目 录

1 基本的正则表达式

1.1 纯文本匹配

sed 编辑器就使用标准的文本字符串筛选数据，如下例所示：

```
1  echo "This is a test" | sed -n '/test/p'
2  echo "This is a test" | sed -n "/this/p"
```

在正则表达式中，文本匹配不局限于完整的单词，如果所定义的文本出现在数据流的任意位置，正则表达式都将匹配，如下例所示：

```
1  echo "This books are expensive" | sed -n '/book/p'
```

1.2 特殊字符

正则表达式认可的特殊字符有：

```
1  .*[]^$ {}\+?|()
```

如果要使用这些特殊字符作为文本字符，需要在特殊符号前加反斜杠。

1.2.1 定位符

和 $ 用于将模式定位到数据流中行的开头和结尾。

定 ^ 义从数据流中文本行开头开始的模式，如果该模式位于文本行的其他任意位置，正则表达式匹配失败。如果要使用，^需要将放 ^ 在正则表达式指定的模式之前。如下例所示：

```
1  echo "This book store" | sed -n '/^book/p'
2  echo "books are greate" | sed -n '/^book/p'
```

需要注意的是，如果将脱字符放 ^ 在模式的其他位置，它就充当普通字符而不再作为特殊字符，如下例所示：

```
1  echo "This ^ is a test" | sed -n '/s ^/p'
```

$ 定义从数据流中文本行结尾处的模式，如果在文本模式之后添加这个特殊符号，表示数据行必须以此文本模式结束，如下例所示：

```
1  echo "This is a good book" | sed -n '/book$/p'
2  echo "This book is good" | sed -n '/book$/p'
```

和 $ 在没有文本的模式中结合使用时可以筛选数据流中的空行，如下例所示：

```
sed '/^$/d' data1
```

1.2.2 点字符

点字符可以匹配除换行符之外的任何单个字符，如下例所示：

```
sed -n '/.at/p' data1
```

1.2.3 字符类

字符类用于限制要匹配的字符。如果要定义字符类，要使用方括号，如下例所示：

```
# 仅匹配cat和hat
sed -n '/[ch]at/p' data1
# 仅匹配0、1、2或3
sed -n '/[0123]/p' data1
```

1.2.4 否定字符类

否定字符类用于查找不在该字符类中的字符，当然这些字符仍然需要符合模式。通过在字符类范围的开头添加脱字符，可以使用否定字符类，如下例所示：

```
# 这个用于匹配除cat和hat之外的所有.at
sed -n '/[^ch]at/p' data1
```

1.2.5 使用范围

可以通过使用短划线符号在字符类中使用一系列字符范围，只需要指定范围内的第一个字符、短划线和范围内的最后一个字符。例子如下：

```
# 用于匹配5位数字
sed -n '/^[0-9][0-9][0-9][0-9][0-9]$/p' data1
# 用于匹配cat、dat、eat、fat、gat或hat
sed -n '/[c-h]at/p' data1
```

还可以在单个字符类中指定多个非连续的范围，如下例所示：

```
# 用于匹配aat、bat、cat、hat、iat、jat、kat、lat和mat
sed -n '/[a-ch-m]at/p' data1
```

1.2.6 星号

在某一个字符之后加一个星号，表示该字符在匹配模式中的文本中不出现或者出现大等于 1 次，如下例所示：

```
echo "ik" | sed -n '/ie*k/p'
echo "iek" | sed -n '/ie*k/p'
echo "ieek" | sed -n '/ie*k/p'
```

还可以将星号应用于字符类，用于指定在文本中多次出现的一组或某一范围的字符，如下例所示：

```
echo "bt" | sed -n '/b[ae]*t/p'
echo "bat" | sed -n '/b[ae]*t/p'
echo "bet" | sed -n '/b[ae]*t/p'
echo "baaaeeet" | sed -n '/b[ae]*t/p'
```

1.3 特殊字符类

BRE 定义了一些特殊的字符类，如下所示：

[[:alpha:]]	匹配任意字母字符
[[:alnum:]]	匹配任意字母数字字符
[[:blank:]]	匹配空格或制表符字符
[[:digit:]]	匹配任意数字
[[:lower:]]	匹配任意小写字母字符
[[:print:]]	匹配任意可打印字符
[[:punct:]]	匹配任意标点符号
[[:space:]]	匹配任意空白字符
[[:upper:]]	匹配任意大写字母字符

2 扩展的正则表达式

2.1 问号

在某一个字符之后加一个问号，表示该字符在匹配模式中的文本中不出现或者只出现 1 次，如下例所示：

```
echo "bt" | sed -n '/be?t/p'
echo "bet" | sed -n '/be?t/p'
echo "beet" | sed -n '/be?t/p'
```

问号还可以加在字符类后面，表示这个字符类在匹配模式中的文本中不出现或者只出现 1 次，如下例所示：

```
echo "bt" | sed -n '/b[ae]?t/p'
echo "bat" | sed -n '/b[ae]?t/p'
echo "bet" | sed -n '/b[ae]?t/p'
echo "baat" | sed -n '/b[ae]?t/p'
echo "beet" | sed -n '/b[ae]?t/p'
echo "baet" | sed -n '/b[ae]?t/p'
```

2.2 加号

在某一个字符之后加一个加号，表示该字符在匹配模式中的文本中出现大等于 1 次，如下例所示：

```
echo "bt" | sed -n '/be+t/p'
echo "bet" | sed -n '/be+t/p'
echo "beet" | sed -n '/be+t/p'
```

类似的，加号也可以和字符类一起使用。

2.3 使用大括号

可以使用大括号指定匹配模式的重复次数，这种方式称为间隔。有两种格式表示间隔：

- m：该正则表达式正好出现 m 次
- m,n：该正则表达式出现最少 m 次，最多 n 次

使用例子如下所示：

```
echo "bt" | sed -n '/be{1,2}t'
echo "bet" | sed -n '/be{1,2}t'
echo "beet" | sed -n '/be{1,2}t'
```

类似的，间隔模式也可以和字符类一起使用。

2.4 管道符号

管道符号允许正则表达式匹配数据流时使用两个或多个模式。如果任何一个模式与数据流文本匹配，该文本通过。如果没有一个模式匹配，数据流文本匹配失败。使用管道符号的格式如下：

```
expr1|expr2
```

使用例子如下：

```
echo "This cat is asleep" | sed '/cat|dog/p'
echo "This dog is asleep" | sed '/cat|dog/p'
```

2.5 将表达式分组

使用圆括号可以将正则表达式模式分组，一个组合将被作为一个标准字符处理。可以像将特殊字符应用于正常字符一样，将特殊字符应用于组合。例子如下：

```
echo "Sat" | sed -n '/Sat\(urday\)?/p'
```

常见的用法是将分组和管道符号结合使用，如下例所示：

```
echo "cat" | sed -n '/\(c|b\)a\(b|t\)/p'
```