数
- k 发送信号给某个进程

```
top - 16:27:12 up 4 days,  7:57,  3 users,  load average: 0.00, 0.00, 0.00
Tasks: 215 total,   1 running, 210 sleeping,   4 stopped,   0 zombie
%Cpu(s):  0.0 us,  0.3 sy,  0.0 ni, 99.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem :  2048444 total,   250336 free,   910500 used,   887608 buff/cache
KiB Swap:  2095100 total,  2010200 free,    84900 used.   896968 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 1324 root      20   0  364360   1648   1584 S   0.3   0.1   2:25.65 VBoxService
 2325 demo      20   0  232264    644    644 S   0.3   0.0  21:15.44 VBoxClient
19274 demo      20   0   97468   4184   3212 S   0.3   0.2   0:01.33 sshd
19936 demo      20   0  157860   3792   3236 R   0.3   0.2   0:00.37 top
    1 root      20   0  119776   5388   3516 S   0.0   0.3   0:17.80 systemd
    2 root      20   0       0      0      0 S   0.0   0.0   0:00.17 kthreadd
    3 root      20   0       0      0      0 S   0.0   0.0   0:37.50 ksoftirqd/0
    5 root       0 -20       0      0      0 S   0.0   0.0   0:00.00 kworker/0:0H
    7 root      20   0       0      0      0 S   0.0   0.0   0:56.02 rcu_sched
    8 root      20   0       0      0      0 S   0.0   0.0   0:00.00 rcu_bh
    9 root      rt   0       0      0      0 S   0.0   0.0   0:00.00 migration/0
   10 root      rt   0       0      0      0 S   0.0   0.0   0:05.08 watchdog/0
```

查找文件命令find

find [-H] [-L] [-P] [FILE...] [TEST EXPRESSION] [ACTION]

- 根据**文件的各个属性**从某个或者某些目录开始搜寻文件系统

find . -size +1G –exec mv {} ~/bigfiles \;

控制是否跟随（展开）符号连接的选项:
-P 不展开符号连接，缺省选项
-L 展开符号连接
-H 仅展开命令行参数中的符号连接

哪里?	参数为一个或多个目录，如果没有指定目录，则表示从当前目录开始
怎么找?	-开始，测试或匹配表达式给出了要比较哪些文件属性，如果没有指定匹配表达式，表示都匹配
干什么?	-开始，动作部分给出了匹配时所采取的动作，如果没有指定，则表示输出（-print）文件名
	找到的文件的名字以起始点dir开始，即文件名为dir/…/file 如果dir为相对路径，则文件名也是相对路径，如果dir为绝对路径，则文件名也是绝对路径

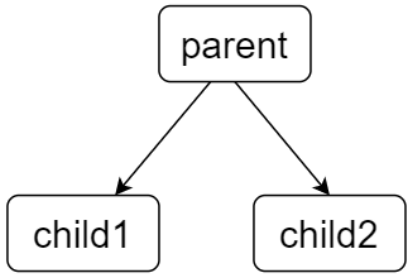
find	从当前目录(.) 开始查找所有的文件并显示文件名
find *	首先通配符扩展起作用，等价于find file1 file2 …，输出文件名为相对路径
find /boot	从/boot目录开始查找，输出文件名为绝对路径

```
dlmao@mars:/$ find boot
boot
boot/initrd.img-5.3.0-46-generic
...
dlmao@mars:/$ find /boot
/boot
/boot/initrd.img-5.3.0-46-generic
/boot/vmlinuz-5.3.0-46-generic
...
```

查找文件命令find：全局选项和测试

-depth	搜索时在遇到目录时，首先检查目录中包含的内容，然后检查该目录。**缺省首先检查目录，然后检查目录中的内容**。 • 采用或不采用本选项，最终输出顺序不一样 • 要**删除目录时，会自动打开本选项**，首先删除目录中的内容，然后删除目录本身
-maxdepth <levels>	搜索时最多到的目录级数。levels为0表示仅测试命令行传递的起始点。levels为1表示仅检查第一级子目录中的内容
-mindepth <levels>	搜索时仅包含目录级数至少为levels的内容
-xdev, -mount	搜索时不要跨越到其他文件系统
-noleaf	扫描非UNIX文件系统(比如CD和windows文件系统)时采用的选项，优化查找速度

```
dlmao@mars:/$ find /etc -mindepth 1 -maxdepth 2 -type d
/etc/gdm3
/etc/gdm3/PrimeOff
/etc/gdm3/PostLogin
...
```




```
$ tree 1
1
├── 11
│   ├── 111
│   │   └── 1.txt
│   └── 1.txt
├── 12
│   └── 121
├── 1.txt
└── 2.txt
```

先序:
parent-child1-child2
后序:
child1-child2-parent

```
$ find 1
1
1/12
1/12/121
1/11
1/11/111
1/11/111/1.txt
1/11/1.txt
1/1.txt
1/2.txt
```

```
$ find 1 -depth
1/12/121
1/12
1/11/111/1.txt
1/11/111
1/11/1.txt
1/11
1/1.txt
1/2.txt
1
```

匹配表达式：文件名、路径名、符号链接

文件名	-name pattern -iname pattern	文件名的基本名(即去掉前面的目录部分)与pattern**完全匹配**；iname表示大小写无关。pattern采用文件通配符，但**可匹配最开始的.**
	-path pattern -ipath pattern	路径名(从搜索起点开始)与pattern**完全匹配**；ipath表示大小写无关。pattern采用**文件通配符，但注意可匹配最开始的.和目录分隔符/**
	-regex expr -iregex expr	路径名(完整文件名)与正则表达式expr**完全匹配**，缺省采用的正则表达式为POSIX基本正则表达式(BRE)。可通过-regextype name采用其他类型的正则表达式(如posix-extended, posix-basic)
连接	-lname pattern -ilname pattern	符号连接中的内容与pattern完全匹配

- shell支持文件通配符扩展 * ？[a-z] [[:digit:]] [^123]
- find命令在匹配基本名、路径名时也可使用文件通配符
  - 相比shell而言，可以匹配最开始的.字符，即可以匹配隐藏文件或目录
  - -path pattern时，可匹配最开始的.和目录分隔符/。如foo*bar可匹配 foo3/bar以及foo3/.bar
  - 建议pattern通过引号引用避免被shell首先解释，或转义，如 –name *.txt

`find /etc -name '*.conf'`	查找/etc目录及各级子目录下以.conf结尾的文件
`find /etc -path '/etc/p*/*.conf'`	查找/etc/目录下，所有以p开头的子目录及其下的各级子目录中以.conf结尾的文件
`find /usr/bin -lname '/etc/*'`	查找/usr/bin目录下，指向/etc目录下的文件的符号连接

find: 匹配表达式

连接	-lname pattern -ilname pattern	符号连接中的内容与pattern匹配
	-links [+-]N	硬连接个数等于n或者超过或者小于n
	-inum N	inode节点编号为N (可通过ls –i或者stat –c %i获得)
	-samefile NAME	inode编号与NAME的inode编号一致，用于查找NAME的硬连接
类型	-type C	文件类型为C，其中C可为d(目录)、f(普通文件)、l(符号连接)、b(块设备)、c(字符设备)、p(命名管道,FIFO)、s(Socket)等
拥有者	-uid N -gid N	文件拥有者uid或文件用户组gid为N,支持 +N -N，表示大于或小于N
	-user uname -group gname	文件拥有者或文件用户组的名字为uname或gname
	-nouser –nogroup	文件拥有者或文件用户组不在/etc/passwd或/etc/group中

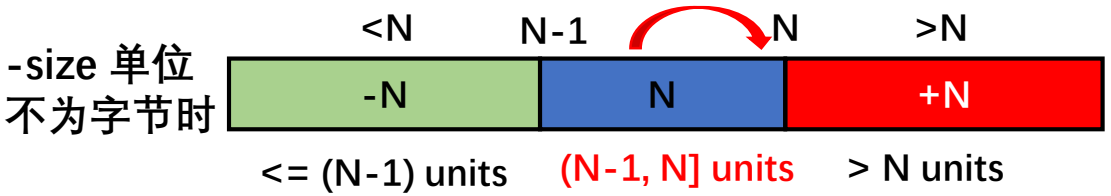
find / -links +1 -type f 2> /dev/null	查找有硬连接的文件(links > 1)
find -samefile 1.txt -ls	当前目录下查找指向1.txt的硬连接文件，长列表显示
find /home –type d -user demo	在/home目录下寻找demo用户拥有的目录
find /home –nouser	在/home目录下寻找无主的文件

匹配表达式：权限

权限	-perm PMODE	PMODE可采取数字或符号方式描述，最前面可为字符-或者/ • 没有-或/，表示文件权限与后面的描述完全一致，如766 • 最前面为-，表示文件权限包含mode中指出的权限；如-744 • 最前面为/，表示文件权限包含了mode中指出的权限中任一权限，如/700
	-readable -writable -executable	**当前用户**是否可以读、写或执行

find /etc -type f -perm -0744	在/etc/目录下寻找普通文件且其权限中用户权限为rwx,组权限至少为r,其他权限至少有r (0744= rwxr—r--)
find /etc -perm 444	在/etc目录下查找权限正好为444(r—r—r--)的文件
find /etc -perm u=r,g=r,o=r	与上条命令一样
find –perm /222 –type f	当前目录下查找可写的普通文件, 222=-w--w--w-
find /usr -executable -perm /7000 -ls	在/usr目录下寻找当前用户可执行且有特殊权限的文件

大小	-empty	普通文件或目录，且其内容为空
	-size [+-]N[bckwMG]	搜寻文件大小大于(小于或等于）N的文件，N后面可以包含一个字符，该字符给出了N的单位，缺省单位为b，表示512字节的块。其他单位还包括c(字节)、w(2字节的word)、k(1024字节)、M(2^{20}字节)、G((2^{30}字节)。注意文件大小比较是round up到相应的大小单位，即10K字节表示[9*1024+1,10*1024]字节



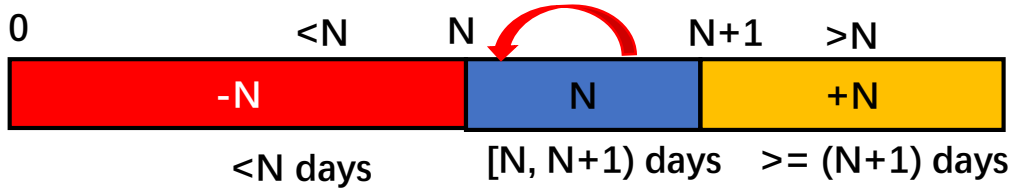
- -size 50k 文件大小为49*1024+1到50*1024字节
- -size +50k 超过50k字节的文件
- -size -50k 文件大小小于等于49*1024字节

-size [+-]N round up到最近的计量单位

`find /etc -empty -type f`	查找空的普通文件
`find /etc -size -1M -type f`	查找空的普通文件, 表示小于等于 (1-1)M，即长度为0的普通文件
`find /etc -not -size +0M -type f`	查找空的普通文件， 不大于0M(小于等于0)
`find /usr/bin -type f  -size 1`	正好占用1个block的小文件(注意不包括空文件)
`find /home -user dlmao -type f -size +10M`	查找/home下用户dlmao拥有大小超过10M的大文件

时间	-atime/-mtime/-ctime [-+]N	文件的访问、修改或状态改变时间正好为N**天(24小时)**，或N天之内，或N天之前。注意文件的时间戳是round down到相应的时间单位，即0表示最近24小时
	-amin/-mmin/-cmin [-+]N	正好N**分钟**或N分钟之内或N分钟之前访问(修改、状态改变)过
	-daystart	表示时间比较不是从现在开始，而是与**当天的结束时刻**比较，注意其仅影响后面出现的时间比较表达式
	-newer file –anewer file -cnewer file -newerXY reference	• 文件的修改(访问或状态改变)时间在file对应的时间之后 • 文件的X(acm)时间比reference的Y(acmt)时间更新。-newermt表示修改时间在reference之后。如-newermt '2018-05-28'

当前时刻-文件时间的间隔满足



-mtime 3 [3*24, 4*24)小时间修改过
-mtime -3 3*24小时之内修改过
-mtime +3 4*24小时之前修改过

find -mtime 0 或**find –mtime -1**	最近24小时内修改过的文件
find -mtime -7	最近一个星期修改过的文件
find -mmin +59	最近(59+1分钟)一个小时之前修改过的文件
find -daystart -atime 1 **find -daystart -not -atime -1 -atime -2**	昨天访问过的文件
find -anewer ~/.bashrc	访问时间在.bashrc之后，一般是这次会话访问的文件
find -newerat '2020-05-12'	2020年5月12日之后访问过的文件

find: 逻辑运算符

-and , -a	两个条件都满足才匹配，连续的两个条件**默认为用and运算**
-or, -o	两个条件任一一个满足才匹配
-not, !	条件为假时匹配
( )	改变逻辑运算的先后顺序。需要在前面添加\,或者引号引用，比如\(或'('

- 一元(not)优先级最高, and第二，or最低。-and可以省略
- 采用左结合，即从左到右顺序
- 二元逻辑运算符 –and –or为短路逻辑运算
  - test1 –and test2: test1为假时后面的test2不会执行
  - test1 –or test2: test1为真时后面的test2不会执行

find ~ \(-type f -and -not -perm 0600 \) -or \(-type d -and -not -perm 0700 \)	查找那些访问权限不是600的文件及访问权限不是700的子目录
find ~ \(-type f ! -perm 0600 \) -o \(-type d ! -perm 0700 \)	与上面命令一致

查找文件命令find: 动作

给出了前面的条件满足时要执行的动作

-print	缺省采用的动作选项。打印当前匹配的路径名，然后输出换行符。
-print0	与print类似，只是文件名之间以空字符(NUL) 分割，其输出一般通过管道由xargs等命令处理
-printf format	按照指定格式输出路径名
-ls	-ls 相当于对于匹配的文件调用 ls –dils （-s选项表示分配的数据块大小），输出的行中最前面是inode节点编号，然后是分配数据块大小
-fprint/fprint0/fls /fprintf FILE	表示将输出保存到指定的文件
-quit	不再继续查找下一个匹配的文件
-prune	如果要匹配的文件为目录，则**跳过该目录**，不会继续进入该目录查找。如果有-depth选项，本选项不起作用,因为其首先查找目录中的内容，最后才是该目录。动作与匹配表达式一样，也有一个结果，-prune的结果总是true

find ~ -size +1M -fprint ~/big.txt	查找超过1M的文件，将其文件名保存到big.txt
find /etc -name 'vim' -print -quit	查找/etc目录，找到第一个名为vim的文件后结束
find . -name .cache -prune **-o \(** -name '*.py' -o -name '*.txt' **\)**	搜索目录下后缀名为.py和.txt的文件，但不包括.cache下的内容。注意**-prune后的逻辑运算符为-o**，利用短路逻辑特性，如果匹配的文件名为.cache，-prune结果为true，因此不会检查后面的条件

查找文件命令find: 动作

给出了前面的条件满足时要执行的动作

-delete	删除文件和目录。如果删除成功该动作返回true。意味着自动打开-depth选项
-exec command ;	执行command，如果该命令执行成功，则该动作返回true。command里面的 {}会替换为当前匹配的路径名。注意分号表示命令的结束，但由于其也是shell元字符，因此要在前面加上\进行转义或通过引号来引用。即\; 或';'
-exec command +	与上面选项类似，只是不是每个匹配文件执行一次命令，而是收集匹配的文件名，最后find结束时，将之前匹配的多个文件名作为参数执行命令一次
-ok command ;	与exec类似，只是执行命令前会询问用户

find /tmp –type f –atime +30 –name '*.bak' –delete	删除后缀为.bak，且31天没有访问的普通文件
find /tmp –type f –atime +30 –name '*.bak' –exec rm -i '{}' \;	删除后缀为.bak，且31天没有访问的普通文件，在删除之前会询问(rm的-i选项)
find . -size +1G -ok mv {} ~/bigfiles \;	查找当前目录下文件大小超过1G的大文件，将文件移动到bigfiles目录，在执行移动命令前询问用户
find . -size +1G -exec mv {} ~/bigfiles +	找到所有匹配的文件名后一次mv，但文件多时可能会超过参数长度限制
mv $(find . -size +1G) ~/bigfiles	采用命令替换实现，但注意文件名有空格会有问题

locate命令：从数据库中查找文件

- 系统定期通过updatedb更新一个数据库（保存在/var/lib/mlocate目录），包含所有文件的路径名
- locate命令查询数据库返回**匹配的路径名**，速度非常快，但不保证文件一定存在

locate [-AbcirS] [--regex] pattern …

pattern给出了匹配的模式，缺省匹配路径名，可使用文件通配符。如果pattern没有包括通配符时相当于 *pattern* 。可以有多个pattern，其中任一个匹配时返回对应的文件名

-A, --all	匹配所有的模式，缺省是匹配多个模式中任一个
-b, --basename	匹配时仅仅比较文件名（即路径名的basename）
-r 或--regex	-r表示采用基本表达式，后者采用扩展正则表达式匹配
-i, --ignore-case	忽略大小写
-c, --count	仅输出匹配的文件个数
-S	查看数据库统计信息

locate '*txt'	查找所有txt结尾的文件，采用文件通配符匹配路径名
locate –c '*jpg'	查看jpg文件的个数
locate -br '^temp$'	查看所有文件名为temp的文件
locate --regex '^/etc/m'	查看所有路径名以/etc/m开始的文件
locate –A bin zip	路径名中包含了字符串bin和zip

回顾

- echo:将命令行的内容放到标准输出
  echo arg1 arg2
- cat:将文件(包括标准输入)中的内容放到标准输出
  cat file1 file2
- 命令替换：执行命令的标准输出的东西放到命令行
  echo $(date; ls)，注意单词分割
- xargs 动态构造命令，将标准输入（一般来自于管道中的前一条命令的标准输出）的内容放到命令行

xargs 从标准输入构造命令并执行

- find命令执行相应的动作时：
  - 一般是每个匹配的文件执行一次动作，效率较低
  - exec + 等允许首先汇集匹配的文件名，最后执行动作，但是exec后面的参数长度可能有限制

```
xargs [-0prt] [-E eof-str] [-I replace-str] [–n max-args]  [command [initial-arguments]]
```

- 构造的命令中参数来自于标准输入，command后可指定一些初始的参数，command没有指定时**缺省为echo**
- 一般根据IFS（缺省为空格、制表和换行符等空格类字符）分割参数

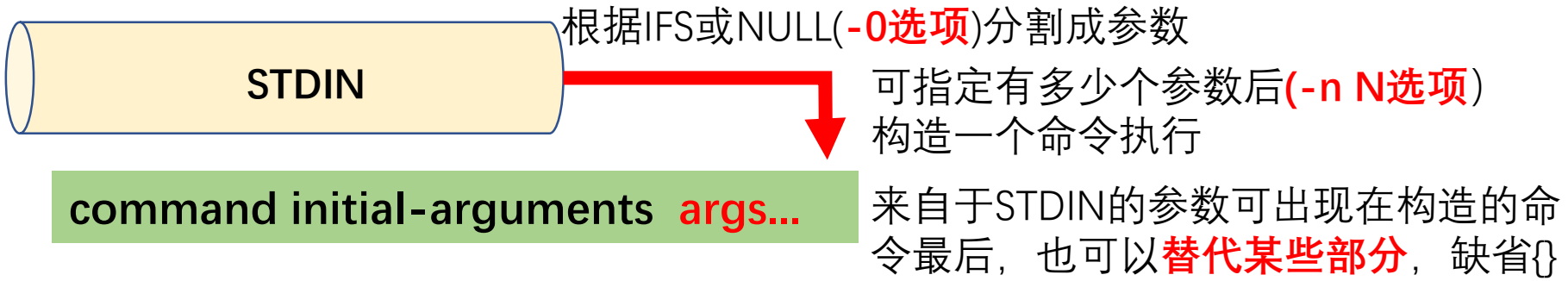
✓选项-0表示用NULL字符来分割参数
  - 考虑到文件名可能包含特殊字符，比如空格
  - find命令的print0的输出采用NULL字符隔开匹配的文件名

✓-n max-args选项表示每次构造的命令只用到**最多max-args个参数**，即执行多次命令

✓-E eof-str选项表示标准输入中看到eof-str时参数结束

xargs 从标准输入构造命令并执行

```
xargs [-0prt] [-E eof-str] [-I replace-str] [–n max-args]  [command [initial-arguments]]
```



-0,--null	分割参数时采用NULL字符，缺省空格类字符(IFS)
-d DELIM	分割参数时采用字符DELIM作为参数分隔符， 比如 -d :
-r,--not-run-if-empty	构造命令时，从标准输入没有读到参数时不执行命令
-p, --interactive	执行命令前询问用户
-t,--verbose	执行命令前将要执行的命令输出到标准错误
-n max-args	标准输入每得到max-args个参数后构造一个命令执行
-E eof-str	标准输入中看到eof-str时参数结束
-I replace-str	占位符为replace-str，构造的命令为 command initial-arguments，其中initial-arguments中的占位符被替换为从标准输入构造的参数
-i	等价于 -I {}

xargs 从标准输入构造命令并执行

(whoami; date) | xargs	多条命令的输出放在同一行
xargs cat <filenames > merge-file	filenames里面给出了多个文件名，将这些文件的内容合并在一起 (构造cat file1 file2 .... > merge-file)
find . -size +1G –exec mv {} ~/bigfiles \;	每找到一个文件mv一次
find . -size +1G -print0 | xargs -0rp mv –t ~/bigfiles	利用mv命令的-t选项，后面的参数来自于标准输入，如果没有找到(-r选项)不执行，-p选项执行前询问用户
find . -size +1G -print0 | xargs -0r -I {} mv {} ~/bigfiles	将超过1G的文件移动到bigfiles目录