转自：<http://blog.chinaunix.net/uid/20422917.html>

让我们看一下 sed 最有用的命令之一，替换命令。使用该命令，可以将特定字符串或匹配的规则表达式用另一个字符串替换。下面是该命令最基本用法的示例：

$ sed -e 's/foo/bar/' myfile.txt

上面的命令将 myfile.txt 中每行第一次出现的 'foo'（如果有的话）用字符串 'bar' 替换，然后将该文件内容输出到标准输出。请注意，我说的是每行第一次出现，尽管这通常不是您想要的。在进行字符串替换时，通常想执行全局替换。也就是说，要替换每行中的所有出现，如下所示：

$ sed -e 's/foo/bar/g' myfile.txt

在最后一个斜杠之后附加的 'g' 选项告诉 sed 执行全局替换。

关于 's///' 替换命令，还有其它几件要了解的事。首先，它是一个命令，并且只是一个命令，在所有上例中都没有指定地址。这意味着，'s///' 还可以与地址一起使用来控制要将命令应用到哪些行，如下所示：

$ sed -e '1,10s/enchantment/entrapment/g' myfile2.txt

上例将导致用短语 'entrapment' 替换所有出现的短语 'enchantment'，但是只在第一到第十行（包括这两行）上这样做。

$ sed -e '/^$/,/^END/s/hills/mountains/g' myfile3.txt

该例将用 'mountains' 替换 'hills'，但是，只从空行开始，到以三个字符 'END' 开始的行结束（包括这两行）的文本块上这样做。

关于 's///' 命令的另一个妙处是 '/' 分隔符有许多替换选项。如果正在执行字符串替换，并且规则表达式或替换字符串中有许多斜杠，则可以通过在 's' 之后指定一个不同的字符来更改分隔符。例如，下例将把所有出现的 /usr/local 替换成 /usr：

$ sed -e 's:/usr/local:/usr:g' mylist.txt

在该例中，使用冒号作为分隔符。如果不指定分隔符，则变成了如下：

$ sed -e 's/usr/local/usrg' mylist.txt

这样就不能执行了

如果需要在规则表达式中指定分隔符字符，可以在它前面加入反斜杠。

规则表达式混乱

目前为止，我们只执行了简单的字符串替换。虽然这很方便，但是我们还可以匹配规则表达式。例如，以下 sed 命令将匹配从 '<' 开始、到 '>' 结束、并且在其中包含任意数量字符的短语。下例将删除该短语（用空字符串替换）：

$ sed -e 's/<.\*>//g' myfile.html

这是要从文件除去 HTML 标记的第一个很好的 sed 脚本尝试，但是由于规则表达式的特有规则，它不会很好地工作。原因何在？当 sed 试图在行中匹配规则表达式时，它要在行中查找最长的匹配。在我的前一篇 sed 文章中，这不成问题，因为我们使用的是 'd' 和 'p' 命令，这些命令总要删除或打印整行。但是，在使用 's///' 命令时，确实有很大不同，因为规则表达式匹配的整个部分将被目标字符串替换，或者，在本例中，被删除。这意味着，上例将把下行：

<b>This</b> is what <b>I</b> meant.

变成：

meant.

我们要的不是这个，而是：

This is what I meant.

幸运的是，有一种简便方法来纠正该问题。我们不输入“'<' 字符后面跟有一些字符并以 '>' 字符结束”的规则表达式，

而只需输入一个“'<' 字符，后面跟有任意数量非 '>' 字符，并以 '>' 字符结束”的规则表达式。这将与最短、而不是最长的可能性匹配。新命令如下：

$ sed -e 's/<[^>]\*>//g' myfile.html

在上例中，'[^>]' 指定“非 '>'”字符，其后的 '\*' 完成该表达式以表示“零或多个非 '>' 字符”。对几个 html 文件测试该命令，将它们管道输出 "more"，然后仔细查看其结果。

更多字符匹配

'[ ]' 规则表达式语法还有一些附加选项。要指定字符范围，只要字符不在第一个或最后一个位置，就可以使用 '-'，如下所示：

'[a-x]\*'

这将匹配零或多个全部为 'a'、'b'、'c'...'v'、'w'、'x' 的字符。另外，可以使用 '[]' 字符类来匹配空格。以下是可用字符类的相当完整的列表：

字符类 描述

[] 字母数字 [a-z A-Z 0-9]

[] 字母 [a-z A-Z]

[] 空格或制表键

[] 任何控制字符

[] 数字 [0-9]

[] 任何可视字符（无空格）

[] 小写 [a-z]

[] 非控制字符

[] 标点字符

[] 空格

[] 大写 [A-Z]

[] 十六进制数字 [0-9 a-f A-F]

尽可能使用字符类是很有利的，因为它们可以更好地适应非英语 locale（包括某些必需的重音字符等等）.

高级替换功能

我们已经看到如何执行简单甚至有些复杂的直接替换，但是 sed 还可以做更多的事。实际上可以引用匹配规则表达式的部分或全部，并使用这些部分来构造替换字符串。作为示例，假设您正在回复一条消息。下例将在每一行前面加上短语 "ralph said: "：

$ sed -e 's/.\*/ralph said: &/' origmsg.txt

输出如下：

ralph said: Hiya Jim,

ralph said:

ralph said:

I sure like this sed stuff!

ralph said:

该例的替换字符串中使用了 '&' 字符，该字符告诉 sed 插入整个匹配的规则表达式。因此，可以将与 '.\*' 匹配的任何内容（行中的零或多个字符的最大组或整行）插入到替换字符串中的任何位置，甚至多次插入。这非常好，但 sed 甚至更强大。

那些极好的带反斜杠的圆括号

's///' 命令甚至比 '&' 更好，它允许我们在规则表达式中定义区域，然后可以在替换字符串中引用这些特定区域。作为示例，假设有一个包含以下文本的文件：

foo bar oni eeny meeny miny larry curly moe jimmy the weasel

现在假设要编写一个 sed 脚本，该脚本将把 "eeny meeny miny" 替换成 "Victor eeny-meeny Von miny" 等等。要这样做，首先要编写一个由空格分隔并与三个字符串匹配的规则表达式。

'.\* .\* .\*'

现在，将在其中每个感兴趣的区域两边插入带反斜杠的圆括号来定义区域：

'\(.\*\) \(.\*\) \(.\*\)'

除了要定义三个可在替换字符串中引用的逻辑区域以外，该规则表达式的工作原理将与第一个规则表达式相同。下面是最终脚本：

$ sed -e 's/\(.\*\) \(.\*\) \(.\*\)/Victor \1-\2 Von \3/' myfile.txt

如您所见，通过输入 '\x'（其中，x 是从 1 开始的区域号）来引用每个由圆括号定界的区域。输入如下：

Victor foo-bar Von oni Victor eeny-meeny Von miny Victor larry-curly Von moe Victor jimmy-the Von weasel

随着对 sed 越来越熟悉，您可以花最小力气来进行相当强大的文本处理。您可能想如何使用熟悉的脚本语言来处理这种问题 -- 能用一行代码轻易实现这样的解决方案吗？

组合使用

在开始创建更复杂的 sed 脚本时，需要有输入多个命令的能力。有几种方法这样做。首先，可以在命令之间使用分号。例如，以下命令系列使用 '=' 命令和 'p' 命令，'=' 命令告诉 sed 打印行号，'p' 命令明确告诉 sed 打印该行（因为处于 '-n' 模式）。

$ sed -n -e '=;p' myfile.txt

无论什么时候指定了两个或更多命令，都按顺序将每个命令应用到文件的每一行。在上例中，首先将 '=' 命令应用到第 1 行，然后应用 'p' 命令。接着，sed 继续处理第 2 行，并重复该过程。虽然分号很方便，但是在某些场合下，它不能正常工作。另一种替换方法是使用两个 -e 选项来指定两个不同的命令：

$ sed -n -e '=' -e 'p' myfile.txt

然而，在使用更为复杂的附加和插入命令时，甚至多个 '-e' 选项也不能帮我们的忙。对于复杂的多行脚本，最好的方法是将命令放入一个单独的文件中。然后，用 -f 选项引用该脚本文件：

$ sed -n -f mycommands.sed myfile.txt

这种方法虽然可能不太方便，但总是管用。

一个地址的多个命令

有时，可能要指定应用到一个地址的多个命令。这在执行许多 's///' 以变换源文件中的字和语法时特别方便。要对一个地址执行多个命令，可在文件中输入 sed 命令，然后使用 '{ }' 字符将这些命令分组，如下所示：

1,20{ s/[Ll]inux/GNU\/Linux/g s/samba/Samba/g s/posix/POSIX/g }

上例将把三个替换命令应用到第 1 行到第 20 行（包括这两行）。还可以使用规则表达式地址或者二者的组合：

1,/^END/{ s/[Ll]inux/GNU\/Linux/g s/samba/Samba/g s/posix/POSIX/g p }

该例将把 '{ }' 之间的所有命令应用到从第 1 行开始，到以字母 "END" 开始的行结束（如果在源文件中没发现 "END"，则到文件结束）的所有行。

附加、插入和更改行

既然在单独的文件中编写 sed 脚本，我们可以利用附加、插入和更改行命令。这些命令将在当前行之后插入一行，在当前行之前插入一行，或者替换模式空间中的当前行。它们也可以用来将多行插入到输出。插入行命令用法如下：

i\ This line will be inserted before each line

如果不为该命令指定地址，那么它将应用到每一行，并产生如下的输出：

This line will be inserted before each line line 1 here

This line will be inserted before each line line 2 here

This line will be inserted before each line line 3 here

This line will be inserted before each line line 4 here

如果要在当前行之前插入多行，可以通过在前一行之后附加一个反斜杠来添加附加行，如下所示：

i\ insert this line\ and this one\ and this one\ and, uh, this one too.

附加命令的用法与之类似，但是它将把一行或多行插入到模式空间中的当前行之后。其用法如下：

a\ insert this line after each line. Thanks! :)

另一方面，“更改行”命令将实际替换模式空间中的当前行，其用法如下：

c\ You're history, original line! Muhahaha!

因为附加、插入和更改行命令需要在多行输入，所以将把它们输入到一个文本 sed 脚本中，然后通过使用 '-f' 选项告诉 sed 执行它们。使用其它方法将命令传递给 sed 会出现问题。