Linux 操作系统及应用
第六章 — 数据处理工具

唐晓晟　李亦农
txs@bupt.edu.cn　hoplee@bupt.edu.cn

Contents

1 高级命令介绍

1.1 cut

- 语法：cut {SECTION SPECIFIERS} [OPTIONS] [FILES]
- 说明：cut命令浏览每行输入，将每行分成节，然后显示指定的节。如果不指定输入，则使用STDIN。
- 节描述符

-bLIST	显示由LIST指定位置上的字节
-cLIST	显示由LIST指定位置上的字符
-fLIST	显示由LIST指定位置上的字段，字段定界符有行起始、行结束和制表符

- 你可以用几种不同的方式来组成LIST参数:
  - 使用逗号表示单个不连续的小节：-f4,7
  - 使用范围表示连续的几个小节：-b2-4
  - 使用n-表示从n小节开始直到行结束的所有小节。
- 选项：

-dDELIMITER	指定节描述符-f所使用的字段定界符
-n	用于支持 Unicode 字符集
-s	和节描述符-f一起使用，用于删除不包含字段分界符的行

- 范例：

```
[Apple] $ more test1.txt
test2alongsentence
this is test1
[Apple] $ cut -f2,3 -d" " test1.txt
test2alongsentence
is test1
[Apple] $ cut -f2 -d" " -s test1.txt
is
[Apple] $ cut -c1-6 test1.txt
test2a
this i
```

1.2 find

- 语法: find [PATHS] [EXPRESSION]
- 说明: find命令检查目录树并计算所遇到的每一个表达式的值。
- 缺省路径是当前路径，缺省表达式是-print

- find根据下列规则来分割路径和表达式，在命令行上第一个-(),!之前的部份为路径，之后的是表达式。
- 表达式分为选项子句、测试子句和动作子句三部分。各个子句之间通过运算符连接，缺省运算符为-and。
- 运算符：

(expr)	圆括号用于改变优先级
! expr	取反运算
-not expr	
expr1 expr2	
expr1 -a expr2	
expr1 -and expr2	短路与运算
expr1 -o expr2	短路或运算
expr1 -or expr2	
expr1, expr2	顺序运算，返回expr2的值

- 选项影响整个find命令，因此它们总是返回真值。

-daystart	当天的起点为00:00:00而不是现在
-depth	子目录深度优先搜索
-help	显示简短帮助信息
-maxdepth n	最多搜索n层子目录，n=0表示测试和动作仅限于命令行列出的目录本身
-mindepth n	最少搜索n层子目录，n=1表示所有除命令行上的目录都要处理
-mount	只检查和指定目录在同一个文件系统下的文件

- find命令的测试：[-1ex]

-amin n	文件在n分钟之前被访问过
-anewer FILE	文件的 LAT 在FILE的 LAT 之后
-atime n	文件在n天之前被访问过
-cmin n	文件状态在n分钟之前被修改过
-cnewer FILE	文件状态的 LMT 在FILE的 LMT 之后
-ctime n	文件状态在n天之前被修改过
-empty	文件为普通文件或目录并且为空
-false	永假
-fstyle TYPE	文件位于指定类型的文件系统上。可以用-printf动作的%F参数显示支持的文件系统类型
-gid n	文件属于 gid 为n的组
-group name	文件属于名字为name的组。允许使用数字形式的 gid 来代替name
-links n	文件有n个链接

-mmin n	文件内容在n分钟之前被修改过
-mtime n	文件内容在n天之前被修改过
-newer FILE	文件的 LMT 在FILE的 LMT 之后
-name PATTERN	文件名与模式PATTERN相匹配
-iname PATTERN	文件名与模式PATTERN相匹配且忽略大小写
-path PATTERN	目录名与模式PATTERN相匹配
-ipath PATTERN	目录名与模式PATTERN相匹配且忽略大小写
-perm MODE	文件的权限与MODE匹配。MODE的形式为+|-nnn，+ 为按位或运算，-为按位与运算
-regex PATTERN	文件的全名与模式PATTERN相匹配
-iregex PATTERN	文件的全名与模式PATTERN相匹配且忽略大小写
-size n[bckw]	文件使用n个单元的空间。b为字节，c为字，k为千字节，w为双字
-type C	文件的类型为C。C的值可为：b块设备文件、c字
-xtype C	
-true	永真
-uid n	文件属于 uid 为n的用户
-user NAME	文件属于名字为NAME的用户。允许使用数字形式的 uid 来代替NAME
-pid n	进程号为n的文件

- 在上述测试中，出现数字n的地方（时间、大小）可以使用+n表示大于n，-n表示小于n，而n则表示正好等于n
- find命令的动作：

-exec CMD \;	在当前目录下执行命令 CMD，返回值
-ok CMD	
-print	输出文件全名至STDOUT
-print0	在输出的文件全名后加一个空字符
-fprint0 FILE	
-fprint FILE	输出文件全名至指定的文件
-printf FORMAT	按给定的格式输出。格式描述字符串
-fprintf FILE FORMAT	

例 1 将当前目录及其子目录下所有扩展名是 c 的文件列出来。

```
[Apple] $ find . -name "*.c"
```

例 2 将当前目录及其下子目录中所有普通文件列出来

```
[Apple] $ find . -type f
```

例 3 将目前目录及其子目录下所有最近 20 分钟内更新过的文件列出

```
[Apple] $ find . -ctime -20
```

例 4 将当前目录及其子目录下所有扩展名不是 c 的文件列出来：

```
[Apple]$ find . -name "*.c" -o print
```

例 5 显示目录的深度及文件的目录名和文件名：

```
[Apple]$ find ch1 -printf "%d      %h/%f\n"
0/ch1
1         ch1/ch1.cpp
1         ch1/ch1.h
1         ch1/ch1_main.cpp
```

例 6 显示所有具有全局可执行权限的文件：

```
[Apple]$ find . -type f -perm -001 -print
./test
./filecounter
./filesplitor
./dtree
```

例 7 使用 find 命令来生成命令行（备份）：

```
[Apple]$ dir2="backup"
[Apple]$ find dir1 -type d -printf \
         "mkdir -p '${dir2}/%P'\n" -o \
         ! -type d -printf \
         "mv '%p' '${dir2}/%p'\n"| sh
```

%P和%p都是输出文件名，%p的输出是从命令行上指定的搜索目录开始，而%P则去掉命令行上指定的目录

```
[Apple]$ tree
.
|-- dir2
|   |-- one.txt
|   `-- two.txt
|-- one.txt
`-- two.txt
[Apple]$ find dir1 -printf "%p\n"
dir1
dir1/one.txt
dir1/two.txt
dir1/dir2
dir1/dir2/one.txt
dir1/dir2/two.txt
[Apple]$ find dir1 -printf "%P\n"
one.txt
two.txt
dir2
dir2/one.txt
dir2/two.txt
```

1.3 info

- 语法: info [OPTIONS] [NODENAMES]
- 说明: info命令启动 GNU 扩展超文本文档系统。系统中每一个小而精的信息被称为一个节点。

1.4 join

- 语法: join [OPTIONS] FILE1 FILE2
- 说明: 将两个文件中拥有相同字段的行合并起来。
- 选项：

-aSIDE	SIDE为 1 或 2 表示FILE1或FILE2。将SIDE侧文件中无法配对的行输出
-eSTRING	用STRING来代表没有的输入字段
-i	字段比较时忽略大小写
-oFIELDLIST	只显示出现在FIELDLIST中的字段。形式为用逗号分隔的SIDE.n
-tCHAR	使用指定的字符作为输入和输出字段分隔符
-vSIDE	仅SIDE侧文件中无法配对的行输出
-1 FIELD	根据FILE1的指定字段来合并行
-2 FIELD	根据FILE2的指定字段来合并行

- 范例：先建立三个示例文本文件：

```
jnames        jsalaries    jhiredt
-----------   ----------   --------------
001 Barry     001 135000   001 04/12/1989
002 Mario     002 80000    002 06/23/1994
003 John      003 45000    003 12/12/1997
004 Harry     004 75000    004 11/17/1997
005 Dorothy   005 90000    005 02/06/1994
006 Jackie    007 68000    006 07/20/1995

[Apple]$ join jnames jsalaries
001 Barry 135000
002 Mario 80000
003 John 45000
004 Harry 75000
005 Dorothy 90000
[Apple]$ join -a1 jnames jsalaries
001 Barry 135000
002 Mario 80000
003 John 45000
004 Harry 75000
005 Dorothy 90000
006 Jackie
[Apple]$ join -a2 jnames jsalaries
001 Barry 135000
002 Mario 80000
003 John 45000
004 Harry 75000
005 Dorothy 90000
007 68000
```

```
[Apple]$ join -o1.2,2.2 jnames jsalaries
Barry 135000
Mario 80000
John 45000
Harry 75000
Dorothy 90000
[Apple]$ join -v1 jnames jsalaries
006 Jackie
[Apple]$ join -v2 jnames jsalaries
007 68000
[Apple]$ join -a1 jnames jsalaries | \
         join - jhiredt
001 Barry 135000 04/12/1989
002 Mario 80000 06/23/1994
003 John 45000 12/12/1997
004 Harry 75000 11/17/1997
005 Dorothy 90000 02/06/1994
006 Jackie 07/20/1995
```

1.5 paste

- 语法: paste [OPTIONS] [FILES]
- 说明: 将多个文件按行合并，在中间加上一个TAB，然后输出到标准输出上。
- 选项：

-s	依次先将第一个文件的所有行联接到一起，然后输出一个回车再处理第二个文件
-d LIST	依次使用LIST中的字符取代TAB作为输出分隔符
-f	遇到第一个EOF 就终止

- 范例：

```
$ cat num2            $ cat let3
1                     a
2                     b
                      c
$ paste num2 let3     $ paste -s num2 let3
1       a             1       2
2       b             a       b       c
        c
$ paste -d '%_' num2 let3 num2
1%a_1
2%b_2
%c_
```

1.6 nl

- 语法: nl [OPTIONS] [FILES]
- 说明: 加上行号后输出文件

1.7 pstree

- 语法: pstree [OPTIONS] [PID|USER]
- 说明: 以树形格式输出进程名，缺省时以init进程为根节点，可以通过指定PID或USER来改变根节点。
- 选项：

-a	显示各进程的命令行。换出的进程显示在括号种
-c	取消压缩等价的进程
-h	高亮当前进程及其子进程
-l	使用加长行显示，避免被屏幕宽度所限
-n	根据进程标识符来排序，缺省时按名称排序
-p	显示进程标识符

- 范例：

```
[Apple]$ pstree -h
init-+-adsl-connect---pppd---pppoe
     |-atd
..............
     |-socks5---2*[socks5]
     |-sshd-+-sshd---bash---ssh
     |      `-sshd---bash---pstree
     |-syslogd
     `-xinetd
```

1.8 sort

- 语法: sort [OPTIONS] [FILES]
- 说明: 将输入文件按行排序、将多个有序文件合并或验证文件是否是有序的。缺省时sort将单个空格作为字段定界符。然后根据最左边的字段对各行排序，如果最左边的字段相同，则检查下一个字段。
- 当给大文件排序时，应保证有足够的临时磁盘空间。
- 选项：

-b	忽略每行开头的空格
-d	以电话簿顺序排序。忽略所有字母、数字和空格之外的字符
-i	忽略 ASCII(32-126 含) 字符集之外的所有字符
-kn[,m]	指定从n到m之前的字段为排序字段
-m	合并两个或更多的有序文件
-n	按数字排序
-o OUTPUT	将排序结果写入OUTPUT文件中
-r	反序
-tC	指定字段分隔符为C
-u	相同的行只输出一次

- 范例：

```
[Apple]$ cat date
1 AC BB CC
2 AB CC DD
3 CA BB CC
5 BE DD EE
4 BD AA AA
[Apple]$ sort date          [Apple]$ sort -k2 date
1 AC BB CC                  2 AB CC DD
2 AB CC DD                  1 AC BB CC
3 CA BB CC                  4 BD AA AA
4 BD AA AA                  5 BE DD EE
5 BE DD EE                  3 CA BB CC

[Apple]$ sort -k2.3 date1 [Apple]$ sort -k2.3r date1
3 CA BB CC                  5 BE DD EE
2 AB CC DD                  4 BD AA AA
1 AC BB CC                  1 AC BB CC
4 BD AA AA                  2 AB CC DD
5 BE DD EE                  3 CA BB CC
[Apple]$ sort -n date1
1 AC BB CC
2 AB CC DD
3 CA BB CC
4 BD AA AA
5 BE DD EE
```

1.9 split

- 语法: split [OPTIONS] [INPUT [PREFIX]]
- 说明: split命令以大小作为划分的标准将文件分割成几个部分，分割后的文件名为 PREFIXaa、PREFIXab、...，缺省的前缀为 x，缺省的后缀长度为 2。
- 选项：

-aN	指定后缀长度为N
-lNUM	将输出文件的大小限制为NUM行
-bNUM[b|k|m]	将输出文件的大小限制为NUM字节，后续的b表示512 字节、k为千字节、m为兆字节
-cNUM[b|k|m]	以整行为单位输出文件并且将输出文件的大小限制为NUM字节/512 字节/千字节/兆字节

1.10 strings

- 语法：strings [OPTIONS] [FILES]
- 说明：此命令用于显示二进制文件中的可打印字符串。缺省时，strings 在文件的初始化小节和已装入小节中寻找所有长度大于 4 个字符的可打印字符序列。
- 选项：

-a	扫描整个文件查找字符串
-f	将文件名加到每个找到的字符串之前
-n LENGTH	指定可打印字符序列的最小长度
-o[o|x|d]	分别用八进制、十六进制和十进制输出每个字符串的位置

1.11 tee

- 语法：tee [OPTIONS] [FILES]
- 说明：T型管道：从STDIN中读取并写到 STDOUT和指定的文件中。
- 选项：

-a	追加写入文件
-i	忽略中断信号

- 范例：

```
[Apple]$ ls -l | tee ls.dat | wc -l
     12
[Apple]$ more ls.dat
total 52
drwxrwxr-x 2 hop hop 4096 Oct 16 21:43 ch1
-rw-rw-r-- 1 hop hop   55 Nov  1 13:27 date1
drwxrwxr-x 3 hop hop 4096 Nov  1 11:33 dir1
-rwxrwxr-x 1 hop hop  195 Oct 31 22:56 dtree
-rwxrwxr-x 1 hop hop 3095 Oct 31 13:30 filecounter
-rwxrwxr-x 1 hop hop 1049 Oct 31 23:03 filesplitor
-rw-rw-r-- 1 hop hop   90 Nov  1 12:15 jhiredt
-rw-rw-r-- 1 hop hop   62 Nov  1 12:14 jnames
-rw-rw-r-- 1 hop hop   61 Nov  1 12:17 jsalaries
-rwxrwxr-x 1 hop hop 1654 Oct 31 12:26 test
-rw-rw-r-- 1 hop hop   33 Nov  1 09:38 test1.txt
```

1.12 tr

- 语法：tr [OPTIONS] CHARSET1 [CHARSET2]
- 说明：对从STDIN中读入的字符进行变换、压缩或删除，然后将结果送至STDOUT，CHARSET1和CHARSET2指定了哪些字符被转换以及它们被转换成什么字符。

- 当CHARSET1和CHARSET2都被指定并且没有使用-d选项时，tr执行转换，每一个与CHARSET1中的字符匹配的输入字符都被转换为CHARSET2中相应的字符。
- tr也可用于将重复出现的字符紧缩为一个字符。
- 选项：

-c	将所有CHARSET1之外的字符转换为CHARSET2中的字符
-d	删除CHARSET1中的字符，不作转换
-s	将连续多个相同的字符紧缩为一个
-t	截断CHARSET1使其与CHARSET2包含同样数目的字符

- 转换时若CHARSET2比CHARSET1长，则忽略多余的字符；反之，则不断重复CHARSET2的最后一个字符直到二者一样长。
- [c*n]表示字符c重复n次，[c*]表示将字符c重复必要的次数以使CHARSET1和CHARSET2长度相等。
- 在CHARSET1和CHARSET2中可使用字符类。
- 范例：

```
[Apple]$ cat tr.dat
ABC DEF GHI
ABC DEF GHI
AAA BBB CCC
[Apple]$ tr "A" "Z" < tr.dat
ZBC DEF GHI
ZBC DEF GHI
ZZZ BBB CCC
[Apple]$ tr "ABC" "[Z*]" < tr.dat
ZZZ DEF GHI
ZZZ DEF GHI
ZZZ ZZZ ZZZ
[Apple]$ tr "ABCD" "YYZZ" < tr.dat
YYZ ZEF GHI
YYZ ZEF GHI
YYY YYY ZZZ
[Apple]$ tr -s "BC" < tr.dat
ABC DEF GHI
ABC DEF GHI
AAA B C
[Apple]$ tr -s "BC" "ZZ" < tr.dat
AZ DEF GHI
AZ DEF GHI
AAA Z Z
[Apple]$ tr -d "\n" < tr.dat ; echo
ABC DEF GHIABC DEF GHIAAA BBB CCC
[Apple]$ tr -c "A" "Z" < tr.dat
AZZZZZZZZZZZAZZZZZZZZZZZZAAAZZZZZZZZZZ[Apple]$
[Apple]$ tr -c "A \n" "Z" < tr.dat
```

```
AZZ ZZZ ZZZ
AZZ ZZZ ZZZ
AAA ZZZ ZZZ
```

1.13 tty

- 语法：tty [OPTIONS]
- 说明：输出连接到STDIN的终端的文件名。
- 选项：

-s	不产生任何输出，只返回一个退出值

1.14 uniq

- 语法：uniq [OPTIONS] [INPUT [OUTPUT]]
- 说明：从已排序的文件中删除重复的行。
- 选项：

-c	在输出每行之前显示该行重复的次数
-d	仅显示重复的行
-fN	在检查行的唯一性前跳过N个字段
-sN	在检查行的唯一性前跳过N个字符
-u	仅显示不重复的行

- 范例：

```
[Apple]$ cat uniq.dat      [Apple]$ uniq uniq.dat
01 12345 67890             01 12345 67890
02 12345 67890             02 12345 67890
03 12345 22222             03 12345 22222
03 12345 22222             04 11111 22222
04 11111 22222
[Apple]$ uniq -d uniq.dat [Apple]$ uniq -f1 -w5 uniq.dat
03 12345 22222             01 12345 67890
                           04 11111 22222
[Apple]$ uniq -u uniq.dat [Apple]$ uniq -f1 uniq.dat
01 12345 67890             01 12345 67890
02 12345 67890             03 12345 22222
04 11111 22222             04 11111 22222
```

1.15 xargs

- 语法：xargs [-0prtx] [-e[eof-str]] [-i[replace-str]] [-l[max-lines]] [-n max-args] [-s max-chars] [-P max-procs] [--null] [--verbose] [--exit] [--max-procs=max-procs] [--version] [--help] [COMMAND [INITIAL-ARGUMENTS]]
- 说明：建立并执行命令行。
- xargs从STDIN读入被空格和新行符分隔的参数，然后以 INITIAL-ARGUMENTS后跟随读入的参数来执行给定的命令（缺省时执行/bin/echo）
- 选项：

-0	输入文件名以空字符结束。取消空格、引号和反斜线的特殊含义
--null	
-e[eof-str]	指定文件结束字符串。缺省为下划线
-i[replace-str]	用从STDIN读入的名字替换指定的字符串
-l[max-lines]	每个命令行最多使用max-lines个非空的输入行，缺省为 1
-n max-args	每个命令行最多使用max-args个参数
-p	提示用户是否执行每个命令行并读入下一输入行
-r	如果STDIN不包含任何非空字符，则不执行命令
-s max-chars	每个命令行最多使用max-chars个字符，包括命令本身、初始参数和标记结束的空字符，缺省上限为 20k 个字符
-t	
--verbose	在命令执行前在STDOUT上输出整个命令行
-x	当-l -n和-s选项指定的值被超出时终止本命令
--exit	
-P max-procs	同时最多运行max-procs个进程，缺省值为 1。若max-procs被指定为 0，则进程数无限制

1.16 示例

统计单词出现频率

对于给定的英文文本文件，统计各个单词出现的频率，并且显示频率最高的 n 个单词及其出现次数。

- 算法步骤：
  1. 将所有大小写英文字母之外的字符转化为一个回车并将多个回车压缩为一个
  2. 将所有大写字母转换为小写
  3. 排序，结果是按字典顺序排好的单词列表，相同的单词挨在一起
  4. 在每个单词之前显示此单词出现的次数
  5. 按照出现的次数降序排序
  6. 取前 n 行输出
  7. 输出时加上序号
- 脚本代码：

```
#!/bin/bash
# Author  : Hop Lee
# Date    : 2002.10
# Purpose : Output the top n words in a given
#           English text file
#
if [ $# -ne 2 -a $# -ne 1 ]; then
  echo "usage: `basename $0' [n] Input_File"
  echo
  exit
fi

if [ $# -eq 1 ]
then
  I_TOP=10
  I_FILE=$1
fi

if [ $# -eq 2 ]
then
  I_TOP=$1
  I_FILE=$2
fi

tr -sc "[A-Z][a-z]" "[\012*]" < $I_FILE | \
tr "[A-Z]" "[a-z]" | \
sort | \
uniq -c | \
sort -k1 -n -r | \
head -$I_TOP | nl
```

2 正则表达式 (Regular Expression)

- 正则表达式是一种记法，她可以让你搜索符合特定规范的文本。
- 我们可以通过构造单个正则表达式来选择、匹配多个字符串。
- 大量的 UNIX 命令使用了正则表达式，其中比较著名的有grep家族、sed、文本处理工具（awk, Perl, Python, Ruby, Tcl等）、文件查看工具（more, page, less等）以及文本编辑器（ed, vi/vim, emacs, jed, jove等）。
- 正则表达式分为两大类：BRE (Basic Regular Expressions) 和 ERE (Extended Regular Expressions).

2.1 正则表达式语法

x?	0 或 1 个 x 字符
x*	0 或 0 个以上 x 字符
.*	0 或 0 个以上任意字符
x+	1 或 1 个以上 x 字符
.+	1 或 1 个以上任意字符

{m}	m 个字符
{m,n}	m 个以上、n 个以下个数的字符
{m,}	m 个以上个数的字符
[]	范围 (若减号在最后，则失去表示区间的意义)
[^]	范围的补集，若^不在开头，则失去其表示补集的特殊意义
^	字符串开头的字符
$	字符串末尾的字符
\d	等价于[0-9]
\d+	等价于[0-9]+
\D	等价于[^0-9]
\D+	等价于[^0-9]+
\w	等价于[a-zA-Z0-9]
\w+	等价于[a-zA-Z0-9]+
\W	等价于[^a-zA-Z0-9]
\W+	等价于[^a-zA-Z0-9]+
\s	等价于[\n\t\r\f]
\s+	等价于[\n\t\r\f]+
\S	等价于[^\n\t\r\f]
\S+	等价于[^\n\t\r\f]+
\b	寻找不以数字和字母为边界的串
\B	寻找以数字和字母为边界的串
a|b|c	含有 a 或 b 或 c 的串
abc	含有 abc 的串
\(pattern\)	依次将找到的串存入标记寄存器 1,2,...,9
/pattern/i	不考虑大小写
\	转义

- 非特殊字符与其自身匹配。
- 在特殊字符前，缀以反斜线，则丧失字符的特殊含义，与其自身匹配。
- 圆点、星号、反斜线在方括号内时，代表它们自己。
- $只有出现在正则表达式最尾部时才有特殊意义，否则与其自身匹配；^只有出现在正则表达式最左边时才有特殊意义，否则与其自身匹配。

2.2 正则表达式运算符

- =~：$string=~/pattern/ 表示对字符串 $string 用模式 /pattern/ 进行匹配运算
- !~：不相匹配运算符
- 单目运算符*、+和?优先级最高，其次是串接符，最后是|。可以用圆括号改变运算的优先级。
- 所有运算都是左结合的。

2.3 常用的函数

tr/srchlist/repllist/	把srchlist转换为repllist
s/pattern/replace/eg	用replace替换pattern

2.4 标记正则表达式

- 标记正则表达式是用\(和\) 括起来的部分。
- 匹配时从左到右扫描并依次将匹配的结果存入19 号寄存器中，可以使用\n 引用寄存器的内容。
- 例：‘\([a-z]\)\([a-z]\)[0-9]\2\1’，这个标记正则表达式将匹配所有由 5 个字符组成、并且中间一位是数字的回文字符串。

2.5 正则表达式示例

- /\b([a-z]+) \1\b/gi 一个单词的连续出现（以空格分隔）；
- /(\w+):\/\/([^/:]+)(:\d*)?([^#]*)/将一个 URL 解析为协议、域、端口及相对路径；
- /^(?:Chapter|Section)[1-9][0-9]{0,1}$/定位章节的位置；
- /[-a-z]/ A 至 z 共 26 个字母再加一个- 号；
- /ter\b/可匹配 chapter，而不能匹配 terminal；
- /\Bapt/可匹配 chapter，而不能匹配 aptitude；
- /Windows(95|98|NT)/可匹配 Windows95 或 Windows98 或 WindowsNT，当找到一个匹配后，从 Windows 后面开始进行下一次的检索匹配；
- ^(?=.*\d)(?=.*[a-z])(?=.*[A-Z]).{8,10}$ 验证强密码。字符数必须在 8 至 10 的范围内。必须包含大小写字母和数字的组合，不能使用特殊字符；
- \d+ 非负整数（正整数和 0）；
- ((-\d+)|(0+)) 非正整数（负整数和 0）；
- [0-9]*[1-9][0-9]* 正整数；
- -[0-9]*[1-9][0-9]* 负整数；
- -?\d+ 整数；
- \d+(\.\d+)? 非负浮点数（正浮点数和 0）；
- ((-\d+(\.\d+)?)|(0+(\.0+)?)) 非正浮点数（负浮点数和 0 ）；
- (([0-9]+\.[0-9]*[1-9][0-9]*)|([0-9]*[1-9][0-9]*\.[0-9]+)|([0-9]*[1-9][0-9]*)) 正浮点数；
- (-(([0-9]+\.[0-9]*[1-9][0-9]*)|([0-9]*[1-9][0-9]*\.[0-9]+)|([0-9]*[1-9][0-9]*))) 负浮点数；
- (-?\d+)(\.\d+)? 浮点数；
- [a-zA-Z][.?!] 表示任意后面跟随句号、问号或感叹号的小写或大写字母；
- [])}]右方括号、右圆括号或右花括号；
- [01][0-9][-/][0-3][0-9][-/][0-9][0-9] 形如 MM-DD-YY 或MM/DD/YY 的日期表示；
- [^aeiou] 任意非元音字母字符；
- 80[234]?86 匹配 “8086”, “80286”, “80386” 或 “80486”

- `(^\... )` 表示 “行首为一个句点，随后是两个字符，然后跟一个空格的字符串”；
- `[0-9]\{3\}-[0-9]\{4\}` 北美地区电话号码；
- `compan(y|ies)` 匹配 “company” 的单数或复数形式；
- `[\u4e00-\u9fa5]` 汉字；
- `[^\x00-\xff]` 任意双字节字符；
- `<[^>]*>` 任意的 HTML 标签；
- `<\(.*\)>.*</\1>|<(.*)/>` 更准确地匹配 HTML 标签；
- `\w+([-+.]\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*` 匹配 Email 地址的正则表达式；
- `([\w-\.]+)@(([0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.)|(([\w-]+\.)+))([a-zA-Z]{2,4}|[0-9]{1,3})(\]?)` 更准确地匹配 Email 地址；
- `http://([\w-]+\.)+[\w-]+(/[\w- ./?%&=]*)?` 匹配 URL；
- `^(http|https|ftp)\://[a-zA-Z0-9\-\.]+\.[a-zA-Z]{2,3}(: [a-zA-Z0-9]*)?/?([a-zA-Z0-9\-\._\?\,\'/\\\+&%\$#\=])*$` 更准确地匹配 URL

3 grep 家族

3.1 grep 家族概述

- grep命令可以在一个文件中搜索指定的字符串。其基本语法为：

  ```
  grep [OPTIONS] [-e] 'PATTERN' [FILENAMES]
  ```

- FILENAMES是一系列用空格格开的文件名，输出结果时会在每个匹配之前显示文件名
- 如果没有FILENAMES参数，则grep命令将扫描标准输入
- 缺省情况下grep将输出匹配指定模式的每一行
- 在PATTERN中如果含有元字符的话必须将其转义，使用' 或 \ 或"
- fgrep命令是grep的一个简化版本，仅用于查找固定的字符串而不允许使用正则表达式。其优点为速度快，且可以指定任意个数的查找字符串。
- egrep命令具有fgrep的一些特征，同时也能使用正则表达式、增加了一些查找技巧，但是不支持grep和ed命令中的标记表达式 (Tag Expression)。

3.2 grep 家族选项总结

仅适用于fgrep ：

-x	严格匹配整行

仅适用于grep ：

-s	隐藏错误信息
-w	仅匹配整个字

仅适用于egrep和fgrep：

-f	匹配指定文件中的字符串

适用于grep，fgrep和egrep：

-h	隐藏文件名的显示
-b	显示匹配行的磁盘块号
-l	仅显示发现匹配的文件名
-c	统计和显示匹配的数量
-n	显示每个匹配行的行号
-v	显示非匹配的行
-I	忽略大小写
-e	匹配后面的以“-”开头的表达式

3.3 grep 家族表达式总结

仅适用于grep和egrep：

\m	转义元字符m
^	行首
$	行尾
.	任意单个字符
[xy^$z]	x,y,^,$,z中的任意一个字符
[^xy^$z]	上式的否定
[a-z]	指定的范围内的任一字符
[^a-z]	上式的否定
r*	零或多个r表达式
rs	匹配r和后续的s（连接）

仅适用于grep：

\(r\)	匹配标记正则表达式
\n	设置为匹配第n个标记表达式（1~9）

仅适用于egrep：

r+	匹配一个或多个r
r?	匹配零个或一个r
r|s	匹配r或s
(r|s)t	匹配rt或st
(r|s)*	匹配零个或多个r|s

3.4 其它 grep 相关命令

agrep 可进行模糊搜索

ngrep 用于搜索网络层数据

pdfgrep 用于搜索pdf文件中的模式

ack 是grep的一个可选替换品，可以对匹配字符串高亮显示，默认递归搜索且能自动忽略无关的文件

sgrep 用于搜索结构化的模式，例如 SGML, C, TeX 等类型的源文件

xgrep 专门针对 XML 文件的搜索工具

3.5 grep 示例

- 示例数据文件：

```
[Apple]$ cat myfile
blueflybird is me.
BLUEFLYBIRD IS ME.
Yes,I'm blueflybird.
of course,Blueflybird is me.
Of Course,BLUEFLYBIRD IS ME.
[BLUEBLUE BLUE by BLUE blue bird bird ]
```

1. 分支、块、原子

```
[Apple]$ egrep "^blueflybird|BLUEFLYBIRD.*E\.$" \
         myfile
blueflybird is me.
BLUEFLYBIRD IS ME.
Of Course,BLUEFLYBIRD IS ME.
```

- 上面的命令包括两个分支，用|分开，意思是查找myfile文件中，以小写blueflybird开头的行，或者是含有大写BLUEFLYBIRD单词并且以E.结尾的行。

2. 匹配行末$

- 如果要查找myfile文件中以e.结尾的行（注意加了转义符\）：

```
[Apple]$ egrep "e\.$" myfile
blueflybird is me.
of course,Blueflybird is me.
```

- 或者匹配以bird.结尾的行（注意$号放在命令后面）：

```
[Apple]$ egrep "bird\.$" myfile
Yes,I'm blueflybird.
```

3. 匹配行首^

- 如果要查找myfile文件中以小写b开头的行：

```
[Apple]$ egrep "^b" myfile
blueflybird is me.
```

- 如想查找以小写b或者大写B开始的行：

```
[Apple]$ egrep "^[bB]" myfile
blueflybird is me.
BLUEFLYBIRD IS ME.
```

4. 匹配任一字符.

   - .通常与其它元字符联合使用，因为.代表任一字符，如下面的命令表示查找以任一字符开头，第二个字符是f的行：

```
[Apple]$ egrep "^.f" myfile
of course,Blueflybird is me.
Of Course,BLUEFLYBIRD IS ME.
```

5. 匹配[]内列出的字符

   - 查找包括有小写of与大写Of的行：

```
[Apple]$ egrep "[oO]f" myfile
of course,Blueflybird is me.
Of Course,BLUEFLYBIRD IS ME.
```

   - 查找不包括小写of与大写Of的行

```
[Apple]$ egrep "[^oO]f" myfile
blueflybird is me.
Yes,I'm blueflybird.
of course,Blueflybird is me.
```

   - 查找包括符号]和大写Y的行

```
[Apple]$ egrep "[]Y]" myfile
BLUEFLYBIRD IS ME.
Yes,I'm blueflybird.
Of Course,BLUEFLYBIRD IS ME.
[BLUEBLUE BLUE by BLUE blue bird bird ]
```

   - 查找所有以大写字母开始的行

```
[Apple]$ egrep "^[A-Z]" myfile
BLUEFLYBIRD IS ME.
Yes,I'm blueflybird.
Of Course,BLUEFLYBIRD IS ME.
```

6. \<完整单词\>

   - 这里的完整单词并不是说真实的英语单词，而是指连续的字母组合中不含有数字、空格、换行符及制表符，如“myfile1”中完整单词是myfile，同样“my name1eye”中，my, name, eye各自是完整单词，而“bbeekks”无序的组合也是完整单词

7. 匹配次数{}、+、?、*

- 为了继续说明匹配次数，我们新建一个文件叫myfile1，内容如下：

```
[Apple]$ cat myfile1
blue
blueblue
bluebluebluе
BLUE
BLUEBLUE
BLUEBLUEBLUEBLUE
BBLLUUEE
BLUEEE
```

- 如果我要查找两个（精确的两个）连续BLUE相连的行

```
[Apple]$ egrep "(BLUE){2}" myfile1
BLUEBLUE
BLUEBLUEBLUEBLUE
```

- 注意:为了查找BLUEBLUE,用RE表示则是(BLUE){2},而不是BLUE{2},如果是BLUE{2},则表示BLUEE
- 警告：本来要查BLUEBLUE，为甚麼上面还列出了三个连续的BLUE 呢？因为BLUEBLUEBLUE包括了BLUEBLUE
- 如果想查找BLUEBLUE那一行，则必须使用定位符，如：

```
[Apple]$ egrep "^(BLUE){2}\>" myfile1
BLUEBLUE
```

- 上面的例子查找以两个连续BLUE开始的完整单词的行
- 如要查找以两个或者两个以上连续BLUE开始的完整单词，则在2后面加上逗号：

```
[Apple]$ egrep "^(BLUE){2,}\>" myfile1
BLUEBLUE
BLUEBLUEBLUEBLUE
```

- 如查找以一至三个连续BLUE开始的完整单词的行：

```
[Apple]$ egrep "(BLUE){1,3}" myfile1
BLUE
BLUEBLUE
BLUEBLUEBLUEBLUE
BLUEEE
```

- 如果要查找以至少 1 个或多个BLUE开始的完整单词行：

```
[Apple]$ egrep "^(BLUE)+\>" myfile1
BLUE
BLUEBLUE
BLUEBLUEBLUEBLUE
```

- 如果要查找以 0 个或多个BLUE开始的完整单词行：

```
[Apple]$ egrep "^(BLUE)*\>" myfile1
BLUE
BLUEBLUE
BLUEBLUEBLUEBLUE
```

- 如果要查找 0 个或 1 个以BLUE开始的完整单词行：

```
[Apple]$ egrep "^(BLUE)?\>" myfile1
BLUE
```

- 非常值得注意的是：上面的命令(BLUE)?\>中增加了\>是否完整单词的测试，如果去掉\>，那么：

```
[Apple]$ egrep "^(BLUE)?" myfile1
blue
blueblue
blueblueblue
BLUE
BLUEBLUE
BLUEBLUEBLUEBLUE
BBLLUUEE
BLUEEE
```

- 由于^(BLUE)?表示以 0 个或 1 个BLUE开始的行，没有测试BLUE是否是完整单词，所以可以匹配任何行，而加了完整单词测试的命令，则只能匹配有 1 个BLUE开始并是完整单词的行匹配。

4 附录：特殊符号的名字

符号	英文名	中文名
~	tilde	波浪号
!	exclamation	感叹号
@	at	at
#	number	井号
$	dollar	美元符
%	percent	百分号
^	caret	control 符
&	ampersand or and	and 符
*	asterisk	星号
⋆	star	星号
+	plus	加号
−	minus	减号
-	hyphen	连字符
_	underline	下划线
—	dash	破折号
'	quote or apostrophe	单引号
"	double quote	双引号
`	back quote	反引号

.	period	英文句号
,	comma	逗号
:	colon	冒号
;	semicolon	分号
[	left bracket	左方括号
]	right bracket	右方括号
(	left parenthesis	左圆括号
)	right parenthesis	右圆括号
{	left curly bracket, or left brace	左花括号
}	right brace	右花括号
《》	French quotes	书名号
...	ellipsis	省略号
|	pipe or or	管道线
‖	parallel	平行
/	virgule, or slash	斜线
\	backslash	反斜线
?	question mark	问号
=	equal	等号
≡	identical	恒等
≅	congruent	全等

The End of Chapter VI.