第十二章、正规表示法与文件格式化处理

1、正规表示法是处理字符串的方法
2、用途：在服务器上删除垃圾广告信件
3、正规表示法也需要支持工具程序来辅助，比如 grep
4、语系对正规表示法的影响：字符和数字都是通过编码表转换来的，不同语系有不同编码表，比如
LANG=C 时：0 1 2 3 4 ... A B C D ... Z a b c d ...z
LANG=zh_TW 时：0 1 2 3 4 ... a A b B c C d D ... z Z
LANG=C，[A-Z]　只有大写英文字母
LANG=zh_TW，[A-Z] 里大小写英文字母都包括进去了
我们一般用的是 C 这个语系
5、特殊符号的代表意义
[:alnum:]　代表英文大小写字符及数字，即 0-9, A-Z, a-z
[:alpha:]　代表任何英文大小写字符，即 A-Z, a-z
[:upper:]　代表大写字符，即 A-Z
[:lower:]　代表小写字符，即 a-z
[:digit:]　代表数字而已，即 0-9

6、grep 的进阶用法：
grep [-A] [-B] [--color=auto] '要搜寻的字符串' filename
-A ：后面可加数字，为 after 的意思，除了列出该行外，后续的 n 行也列出来；
-B ：就是前面的 n 行
--color=auto 可将要搜寻的字符串用颜色标注出来

```
[root@localhost ~]# dmesg | grep 'eth'
[    2.014595] e1000 0000:02:01.0 eth0: (PCI:66MHz:32-bit) 00:0c:29:1f:6b:af
[    2.014606] e1000 0000:02:01.0 eth0: Intel(R) PRO/1000 Network Connection
```

加上行号和颜色（奇怪，默认就标注颜色的吗）

```
[root@localhost ~]# dmesg | grep -n --color=auto 'eth'
1749:[    2.014595] e1000 0000:02:01.0 eth0: (PCI:66MHz:32-bit) 00:0c:29:1f:6b:af
1750:[    2.014606] e1000 0000:02:01.0 eth0: Intel(R) PRO/1000 Network Connection
```

同时显示出前两行和后三行

```
[root@localhost ~]# dmesg | grep -n -A3 -B2 --color=auto 'eth'
1747-[    1.903866] sr 2:0:0:0: Attached scsi CD-ROM sr0
1748-[    1.997620] random: fast init done
1749:[    2.014595] e1000 0000:02:01.0 eth0: (PCI:66MHz:32-bit) 00:0c:29:1f:6b:af
1750:[    2.014606] e1000 0000:02:01.0 eth0: Intel(R) PRO/1000 Network Connection
1751-[    2.815637] random: crng init done
1752-[    3.099134] EXT4-fs (dm-0): mounting ext3 file system using the ext4 subsyste
1753-[    3.110112] EXT4-fs (dm-0): mounted filesystem with ordered data mode. Opts:
```

找到前面困惑的解答了，原来默认设定了 grep --color=auto 的命令别名为 grep

```
[root@localhost ~]# alias
alias cp='cp -i'
alias egrep='egrep --color=auto'
alias fgrep='fgrep --color=auto'
alias grep='grep --color=auto'
alias l.='ls -d .* --color=auto'
alias ll='ls -l --color=auto'
alias ls='ls --color=auto'
alias mv='mv -i'
alias rm='rm -i'
alias which='alias | /usr/bin/which --
```

1、搜寻字符串，比如 the

```
[root@localhost ~]# grep -n 'the' regular_express.txt
8:I can't finish the test.^M
12:the symbol '*' is represented as start.
15:You are the best is mean you are the no. 1.
16:The world <Happy> is the same with "glad".
18:google is the best tools for search keyword.
```

2、反向选择，即没有“the”这个字符串的行，加上参数-v 即可

```
[root@localhost ~]# grep -vn 'the' regular_express.txt
1:"Open Source" is a good mechanism to develop programs
2:apple is my favorite food.
3:Football game is not use feet only.
4:this dress doesn't fit me.
5:However, this dress is about $ 3183 dollars.^M
6:GNU is free air not free beer.^M
```

3、还是搜寻 the，这次无论大小写（加上参数-i）

```
[root@localhost ~]# grep -in 'the' regular_express.txt
8:I can't finish the test.^M
9:Oh! The soup taste good.^M
12:the symbol '*' is represented as start.
14:The gd software is a library for drafting programs.^
15:You are the best is mean you are the no. 1.
```

4、利用[]来搜寻集合字符，比如想搜寻 test 和 taste，由于很相似，所以可以这样

```
[root@localhost ~]# grep -n 't[ae]st' regular_express.txt
8:I can't finish the test.^M
9:Oh! The soup taste good.^M
```

5、如果想要搜寻 oo 字符串同时不影响它前面有字母 g 的话：

```
[root@localhost ~]# grep -n '[^g]oo' regular_express.txt
2:apple is my favorite food.
3:Football game is not use feet only.
18:google is the best tools for search keyword.
19:gooooooogle yes!
```

6、如果不想 oo 前面是小写字母，则这个需要搜寻的字符串可以写成 ‘[^a-z]oo’

不想有数字，写成 ‘[^0-9]oo’

以上两句话在 zh_TW 语系里不适用，而普遍通用的写法是：

‘[^[:lower:]]oo’

‘[^[:digit:]]oo’

7、行首字符 ^

^ 符号，在字符集合符号(括号[])内与外是不同的！ 在 [] 内代表『反向选择』，在[]外则代

表定位在行首的意义

找出行首为 the 的那些行

```
[root@localhost ~]# grep -n '^the' regular_express.txt
12:the symbol '*' is represented as start.
```

找出行首为小写字母的那些行

```
[root@localhost ~]# grep -n '^[:lower:]' regular_express.txt
grep: 字符类的语法是 [[:space:]],而非 [:space:]
[root@localhost ~]# grep -n '^[[:lower:]]' regular_express.txt
2:apple is my favorite food.
4:this dress doesn't fit me.
10:motorcycle is cheap than car.
12:the symbol '*' is represented as start.
18:google is the best tools for search keyword.
19:gooooooogle yes!
20:go! go! Let's go.
```

找出行首不是英文字母的那些行：

```
[root@localhost ~]# grep -n '^[^[:alpha:]]' regular_express.txt
1:"Open Source" is a good mechanism to develop programs.
21:# I am VBird
```

8、行尾字符$

搜寻结尾为“.”的那些行（哈哈，这些红色的小点），这里之所以加上\是因为小数点是特殊符号，需要转义符

```
[root@localhost ~]# grep -n '\.$' regular_express.txt
1:"Open Source" is a good mechanism to develop programs.
2:apple is my favorite food.
3:Football game is not use feet only.
4:this dress doesn't fit me.
10:motorcycle is cheap than car.
11:This window is clear.
```

搜寻空行

grep -n ‘^$’ re....txt

如果想不显示空行和以 ‘#’ 开头的行，则可以：

grep -v '^$' /etc/syslog.conf | grep -v '^#'

9、 . (小数点)：代表『一定有一个任意字符』的意思；

* (星星号)：代表『重复前一个 字符 到无穷多次』的意思，为组合形态

如果想找出 g??d 的字符串：

```
[root@localhost ~]# grep -n 'g..d' regular_express.txt
1:"Open Source" is a good mechanism to develop programs.
9:Oh! The soup taste good.
16:The world <Happy> is the same with "glad".
```

如果想找出列有 oo,ooo,oooo 等数据（即含有两个及更多 o 的字符串）：

```
[root@localhost ~]# grep -n 'ooo*' regular_express.txt
1:"Open Source" is a good mechanism to develop programs.
2:apple is my favorite food.
3:Football game is not use feet only.
9:Oh! The soup taste good.
18:google is the best tools for search keyword.
19:gooooooogle yes!
```

这里用到的是 ‘ooo*’ ，因为*前的那个 o 的意义是告诉*，后面接任意数量的 o，即至少有两个 o

那么如果想要字符串开头结尾都为 g，中间至少有一个 o 呢？

```
[root@localhost ~]# grep -n 'goo*g' regular_express.txt
18:google is the best tools for search keyword.
19:gooooooogle yes!
```

那如果是 g 作为开头结尾，中间有任意字符或者没字符都可以呢？

```
[root@localhost ~]# grep -n 'g.*g' regular_express.txt
1:"Open Source" is a good mechanism to develop programs.
14:The gd software is a library for drafting programs.
18:google is the best tools for search keyword.
19:gooooooogle yes!
20:go! go! Let's go.
[root@localhost ~]#
```

这里.*就是任意数量的任意字符的意思

10、找出含有 oo 的字符串

```
[root@localhost ~]# grep -n 'o\{2\}' regular_express.txt
1:"Open Source" is a good mechanism to develop programs.
2:apple is my favorite food.
3:Football game is not use feet only.
9:Oh! The soup taste good.
18:google is the best tools for search keyword.
19:gooooooogle yes!
```

限制 g　g 之间有 2 到 5 个 o：

```
[root@localhost ~]# grep -n 'go\{2,5\}g' regular_express.txt
18:google is the best tools for search keyword.
```

限制 g　g 之间有两个以上的 o

```
[root@localhost ~]# grep -n 'go\{2,\}g' regular_express.txt
18:google is the best tools for search keyword.
19:gooooooogle yes!
```

11、汇总

^word　意义：待搜寻的字符串(word)在行首！

word$　意义：待搜寻的字符串(word)在行尾！

.　　意义：代表『一定有一个任意字符』的字符！

\　　意义：跳脱字符，将特殊符号的特殊意义去除！

*　　意义：重复零个到无穷多个的前一个 RE 字符

[list]　意义：字符集合的 RE 字符，里面列出想要截取的字符！

[n1-n2]　意义：字符集合的 RE 字符，里面列出想要截取的字符范围！

[^list]　意义：字符集合的 RE 字符，里面列出不要的字符串或范围！

\{n,m\}　意义：连续 n 到 m 个的『前一个 RE 字符』

12、在不支持正规表示法的 ls 这个工具中，使用 ls -l *表示列出任意文件名的文件，而 ls -l a*　则表示列出以 a 开头的任意文件。

但在正规表示法中，要找到以 a 开头的文档，必须搭配支持正规表示法的工具：

ls | grep -n ‘^a.*’

13、sed

以行为单位的新增/删除功能

删除 2~5 行（当然只是显示的时候删除，并不删除文件内容）

```
[root@localhost ~]# cat -n /etc/passwd | sed '2,5d'
     1  root:x:0:0:root:/root:/bin/bash
     6  sync:x:5:0:sync:/sbin:/bin/sync
     7  shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
     8  halt:x:7:0:halt:/sbin:/sbin/halt
     9  mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
    10  operator:x:11:0:operator:/root:/sbin/nologin
```

删除 3 到最后一行：

cat -n /etc/passwd | sed '3,$d'

在第二行后面（即加进去变成第三行）新增字符串：

```
[root@localhost ~]# cat -n /etc/passwd | sed '2a The Dark Knight Rise'
     1  root:x:0:0:root:/root:/bin/bash
     2  bin:x:1:1:bin:/bin:/sbin/nologin
The Dark Knight Rise
     3  daemon:x:2:2:daemon:/sbin:/sbin/nologin
     4  adm:x:3:4:adm:/var/adm:/sbin/nologin
```

加在第二行前：

```
[root@localhost ~]# cat -n /etc/passwd | sed '2i The Dark Knight Rise'
     1  root:x:0:0:root:/root:/bin/bash
The Dark Knight Rise
     2  bin:x:1:1:bin:/bin:/sbin/nologin
     3  daemon:x:2:2:daemon:/sbin:/sbin/nologin
```

如果要加两行以上，则每一行用\隔开

```
[root@localhost ~]# cat -n /etc/passwd | sed '2i The Dark Knight\
> Rise'
     1  root:x:0:0:root:/root:/bin/bash
The Dark Knight
Rise
     2  bin:x:1:1:bin:/bin:/sbin/nologin
```

以行为单位取代与显示：

将第 2-5 行的内容替代成为『No 2-5 number』

```
[root@localhost ~]# cat -n /etc/passwd | sed '2,5c No 2-5 Number'
     1  root:x:0:0:root:/root:/bin/bash
No 2-5 Number
     6  sync:x:5:0:sync:/sbin:/bin/sync
```

显示第 5-7 行

```
[root@localhost ~]# cat -n /etc/passwd | sed -n '5,7p'
     5  lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
     6  sync:x:5:0:sync:/sbin:/bin/sync
     7  shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
```

此外 sed 还有很多功能，像部分数据的搜寻并取代、直接修改文档内容等，就不谈了

延伸型正规表示法也不管了

14、文件的格式化与相关处理

格式化打印 printf

```
[root@localhost ~]# printf '%s\t %s\t %s\t %s\t %s\t \n' $(cat printf.txt)
Name     Chinese         English         Math    Average
DmTsai   80      60      92      77.33
VBird    75      55      80      70.00
Ken      60      90      70      73.33
```

格式是 string Tab string Tab string Tab string Tab string Tab 换行符

```
[root@localhost ~]# last -n 5
root     pts/2        192.168.142.1    Tue Mar 27 19:17   still logged in
root     pts/1        192.168.142.1    Tue Mar 27 18:29   still logged in
root     pts/0        192.168.142.1    Tue Mar 27 14:40   still logged in
root     pts/1        192.168.142.1    Tue Mar 27 12:10 - 16:12  (04:02)
root     pts/0        192.168.142.1    Tue Mar 27 10:57 - 14:33  (03:35)
```

以空格或 Tab 将字段数据隔开，取出第一栏和第三栏

```
[root@localhost ~]# last -n 5 | awk '{print $1 "\t" $3}'
root    192.168.142.1
root    192.168.142.1
root    192.168.142.1
root    192.168.142.1
root    192.168.142.1

wtmp    Wed
```

$0 指的是一整行

awk 的内置变量

变量名称	代表意义
NF	每一行 ($0) 拥有的字段总数
NR	目前 awk 所处理的是『第几行』数据
FS	目前的分隔字符，默认是空格键

last -n 5 列出每一行的账号、当前的行数、该行有多少字段

```
[root@localhost ~]# last -n 5 | awk '{print $1 "\tlines:" NR "\tpartitions:" NF}'
root    lines:1 partitions:10
root    lines:2 partitions:10
root    lines:3 partitions:10
root    lines:4 partitions:10
root    lines:5 partitions:10
        lines:6 partitions:0
```

要查阅 passwd 的第三栏小于 10 以下的数据，并且仅列出账号与第三栏

```
[root@localhost ~]# cat /etc/passwd | awk '{FS=":"} $3 < 10 {print $1 "\t" $3}'
root:x:0:0:root:/root:/bin/bash
bin     1
daemon  2
adm     3
lp      4
sync    5
shutdown        6
halt    7
mail    8
```

实话说这个操作太复杂了
还有更复杂的，什么计算总额都有，哇，跳过了

diff 对比文档之间的差异（以行为单位），一般用在 ASCII 纯文本文档上，此外 cmp 能对比非存文本文档

```
[root@localhost ~]# mkdir -p /tmp/test
[root@localhost ~]# cd /tmp/test
[root@localhost test]# cp /etc/passwd passwd.old
[root@localhost test]# cat /etc/passwd | sed -e '4d' -e '6c no six line' > passwd.new
[root@localhost test]# diff passwd.old passwd.new
4d3
< adm:x:3:4:adm:/var/adm:/sbin/nologin
6c5
< sync:x:5:0:sync:/sbin:/bin/sync
---
> no six line
```

用 sed 来修改生成新的 passwd 文件（这个上面省略了没看，其实没必要这么麻烦用这个命令改吧，用 vi 就很好啊）。

diff [-bBi] from-file to-file

对比两个文档，

4d3 意思是左边文档的第四行被删除了，基准是右边的第三行

< adm:x:3:4:adm:/var/adm:/sbin/nologin 意思是，这是左边被删除的那一行

6c5 意思是左边文档的第六航被取代成右边文档的第五行

< sync:x:5:0:sync:/sbin:/bin/sync 意思是左边(<)文档第六行的内容

>no six lin 是右边文档第五行的内容

diff 还能对比两个不同的目录

```
[root@localhost test]# diff /etc /home
只在 /etc 存在：abrt
只在 /etc 存在：adjtime
只在 /home 存在：alex
只在 /etc 存在：aliases
```

diff 以行为单位对比，cmp 以字节为单位对比（多用于对比文档）

cmp 预设只输出第一个找到的不同点

```
[root@localhost test]# cmp passwd.old passwd.new
passwd.old passwd.new 不同：第 106 字节，第 4 行
```

patch

制作补丁文件

```
[root@localhost test]# diff passwd.old passwd.new > passwd.patch
[root@localhost test]# cat passwd.patch
4d3
< adm:x:3:4:adm:/var/adm:/sbin/nologin
6c5
< sync:x:5:0:sync:/sbin:/bin/sync
---
> no six line
```

如果要用这个补丁文件更新旧版文件（old），可以这样：

patch -pN < patch_file <==更新

patch -R -pN < patch_file <==还原

-p ：后面可以接『取消几层目录』的意思，一般接 0

-R ：将新文件还原成旧版本

然后就出现了这个问题，原因是上面漏了参数

```
[root@localhost test]# patch -p0 < passwd.patch
patch: **** Only garbage was found in the patch input.
[root@localhost test]# cat passwd.patch
```

现在好了

```
[root@localhost test]# diff -Naur passwd.old passwd.new > passwd.patch
[root@localhost test]# patch -p0 < passwd.patch
patching file passwd.old
[root@localhost test]# diff passwd.old passwd.new
[root@localhost test]#
```

复原旧版文档：

```
[root@localhost test]# patch -R -p0 <passwd.patch
patching file passwd.old
[root@localhost test]# ll
总用量 12
-rw-r--r--. 1 root root 2232 3月  27 23:19 passwd.new
-rw-r--r--. 1 root root 2289 3月  27 23:53 passwd.old
-rw-r--r--. 1 root root  480 3月  27 23:51 passwd.patch
[root@localhost test]#
```

为什么这里会使用 -p0 呢？因为我们在比对新旧版的数据时是在同一个目录下， 因此不需要减去目录

习题：

1、利用正规表示法找出系统中含有某些特殊关键词的档案，举例来说，找出在 /etc 底下含有星号 (*) 的档案与内容：

```
[root@localhost ~]# grep '\*' /etc/*
grep: /etc/abrt: 是一个目录
匹配到二进制文件 /etc/aliases.db
grep: /etc/alsa: 是一个目录
grep: /etc/alternatives: 是一个目录
grep: /etc/audisp: 是一个目录
grep: /etc/audit: 是一个目录
/etc/auto.master:# Include /etc/auto.master.d/*.autofs
grep: /etc/auto.master.d: 是一个目录
grep: /etc/avahi: 是一个目录
grep: /etc/bash_completion.d: 是一个目录
/etc/bashrc:    xterm*|vte*)
/etc/bashrc:          PROMPT_COMMAND='printf "\033]0;%s@%s:%s
ME/~}"'
/etc/bashrc:    screen*)
/etc/bashrc:          PROMPT_COMMAND='printf "\033k%s@%s:%s\0
ME/~}"'
/etc/bashrc:    *)
```

但以上命令也只是能搜寻/etc 目录下第一层子目录的数据，如果要包含次目录数据，就要 grep '*' $(find /etc -type f)

2、找出在 /etc 底下，只要含有 XYZ 三个字符的任何一个字符的那一行：

grep [XYZ] /etc/*

3、将 /etc/termcap 内容取后，(1)去除开头为 # 的行 (2)去除空白行 (3)取出开头为英文字母的那几行 (4)最终统计总行数该如何进？

```
grep -v ‘^#’ /etc/termcap | grep -v ‘$’ | grep ‘^[[:alpha:]]’ |wc -l
```