Linux

华清远见® FARSIGHT 嵌入式培训专家

第3章 正则表达式

由于很多 Linux Shell 编程的工具和命令普遍使用到了正则表达式，如 grep、sed 和 awk 等，因此，不理解正则表达式就无法理解和熟练地使用 Shell 编程的工具和命令。为此，本章首先深入介绍正则表达式的基础知识，详细讨论基本正则表达式和扩展正则表达式中元字符的意义和用法。然后，介绍 Shell 在搜索匹配文件时经常使用的机制——通配，通配所用到的元字符与正则表达式存在细微差别，本章结合例子逐个讨论通配的元字符。最后，介绍 Linux 系统中使用广泛的 grep 命令，着重讨论 grep 命令的基本用法，及如何与正则表达式相结合，以便更加灵活地进行文本搜索，此外，还将简单介绍 grep 命令族中的其他两个命令 egrep 和 fgrep。

3.1 正则表达式基础

Linux Shell 以一串字符作为表达式向系统传达意思。元字符（Metacharacters）是用来阐释字符表达式意义的字符，简言之，元字符就是描述字符的字符，它用于对字符表达式的内容、转换及各种操作信息进行描述。正则表达式是由一串字符和元字符构成的字符串，简称 RE（Regular Expression）。正则表达式的主要功能是文本查询和字符串操作，它可以匹配文本的一个字符或字符集合。

在 Linux 系统中，程序设计语言 Java、Perl 和 Python 等，Shell 工具 sed、awk 和 grep 等，MySQL 和 PostgreSQL 等数据库服务器都使用了正则表达式，图 3-1 描述了正则表达式用于数据流处理的过程，实际上，正则表达式完成了数据过滤，将不满足正则表达式定义的数据拒绝掉，剩下与正则表达式匹配的数据。

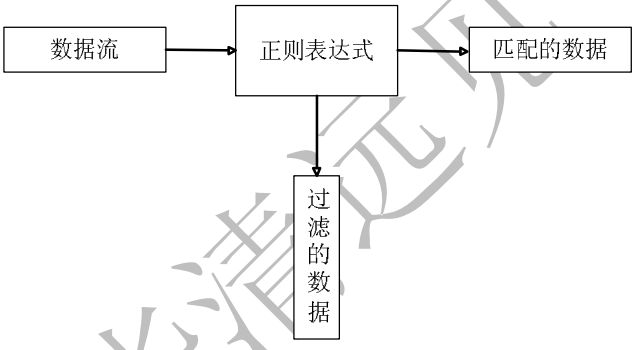

图 3-1　正则表达式处理数据过程

正则表达式的基本元素包括普通字符和元字符，例如，a、b、1、2 等字符属于普通字符，普通字符可以按照字面意思理解，如：a 只能理解为英文的小写字母 a，没有其他隐藏含义。而*、^、[]等元字符，Shell 赋予了它们超越字面意思的意义，如：*符号的字面意义只是一个符号，而实际上却表示了重复前面的字符 0 次或多次的隐藏含义。因此，掌握正则表达式基本元素主要是对正则表达式中元字符意义的掌握。

POSIX 标准将正则表达式分为两类：基本的正则表达式和扩展的正则表达式，大部分 Linux 应用和工具仅支持基本的正则表达式，因而，本节所述的正则表达式基础是掌握正则表达式的关键，表 3-1 列出了基本的正则表达式中元字符集合及其意义。

表 3-1　基本的正则表达式元字符集合及其意义

符　号	意　义
*	0 个或多个在*字符之前的那个普通字符
.	匹配任意字符
^	匹配行首，或后面字符的非

$	匹配行尾

续表

符　　号	意　　义
[]	匹配字符集合
\	转义符，屏蔽一个元字符的特殊意义
\<\>	精确匹配符号
\{n\}	匹配前面字符出现 n 次
\{n,\}	匹配前面字符至少出现 n 次
\{n,m\}	匹配前面字符出现 n～m 次

下面逐个介绍正则表达式元字符的意义和用法，并举一些例子结合使用元字符集合。

1.“*”符号

“*”符号用于匹配前面一个普通字符的 0 次或多次重复，如：

```
#例 3-1：*符号的意义
hel*o
```

“*”符号前面的普通字符是 l，*字符就表示匹配 l 字符 0 次或多次，如字符串 helo、hello、hellllllo 都可以由 hel*o 来表示。

2.“.”符号

点号“.”用于匹配任意一个字符，如：

```
#例 3-2：.符号的意义
…73.
```

由于“.”符号只能匹配一个字符，因此，上述字符串表示前面三个字符为任意字符，第 4 和第 5 个字符是 7 和 3，最后一个字符为任意字符，如 xcb738、4J973U 都能匹配上述字符串。值得注意的是，“.”符号可以匹配一个空格，因此，x b738、ui 73e 也能匹配上述字符串。

3.“^”符号

“^”符号用于匹配行首，表示行首的字符是“^”字符后面的那个字符，如：

```
#例 3-3：^符号的意义
^cloud
```

这表示匹配以 cloud 开头的行。结合上面介绍的“*”符号和“.”符号，再举一个例子：

```
#例 3-4：^符号、.符号和*符号结合使用
^…X86*
```

该字符串表示行首的三个字符为任意字符（可以是空格），第 4～6 个字符为 X86，第 7 个字符开始可以重复匹配 6，如：866X86666、8 6X86 都可以匹配上述字符串。

4.“$”符号

$符号匹配行尾，$符号放在匹配字符之后，与“^”符号的功能和用法都相反，如：

```
#例 3-5：$符号的意义
micky$
```

该正则表达式表示匹配以 micky 结尾的所有行。一个特殊的正则表达式是匹配所有空行的表达式，为：

```
#例 3-6：空行的表示方法
^$
```

该正则表达式既匹配行首，又匹配行尾，中间没有任何字符，因此，为空行。读者需要

牢记例 3-6 所示的空行表示方法，很多命令都用到这个正则表达式来表示空行。

如果需要匹配只包含一个字符的行，如下面的例 3-7 所示：

```
#例 3-7：包含一个字符的行
^.$
```

5.“[]”符号

方括号[]匹配字符集合，该符号支持穷举方法列出字符集合的所有元素，也支持使用“-”符号表示字符集合范围，表明字符集合范围从“-”左边字符开始，到“-”右边字符结束。如果要匹配任意一个数字，可以使用如例 3-8 所示的两种方法，前一种穷举了阿拉伯数字，后一种用数字范围表示，显得比较简洁。

```
#例 3-8：匹配任意一个数字
[0123456789]
[0-9]
```

“[]”也可以用做字母匹配，例 3-9 给出了匹配字母的例子：

```
#例 3-9：匹配字母
[a-z]                                    #所有小写字母
[A-Z]                                   #所有大写字母
[b-p]                                    #小写字母 b～p
```

Linux 系统对大小写是敏感的，并且支持字母排序，因此，Linux 中有大写字母序列和小写字母序列，两者是分开的，例 3-9 中 a～z 表示所有的小写字母，A～Z 表示所有的大写字母，而 b～p 表示从 b 到 p 之间所有的小写字母。

我们知道，“^”符号表示匹配行首，但是，“^”符号放到“[]”符号中就不再表示匹配行首了，而是表示取反符号，请看下面的例 3-10。

```
#例 3-10：^表示取反
[^b-d]
```

例 3-10 的正则表达式匹配不在 b～d 范围之内的所有字符，此时，符号“^”不再表示匹配行首，而是取反符号，不在 b～d 范围内的字符实际上涵盖了除了小写字母 b、c 和 d 之外的所有字符（包括其他字母、数字、空格等）。再举一个“[]”符号和“*”符号结合的例子。

```
#例 3-11：匹配所有的英文单词
[A-Za-z] [A-Za-z]*
```

例 3-11 的正则表达式表示以任意一个字母开头，再以任意字母进行 0 次或任意次重复，实际上，这个正则表达式可以匹配任意英文单词。

6.“\”符号

“\”符号是转义符，用于屏蔽一个元字符的特殊意义，即以字面含义来解释“\”符号后面的元字符，如：

```
#例 3-12：转义符
\.
```

反斜杠后面的字符“.”是元字符，经过转义后，“.”不再表示任意一个字符，而是一个普通字符句号“.”。转义符“\”是引用符的一种，我们将在 6.2.3 节深入讨论它，在此不再展开论述。

7.“\<\>”符号

“\<\>”符号是精确匹配符号，该符号利用“\”符号屏蔽“<>”符号，如：

```
#例 3-13：精确匹配
\<the\>
```

该正则表达式精确匹配 the 这个单词，而不匹配包含 the 字符的单词，如 them、there、another 等。

8.“\{\}”系列符号

“\{\}”系列符号与“*”符号类似，都是表示前一个字符的重复。但是，“*”符号表示重复 0 次或任意次，而“\{\}”系列符号可以指定重复次数，“\{\}”系列符号包括以下三种形式。

- \{n\}：匹配前面字符出现 n 次。
- \{n,\}：匹配前面字符至少出现 n 次。
- \{n,m\}：匹配前面字符出现 n～m 次。

请看下面的例 3-14。

```
#例 3-14：\{\}系列符号的用法
JO\{3\}B                                    #重复字符 O 3 次
JO\{3,\}B                                   #重复字符 O 至少 3 次
JO\{3,6\}B                                  #重复字符 O 3～6 次
```

JO\{3\}B 表示重复字符 O 3 次，匹配值为：JOOOB。

JO\{3,\}B 表示重复字符 O 至少 3 次，JOOOB、JOOOOB、JOOOOOB 等字符串都可由该正则表达式来匹配。

JO\{3,6\}B 表示重复字符 O 至少 3 次，至多 6 次，JOOOB、JOOOOOOB 等字符串都满足，但是 JOOB、JOOOOOOOB 等字符串就不满足。

再举一个例子：

```
#例 3-15：精确匹配 5 个小写字母
[a-z] \{5\}
```

例 3-15 中的表达式表示精确匹配 5 个小写英文字母，比如 hello、house 等。

3.2 正则表达式的扩展

除了表 3-1 列出的正则表达式的元字符之外，awk 和 Perl 等 Linux 工具还支持正则表达式扩展出来的一些元字符，这些元字符如表 3-2 所示。

表 3-2 扩展的正则表达式元字符及其意义

符 号	意 义
?	匹配 0 个或 1 个在其之前的那个普通字符
+	匹配 1 个或多个在其之前的那个普通字符
()	表示一个字符集合或用在 expr 中
|	表示“或”，匹配一组可选的字符

下面详细介绍扩展的正则表达式元字符及其用法。

1.“?”符号

匹配“?”符号之前的那个字符 0 次或 1 次，如：

```
#例 3-16：?符号的意义
```


```
JO?B
```

该表达式表示匹配 O 字符 0 次或 1 次，即匹配 JOB 或 JOOB。需要注意的是，“?”字符至多可以匹配 1 个字符。

2.“+”符号

与“*”符号类似，都是匹配其前面的那个字符多次，但是，“*”符号可以匹配 0 次，而“+”符号至少匹配 1 次，如：

```
#例 3-17：+符号的意义
S+EU
```

该表达式表示匹配 S 1 次或任意次，SSEU、SSSSEU 等字符串都可由该表达式进行匹配，而 SEU 却不能由 S+EU 来匹配。

3.“()”符合和“|”符号

“()”符号通常与“|”符号结合使用，表示一组可选字符的集合，如：

```
#例 3-18：()符号和|符号的意义
re(a|e|o)d
```

该表达式中的(a|e|o)表示在字符 a、e 和 o 中选择任意一个字符，即 read、reed、reod 都可由该表达式进行匹配。

事实上，()符号很少使用到，因为“[]”符号完全能够替代“()”符号表示一组可选字符的集合，re(a|e|o)d 就等价于 re[aeo]d。

“|”符号也可以表示多个正则表达式的“或”关系，基本格式为：

```
RE1 | RE2 | RE3 |…
```

上述格式中，RE1、RE2 和 RE3 表示正则表达式。

“|”符号在扩展的正则表达式中表示“或”意义，遗憾的是，“|”符号的这种用法却很少被人记住，“|”符号最著名的是其管道符用法，这将在第 10 章中详细介绍。

3.3 通配

bash Shell 本身不支持正则表达式，使用正则表达式的是 Shell 命令和工具，如 grep、sed、awk 等，这些命令将在本书后续章节详细介绍。但是，bash Shell 可以使用正则表达式中的一些元字符实现通配（Globbing）功能，通配是把一个包含通配符的非具体文件名扩展存储在计算机、服务器或者网络上的一批具体文件名的过程。最常用的通配符包括正则表达式元字符：?、*、[]、{}、^等。这些元字符在通配中的意义与正则表达式中的意义不完全一致，*符号不再表示其前面字符的重复，而是表示任意位的任意字符，?字符表示一个任意字符，^符号在通配中不代表行首，而是代表取反。

例如，如果一个用户不知道在一个扩展名为.rtf 的文件名中一个人的首名是如何拼写的，是 Philip 还是 Phillip。这个用户可以输入：

```
Phi*ip.rtf
```

下面举几个例子来说明通配的使用和通配元字符的意义，这些例子都用 ls 命令进行通配，ls 命令是 Linux 下最常用的命令之一，它用于列出目录下的文件，它可以有很多选项，ls -l 表示列出文件的详细信息，ll 命令等价于 ls -l 命令。/usr/local/globus 目录下的所有文件如下

所示：

```
[root@zawu globus]# ll                          #globus 目录下文件和目录的详细信息
总计 64
-rw-r--r--. 1 root root 1420 2007-06-21 00.pem
-rw-r--r--. 1 root root 2718 2007-06-21 08.pem
-rw-r--r--. 1 root root 2724 2007-06-05 11.pem
-rw-r--r--. 1 root root   81 10-22 14:57 append.sed
-rw-r--r--. 1 root root   72 10-15 14:11 argv.awk
-rw-r--r--. 1 root root   79 10-14 23:36 array.awk
drwxr-xr-x. 4 root root 4096 10-26 11:52 BUILD
-rw-r--r--. 1 root root   62 09-22 13:15 delete.sed
-rw-r--r--. 1 root root   73 10-15 15:49 environ.awk
-rw-r--r--. 1 root root  234 10-15 14:59 findphone.awk
-rw-r--r--. 1 root root  234 10-15 14:57 finephone.awk
-rw-r--r--. 1 root root   97 10-22 11:58 null-undeclear
-rw-r--r--. 1 root root   73 10-13 00:25 pass.awk
-rw-r--r--. 1 root root   78 10-22 15:27 scr2.awk
-rw-r--r--. 1 root root  121 10-24 11:41 stephane
-rw-r--r--. 1 root root  164 10-22 15:50 sturecord
[root@zawu globus]#
```

/usr/local/globus 目录下包含一个子目录 BUILD，子目录的详细信息以 d 开头，d 表示 directory 的意思，其他以横杠（-）开头的都是文件。

如果我们仅需要列出/usr/local/globus 目录下以.awk 结尾的文件，就可以使用*.awk 匹配所有以.awk 结尾的文件，如下面的例 3-19 所示。

```
#例 3-19：列出以.awk 结尾文件的详细信息
[root@zawu globus]# ls -l *.awk
-rw-r--r--. 1 root root  72 10-15 14:11 argv.awk
-rw-r--r--. 1 root root  79 10-14 23:36 array.awk
-rw-r--r--. 1 root root  73 10-15 15:49 environ.awk
-rw-r--r--. 1 root root 234 10-15 14:59 findphone.awk
-rw-r--r--. 1 root root 234 10-15 14:57 finephone.awk
-rw-r--r--. 1 root root  73 10-13 00:25 pass.awk
-rw-r--r--. 1 root root  78 10-22 15:27 scr2.awk
[root@zawu globus]#
```

如果我们需列出以 0 开头、后面跟 1 个字符且以.pem 为后缀的文件，可以使用 0?.pem 来匹配这些文件，如下面的例 3-20 所示。

```
#例 3-20：列出以 0 开头、后面跟 1 个字符且以.pem 为后缀的文件
[root@zawu globus]# ls -l 0?.pem
-rw-r--r--. 1 root root 1420 2007-06-21 00.pem
-rw-r--r--. 1 root root 2718 2007-06-21 08.pem
[root@zawu globus]#
```

下面举一个用[]符号进行通配的例子，若我们需列出在 a～h 范围内以字母开头并以.awk 结尾的文件，我们可以用[a-h]*.awk 来匹配这些文件，如下面的例 3-21 所示。

```
#例 3-21：列出以 a～h 范围内字母开头，以.awk 结尾的文件
[root@zawu globus]# ls -l [a-h]*.awk
-rw-r--r--. 1 root root  72 10-15 14:11 argv.awk
-rw-r--r--. 1 root root  79 10-14 23:36 array.awk
-rw-r--r--. 1 root root  73 10-15 15:49 environ.awk
-rw-r--r--. 1 root root 234 10-15 14:59 findphone.awk
-rw-r--r--. 1 root root 234 10-15 14:57 finephone.awk
[root@zawu globus]#
```

例 3-21 的结果确实仅列出以 a～h 范围内字母开头且以.awk 结尾的文件，pass.awk 和 scr2.awk 并未列出。如果我们要列出以 a～h 范围内字母开头且句点后不是以.awk 结尾的文件，可以使用[a-h]*.[^awk]*来匹配这些文件，句点后面方括号内使用“^”符号表示取反，即除去 a、w 和 k 这三个字母，而且最后一个*符号必不可少，否则句点后仅匹配一个字符，例 3-22 给出了[a-h]*.[^awk]*的匹配结果。

```
#例 3-22：列出以 a～h 范围内字母开头，不以.awk 结尾的文件
[root@zawu globus]# ls -l [a-h]*.[^awk]*
-rw-r--r--. 1 root root 81 10-22 14:57 append.sed
-rw-r--r--. 1 root root 62 09-22 13:15 delete.sed
[root@zawu globus]#
```

由例 3-22 可知，[]符号的意义与正则表达式中[]符号的意义一样，那么通配中的花括号“{}”表示何种意义呢？正则表达式中只有在花括号前加上转义符的用法，即\{\}，用于限制匹配字符的个数。但是，通配中的{}符号表示一组表达式的集合，如：

```
{[a-h]*.awk ,0?.pem}
```

上述通配表示满足[a-h]*.awk 或 0?.pem 的所有文件，下面的例 3-23 给出了这一通配的执行结果。

```
#例 3-23：列出匹配[a-h]*.awk 或 0?.pem 的所有文件
[root@zawu globus]# ls -l {[a-h]*.awk,0?.pem}
-rw-r--r--. 1 root root 1420 2007-06-21 00.pem
-rw-r--r--. 1 root root 2718 2007-06-21 08.pem
-rw-r--r--. 1 root root   72 10-15 14:11 argv.awk
-rw-r--r--. 1 root root   79 10-14 23:36 array.awk
-rw-r--r--. 1 root root   73 10-15 15:49 environ.awk
-rw-r--r--. 1 root root  234 10-15 14:59 findphone.awk
-rw-r--r--. 1 root root  234 10-15 14:57 finephone.awk
[root@zawu globus]#
```

上述例 3-23 的结果中既有满足 0?.pem 的 00.pem 和 08.pem 两个文件，也有满足[a-h]*.awk 的 argv.awk 等文件。注意，{}符号内的表达式是“或”的关系，即只要有{}符号内的一个表达式的文件，就能被列出。

通配的结果由计算机搜索大量的文件或者目录进行匹配而输出，通配对处理能力和内存资源有很高的需求。黑客输入包含通配符的文件名故意让服务器重复和连续不断地进行通配可能引起的拒绝服务攻击。因此，大型服务器经常通过限制服务器执行通配功能的次数、限制一个具体用户每次输入的通配符或者如果通配符太普通，则拒绝执行通配等方法来提高服务器的安全性。

内部变量 GLOBIGNORE 保存了通配时所忽略的文件名集合，9.1 节将详细介绍 GLOBIGNORE 变量的用法，应该说，?、*、[]、{}、^五个符号和 GLOBIGNORE 变量构成了 Shell 通配的所有内容。

3.4 grep 命令

GREP 是 Global search Regular Expression and Print out the line 的简称，即全面搜索正则

表达式并把行打印出来。GREP 是一种强大的文本搜索工具，它能使用正则表达式搜索文本，并把匹配的行打印出来，grep 也是 Linux 中最广泛使用的命令之一。本节重点介绍 grep 命令，以及 grep 命令与正则表达式结合使用，并简略介绍 grep 命令族中的其他命令用法。

3.4.1 grep 命令基本用法

grep 命令是支持正则表达式的一个多用途文本搜索工具，grep 的一般格式为：

```
grep [选项][模式][文件…]
```

grep 命令由选项、模式和文件三部分组成，它在一个或多个文件中搜索满足模式的文本行，模板后的所有字符串被看做文件名，文件名可以有多个，搜索的结果被打印到屏幕，不影响原文件的内容。grep 命令的选项用于对搜索过程进行补充说明，grep 命令的选项及其意义如表 3-3 所示。

表 3-3 grep 命令选项及其意义

选　项	意　义
-c	只输出匹配行的数量
-i	搜索时忽略大小写
-h	查询多文件时不显示文件名
-l	只列出符合匹配的文件名，而不列出具体的匹配行
-n	列出所有的匹配行，并显示行号
-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
-w	匹配整词
-x	匹配整行
-r	递归搜索，不仅搜索当前工作目录，而且搜索子目录
-q	禁止输出任何结果，以退出状态表示搜索是否成功
-b	打印匹配行距文件头部的偏移量，以字节为单位
-o	与-b 选项结合使用，打印匹配的词距文件头部的偏移量，以字节为单位
-E	支持扩展的正则表达式
-F	不支持正则表达式，按照字符串的字面意思进行匹配

grep 命令的模式十分灵活，可以是字符串，也可以是变量，还可以是正则表达式。需要说明的是，无论模式是何种形式，只要模式中包含空格，就需要使用双引号将模式引起来，如果不加双引号，空格后的单词容易被误认为是文件名，如普通字符串为“hello world”，grep hello world 命令就将 world 认为是文件名，因此，grep “hello world” filename 才是正确的写法。大部分情况下，使用单引号将模式引起来也是可以的，第 6 章将详细讨论双引号和单引号的区别，在此，我们只简单告诉读者：一旦模式中包含空格，就需要使用双引号或单引号将模式引起来。下面的例 3-24 可以充分验证这个观点。


```
#例 3-24：模式包含空格时，是否使用双引号的区别
#搜索 00.pem 文件中包含 certificate 字符串的行，不需要引号引起模式
[root@zawu globus]# grep certificate 00.pem
The above string is known as your user certificate subject, and it
To install this user certificate, please save this e-mail message
If you have any questions about the certificate contact

#若需要搜索 00.pem 文件中包含 user certificate 字符串的行
#我们看一看不用引号将 user certificate 引起来的结果
[root@zawu globus]# grep user certificate 00.pem
grep: certificate: 没有那个文件或目录
#Shell 将 certificate 解析为文件名，提示没有此文件的错误
#下面给出 00.pem 文件中包含 user 字符串的行
00.pem:The above string is known as your user certificate subject, and it
00.pem:uniquely identifies this user.
00.pem:To install this user certificate, please save this e-mail message
00.pem:/home/globus/.globus/usercert.pem

#用引号将 user certificate 引起来后得到正确的结果
[root@zawu globus]# grep "user certificate" 00.pem
The above string is known as your user certificate subject, and it
To install this user certificate, please save this e-mail message
[root@zawu globus]#
```

例 3-24 首先搜索 00.pem 文件中包含 certificate 字符串的行，由于模式 certificate 中不包含空格，因此，是否用引号引起模式对 grep 命令不产生影响。当我们要搜索 00.pem 文件中包含 user certificate 字符串的行时，不用双引号将 user certificate 括起来时，Shell 提示没有 certificate 这个文件或目录，然后，给出 00.pem 文件中包含 user 字符串的行，这说明 Shell 将 grep user certificate 00.pem 这条命令解释为在 certificate 和 00.pem 两个文件中搜索包含 user 字符串的行，这显然与我们的初衷不符。而当我们用双引号将 user certificate 括起来后，就得到了正确的结果。

grep 支持多文件查询，请看下面的例 3-25。

```
#例 3-25：演示 grep 的多文件查询
[root@zawu globus]# grep Certificate 00.pem 08.pem
00.pem:This is a Certificate Request file:
00.pem:Certificate Subject:
08.pem:Certificate:
[root@zawu globus]#
```

上例搜索 00.pem 和 08.pem 两个文件中包含 Certificate 字符串的行，命令逐个给出待搜索的文件，结果打印出所有包含 Certificate 字符串的行，并以文件名开头。

grep 命令指定多个文件时可以使用通配，这样就不必逐个给出待搜索的文件了，例 3-25 的命令可以改成如例 3-26 所示的等价命令。

```
#例 3-26：用通配表示多文件
[root@zawu globus]# grep Certificate 0?.pem
00.pem:This is a Certificate Request file:
00.pem:Certificate Subject:
08.pem:Certificate:
[root@zawu globus]#
```

上例利用 0?.pem 代替了 00.pem 和 08.pem 两个文件，显得十分简洁。

表 3-3 已经列出了 grep 命令的选项，下面我们结合具体例子逐个说明 grep 选项的含义和用法。

1．-c 选项

-c 选项表示输出匹配字符串行的数量，默认情况下，grep 命令打印出包含模式的所有行，一旦加上-c 选项，就只显示包含模式行的数量，下面给出一个使用-c 选项的例子。

```
#例 3-27：grep -c 的用法
[root@zawu globus]# grep -c Certificate *.pem
00.pem:2                                #00.pem 文件中有 2 行包含 Certificate
08.pem:1
11.pem:1
[root@zawu globus]#
```

例 3-27 对当前目录下所有.pem 的文件查找 Certificate 关键字，指定文件使用了通配，结果 00.pem:2 表示 00.pem 中有 2 行包含关键字 Certificate，08.pem 和 11.pem 各有 1 行包含关键字 Certificate。

2．-n 选项

-n 选项列出所有的匹配行，并显示行号。默认情况下，grep 搜索单个文件时，只显示每行的内容，搜索多个文件时，显示文件名及每行的内容，加上-n 选项后，将在行内容前附加显示行号，下面给出一个使用-n 选项的例子。

```
#例 3-28：grep -n 的用法
[root@zawu globus]# grep -n Certificate *.pem
00.pem:1:This is a Certificate Request file:              #00.pem 文件的第 1 行
00.pem:7:Certificate Subject:
08.pem:1:Certificate:
11.pem:1:Certificate:
[root@zawu globus]#
```

例 3-28 仍然对当前目录下所有.pem 的文件查找 Certificate 关键字，结果 00.pem:1:表示 00.pem 文件的第 1 行包含 Certificate 关键字，并列出 00.pem 的第 1 行具体内容。对于 08.pem 和 11.pem 的搜索也同样显示了行号。

3．-v 选项

-v 选项显示不包含模式的所有行，下面给出一个使用-v 选项的例子。

```
#例 3-29：grep -v 的用法
[root@zawu globus]# grep -vc Certificate *.pem            #同时使用-v 和-c 选项
00.pem:39                       #00.pem 文件中有 39 行不包含 Certificate 字符串
08.pem:50
11.pem:50
[root@zawu globus]#
```

例 3-29 结合使用-v 和-c 参数列出 00.pem 文件中不包含 Certificate 关键字的行数，结果 00.pem:39 表示 00.pem 文件中有 39 行不包含 Certificate 字符串。例 3-29 还说明 grep 命令可同时使用多个选项。

4．-i 选项

默认情况下，grep 命令对大小写是敏感的，如果加上-i 选项就表示 grep 命令不区分大小写，下面给出一个使用-i 选项的例子。

```
#例 3-30：grep -i 的用法
[root@zawu globus]# grep -i certificate 00.pem
```


```
This is a Certificate Request file:
Certificate Subject:
The above string is known as your user certificate subject, and it
To install this user certificate, please save this e-mail message
If you have any questions about the certificate contact
-----BEGIN CERTIFICATE REQUEST-----
-----END CERTIFICATE REQUEST-----
[root@zawu globus]#
```

上例在 00.pem 文件中搜索不区分大小写的 certificate 字符串的所有行，可以看出，结果匹配到 Certificate、CERTIFICATE 等关键字。

5．-h 选项

-h 选项表示查询多文件时不显示文件名，默认情况下，grep 命令查询多个文件时，在匹配行之前显示文件名，加上-h 选项后，grep 命令将不再显示文件名。下面给出一个使用-h 选项的例子。

```
#例 3-31：grep -h 的用法
[root@zawu globus]# grep -h Certificate *.pem
This is a Certificate Request file:              #在匹配行前不再显示文件名了
Certificate Subject:
Certificate:
Certificate:
[root@zawu globus]#
```

例 3-31 仍然是在当前目录下所有.pem 的文件中查找 Certificate 字符串，加上-h 后，结果就只显示匹配行的内容，而不显示文件名。注意例 3-31 与例 3-28 中 grep 打印结果的区别。

6．-l 选项

-l 选项表示只列出符合匹配的文件名，而不列出具体匹配行，下面给出一个使用-l 选项的例子。

```
#例 3-32：grep -l 的用法
[root@zawu globus]# grep -l Certificate *
00.pem                                           #只显示包含 Certificate 字符串的文件名
08.pem
11.pem
[root@zawu globus]#
```

例 3-32 的命令用于搜索当前目录下所有的文件中包含 Certificate 字符串的文本行，加上-l 选项后，在结果中不再显示具体的匹配行，只列出包含 Certificate 字符串的文件名。

7．-s 选项

-s 选项表示不显示不存在或无匹配文本的错误信息，默认情况下，grep 在待搜索文件不存在或搜索不到符合模式的文本行时将打印错误信息。下面给出一个使用-s 选项前后比较的例子。

```
#例 3-33：grep -s 的用法
[root@zawu globus]# grep user certificate 00.pem   #未使用-s 选项
grep: certificate: 没有那个文件或目录               #打印了错误信息
00.pem:The above string is known as your user certificate subject, and it
00.pem:uniquely identifies this user.
00.pem:To install this user certificate, please save this e-mail message
00.pem:/home/globus/.globus/usercert.pem
[root@zawu globus]# grep -s user certificate 00.pem      #使用-s 选项后，不打印错误信息
00.pem:The above string is known as your user certificate subject, and it
00.pem:uniquely identifies this user.
```

```
00.pem:To install this user certificate, please save this e-mail message
00.pem:/home/globus/.globus/usercert.pem
[root@zawu globus]#
```

例 3-33 给出同样的命令使用-s 选项前后的结果，两条命令都是在 certificate 和 00.pem 两个文件中搜索 user 字符串，当未使用-s 选项时，提示 certificate 文件不存在的错误信息，但是，在 grep 后加上-s 选项后，就不再打印错误信息了。

8．-r 选项

默认情况下，grep 命令只对当前目录下的文件进行搜索，而不对子目录中的文件进行搜索。-r 选项表示递归搜索，不仅搜索当前目录，而且搜索子目录。下面举一个-r 选项的例子。

```
#例 3-34：grep -r 的用法
[root@zawu globus]# grep -r CERTIFICATE *
00.pem:-----BEGIN CERTIFICATE REQUEST-----
00.pem:-----END CERTIFICATE REQUEST-----
08.pem:-----BEGIN CERTIFICATE-----
08.pem:-----END CERTIFICATE-----
11.pem:-----BEGIN CERTIFICATE-----
11.pem:-----END CERTIFICATE-----
#以下是对子目录 BUILD 中文件的搜索结果
BUILD/globus_simple_ca_1c5c054a_setup-0.19/1c5c054a.0:-----BEGIN CERTIFICATE-----
BUILD/globus_simple_ca_1c5c054a_setup-0.19/1c5c054a.0:-----END CERTIFICATE-----
BUILD/66.pem:-----BEGIN CERTIFICATE REQUEST-----
BUILD/66.pem:-----END CERTIFICATE REQUEST-----
[root@zawu globus]#
```

例 3-34 对当前目录递归搜索 CERTIFICATE 字符串，不仅列出当前目录下文件的搜索结果，还列出了子目录 BUILD 下文件包含 CERTIFICATE 字符串的文本行。

9．-w 和-x 选项

grep 命令的模式是支持正则表达式的，正则表达式的元字符将被解释成特殊的含义，-w 选项表示匹配整词，即以模式的字面含义去解析它。因此，grep 命令使用-w 选项后，元字符不再被解释为特殊含义，下面的例子说明了-w 选项的功能：

```
#例 3-35：grep -w 的用法
[root@zawu globus]# grep cer* 00.pem            #搜索包含以 cer 开头字符串的文本行
#有 4 行文本满足条件
The above string is known as your user certificate subject, and it
To install this user certificate, please save this e-mail message
/home/globus/.globus/usercert.pem
If you have any questions about the certificate contact

#加上-w 选项后，表示搜索包含 cer*字符串的文本行
[root@zawu globus]# grep -w cer* 00.pem
[root@zawu globus]#                          #没有满足该条件的文本行
```

上例两条命令的模式都为 cer*，当未用-w 选项时，模式中的*被解析为任意字符，即搜索包含以 cer 开头字符串的文本行，00.pem 文件共有 4 行文本满足该条件；加上-w 选项后，*被解析为字面含义，表示搜索包含 cer*字符串的文本行，00.pem 文件不包含 cer*这一完整的字符串。因此，无任何结果输出。

-x 选项是匹配整行，即只有当文件中有整行内容与模式匹配时，grep 命令才输出该行结果，下面的例 3-36 说明 grep 命令的-w 和-x 选项的区别。

```
#例 3-36：说明 grep 命令的-w 和-x 选项的区别
```


```
[root@jselab shell-book]# cat world.txt              #第 1 条命令：查看 world.txt 内容
Hello World
World
World Cup
African
One One World
#第 2 条命令：搜索包含单词“World”的文本行
[root@jselab shell-book]# grep -w 'World' world.txt
Hello World                    #所有包含单词“World”的文本行都被输出
World
World Cup
One One World
#第 3 条命令：搜索整行文本是单词“World”的行
[root@jselab shell-book]# grep -x 'World' world.txt
World                      #只有此行满足条件
[root@jselab shell-book]#
```

例 3-36 中，world.txt 有 5 行文本，第 2 条命令在 world.txt 文件中搜索包含单词“World”的文本行，结果列出 4 行满足条件的文本行。但是，第 3 条命令在 world.txt 文件中搜索整行文本是单词“World”的行时，只有一行满足条件。可以看出，-w 选项搜索的是整词匹配，而-x 选项搜索的是整行匹配。

10．-q 选项

从上面的讲解中知道，grep 命令默认情况下是输出结果的，但是，grep 命令后一旦加上 -q 选项，grep 将不再输出任何结果，而是以退出状态表示搜索是否成功，退出状态 0 表示搜索成功，退出状态 1 表示未搜索到满足模式的文本行，退出状态 2 表示命令或程序由于错误而未能执行。有关退出状态的内容将在第 7 章介绍，读者将 grep -q 选项与退出状态结合起来阅读，能更深入地理解-q 选项。下面我们举一个例子说明 grep -q 选项的含义。

```
#例 3-37：演示 grep -q 选项
#第 1 条命令：grep 命令搜索成功
[root@jselab shell-book]# grep -q -x 'World' world.txt
[root@jselab shell-book]# echo $?
0                                        #退出状态是 0
#第 3 条命令：grep 命令未搜索到满足模式的文本行
[root@jselab shell-book]# grep -q -x 'World African' world.txt
[root@jselab shell-book]# echo $?
1                                        #退出状态是 1
#第 5 条命令：grep 命令执行失败
[root@jselab shell-book]# grep -q -x 'World African' world
grep: world: 没有那个文件或目录
[root@jselab shell-book]# echo $?
2                                       #退出状态是 2
[root@jselab shell-book]#
```

例 3-37 列举了三种场景，第 1 条命令 grep 在 world.txt 文件中搜索整行文本是单词“World”的行，由例 3-36 能找到满足模式的行，因而，退出状态是 0（echo $?命令用于输出上条命令的退出状态，第 7 章将会讲述）；第 3 条命令 grep 在 world.txt 文件中搜索整行文本“World African”的行，由于此时 grep 命令未搜索到满足模式的文本行，因而退出状态是 1；第 5 条命令中的 world 文件不存在，grep 发生语法错误，因而退出状态是 2。

11．-b 和-o 选项

grep -b 选项打印匹配行距文件头部的偏移量，以字节为单位。如果在-b 选项后再加上-o 选项，grep 命令将打印匹配的词距文件头部的偏移量。下面的例 3-38 演示了 grep 的-b 和-o 选项。

```
#例 3-38：演示 grep 命令的-b 和-o 选项
#第 1 条命令：打印匹配行距文件头部的偏移量
[root@jselab shell-book]# grep -b -w 'World' world.txt
0:Hello World
12:World
18:World Cup
36:One One World
#第 2 条命令：打印匹配词距文件头部的偏移量
[root@jselab shell-book]# grep -b -o -w 'World' world.txt
6:World
12:World
18:World
44:World
[root@jselab shell-book]#
```

例 3-38 中的第 1 条命令打印匹配行距文件头部的偏移量，由于“Hello World”是 world.txt 的第一行，因而，该行距离文件头部的偏移量是 0 字节；然而，当第 2 条命令加上-o 选项打印匹配词距文件头部的偏移量时，第一行“Hello World”的词 World 距离文件头部是 6 字节。

grep 命令的-E 和-F 选项分别等价于 grep 命令族中的 egrep 和 fgrep 命令，我们将在 3.4.3 节介绍，有关 grep 命令选项、模式和文件的介绍到此为止，下一节将举例介绍 grep 和正则表达式的结合使用。

3.4.2 grep 和正则表达式结合使用的一组例子

将带元字符的正则表达式用于 grep 命令能够更灵活地匹配信息，使用时需要使用单引号将正则表达式引起来，以免发生一些不可预知的错误。下面通过一组例子来介绍 grep 与正则表达式结合的用法，以加强读者对 grep 命令的认识，而且有利于对正则表达式的巩固。

1. 匹配行首

元字符“^”表示行首，若需要匹配.pem 为后缀文件中以横杠（-）开头的行，可输入如下所示的命令：

```
#例 3-39：grep 查找以-符号开头的行
[root@zawu globus]# grep ^- *.pem
00.pem:-----BEGIN CERTIFICATE REQUEST-----
00.pem:-----END CERTIFICATE REQUEST-----
08.pem:-----BEGIN CERTIFICATE-----
08.pem:-----END CERTIFICATE-----
11.pem:-----BEGIN CERTIFICATE-----
11.pem:-----END CERTIFICATE-----
[root@zawu globus]#
```

下面的例 3-40 结合 grep 和正则表达式搜索空白行，第 1 个命令的意义为匹配 00.pem 文件中空白行的行数，显示结果说明 00.pem 文件中有 14 行空白行。第 2 个命令为匹配 00.pem 文件中非空白行的行数，此时使用[^$]符号表示空白行范围，前面加上“^”符号取反，显然，^^$表达式是错误的，因为 grep 将第 1 个“^”理解为行首，显示结果说明 00.pem 文件中有 27 行空白行。

```
#例 3-40：查找空白行
```


```
#第 1 条命令：搜索 00.pem 中的空白行，只打印行数
[root@zawu globus]# grep -c ^$ 00.pem
14
#第 2 条命令：搜索 00.pem 中的非空白行，只打印行数
[root@zawu globus]# grep -c ^[^$] 00.pem
27
[root@zawu globus]#
```

2．设置大小写

利用-i 符号可以使 grep 命令不区分大小写，当然也可利用[]符号来实现这一功能。下面给出用[]符号设置大小写的例子。

```
#例 3-41：用[]符号设置大小写
[root@zawu globus]# grep -n [Cc]ertificate 00.pem
1:This is a Certificate Request file:
7:Certificate Subject:
11:The above string is known as your user certificate subject, and it
14:To install this user certificate, please save this e-mail message
26:If you have any questions about the certificate contact
[root@zawu globus]#
```

上例匹配 00.pem 文件中 certificate 和 Certificate 两个关键字的行，并显示匹配的行号。如果不区分大小写来查找 00.pem 中 Certificate 关键字的行，我们就可以使用下面的两条等价命令：

```
grep "certificate" 00.pem
grep '[Cc][Ee][Rr][Tt][Ii][Ff][Ii][Cc][Aa][Tt][Ee]' 00.pem
```

3．匹配重复字符

匹配重复字符通常可以利用“.”符号和“*”符号来实现。首先举一个“.”符号的例子。

```
#例 3-42：grep 和.符号
[root@zawu globus]# grep ^/..../ 00.pem
/home/globus/.globus/usercert.pem
[root@zawu globus]#
```

上例搜索 00.pem 文件中以/字符开始、中间 4 个任意字符、第 6 个字符仍为/的行，显示结果/home/满足匹配条件。

然后给出一个“*”符号的例子。

```
#例 3-43：grep 和*符号
[root@zawu globus]# grep ^-*B 00.pem
-----BEGIN CERTIFICATE REQUEST-----
BAIwADANBgkqhkiG9w0BAQQFAAOBgQBoHRUaaB/Tyu+LuALwnT3Muw/0jDIYxc5a
[root@zawu globus]#
```

上例搜索以“-”开头，重复“-”符号任意次，然后是 B 字符的行，第 1 条结果 B 表示“-”符号重复 5 次满足匹配条件，第 2 条结果 B 表示“-”符号重复 0 次，仍然符合“*”符号的语法。因此，满足匹配条件。

4．转义符

如果匹配的目标字符串中包含元字符，则需要利用转义符“\”屏蔽其意义。如果需要搜索包含 seu.edu.cn 字符串的行，由于句号“.”字符是元字符，所以，需要在“.”符号之前加上“\”符号进行转义。如果将命令写成

```
grep 'seu.edu.cn' 00.pem
```

则是匹配 seu 和 edu、edu 和 cn 之间存在任意单个字符的行，如 seuxeduxcn 能够满足条件。下面给出搜索包含 seu.edu.cn 字符串的行的例子。

```
#例 3-44：grep 和转义符
[root@zawu globus]# grep seu\.edu\.cn 00.pem
It should be mailed to xd_ni@seu.edu.cn
   /O=Grid/OU=GlobusTest/OU=simpleCA-seugrid1.seu.edu.cn/OU=seu.edu.cn/CN=globus
the Globus Simple CA at xd_ni@seu.edu.cn
[root@zawu globus]#
```

由上例的结果可以看出，转义符使得元字符“.”符号被解析为字面含义，打印出了包含 seu.edu.cn 字符串的行。

横杠（-）字符较为特别，它虽然不属于正则表达式元字符，但是，由于“-”字符是引出 grep 命令选项的特殊字符，所以，当模式以“-”符号开头时，需要用转义符将其转义，请看下面的例 3-45。

```
#例 3-45：-字符在 grep 命令中的特殊性
[root@zawu globus]# grep -\{5\} 00.pem                    #模式以-符号开头
grep：无效选项 -- {                                       #提示错误，grep 将模式解析为选项
Usage: grep [OPTION]... PATTERN [FILE]...
Try `grep --help' for more information.
[root@zawu globus]# grep '-\{5\}' 00.pem                  #将模式用引号括起也解决不了问题
grep：无效选项 -- \
Usage: grep [OPTION]... PATTERN [FILE]...
Try `grep --help' for more information.
```

上例原本是要搜索“-”符号重复 5 次的文本行，模式当然以“-”符号开头，结果 grep 将模式解析为选项，Shell 提示无效选项错误。尽管我们将模式用引号引起来，但是仍然得到相同的错误。正确的用法应该如下所示：

```
[root@zawu globus]# grep '\-\{5\}' 00.pem
-----BEGIN CERTIFICATE REQUEST-----
-----END CERTIFICATE REQUEST-----
[root@zawu globus]#
```

上面的命令在“-”符号前加上转义符，并用引号将模式引起来，得到了正确的结果。注意，这里模式上的引号十分重要，如果不加引号，仍然提示无效选项错误。

```
[root@zawu globus]# grep \-\{5,\} 00.pem
grep：无效选项 -- {
Usage: grep [OPTION]... PATTERN [FILE]...
Try `grep --help' for more information.
[root@zawu globus]#
```

本章上机提议第 8 题建议读者执行与此相关的几条有趣的命令，读者如果执行它们并分析其中原因，一定能对转义符、“-”符号和 grep 的执行过程有更深的理解。

5. POSIX 字符类

为了保持不同国家的字符编码的一致性，POSIX（Portable Operating System Interface）增加了特殊的字符类，以[:classname]的格式给出，grep 命令支持 POSIX 字符类，首先将 POSIX 类及其意义列于表 3-4 中。

表 3-4 POSIX 字符类

类 名	意 义
[:upper:]	表示大写字母[A～Z]
[:lower:]	表示小写字母[a～z]

[:digit:]	表示阿拉伯数字[0～9]
[:alnum:]	表示大小写字母和阿拉伯数字[0～9 a～z A～Z]
[:space:]	表示空格或 Tab 键

续表

类　　名	意　　义
[:alpha:]	表示大小写字母[a～z A～Z]
[:cntrl:]	表示 Ctrl 键
[:graph:]或[:print:]	表示 ASCII 码 33～126 之间的字符
[:xdigit:]	表示 16 进制数字[0～9 A～F a～f]

下面举几个例子来说明 POSIX 字符类的用法，首先，给出一个利用 POSIX 字符类搜索以大写字母开头的行：

```
#例 3-46：利用 POSIX 字符类搜索以大写字母开头的行
[root@zawu globus]# grep ^[[:upper:]] 00.pem
This is a Certificate Request file:
It should be mailed to xd_ni@seu.edu.cn
Certificate Subject:
The above string is known as your user certificate subject, and it
To install this user certificate, please save this e-mail message
If you have any questions about the certificate contact
MIIB4zCCAUwCAQAwcTENMAsGA1UEChMER3JpZDETMBEGA1UECxMKR2xvYnVzVG
CxMKc2V1LmVkdS5jbjEPMA0GA1UEAxMGZ2xvYnVzMIGfMA0GCSqGSIb3DQEBAQUA
A4GNADCBiQKBgQCw50H88r7sAGjQGLZTmMxTiw9AgDqppBMhP6Fg3eJTqBvqBdha
YTRt1eSBT/AJUi3rTDRIABJPgu8cZkwb1AE8uEJSeCKwgk3J9QHK2NcZXwIDAQAB
BAIwADANBgkqhkiG9w0BAQQFAAOBgQBoHRUaaB/Tyu+LuALwnT3Muw/0jDIYxc5a
YaA4dWCB6/2yVYyfmyRCNox3rIsyUvqL9p81d/hpNiAB/0OazMBialq5Gcpaansd
[root@zawu globus]#
```

上例的命令使用 POSIX 字符类作为模式，[:upper:]表示大写字母集合，再用一层方括号将[:upper:]括起，表示匹配字符集合，得到的结果确实是所有以大写字母开头的文本行。

再举一个搜索以空格开头文本行的例子：

```
#例 3-47：搜索以空格开头文本行
[root@zawu globus]# grep ^[[:space:]] 00.pem
   /O=Grid/OU=GlobusTest/OU=simpleCA-seugrid1.seu.edu.cn/OU=seu.edu.cn/CN=globus
     You need not edit this message in any way. Simply
     save this e-mail message to the file.
[root@zawu globus]#
```

上例的命令仍以 POSIX 字符类作为模式，[:space:]用法与[:upper:]类似。事实上，POSIX 字符类作为模式的用法都类似，使用时只要注意用方括号将 POSIX 字符类括起来即可。

6．精确匹配

3.1 节讲述正则表达式时未曾讲述“\<\>”符号，该符号用于精确匹配，在此结合 grep 命令对此符号进行介绍。

首先给出一个说明性的例子，我们结合例子阐释何为精确匹配，以及“\<\>”符号的用法，例子如下所示：

```
#例 3-48：精确匹配
[root@zawu globus]# cat re01                              #显示 re01 文件的内容
```

```
Line1:there are four lines in this file
Line2:this the line 2
Line3:this is another line
Line4:this is line4
[root@zawu globus]# grep the re01                    #列出所有包含 the 字符串的行
Line1:there are four lines in this file              #there 中包含了 the
Line2:this the line 2
Line3:this is another line                           #another 中包含了 the
[root@zawu globus]# grep "\<the\>" re01              #精确匹配 the 这个单词
Line2:this the line 2                                #只要第 2 行有此单词
[root@zawu globus]#
```

上例创建名为 re01 的示例文件，该文件中有 4 行字符串，cat re01 命令列出了这 4 行字符串内容，关于 cat 命令的用法，可以参考 10.1.2 节，在此不介绍。未用“\<\>”符号时为模糊匹配，列出所有包含 the 字符串的行：Line 1、Line 2 和 Line 3，因为 Line 1 中包含 there，Line 2 中包含 the，Line 3 中包含 another。用“\<\>”符号精确匹配 the 这个单词，因而，只列出了 Line 2，注意\<the\>上的引号必不可少。

事实上，grep 命令的-w 选项也可用于精确匹配，下面的命令等价于上例的 grep "\<the\>" re01 命令：

```
[root@zawu globus]# grep -w the re01                 #利用-w 选项实现精确匹配
Line2:this the line 2                                #结果仍是第 2 行
[root@zawu globus]#
```

7．或字符

或字符“|”是扩展的正则表达式中定义的，grep 需要加上-E 选项才能支持它，下面给出 grep 命令使用“|”字符的例子。

```
#例 3-49：grep 命令与|字符
[root@zawu globus]# grep -E "OU|seu" 00.pem          #带-E 选项的 grep 执行成功
It should be mailed to xd_ni@seu.edu.cn
    /O=Grid/OU=GlobusTest/OU=simpleCA-seugrid1.seu.edu.cn/OU=seu.edu.cn/CN=globus
the Globus Simple CA at xd_ni@seu.edu.cn
[root@zawu globus]# grep "OU|seu" 00.pem
[root@zawu globus]#
```

例 3-49 的两个命令用于匹配带有 OU 或 seu 字符串的行，grep 带上-E 选项后得到正确的结果。而 grep 没有带-E 选项时，返回结果为空，这是因为 grep 命令将“|”字符解析为字面意义。注意，OU|seu 上的引号必不可少。

grep 命令与正则表达式结合使用极大地增加了命令应用的灵活性，同时也增大了使用命令的难度，读者只有在本节例子的基础上，多上机练习，多思考分析，才有可能熟练而灵活地运用 grep 命令。

3.4.3 grep 命令族简介

Linux 系统支持三种形式的 grep 命令，通常将这三种形式的 grep 命令称为 grep 命令族，这三种形式具体为：

- grep：标准 grep 命令，支持基本正则表达式，上面两小节已经对此命令进行了详细的讨论。
- egrep：扩展 grep 命令，支持基本和扩展正则表达式。

- fgrep：快速 grep 命令，不支持正则表达式，按照字符串的字面意思进行匹配。

egrep 命令与 grep –E 等价，fgrep 命令与 grep –F 等价，在某些 Linux 发行版中，egrep 和 fgrep 都是 grep 命令的别名，分别将其符号链接到 grep –E 和 grep –F 命令。下面举两个例子来说明 egrep 和 fgrep 命令。首先，举一个 egrep 命令的例子，如下：

```
#例 3-50：egrep 命令的用法
[root@zawu globus]# egrep "seu.edu|certificate" 00.pem          #第 1 条命令
It should be mailed to xd_ni@seu.edu.cn
   /O=Grid/OU=GlobusTest/OU=simpleCA-seugrid1.seu.edu.cn/OU=seu.edu.cn/CN=globus
The above string is known as your user certificate subject, and it
To install this user certificate, please save this e-mail message
If you have any questions about the certificate contact
the Globus Simple CA at xd_ni@seu.edu.cn
[root@zawu globus]# egrep "^-+B" 00.pem                         #第 2 条命令
-----BEGIN CERTIFICATE REQUEST-----
[root@zawu globus]#
```

上例第 1 条命令 egrep 使用了扩展的正则表达式元字符“|”符号，表示搜索包含 seu.edu 字符串或 certificate 字符串的文本行，egrep 成功地解析了元字符“|”符号的含义，得到正确的结果。第 2 条命令使用了“+”符号，意义为查找 00.pem 文件中以“-”符号开头、重复任意次且是 B 字符的行。请注意“^-+B”和“^-*B”的区别，“+”符号表示“-”符号至少重复一次，不匹配以 B 字符开头的行。

其次，我们再举一个 fgrep 命令的例子。

```
#例 3-51：fgrep 命令的用法
[root@zawu globus]# fgrep ^-*B 00.pem          #第 1 条命令，fgrep 命令不支持正则表达式
[root@zawu globus]# fgrep certificate 00.pem   #第 2 条命令，fgrep 命令支持普通字符串
The above string is known as your user certificate subject, and it
To install this user certificate, please save this e-mail message
If you have any questions about the certificate contact
[root@zawu globus]#
```

上例的第 1 条命令是 fgrep 后加正则表达式，无任何返回结果，这说明 fgrep 不支持正则表达式；第 2 条命令 fgrep 后面加普通字符串，返回正确结果，这说明 fgrep 支持普通字符串。

egrep 和 fgrep 命令极少使用，因为 grep 命令功能已十分强大，足以替代 egrep 和 fgrep 命令。因此，读者只需对 egrep 和 fgrep 命令有所了解即可。

3.5 本章小结

本章介绍了 Shell 命令和工具所涉及的基本文法——正则表达式，重点介绍了基本正则表达式和扩展正则表达式中元字符的意义和用法。在此基础上，介绍了 Shell 的通配功能，结合例子逐个讨论了通配的元字符。本章还介绍了 Linux 系统中使用广泛的 grep 命令，着重讨论了 grep 命令的基本用法，及如何与正则表达式相结合以更加灵活地进行文本搜索，此外，还简单介绍了 grep 命令族中的其他两个命令 egrep 和 fgrep。

3.6 上机提议

1. 分析下面的正则表达式表达了什么含义。

2.

```
（1）kK*
（2）k\{6,8\}
（3）k\{6,\}
（4）k\{10\}
（5）^NEW YEAR$
（6）^$
（7）[0-9][0-9][a-z]
（8）[A-H]\{1,3\},[0-9]\{5\}
（9）^\...
（10）[^p-z]*\.
```

2. 利用通配功能列出某目录下所有以数字开头、最后 3 位是句点和 2 个任意字母的文件名。

3. 利用通配功能列出 Windows 下文本格式文件，即以.doc、.txt、.ppt、.docx 和.pptx 等结尾的文件。

4. 在上题的基础上，不区分大小写搜索 Windows 下文本格式文件包含 chapter 字符串的文本行内容及其行号，要求写出等价的两种形式的命令。

5. 下面的第（1）条命令是 3.4.1 节讲述 grep 命令的-r 选项时的示例命令，执行下面的第（2）条命令，观察该命令是否仍将对 BUILD 子目录进行搜索，分析其原因。

```
(1) grep -r CERTIFICATE *
(2) grep -r CERTIFICATE *.pem
```

6. 重做 3.4.2 节搜索空白行和非空白行的例子，并试验用^^$匹配非空白行这种错误用法，即执行下面的三条命令：

```
(1) grep -c ^$ 00.pem
(2) grep -c ^[^$] 00.pem
(3) grep -c ^^$ 00.pem
```

7. 统计当前目录及其子目录下的所有文件所包含空白行的行数。再写一个命令统计当前目录及其子目录下的所有文件包含非空白行的行数。

8. 结合 3.4.2 节对“-”符号的阐述，执行下面几条命令，观察是否仍然提示无效选项错误，分析其中的原因：

```
grep -n -\{5,\} 00.pem
grep -n '-\{5,\}' 00.pem
```

9. 对于 3.4.2 节精确匹配示例中的 re01 文件，依次执行下面的 4 条命令，前 3 条命令是 3.4.2 节曾讲述过的，第 4 条命令是错误的命令，分析第 4 条命令得不到正确结果的原因。


```
(1) grep the re01
(2) grep "\<the\>" re01
(3) grep -w the re01
(4) grep \<the\> re01
```

10．分别用 grep -E 和 egrep 两个命令，结合扩展的正则表达式实现在 00.pem 中查询以冒号（:）结尾，或以非英文字母结尾的文本行。