归档、文件内容查找和发送信号

主要内容

- 压缩和归档
- 文件内容查找
- 发送信号

压缩命令gzip/gunzip和bzip2/bunzip2

gzip [options] [FILES…]

- gzip命令采用Lempel-Ziv算法进行压缩，产生的文件后缀一般为.gz。压缩文件的访问模式和时间一般与源文件一致。
- 如果没有指定文件，表示标准输入，输出到标准输出
- 解压缩可采用gunzip或gzip –d
- gzip还支持解压缩zip/compress命令创建的压缩文件
- bzip2/bunzip2采用更高质量压缩算法

```
dlmao@greensea:~/root$ cp /etc/hosts* .
dlmao@greensea:~/root$ ls -l
total 4
-rw-r--r-- 1 dlmao dlmao 410 May 20 23:17 hosts
-rw-r--r-- 1 dlmao dlmao 411 May 20 23:17 hosts.allow
-rw-r--r-- 1 dlmao dlmao 711 May 20 23:17 hosts.deny
dlmao@greensea:~/root$ gzip hos*
dlmao@greensea:~/root$ ls -l
total 0
-rw-r--r-- 1 dlmao dlmao 282 May 20 23:17 hosts.allow.gz
-rw-r--r-- 1 dlmao dlmao 441 May 20 23:17 hosts.deny.gz
-rw-r--r-- 1 dlmao dlmao 288 May 20 23:17 hosts.gz
dlmao@greensea:~/root$ gzip -d hos*
```

选项	含义
-c	保留原有文件，输出到stdout
-f	强制覆盖文件
-r	递归方式
-t	检验压缩文件的完整性
-v	显示详细信息
-l	列出压缩文件信息
-d	解压缩，相当于gunzip

gzip –r old	递归压缩目录old中的所有文件，原来的文件被压缩后的文件代替
gzip –rd old	递归解压old目录下的所有压缩文件

压缩命令gzip/gunzip和bzip2/bunzip2

gzip [options] [FILES…]

- gzip命令采用Lempel-Ziv算法进行压缩，产生的文件后缀一般为.gz。压缩文件的访问模式和时间一般与源文件一致
- zcat等价于gunzip –c或gzip –cd， 类似地还有zless

```
~/root$ gzip hosts
~/root$ ls hosts*
hosts.allow  hosts.deny  hosts.gz
~/root$ gzip -cd hosts.gz > hosts2
~/root$ ls hosts*
hosts.allow  hosts.deny  hosts.gz  hosts2
```

选项	含义
-c	保留原有文件，输出到stdout
-f	强制覆盖文件
-r	递归方式
-t	检验压缩文件的完整性
-v	显示详细信息
-l	列出压缩文件信息
-d	解压缩，相当于gunzip

归档命令tar

tar mode [options] [pathname…]

- tar命令将多个文件和目录**打包成一个文件**，通过f选项指定归档文件，一般后缀名为.tar，如果不指定，表示为标准输入或者输出
- 第一个参数为模式mode，指定所要完成的功能。模式前的**连字符可选**，如tar cvf tarfile dirs或tar -cvf tarfile dirs.
- 在归档时，后面的pathname参数给出了归档哪些文件或目录，而在untar时表示仅提取哪些文件或目录

选项	含义
-v	显示详尽信息
-f	指定归档文件
-j -z	采用bzip2或gzip进行压缩或者解压缩
-p	保留文件权限属性
-N date-or-file	仅仅归档修改时间在指定时间之后的文件
--exclude=file	归档时不包含file
-T, --files-from FILE	创建或者提取时，文件名来自于FILE而不是参数

模式	含义
-c	创建归档文件
-r	附加文件到归档文件中
-A	附加tar文件到归档文件中
-x	从归档文件中提取文件
-t	列出归档文件中的内容

- 创建归档文件 tar cvf tarfile dirs
  - 创建时采用相对路径，即移除路径名**最前面的斜杠**
- 查看归档文件内容：tar tvf tarfile
- 提取全部文件： tar xvf tarfile，或提取其中某些文件 tar xvf tarfile files

归档命令tar

tar cv old | gzip > old.tar.gz tar cvf - old | gzip>old.tar.gz	打包old，将其输出到标准输出，然后gzip压缩后重定向到old.tar.gz
tar cvzf old2.tar.gz old	首先打包目录old成一个文件，然后使用gzip压缩，保存在old2.tar.gz
tar tvzf old2.tar.gz	列出old2.tar.gz里面包含的文件信息
tar xzvf old.tar.gz gzip -c -d old.tar.gz | tar xvf -	解压缩并解包压缩包中的所有文件，会在当前目录创建目录(压缩时后面的目录名)….
find /bin -name '*zip*' | tar cvzf bin-zip.tar.gz -T - find /bin -name '*zip*' -print0 | xargs -r0 tar rzvf bin-zip.tar.gz	查找/bin目录中文件名中有zip的文件， tar命令表示打包来自于标准输入(前面find找到)的文件，然后压缩写到bin-zip.tar.gz文件 。

主要内容

- 压缩和归档
- 文件内容查找
- 发送信号

grep(global regular expression print)

egrep/fgrep/grep [OPTIONS] PATTERN [FILE…] 选取包含特定模式的行

grep从标准输入(如果没有文件参数或文件参数为-)或文件中以行为单位读取，如果该行匹配给定的模式(正则表达式)，则将该行输出

- 指定模式有三种途径，可通过-e或-f 选项指定（-e和-f可多个，可组合使用）；如果没有采用-e或-f选项，则第一个参数为要匹配的模式

模式	-e PATTERN	指定匹配的模式，可以有多个-e选项，匹配任一一个
	-f FILE	从文件中（每行对应一个模式）读取匹配的模式
	-r, --recursive	递归搜索目录及子目录的文件
	-s, --no-messages	错误信息不输出，也可使用 2>/dev/null

```
dlmao@mars:~$ lsblk |grep sd
sda       8:0    0    20G  0 disk
└─sda1    8:1    0    20G  0 part /
...
dlmao@mars:~$ grep root /etc/{passwd,group}
/etc/passwd:root:x:0:0:root:/root:/bin/bash
/etc/group:root:x:0:
dlmao@mars:~$ lsblk |grep -e sd -e sr
...
```

```
dlmao@mars:~$ echo -e 'root\nmail' | grep -f -
/etc/passwd
root:x:0:0:root:/root:/bin/bash
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
dlmao@mars:~$ grep -rs noremap /usr/share/vim
...
```

grep(global regular expression print)

egrep/fgrep/grep [OPTIONS] PATTERN [FILE…] 选取包含特定模式的行

- 模式有4种类型，分别是固定模式(固定字符串)、基本和扩展正则表达式、perl语言的正则表达式

模式种类	-G, --basic-regexp	采用基本正则表达式BRE，缺省选项
	-E, --extended-regexp	采用扩展正则表达式ERE，egrep相当于grep -E
	-F, --fixed-strings	采用固定模式，fgrep相当于grep –F
	-P, --perl-regexp	采用perl语言的正则表达式，更强大
匹配控制	-i, --ignore-case	忽略大小写
	-v, --invert-match	输出与模式不匹配的行，缺省为输出模式匹配的行
	-w, --word-regexp	匹配整个单词(字母数字下划线)
	-x, --line-regexp	匹配整行

```
$ echo -e 'note\nnote*' | grep  'note*'
note
note*
```

```
$ echo -e 'note\nnote*' | grep  -F 'note*'
note*
$ grep -i options /usr/sbin/locale-gen
#  Handle command-line options
Usage: locale-gen [OPTIONS]
Options:
$ ps ax |grep ssh
  908 ?        Ss     0:00 /usr/sbin/sshd -D
 1431 ?        Ss     0:00 sshd: dlmao [priv]
 1549 ?        S      0:01 sshd: dlmao@pts/0
 2649 pts/0    R+     0:00 grep --color=auto ssh
$ ps ax | grep ssh | grep -v grep
 908 ?        Ss     0:00 /usr/sbin/sshd -D
 1431 ?        Ss     0:00 sshd: dlmao [priv]
 1549 ?        S      0:01 sshd: dlmao@pts/0
$ echo -e 'note\nnotes' | grep -w 'note'
note
```

egrep/fgrep/grep [OPTIONS] PATTERN [FILE…]

- 缺省情况下会输出匹配的行。如果多个文件，则每行还包括了文件名。
- 可以控制是否输出文件名、行号。也可仅输出匹配的行的个数。也可仅仅输出有匹配或者没有匹配的文件名等

```
$ echo 'pi=3.14, e=2.718'
|grep -o -E '[0-9]+\.[0-9]*'
3.14
2.718
```

输出控制模式	-o, --only-matching	在输出匹配行时仅仅包括匹配的内容
	-h, --no-filename	输出时不包含文件名，一个文件时缺省模式
	-H, --with-filename	输出时也包含文件名，多个文件时缺省模式
	-n, --line-number	输出时包含行号，缺省不包括行号
	-c, --count	仅输出匹配的行数
	-l, --files-with-matches	仅输出那些有匹配的文件名，不包含匹配行
	-L, --files-without-match	仅输出那些没有匹配的文件名，不包含匹配行
	-q, --quiet	不输出任何东西，可检查状态码决定是否有匹配

```
~$ cd /usr/lib/python3.6
```

```
/usr/lib/python3.6 $ grep -n '^def' lzma.py
263:def open(filename, mode="rb", *,
310:def compress(data...
322:def decompress(data, ...
/usr/lib/python3.6$ grep -c '^def' z*.py
zipapp.py:6
zipfile.py:9
/usr/lib/python3.6$ grep -c '^def' zipfile.py
9
```

```
/usr/lib/python3.6$ grep  'import re' l*.py
locale.py:import re
/usr/lib/python3.6$ grep -l 'import re' l*.py
locale.py
/usr/lib/python3.6$ grep -L 'import re' l*.py
linecache.py
lzma.py
/usr/lib/python3.6$ grep -q '^def' lzma.py
/usr/lib/python3.6$ echo $?
0
```

回顾: 通配符扩展(wildcard match)

元字符	通配符扩展	正则表达式
*	匹配0个或多个任意字符，不匹配隐藏文件名中最前面的点.	重复0到多次
?	匹配1个字符，不匹配隐藏文件名中最前面的点.	重复0到1次
[characters]	匹配任意一个属于characters的字符，characters可使用-指定一定范围的字符如 [0-9a-z]	字符集，类似， • 正则表达式使用^表示否定 • 预定义字符类只能出现在[ ]内部
[^characters] [!characters]	匹配任意一个不在characters里面的字符，^和！都代表非(not), 建议采用!	
[[:class:]]	匹配任意一个属于指定的预定义字符类的字符，预定义字符类只能出现在[ ]内部	

类	含义	相当于
[:lower:]	小写字母	a-z
[:upper:]	大写字母	A-Z
[:alpha:]	大小写字母	A-Za-z]
[:alnum:]	大小写字母和数字	A-Za-z0-9
[:digit:]	数字	0-9

类	含义	相当于
[:punct:]	标点符号	
[:blank:]	空格和制表符	\t
[:space:]	空格类字符	\t\r\n\v\f
[:xdigit]	十六进制数字	0-9A-Fa-f
[:word:]	单词字符	alnum加上_

扩展正则表达式：描述单个字符

字符集	[list]	匹配list中的任意一个字符。可指定范围，如a-z匹配a到z之间的字符。注意在[]中^在最前面出现，-在中间出现有特殊的含义。	[abc] [a0-9z] []][^-]
	[^list]	匹配不在list中出现的单个字符	[^0-9]
	预定义字符类	依赖于LC_CTYPE，比如[:lower:] [:upper:] 等。**这些预定义字符类必须出现在[ ]内部**	[[:xdigit:]]
	简约字符集	\s 空格类字符 [[:space:]] \S非空格类字符 [^[:space:]] \w 单词类字符 [_[:alnum:]] \W 非单词类字符 [^_[:alnum:]] **注意简约字符集在中括号里面不可用**	\shello\s \s\w\w\w\s
	.	匹配任意单个字符	b.t
	\	引用或转义：后面的元字符为普通字符（字面量）	\.或[.]

```
$ echo -e 'taste\ntest' | grep 't[ae]st'
taste
test
$ ps ax |grep [s]sh
  908 ?        Ss     0:00 /usr/sbin/sshd -D
...
$ echo -e '[1] Linux\n(2) C' |grep '[[(][1-9][])]'
[1] Linux
(2) C
$ ls /usr/bin | grep '[^bg]zip'
funzip
...
```

```
$ echo -e '12\nff\n'|grep '[[:xdigit:]][0-9a-fA-F]'
12
ff
$ echo -e  '1 linux\n2 c' | grep '[1-9]\s\w'
1 linux
2 c
$ ls /usr/bin |grep '.zip'
funzip
gpg-zip
...
$ echo -e '1. Linux\n2. C' |grep '[1-9]\.'
1. Linux
2. C
```

ps ax | grep ssh | grep -v grep

扩展正则表达式：边界匹配和重复

边界匹配	^	anchor: 匹配行首	^The
	$	anchor:匹配行尾	food$
	\< 或\b	anchor:匹配单词开始	\<food\>
	\>或\b	anchor:匹配单词结束	\bfood\b
	\B	anchar:匹配单词中间	\Bood

```
$ grep '^bat$' /usr/share/dict/words
bat
$ grep '\<bat\>' /usr/share/dict/words
bat
bat's
$ grep '\bbat' /usr/share/dict/words
bat
bat's
batch
...
$ grep '\Bbat\b' /usr/share/dict/words
Rabat
Rabat's
...
$ grep -E  '^([a-z]er){2,}' /usr/share/dict/words
berserk
...
```

	( )	组：括号内的子模式作为整体看待	(fo){1,3}
重复匹配符	{n,m}	前面的字符或组重复n次到m次	a{1,3}
	{n,}	重复至少n次	(fo){1,}
	{n}	重复正好n次 {n,n}	a{3}
	{,m}	最多重复m次 {0,m}	a{,3}
	*	重复0次或多次 {0,}	a*
	+	重复1次或多次 {1,}	a+
	?	重复0次或1次{0,1}	colou?r
	|	选择：多个子模式中任一一个匹配	foo.(org|com|net)

```
$ echo -e '2020-5-1\n2020-05-01' | grep -E
'[0-9]{4}-[0-9]{1,2}-[0-9]{1,2}'
2020-5-1
2020-05-01
$ echo '1   cat' | grep -E '[1-9]\s+cat'
1   cat
$ grep -v '^\s*$' /etc/hosts
# 去掉空行以及仅仅包含空格类字符的行
$ grep -E '^(bat|cat)' /usr/share/dict/words
bat
...
```

反向引用： back-reference \N

- 通过( )包括的组有一个编号: 按照从左到右的左括号的顺序从1开始编号
- 分组引用\N 表示引用前面的编号为N的分组匹配的内容
  - 只能引用之前已经匹配的分组的编号，不能引用后面的分组的编号

```
$ echo 'Paris in the the spring' | grep -E '(\b\w+\b)\s+\1'
Paris in the the spring
```

贪婪匹配

- 针对重复匹配符，采用贪婪匹配算法
- **贪婪匹配算法**是指前导字符或子模式**尽可能多（leftmost or largest)地重复**，只有当这种重复引起整个正则表达式匹配失败的情况下，引擎才会进行回溯（尝试稍短的匹配）

```
$ echo 'a="linux" b="MAC"' | grep -Eo '".+"'
"linux" b="MAC"
$ echo 'a="linux" b="MAC"' | grep -Eo '"[^"]+"'
"linux"
"MAC"
```

[^"]+"	尽可能多地匹配一个以上非"字符，后面为"

基本和扩展正则表达式

- grep缺省(-G)采用基本正则表达式(BRE)，而grep –E 使用扩展正则表达式(ERE)
- 主要的区别：分组、重复（除了*外）以及选择符是否要进行转义

ERE	BRE	含义	emacs	vim
{ }	\{ \}	重复多次，如 a\{1,3\}	不支持	\{ \}，还支持\{1,3}
( )	\(\)	组，如\(ab\)\{1,3\}	\(\)	\(\)
?	\?	匹配0或1次, 如colou\?r	?	\?
+	\+	匹配1次以上, 如\s\+	+	\+
|	|	匹配子模式中的其中一个	|	|
			不支持预定义字符类	支持更多的简约字符集，比如\d \D :help pattern-atom 建议使用\v，表示采用ERE，比如/\vpattern

- locate命令支持基本正则表达式
- find命令
  - 缺省采用emacs格式，它不支持预定义字符类，也不支持通过花括号指定重复次数，采用\(\)表示分组，采用\|表示选择，支持 + ?，也支持\b \B \< \> \s \S等
  - 建议使用 -regextype egrep -regex pattern

主要内容

- 压缩和归档
- 文件内容查找
- 发送信号

kill命令：向进程发送信号

```
dlmao@mars:~$ kill -0 1
-bash: kill: (1) - Operation not permitted
dlmao@mars:~$ echo $?
1
```

- 进程如何退出？
  - 正常退出
  - 控制终端退出时发送信号SIGHUP给所有前台和后台作业
  - 在前台时可Ctrl-C发送中断信号(INT)结束进程或Ctrl-\发送QUIT信号强制终止进程
  - 在前台时可Ctrl-Z发送TSTP信号以暂停进程，这样shell变为前台作业

kill [-signal] pid… | jobid…　　发送信号给进程或作业，缺省发送SIGTERM信号

- 参数可以为进程ID，发信号给哪些进程
  - pid=0表示当前进程组内的所有进程；pid=-1（**谨慎使用！**）表示允许发送信号的所有进程(init除外)；pid<-1表示进程组为-pid的所有进程
- 参数也可为作业号（%n %command %?name），发信号给哪个作业的进程
- 信号可以为数字形式，也可为信号名称，也可为信号简称（不包括前面的SIG部分）
  - 普通用户可以发送信号给自己的进程，超级用户可以发送信号给所有的进程
  - 信号0不实际发送信号，但是进行错误检查，是否存在进程，是否有权限等

```
kill %2
kill -15 4124
kill -SIGTERM 4124
kill -TERM 4124
```

```
$ kill -l     #不同系统支持的信号以及其对应的编号可能有所不同
 1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL       5) SIGTRAP
 6) SIGABRT      7) SIGBUS       8) SIGFPE       9) SIGKILL     10) SIGUSR1
11) SIGSEGV     12) SIGUSR2     13) SIGPIPE     14) SIGALRM     15) SIGTERM
16) SIGSTKFLT   17) SIGCHLD     18) SIGCONT     19) SIGSTOP     20) SIGTSTP
21) SIGTTIN     22) SIGTTOU     23) SIGURG      24) SIGXCPU     25) SIGXFSZ
```

```
dlmao@mars:~$ sleep 300&
[2] 4124
dlmao@mars:~$ kill 4124
dlmao@mars:~$
[2]-  Terminated
sleep 300
```

kill命令：向进程发送信号

- 执行程序可以定义自己的信号处理程序，但是KILL和STOP信号不能忽略

编号	名称	缩写	缺省含义
1	SIGHUP	HUP	控制终端退出信号，结束进程；守护进程用于重启程序，重新读取配置文件等
2	SIGINT	INT	终端中断信号，用户按了Ctrl-C
3	SIGQUIT	QUIT	终端退出信号，用户按了Ctrl-\，保存core文件
9	SIGKILL	KILL	立即终止进程，不能忽略
15	SIGTERM	TERM	缺省终止信号，kill命令没有指定信号时发送该信号，请求终止
20	SIGTSTP	TSTP	交互式停止信号，用户按Ctrl-Z，缺省会暂停进程
19	SIGSTOP	STOP	挂起进程，不能忽略，暂停进程
18	SIGCONT	CONT	恢复挂起进程
	SIGTTIN	TTIN	后台进程试图从终端中读，缺省暂停进程
	SIGTTOU	TTOU	后台进程试图往终端写(stty tostop)，缺省暂停进程

kill命令

```
dlmao@mars:~$ cat | sort&
[1] 4231
dlmao@mars:~$ ps -j
  PID  PGID   SID TTY          TIME CMD
 3347  3347  3347 pts/2    00:00:01 bash
 4230  4230  3347 pts/2    00:00:00 cat
 4231  4230  3347 pts/2    00:00:00 sort
 4232  4232  3347 pts/2    00:00:00 ps

[1]+  Stopped                 cat | sort
dlmao@mars:~$ kill -TERM -4230

[1]+  Stopped                 cat | sort
dlmao@mars:~$
[1]+  Terminated              cat | sort
dlmao@mars:~$
```

```
dlmao@mars:~$ cat | sort &
[1] 4216
dlmao@mars:~$ ps
  PID TTY          TIME CMD
 3347 pts/2    00:00:01 bash
 4215 pts/2    00:00:00 cat
 4216 pts/2    00:00:00 sort
 4217 pts/2    00:00:00 ps

[1]+  Stopped                 cat | sort
dlmao@mars:~$ kill 4215 4216
dlmao@mars:~$ kill -9 4215 4216

[1]+  Stopped                 cat | sort
dlmao@mars:~$ ps
  PID TTY          TIME CMD
 3347 pts/2    00:00:01 bash
 4219 pts/2    00:00:00 ps
[1]+  Killed                  cat | sort
```

killall命令

```
dlmao@mars:/usr$ ps ax |grep [s]shd
  908 ?        Ss     0:00 /usr/sbin/sshd -D
 1431 ?        Ss     0:00 sshd: dlmao [priv]
 1549 ?        S      0:05 sshd: dlmao@pts/0
```

- killall与kill类似，只是**通过名字来指定进程**，如果名字包含了路径分隔符，则仅仅给该程序对应的进程发送信号

killall [options] [-signal] name

-i 交互式模式
-I 忽略大小写
-u user 只给用户user的进程发送信号

- pidof name 通过名字找到对应的进程ID

```
dlmao@mars:~$ sleep 100& sleep 200& sleep 300&
[1] 4259
[2] 4260
+ sleep 100
[3] 4261
dlmao@mars:~$ + sleep 300
+ sleep 200
dlmao@mars:~$ pidof sleep
4259 4260 4261
dlmao@mars:~$ killall -9 sleep
+ killall -9 sleep
[1]   Killed                  sleep 100
[2]-  Killed                  sleep 200
[3]+  Killed                  sleep 300
dlmao@mars:~$
```

查找进程或者给进程发信号pgrep和pkill

pgrep [options] pattern　基于pattern(扩展正则表达式）查找匹配的进程，返回进程的pid

pkill [options] pattern　与pgrep类似，只是给进程发送信号

-signal	pkill发送的信号
-u /-g <uid/gid>	指定用户和用户组的进程
-d delimiter	pgrep返回的进程PID之间的分隔符，缺省为\n
-l, --list-name -a, --list-full	除了pid外还包含进程名或者完整的命令行
-c, --count	仅仅返回进程个数

```
dlmao@mars:/usr$ ps -p $(pgrep -d, ssh)
  PID TTY          TIME CMD
  908 ?        00:00:00 sshd
 1431 ?        00:00:00 sshd
 1549 ?        00:00:06 sshd
```

```
dlmao@mars:/usr$ sleep 200& sleep 300&
[1] 4591
[2] 4592
dlmao@mars:/usr$ pgrep 'sleep|ssh'
908
1431
1549
4591
4592
```

```
dlmao@mars:/usr$ pgrep -a -u `id -un` 'sleep|ssh'
1549 sshd: dlmao@pts/0
4591 sleep 200
4592 sleep 300
dlmao@mars:/usr$ pkill -SIGSTOP sleep

[1]-  Stopped                 sleep 200

[2]+  Stopped                 sleep 300
dlmao@mars:/usr$ pkill -9 sleep
[1]-  Killed                  sleep 200
[2]+  Killed
```