第8章 grep 家 族

相信grep是UNIX和LINUX中使用最广泛的命令之一。 grep（全局正则表达式版本）允许对文本文件进行模式查找。如果找到匹配模式， grep打印包含模式的所有行。 grep支持基本正则表达式，也支持其扩展集。 grep有三种变形，即：

Grep：标准grep命令，本章大部分篇幅集中讨论此格式。

Egrep：扩展grep，支持基本及扩展的正则表达式，但不支持 \q模式范围的应用，与之相对应的一些更加规范的模式，这里也不予讨论。

Fgrep：快速grep。允许查找字符串而不是一个模式。不要误解单词 fast，实际上它与grep速度相当。

在本章中我们将讨论：

- grep（参数）选项。
- 匹配grep的一般模式。
- 只匹配字母或数字，或两者混用。
- 匹配字符串范围。

实际上应该只有一个grep命令，但不幸的是没有一种简单形式能够统一处理 grep的三种变形，将之合而为一，并保持 grep单模式处理时的速度。 GNU grep虽然在融合三种变形上迈进了一大步，但仍不能区分元字符的基本集和扩展集。上一章只讨论了基本的正则表达式，但在查看grep时也涉及到一些扩展模式的匹配操作。然而，首先还是先讨论一下在 grep和fgrep及egrep中均可使用的grep模式吧。

开始讨论之前，先生成一个文件，插入一段文本，并在每列后加入 <Tab>键，grep命令示例中绝大多数将以此为例，其命名为 data..f。生成一个文件，但不知其含义，将是一件很枯燥的事。那么先来看看data.f的记录结构。

第1列：城市位置编号。

第2列：月份。

第3列：存储代码及出库年份。

第4列：产品代号。

第5列：产品统一标价。

第6列：标识号。

第7列：合格数量。

```
$ pg data.f
48   Dec   3BC1997    LPSX   68.00 LVX2A 138
483  Sept  5AP1996    USP    65.00 LVX2C 189
47   Oct   3ZL1998    LPSX   43.00 KVM9D 512
219  dec   2CC1999    CAD    23.00 PLV2C 68
484  nov   7PL1996    CAD    49.00 PLV2C 234
483  may   5PA1998    USP    37.00 KVM9D 644
216  sept  3ZL1998    USP    86.00 KVM9E 234
```

8.1 grep

grep一般格式为：

grep [选项]基本正则表达式[文件]

这里基本正则表达式可为字符串。

8.1.1 双引号引用

在grep命令中输入字符串参数时，最好将其用双引号括起来。例如：“ mystring ”。这样做有两个原因，一是以防被误解为 shell命令，二是可以用来查找多个单词组成的字符串，例如：“ jet plane ”，如果不用双引号将其括起来，那么单词 plane将被误认为是一个文件，查询结果将返回“ 文件不存在 ”的错误信息。

在调用变量时，也应该使用双引号，诸如： grep“ $MYVAR ”文件名，如果不这样，将没有返回结果。

在调用模式匹配时，应使用单引号。

8.1.2 grep选项

常用的grep选项有：

-c 只输出匹配行的计数。

-i 不区分大小写（只适用于单字符）。

-h 查询多文件时不显示文件名。

-l 查询多文件时只输出包含匹配字符的文件名。

-n 显示匹配行及行号。

-s 不显示不存在或无匹配文本的错误信息。

-v 显示不包含匹配文本的所有行。

8.1.3 查询多个文件

如果要在当前目录下所有.doc文件中查找字符串“ sort ”，方法如下：

```
$ grep "sort"*.doc
```

或在所有文件中查询单词“ sort it ”

```
$ grep "sort it" *
```

现在讲述在文本文件中grep选项的用法。

8.1.4 行匹配

```
$ grep -c "48"data.f
$4
```

grep返回数字4，意义是有4行包含字符串“ 48 ”。

现在显示包含“ 48 ”字符串的4行文本：

```
$ grep "48" data.f
48      Dec     3BC1997  LPSX    68.00   LVX2A   138
483     Sept    5AP1996  USP     65.00   LVX2C   189
484     nov     7PL1996  CAD     49.00   PLV2C   234
```

```
483     may     5PA1998  USP     37.00   KVM9D   644
```

8.1.5 行数

显示满足匹配模式的所有行行数：

```
$ grep -n "48" data.f
1:48    Dec     3BC1997  LPSX    68.00   LVX2A   138
2:483   Sept    5AP1996  USP     65.00   LVX2C   189
5:484   Nov     7PL1996  CAD     49.00   PLV2C   234
6:483   May     5PA1998  USP     37.00   KVM9D   644
```

行数在输出第一列，后跟包含48的每一匹配行。

8.1.6 显示非匹配行

显示所有不包含48的各行：

```
$ grep -v "48" data.f
47      Oct     3ZL1998  LPSX    43.00   KVM9D   512
219     Dec     2CC1999  CAD     23.00   PLV2C    68
216     Sept    3ZL1998  USP     86.00   KVM9E   234
```

8.1.7 精确匹配

可能大家已注意到，在上一例中，抽取字符串“ 48 ”，返回结果包含诸如484和483等包含“ 48 ”的其他字符串，实际上应精确抽取只包含 48的各行。注意在每个匹配模式中抽取字符串后有一个<Tab>键，所以应操作如下：

```
$ grep "48<tab>" data.f
48      Dec     3BC1997  LPSX    68.00   LVX2A   138
```

<Tab>表示点击tab键。

使用grep抽取精确匹配的一种更有效方式是在抽取字符串后加 \>。假定现在精确抽取 48，方法如下：

```
$ grep '48\>' data.f
48      Dec     3BC1997  LPSX    68.00   LVX2A   138
```

8.1.8 大小写敏感

缺省情况下，grep是大小写敏感的，如要查询大小写不敏感字符串，必须使用 -i开关。在data.f文件中有月份字符 Sept，既有大写也有小写，要取得此字符串大小写不敏感查询，方法如下：

```
$ grep -i "sept" data.f
483     Sept    5AP1996  USP     65.00   LVX2C   189
216     sept    3ZL1998  USP     86.00   KVM9E   234
```

8.2 grep和正则表达式

使用正则表达式使模式匹配加入一些规则，因此可以在抽取信息中加入更多选择。使用正则表达式时最好用单引号括起来，这样可以防止 grep中使用的专有模式与一些 shell命令的特殊方式相混淆。

8.2.1 模式范围

假定要抽取代码为484和483的城市位置，上一章中讲到可以使用 []来指定字符串范围，这里用48开始，以3或4结尾，这样抽出484或483。

```
$ grep '48[34]' data.f
483     Sept    5AP1996  USP     65.00   LVX2C   189
484     nov     7PL1996  CAD     49.00   PLV2C   234
483     may     5PA1998  USP     37.00   KVM9D   644
```

8.2.2 不匹配行首

如果要抽出记录，使其行首不是48，可以在方括号中使用^记号，表明查询在行首开始。

```
$ grep '^[^48]' data.f
219     dec     2CC1999 CAD     23.00   PLV2C    68
216     sept    3ZL1998 USP     86.00   KVM9E    234
```

8.2.3 设置大小写

使用-i开关可以屏蔽月份Sept的大小写敏感，也可以用另一种方式。这里使用 []模式抽取各行包含Sept和sept的所有信息。

```
$ grep '[Ss]ept' data.f
483     Sept    5AP1996 USP     65.00   LVX2C    189
216     sept    3ZL1998 USP     86.00   KVM9E    234
```

如果要抽取包含Sept的所有月份，不管其大小写，并且此行包含字符串 483，可以使用管道命令，即符号“|”左边命令的输出作为“|”右边命令的输入。举例如下：

```
$ grep '[Ss]ept' data.f | grep 483
483     Sept    5AP1996  USP     65.00   LVX2C    189
```

不必将文件名放在第二个grep命令中，因为其输入信息来自于第一个 grep命令的输出。

8.2.4 匹配任意字符

如果抽取以L开头，以D结尾的所有代码，可使用下述方法，因为已知代码长度为 5个字符：

```
$ grep 'K...D' data.f
47      Oct     3ZL1998  LPSX    43.00   KVM9D    512
483     may     5PA1998  USP     37.00   KVM9D    644
```

将上述代码做轻微改变，头两个是大写字母，中间两个任意，并以 C结尾：

```
$ grep '[A-Z][A-Z]..C' data.f
483     Sept    5AP1996  USP     65.00   LVX2C    189
219     dec     2CC1999  CAD     23.00   PLV2C     68
484     nov     7PL1996  CAD     49.00   PLV2C    234
```

8.2.5 日期查询

一个常用的查询模式是日期查询。先查询所有以 5开始以1996或1998结尾的所有记录。使用模式5..199[6,8]。这意味着第一个字符为5，后跟两个点，接着是199，剩余两个数字是6或8。

```
$ grep '5..199[6,8]' data.f
483     Sept    5AP1996  USP     65.00   LVX2C    189
483     may     5PA1998  USP     37.00   KVM9D    644
```

查询包含1998的所有记录的另外一种方法是使用表达式 [0-9]\{3\}[8]，含义是任意数字重复3次，后跟数字8，虽然这个方法不像上一个方法那么精确，但也有一定作用。

```
$ grep '[0-9]\ {3\}[8]' data.f
47      Oct     3ZL1998  LPSX    43.00   KVM9D    512
483     may     5PA1998  USP     37.00   KVM9D    644
216     sept    3ZL1998  USP     86.00   KVM9E    234
```

8.2.6 范围组合

必须学会使用[]抽取信息。假定要取得城市代码，第一个字符为任意字符，第二个字符在0到5之间，第三个字符在0到6之间，使用下列模式即可实现。

```
$  grep '[0-9][0-5][0-6]' data.f
48      Dec     3BC1997  LPSX    68.00   LVX2A    138
483     Sept    5AP1996  USP     65.00   LVX2C    189
47      Oct     3ZL1998  LPSX    43.00   KVM9D    512
219     dec     2CC1999  CAD     23.00   PLV2C     68
484     nov     7PL1996  CAD     49.00   PLV2C    234
483     may     5PA1998  USP     37.00   KVM9D    644
216     sept    3ZL1998  USP     86.00   KVM9E    234
```

这里返回很多信息，有想要的，也有不想要的。参照模式，返回结果是正确的，因此这里还需要细化模式，可以以行首开始，使用 ^符号：

```
$  grep '^[0-9][0-5][0-6]' data.f
216     sept    3ZL1998  USP     86.00   KVM9E    234
```

这样可以返回一个预期的正确结果。

8.2.7 模式出现机率

抽取包含数字4至少重复出现两次的所有行，方法如下：

```
$ grep '4\{2,\}' data.f
483     may     5PA1998  USP     37.00   KVM9D    644
```

上述语法指明数字4至少重复出现两次。

同样，抽取记录使之包含数字999（三个9），方法如下：

```
$ grep '9\{3,\}' data.f
219     dec     2CC1999  CAD     23.00   PLV2C     68
```

如果要查询重复出现次数一定的所有行，语法如下，数字 9重复出现两次：

```
$ grep '9\{2\}' data.f
```

有时要查询重复出现次数在一定范围内，比如数字或字母重复出现 2到6次，下例匹配数字8重复出现2到6次，并以3结尾：

```
$ grep '6\{2,6}3'  myfile

83              – no match
888883          – match
8884            – no match
```

88883 – match

8.2.8 使用grep匹配“与”或者“或”模式

grep命令加-E参数，这一扩展允许使用扩展模式匹配。例如，要抽取城市代码为 219或216，方法如下：

```
$ grep -E '219|216' data.f
219     dec     2CC1999  CAD     23.00   PLV2C    68
216     sept    3ZL1998  USP     86.00   KVM9E   234
```

8.2.9 空行

结合使用^和$可查询空行。使用-n参数显示实际行数：

```
$ grep '^$' myfile
```

8.2.10 匹配特殊字符

查询有特殊含义的字符，诸如$. ' " * [] ^ | \ + ? ,必须在特定字符前加\。假设要查询包含“.”的所有行，脚本如下：

```
$ grep '\.' myfile
```

或者是一个双引号：

```
$ grep '\"' myfile
```

以同样的方式，如要查询文件名conftroll.conf（这是一个配置文件），脚本如下：

```
$ grep 'conftroll\.conf' myfile
```

8.2.11 查询格式化文件名

使用正则表达式可匹配任意文件名。系统中对文本文件有其标准的命名格式。一般最多六个小写字符，后跟句点，接着是两个大写字符。例如，要在一个包含各类文件名的文件filename.deposit中定位这类文件名，方法如下：

```
$ grep '[^a-z]\{1,6/}\.[^A-Z]\{1,2/}' filename.deposit
yrend.AS        – match
mothdf          – nomatch
soa.PP          – match
qp.RR           – match
```

8.2.12 查询IP地址

查询DNS服务是日常工作之一，这意味着要维护覆盖不同网络的大量 IP地址。有时地址IP会超过2000个。如果要查看nnn.nnn网络地址，但是却忘了第二部分中的其余部分，只知有两个句点，例如nnn.nn..。要抽取其中所有nnn.nnn IP地址，使用[0-9]\{3\}\.[0-0\{3\}\。含义是任意数字出现3次，后跟句点，接着是任意数字出现3次，后跟句点。

```
$ grep '[0-9]\{3\}\.[0-0\[3\}\.' ipfile
```

8.3 类名

grep允许使用国际字符模式匹配或匹配模式的类名形式。

表8-1 类名及其等价的正则表达式

类	等价的正则表达式	类	等价的正则表达式
[[:upper:]]	[A-Z]	[[:alnum:]]	[0-9a-zA-Z]
[[:lower:]]	[a-z]	[[:space:]]	空格或tab键
[[:digit:]]	[0-9]	[[:alpha:]]	[a-zA-Z]

现举例说明其使用方式。要抽取产品代码，该代码以 5开头，后跟至少两个大写字母。使用的脚本如下：

```
$ grep '5[[:upper:]][[:upper:]]' data.f
483      Sept    5AP1996  USP     65.00    LVX2C  189
483      may     5PA1998  USP     37.00    KVM9D  644
```

如果要抽取以P或D结尾的所有产品代码，方法如下：

```
$ grep '[[:upper:]][[:upper:]] [P,D]' data.f
483      Sept    5AP1996  USP     65.00    LVX2C  189
219      dec     2CC1999  CAD     23.00    PLV2C  68
484      nov     7PL1996  CAD     49.00    PLV2C  234
483      may     5PA1998  USP     37.00    KVM9D  644
216      sept    3ZL1998  USP     86.00    KVM9E  234
```

使用通配符*的匹配模式

现在讲述grep中通配符*的使用。现有文件如下：

```
$ pg testfile
looks
likes
looker
long
```

下述grep模式结果显示如下：

```
$ grep 'l.*s' testfile
looks
likes

$ grep 'l.*k.' testfile
looks
likes

$ grep 'ooo*' testfile
looks
```

如在行尾查询某一单词，试如下模式：

```
$ grep 'device$' *
```

这将在所有文件中查询行尾包含单词 device的所有行。

8.4 系统grep命令

使用已学过的知识可以很容易通过 grep命令获得系统信息。下面几个例子中，将用到管

道命令，即符号|，使用它左边命令的输出结果作为它右边命令的输入。

8.4.1 目录

如果要查询目录列表中的目录，方法如下：

```
$ ls -l | grep '^d'
```

如果在一个目录中查询不包含目录的所有文件，方法如下：

```
$ ls -l | grep '^[^d]'
```

要查询其他用户和其他用户组成员有可执行权限的目录集合，方法如下：

```
$ ls -l | grep '^d.....x..x'
```

8.4.2 passwd文件

```
$ grep "louise" /etc/passwd
louise:lxAL6GW9G.ZyY:501:501:Accounts Sect 1C:/home/accts/louise:
/bin/sh
```

上述脚本查询/etc/passwd文件是否包含louise字符串，如果误输入以下脚本：

```
$ grep "louise" /etc/password
```

将返回grep命令错误代码'No such file or directory'。

上述结果表明输入文件名不存在，使用 grep命令-s开关，可屏蔽错误信息。

```
$ grep -s "louise" /etc/password
$
```

返回命令提示符，而没有文件不存在的错误提示。

如果grep命令不支持-s开关，可替代使用以下命令：

```
$ grep "louise" /etc/password >/dev/null 2>&1
```

脚本含义是匹配命令输出或错误（ 2>$1），并将结果输出到系统池。大多数系统管理员称/dev/null为比特池，没关系，可以将之看成一个无底洞，有进没有出，永远也不会填满。

上述两个例子并不算好，因为这里的目的只想知道查询是否成功。本书后面部分将讨论grep命令的exit用法，它允许查询并不成功返回。

如要保存grep命令的查询结果，可将命令输出重定向到一个文件。

```
$ grep "louise" /etc/passwd >/tmp/passwd.out
```

脚本将输出重定向到目录/tmp下文件passwd.out中。

8.4.3 使用ps命令

使用带有ps x命令的grep可查询系统上运行的进程。 ps x命令意为显示系统上运行的所有进程列表。要查看DNS服务器是否正在运行（通常称为named），方法如下：

```
$ ps ax | grep "named"
PID  TTY  STAT TIME  CMD
211  ?    S    4.56  named
303  3    S    0.00  grep named
```

输出也应包含此grep命令，因为grep命令创建了相应进程， ps x将找到它。在grep命令中使用-v选项可丢弃ps命令中的grep进程。

```
$ ps ax | grep named |grep -v "grep"
211  ?    S    4.56  named
```

如果ps x不适用于用户系统，替代使用 ps -ef。

8.4.4 对一个字符串使用grep

grep不只应用于文件，也可应用于字符串。为此使用 echo字符串命令，然后对 grep命令使用管道输入。

```
$ STR="Mary Joe Peter Pauline"
$ echo $STR | grep "Mary"
Mary Joe Peter Pauline
```

匹配成功实现。

```
$ echo $STR | grep "Simon"
```

因为没有匹配字符串，所以没有输出结果。

8.5 egrep

egrep代表expression或extended grep，适情况而定。egrep接受所有的正则表达式， egrep的一个显著特性是可以以一个文件作为保存的字符串，然后将之传给 egrep作为参数，为此使用-f开关。如果创建一个名为grepstrings的文件，并输入484和47：

```
$ pg grepstrings
484
47
$ egrep -f grepstrings data.f
```

上述脚本匹配data.f中包含484或47的所有记录。当匹配大量模式时， -f开关很有用，而在一个命令行中敲入这些模式显然极为繁琐。

如果要查询存储代码32L或2CC，可以使用（|）符号，意即“ | ”符号两边之一或全部。

```
$ egrep '(3ZL|2CC)' data.f
47      Oct     3ZL1998  LPSX    43.00   KVM9D   512
219     dec     2CC1999  CAD     23.00   PLV2C   68
216     sept    3ZL1998  USP     86.00   KVM9E   234
```

可以使用任意多竖线符“ | ”，例如要查看在系统中是否有帐号 louise、matty或pauline ，使用who命令并管道输出至egrep。

```
$ who | egrep (louise|matty|pauline)
 louise pty8
 matty tty02
pauline pty2
```

还可以使用^符号排除字符串。如果要查看系统上的用户，但不包括 matty和pauline，方法如下：

```
$ who | egrep -v '^(matty|pauline)'
```

如果要查询一个文件列表，包括 shutdown、shutdowns、reboot和reboots，使用egrep可容易地实现。

```
$ egrep '(shutdown | reboot) (s)?'  *
```

8.6 小结

希望大家已经理解了grep的灵活性，它是一个很强大而流行的工具，像其他许多 UNIX工具一样，已经被使用在 DOS中。如果要通过文件快速查找字符串或模式， grep是一个很好的选择。简单地说， grep是shell编程中很重要的工具，在本书后面部分使用其他 UNIX工具和进行变量替换时将发现这一点。