通用线程 -- sed 实例，第 1 部分[转帖]
http://www.chinaunix.net 作者:sd-feng 发表于：2002-11-06 17:02:26

在本文章系列中，Daniel Robbins 将为您演示如何使用功能十分强大（但常被遗忘）的 UNIX 流编辑器 sed。sed 是用批处理方式编辑文件或以十分有效的方式创建 shell 脚本以修改现有文件的理想工具。

挑选编辑器
在 UNIX 世界中有很多文本编辑器可供我们选择。思考一下 -- vi、emacs 和 jed 以及很多其它工具都会浮现在脑海中。我们都有自己已逐渐了解并且喜爱的编辑器（以及我们喜爱的组合键）。有了可信赖的编辑器，我们可以轻松处理任何数量与 UNIX 有关的管理或编程任务。

虽然交互式编辑器很棒，但却有其限制。尽管其交互式特性可以成为强项，但也有其不足之处。考虑一下需要对一组文件执行类似更改的情形。您可能会本能地运行自己所喜爱的编辑器，然后手工执行一组烦琐、重复和耗时的编辑任务。然而，有一种更好的方法。

进入 sed

如果可以使编辑文件的过程自动化，以便用“批处理”方式编辑文件，甚至编写可以对现有文件进行复杂更改的脚本，那将太好了。幸运的是，对于这种情况，有一种更好的方法 -- 这种更好的方法称为 "sed"。

sed 是一种几乎包括在所有 UNIX 平台（包括 Linux）的轻量级流编辑器。sed 有许多很好的特性。首先，它相当小巧，通常要比您所喜爱的脚本语言小很多倍。其次，因为 sed 是一种流编辑器，所以，它可以对从如管道这样的标准输入接收的数据进行编辑。因此，无需将要编辑的数据存储在磁盘上的文件中。因为可以轻易将数据管道输出到 sed，所以，将 sed 用作强大的 shell 脚本中长而复杂的管道很容易。试一下用您所喜爱的编辑器去那样做。

GNU sed
对 Linux 用户来说幸运的是，最好的 sed 版本之一恰好是 GNU sed，其当前版本是 3.02。每一个 Linux 发行版都有（或至少应该有）GNU sed。GNU sed 之所以流行不仅因为可以自由分发其源代码，还因为它恰巧有许多对 POSIX sed 标准便利、省时的扩展。另外，GNU 没有 sed 早期专门版本的很多限制，如行长度限制 -- GNU 可以轻松处理任意长度的行。

最新的 GNU sed
在研究这篇文章之时我注意到：几个在线 sed 爱好者提到 GNU sed 3.02a。奇怪的是，在 ftp.gnu.org（有关这些链接，请参阅参考资料）上找不到 sed 3.02a，所以，我只得在别处寻找。我在 alpha.gnu.org 的 /pub/sed 中找到了它。于是我高兴地将其下载、编译然后安装，而几分钟后我发现最新的 sed 版本却是 3.02.80 -- 可在 alpha.gnu.org 上 3.02a 源代码旁边找到其源代码。安装完 GNU sed 3.02.80 之后，我就完全准备好了。

正确的 sed
在本系列中，将使用 GNU sed 3.02.80。在即将出现的本系列后续文章中，某些（但非常少）最高级的示例将不能在 GNU sed 3.02 或 3.02a 中使用。如果您使用的不是 GNU sed，那么结果可能会不同。现在为什么不花些时间安装 GNU sed 3.02.80 呢？那样，不仅可以为本系列的余下部分作好准备，而且还可以使用可能是目前最好的 sed。

sed 示例

sed 通过对输入数据执行任意数量用户指定的编辑操作（“命令”）来工作。sed 是基于行的，因此按顺序对每一行执行命令。然后，sed 将其结果写入标准输出 (stdout)，它不修改任何输入文件。

让我们看一些示例。头几个会有些奇怪，因为我要用它们演示 sed 如何工作，而不是执行任何有用的任务。然而，如果您是 sed 新手，那么理解它们是十分重要的。下面是

第一个示例：

```
$ sed -e 'd' /etc/services
```

如果输入该命令，将得不到任何输出。那么，发生了什么？在该例中，用一个编辑命令 'd' 调用 sed。sed 打开 /etc/services 文件，将一行读入其模式缓冲区，执行编辑命令（“删除行”），然后打印模式缓冲区（缓冲区已为空）。然后，它对后面的每一行重复这些步骤。这不会产生输出，因为 "d" 命令除去了模式缓冲区中的每一行！

在该例中，还有几件事要注意。
首先，根本没有修改 /etc/services。这还是因为 sed 只读取在命令行指定的文件，将其用作输入 -- 它不试图修改该文件。

第二件要注意的事是 sed 是面向行的。'd' 命令不是简单地告诉 sed 一下子删除所有输入数据。相反，sed 逐行将 /etc/services 的每一行读入其称为模式缓冲区的内部缓冲区。一旦将一行读入模式缓冲区，它就执行 'd' 命令，然后打印模式缓冲区的内容（在本例中没有内容）。我将在后面为您演示如何使用地址范围来控制将命令应用到哪些行 -- 但是，如果不使用地址，命令将应用到所有行。

第三件要注意的事是括起 'd' 命令的单引号的用法。养成使用单引号来括起 sed 命令的习惯是个好注意，这样可以禁用 shell 扩展。

另一个 sed 示例

下面是使用 sed 从输出流除去 /etc/services 文件第一行的示例：

```
$ sed -e '1d' /etc/services | more
```

如您所见，除了前面有 '1' 之外，该命令与第一个 'd' 命令十分类似。如果您猜到 '1' 指的是第一行，那您就猜对了。与第一个示例中只使用 'd' 不同的是，这一次使用的 'd' 前面有一个可选的数字地址。通过使用地址，可以告诉 sed 只对某一或某些特定行进行编辑。

地址范围

现在，让我们看一下如何指定地址范围。在本例中，sed 将删除输出的第 1 到 10 行：

```
$ sed -e '1,10d' /etc/services | more
```

当用逗号将两个地址分开时，sed 将把后面的命令应用到从第一个地址开始、到第二个地址结束的范围。在本例中，将 'd' 命令应用到第 1 到 10 行（包括这两行）。所有其它行都被忽略。

带规则表达式的地址

现在演示一个更有用的示例。假设要查看 /etc/services 文件的内容，但是对查看其中包括的注释部分不感兴趣。如您所知，可以通过以 '#' 字符开头的行在 /etc/services 文件中放置注释。为了避免注释，我们希望 sed 删除以 '#' 开始的行。以下是具体做法：

```
$ sed -e '/^#/d' /etc/services | more
```

试一下该例，看看发生了什么。您将注意到，sed 成功完成了预期任务。现在，让我们分析发生的情况。

要理解 '/^#/d' 命令，首先需要对其剖析。首先，让我们除去 'd' -- 这是我们前面所使用的同一个删除行命令。新增加的是 '/^#/' 部分，它是一种新的规则表达式地址。规则表达式地址总是由斜杠括起。它们指定一种 模式，紧跟在规则表达式地址之后的命令将仅适用于正好与该特定模式匹配的行。

因此，'/^#/' 是一个规则表达式。但是，它做些什么呢？很明显，现在该复习规则表达式了。

规则表达式复习

可以使用规则表达式来表示可能会在文本中发现的模式。您在 shell 命令行中用过 '*' 字符吗？这种用法与规则表达式类似，但并不相同。下面是可以在规则表达式中使用的特殊字符：

字符 描述
　　与行首匹配
　　与行末尾匹配
　　与任一个字符匹配
　　将与前一个字符的零或多个出现匹配
[ ] 与 [ ] 之内的所有字符匹配

感受规则表达式的最好方法可能是看几个示例。所有这些示例都将被 sed 作为合法地址接受，这些地址出现在命令的左边。下面是几个示例：

规则
表达式 描述
/./ 将与包含至少一个字符的任何行匹配
/../ 将与包含至少两个字符的任何行匹配
/^#/ 将与以 '#' 开始的任何行匹配
/^$/ 将与所有空行匹配
/}^/ 将与以 '}'（无空格）结束的任何行匹配
/} *^/ 将与以 '}' 后面跟有零或多个空格结束的任何行匹配
/[abc]/ 将与包含小写 'a'、'b' 或 'c' 的任何行匹配
/^[abc]/ 将与以 'a'、'b' 或 'c'开始的任何行匹配

在这些示例中，鼓励您尝试几个。花一些时间熟悉规则表达式，然后尝试几个自己创建的规则表达式。

可以如下使用 regexp：

```
$ sed -e '/regexp/d' /path/to/my/test/file | more
```

这将导致 sed 删除任何匹配的行。然而，通过告诉 sed 打印 regexp 匹配并删除不匹配的内容，而不是与之相反的方法，会更有利于熟悉规则表达式。可以用以下命令这样做：

```
$ sed -n -e '/regexp/p' /path/to/my/test/file | more
```

请注意新的 '-n' 选项，该选项告诉 sed 除非明确要求打印模式空间，否则不这样做。您还会注意到，我们用 'p' 命令替换了 'd' 命令，如您所猜想的那样，这明确要求 sed 打印模式空间。就这样，将只打印匹配部分。

有关地址的更多内容
目前为止，我们已经看到了行地址、行范围地址和 regexp 地址。但是，还有更多的可能。我们可以指定两个用逗号分开的规则表达式，sed 将与所有从匹配第一个规则表达式的第一行开始，到匹配第二个规则表达式的行结束（包括该行）的所有行匹配。例如，以下命令将打印从包含 "BEGIN" 的行开始，并且以包含 "END" 的行结束的文本块：

$ sed -n -e '/BEGIN/,/END/p' /my/test/file | more

如果没发现 "BEGIN"，那么将不打印数据。如果发现了 "BEGIN"，但是在这之后的所有行中都没发现 "END"，那么将打印所有后续行。发生这种情况是因为 sed 面向流的特性 -- 它不知道是否会出现 "END"。

C 源代码示例

如果只要打印 C 源文件中的 main() 函数，可输入：

$ sed -n -e '/main[[:space:]]*(/,/^}/p' sourcefile.c | more

该命令有两个规则表达式 '/main[[:space:]]*(/' 和 '/^}/'，以及一个命令 'p'。第一个规则表达式将与后面依次跟有任意数量的空格或制表键以及开始圆括号的字符串 "main" 匹配。这应该与一般 ANSI C main() 声明的开始匹配。

在这个特别的规则表达式中，出现了 '[[:space:]]' 字符类。这只是一个特殊的关键字，它告诉 sed 与 TAB 或空格匹配。如果愿意的话，可以不输入 '[[:space:]]'，而输入 '['，然后是空格字母，然后是 -V，然后再输入制表键字母和 ']' -- Control-V 告诉 bash 要插入“真正”的制表键，而不是执行命令扩展。使用 '[[:space:]]' 命令类（特别是在脚本中）会更清楚。

好，现在看一下第二个 regexp。'/^}' 将与任何出现在新行行首的 '}' 字符匹配。如果代码的格式很好，那么这将与 main() 函数的结束花括号匹配。如果格式不好，则不会正确匹配 -- 这是执行模式匹配任务的一件棘手之事。

因为是处于 '-n' 安静方式，所以 'p' 命令还是完成其惯有任务，即明确告诉 sed 打印该行。试着对 C 源文件运行该命令 -- 它应该输出整个 main() { } 块，包括开始的 "main()" 和结束的 '}'。

下一篇
既然已经触及了基本知识，我们将在后两篇文章中加快步伐。如果想看一些更丰富的 sed 资料，请耐心一些 -- 马上就有！同时，您可能想查看下列 sed 和规则表达式资源。

通用线程 -- sed 实例，第 2 部分[转贴]
http://www.chinaunix.net 作者:sd-feng 发表于：2002-11-06 17:13:12

sed 是十分强大和小巧的文本流编辑器。在本文章系列的第二篇中，Daniel Robbins 为您演

示如何使用 sed 来执行字符串替换、创建更大的 sed 脚本以及如何使用 sed 的附加、插入和更改行命令。
sed 是很有用（但常被遗忘）的 UNIX 流编辑器。在以批处理方式编辑文件或以有效方式创建 shell 脚本来修改现有文件方面，它是十分理想的工具。本文是前一篇介绍 sed 文章的续篇。

替换！

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，将它们管道输出到 "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 '[:space:]' 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
[:alnum:] 字母数字 [a-z A-Z 0-9]
[:alpha:] 字母 [a-z A-Z]
[:blank:] 空格或制表键

[:cntrl:] 任何控制字符
[:digit:] 数字 [0-9]
[:graph:] 任何可视字符（无空格）
[:lower:] 小写 [a-z]
[:print:] 非控制字符
[:punct:] 标点字符
[:space:] 空格
[:upper:] 大写 [A-Z]
[:xdigit:] 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。

下一篇
在下一篇、也是本 sed 系列的最后一篇文章中，我将为您演示许多使用 sed 来完成不同类型任务的极佳实例。我将不仅为您显示脚本做些什么，还显示为什么那样做。完成之后，您将掌握更多有关如何在不同项目中使用 sed 的极佳知识。到时候见！

通过几个例子看 sed 的模式空间与保持空间
http://www.chinaunix.net 作者:binary 发表于：2003-09-08 11:19:31

近来看到几个不是很常见，但是比较有用的 sed 例子，都和 hold space 与 pattern space 有关，于是对几个例子做了自己认为正确的解释，贴出来与大家共享并请指正，继续讨论

例子一

[color=blue:63cce96b38]sed G[/color:63cce96b38]
在文件每一行下面输出一个空行

[code:1:63cce96b38]
$ cat foo
11111111111111
22222222222222
33333333333333
44444444444444
55555555555555

$ sed G foo
11111111111111

22222222222222

33333333333333

44444444444444

55555555555555

[/code:1:63cce96b38]
解释：

sed 中 G 的用法
The G function appends the contents of the holding area to the contents of the pattern space. The former and new contents are separated by a newline. The maximum number of addresses is two.

hold space : 保持空间（或者叫保留空间、缓冲区）,初始为空
pattern space ： 模式空间

在上面的例子中，将为空的 hold space 附加到文件的每一行后面，所以结果是每一行后面多了一个空行

引申出：
[color=blue:63cce96b38]sed '/^$/d;G'[/color:63cce96b38]
在文件的每一个非空行下面输出一个空行
[color=blue:63cce96b38]sed '/^$/d;G;G'[/color:63cce96b38]
在文件的每一个非空行下面输出两个空行

[code:1:63cce96b38]
$ cat foo
11111111111111
22222222222222

33333333333333
44444444444444
55555555555555

$ sed '/^$/d;G' foo
11111111111111

22222222222222

33333333333333

44444444444444

55555555555555

[/code:1:63cce96b38]

注：有时会有一些由空格符或者 TAB 组成的空行，前面的正则式 [color=green:63cce96b38]^$[/color:63cce96b38] 就不能匹配到这样的行，则可以这样

[color=blue:63cce96b38]sed '/[[:space:]]/d;G'[/color:63cce96b38]

例子二

[color=blue:63cce96b38]sed '/regex/{x;p;x;}'[/color:63cce96b38]
在匹配 regex 的所有行前面插入一个空行
[code:1:63cce96b38]
$ cat foo
11111111111111
22222222222222
test33333333333333
44444444444444
55555555555555

$ sed '/test/{x;p;x;}' foo
11111111111111
22222222222222

test33333333333333
44444444444444
55555555555555
[/code:1:63cce96b38]

解释：
sed 中 x 的用法
The exchange function interchanges the contents of the pattern space and the holding area. The maximum number of addresses is two.
即交换保持空间 hold space 和模式空间 pattern space 的内容

sed 中 p 的作用是把模式空间复制到标准输出。

分析一下该命令执行过程中保持空间和模式空间的内容

命令	保持空间		模式空间		
x	执行前:null	执行后:test\n	执行前:test\n	执行后:null	
p	执行前:null	执行后:test\n	执行前:test\n	执行后:null	输出一个空行
x	执行前:test\n	执行后:null	执行前:null	执行后:test\n	

(注：把 test 所在的行简写为 test 了)

引申：
可以试验一下 sed '/test/{x;p;}' foo 或者 sed '/test/{p;x;}' foo 等，看看结果，体会两个空间的变化

相应的：
[color=blue:63cce96b38]sed '/regex/G'[/color:63cce96b38] 是在匹配 regex 的所有行下面输出一个空行
[color=blue:63cce96b38]sed '/regex/{x;p;x;G;}'[/color:63cce96b38] 是在匹配 regex 的所有行前面和下面都输出一个空行

例子三

[color=blue:63cce96b38]sed 'n;G;'[/color:63cce96b38]
在文件的偶数行下面插入一个空行
[code:1:63cce96b38]
$ cat foo
11111111111111
22222222222222

33333333333333
44444444444444
55555555555555

$ sed 'n;G;' foo
11111111111111
22222222222222

33333333333333
44444444444444

55555555555555
[/code:1:63cce96b38]
解释：
sed 中 n 的用法：将模式空间拷贝于标准输出。用输入的下一行替换模式空间。

执行 n 以后将第一行输出到标准输出以后，然后第二行进入模式空间，根据前面对 G 的解释，会在第二行后面插入一个空行，然后输出；再执行 n 将第三行输出到标准输出，然后第四行进入模式空间，并插入空行，依此类推

相应的：
[color=blue:63cce96b38]sed 'n;n;G'[/color:63cce96b38] 表示在文件的第 3,6,9,12,... 行后面插入一个空行
[color=blue:63cce96b38]sed 'n;n;n;G'[/color:63cce96b38] 表示在文件的第 4,8,12,16,... 行后面插入一个空行
[color=blue:63cce96b38]sed 'n;d'[/color:63cce96b38] 表示删除文件的偶数行

例子四

[color=blue:63cce96b38]sed '$!N;$!D'[/color:63cce96b38]
输出文件最后 2 行，相当于 tail -2 foo

[code:1:63cce96b38]
$ cat foo
11111111111111
22222222222222
33333333333333
44444444444444
55555555555555

$ sed '$!N;$!D' foo
44444444444444

55555555555555

[/code:1:63cce96b38]

解释：

D 删除模式空间内第一个 newline 字母 \n 前的资料。
N 把输入的下一行添加到模式空间中。

sed '$!N;$!D' ：对文件倒数第二行以前的行来说，N 将当前行的下一行放到模式空间中以后，D 就将模式空间的内容删除了；到倒数第二行的时候，将最后一行附加到倒数第二行下面，然后最后一行不执行 D ，所以文件的最后两行都保存下来了。

还有 N 的另外一种用法
[code:1:63cce96b38]
$ sed = foo | sed N
1
11111111111111
2
22222222222222
3
33333333333333
4
44444444444444
5
55555555555555

$ sed = foo | sed 'N;s/\n/ /'
1 11111111111111
2 22222222222222
3 33333333333333
4 44444444444444
5 55555555555555
[/code:1:63cce96b38]
解释：
N 的作用是加上行号，可以用于格式化输出文件

例子五

[color=blue:63cce96b38]sed '1!G;h;$!d'[/color:63cce96b38]
[color=blue:63cce96b38]sed -n '1!G;h;$p'[/color:63cce96b38]

将文件的行反序显示，相当于 tac 命令(有些平台没有这个命令)

[code:1:63cce96b38]
$ cat foo
11111111111111
22222222222222
33333333333333

$ sed '1!G;h;$!d' foo
33333333333333
22222222222222
11111111111111

$ sed -n '1!G;h;$p' foo
33333333333333
22222222222222
11111111111111
[/code:1:63cce96b38]

解释：
sed 中 h 用法：h
The h (hold) function copies the contents of the pattern space into a holding area, destroying any previous contents of the holding area.
意思是将模式空间的内容保存到保持空间中去

sed 中的 d 表示删除模式空间。

1!G 表示除了第一行以外，其余行都执行 G 命令；$!d 表示除了最后一行以外，其余行都执行 d 命令。

看一下[color=blue:63cce96b38]sed '1!G;h;$!d'[/color:63cce96b38]命令执行过程中保持空间与模式空间的变化:

命令		保持空间		模式空间	
第一行	h;d	执行前:null	执行后:1111\n	执行前:1111\n	执行后:null
第二行	G;h;d	执行前:1111	执行后:2222\n1111\n	执行前:2222\n	执行后:null
第二行	G;h	执行前:2222\1111\n	执行后:3333\n2222\n\1111\n	执行前:3333\n	执行后:3333\n2222\n\1111\n

(注：把各个行简写了)

这样输出以后就是文件的反序了。

题外话：在 vi 中对一个文件进行反序显示的命令是 [color=blue:63cce96b38]
:g/./m0
[/color:63cce96b38] ，意思是按照文件正常顺序每找到一行，就把该行放到文件的最上面一行去，这样循环一下正好把文件的行反序显示了。

这只是自己对模式空间与保持空间的一点点理解，个人觉得在这个方面还有很多东西要学，请大家拍砖。

---

admirer 回复于：2003-09-08 11:34:30

---

binary 回复于：2003-09-08 20:51:26
发现一个错误
[color=blue:7cee071f20]sed 'N;D;'　　输出文件最后 1 行，相当于 tail -1 [/color:7cee071f20]
刚才试验了一下是不对的，结果是没有输出

我在 windows 平台调试的这个命令是 tail -1 的结果，但是忘了把这个命令在 UNIX 上面试验一下了，致歉！

---

yoursmile 回复于：2003-10-03 18:55:15
[quote:a9fae792bf="binary"]
注：有时会有一些由空格符或者 TAB 组成的空行，前面的正则式 ^$ 就不能匹配到这样的行，则可以这样

sed '/[[:space:]]/d;G'
[/quote:a9fae792bf]

[code:1:a9fae792bf]$cat foo
11111111111111
22222222222222
　(这一行有三个空格键)
33333333333333
44444444444444
55555555555555[/code:1:a9fae792bf]
[code:1:a9fae792bf]

$sed '/[[:space:]]/d;G' foo

11111111111111
[/code:1:a9fae792bf]

好象和楼主说的不一样。。。。。

---

admirer 回复于：2003-10-03 19:32:12
[quote:80f24fb8dd="yoursmile"]

好象和楼主说的不一样。。。。。

[/quote:80f24fb8dd]
应该可以的！
不过为了保险，还是这样写较好一些：
[code:1:80f24fb8dd]sed '/^[[:space:]]\{1,\}$/d;$!G' filename[/code:1:80f24fb8dd]

---

yoursmile 回复于：2003-10-03 19:50:41
[quote:765a15f568="admirer"][/quote:765a15f568]

老大，你的那个语句我看的懂。也能正确执行。

[code:1:765a15f568]sed '/[[:space:]]/d;G' [/code:1:765a15f568]

这个语句我就看不懂了。先删除掉含有一个空格的行？
可是结果并不是这样的。
即使那个行只有一个空格。

补充：我在 red hat linux 7.3 的操作环境下。

---

admirer 回复于：2003-10-03 19:59:25
[quote:5b90ccb28f="yoursmile"]
这个语句我就看不懂了。先删除掉含有一个空格的行？
可是结果并不是这样的。
即使那个行只有一个空格。

补充：我在 red hat linux 7.3 的操作环境下。[/quote:5b90ccb28f]
[b:5b90ccb28f]sed '/[[:space:]]/d;G' [/b:5b90ccb28f]
这段代码在文件中行内包含空格时肯定会删除的，而不会理会它是不是还有其他内容！所以只能处理特定的文件。

---

yoursmile 回复于：2003-10-03 20:17:19
[quote:013ba0ce56="admirer"]sed '/[[:space:]]/d;G'
这段代码在文件中行内包含空格时肯定会删除的，而不会理会它是不是还有其他内容！所以只能处理特定的文件。 [/quote:013ba0ce56]

原来如此，终于明白了。
搞定！~~

原来是我在复制的时候，多复制了每行后面的空格。
造成 foo 文件有的行最后有空格。
谢谢[size=24:013ba0ce56]admirer[/size:013ba0ce56]~~

---

admirer 回复于：2003-10-03 20:40:59
[quote:98aea365ec="yoursmile"]

[/quote:98aea365ec]
同感。我也常犯这样的错误，末尾的空格往往会神不知鬼不觉的给人一些难堪！

---

c1l2d3 回复于：2003-10-05 10:13:12
好喔！又学到不少

如何用 sed 删除由空格组成的空行？

http://www.chinaunix.net 作者:binary 发表于：2002-07-26 17:13:55

我知道可以用 sed /^$/d 删除文件中的空行，但是不能删除由空格组成的空行，如果某一行上只有几个空格，前面的命令是不能匹配该行的。

记得原来看到过删除这种空行的 sed 命令，是匹配/^/和/$/之间没有字符的做法，可是试了一下没有成功，哪位知道的请赐教？

【发表回复】【查看 CU 论坛原帖】【关闭】

---

valentine 回复于：2002-07-27 19:59:27

```
sed /^[             ]*$/d tt
#the bracket contains a SPACE and a TAB
```

---

laoju 回复于：2002-07-28 11:01:46

```
sed /^[[:space:]]*$/d
```

---

valentine 回复于：2002-07-28 11:31:15

[[:space:]] 比 [[:blank:]]或[\t] 处理的情况更多一些.不错.

---

laoxia 回复于：2002-11-26 07:14:03

高人指点啊，怎么跑不出来啊

```
$ more text
s_f
f_t
s_f
f_t

$ sed '/^[[:space:]]*$/d' text
s_f
f_t
s_f
f_t
```

---

binary 回复于：2002-11-26 10:15:22
你的文件中没有空行，所以运行后文件内容没有变化

---

vavaForet 回复于：2002-11-27 16:45:32
有趣！~~

sed 中如何替换出新行来

http://www.chinaunix.net 作者:bjchenxu 发表于：2002-08-16 15:12:09

[这个贴子最后由 bjchenxu 在 2002/08/19 12:24pm 编辑]

操作系统 SunOS 5.9 sun4u sparc SUNW,Ultra-250
原文件名 test 仅一行
hellohellohello

要求结果：
hello
hello
hello

我使用
sed 's/hello/hello^M/g' test #^M 为 ctrl_V ctrl_M
不能实现，请问有何错误

【发表回复】【查看 CU 论坛原帖】【关闭】

---

valentine 回复于：2002-08-16 15:22:35
sed 是以行为处理单位的.
随便处理一下吧:
cat tt|sed 's/hello/hello@/g'|tr '@' '\n'

---

binary 回复于：2002-08-16 15:31:07
版主真是信手捻来皆成贴啊

---

bjchenxu 回复于：2002-08-16 15:33:22
感谢感谢，回答得太快太好了
看来仅有 sed 是无法生成新行的，必须 tr 的帮助

---

valentine 回复于：2002-08-16 17:15:08
问的是经典问题.

---

goodboy 回复于：2002-08-19 11:00:50
斑竹，要我有一行，为 abcdefghijklmnopqrstuvwxy
按照每五个字符就换行怎么做？

---

valentine 回复于：2002-08-19 11:39:37
用 awk 处理一下吧.

---

bjchenxu 回复于：2002-08-19 12:20:39
如果使用 VI 的话
:%s/...../&^M/g
注意：^M 是由 ctrl_v ctrl_m 产生的

---

bjchenxu 回复于：2002-08-19 14:02:38
如果使用 sed 的话
sed 's/\(.....\)/\1@/g' file | tr '@' '\n'
如果想每 m 个字符换行
sed 's/\(.\{m,m\}\)/\1@/g' file | tr '@' '\n'
献丑了

---

valentine 回复于：2002-08-19 14:43:43
bjchenxu 回答的不错.

---

bjchenxu 回复于：2002-08-19 14:46:57
没有版主的提示，也就没有这个答案，呵呵
再次感谢 valentine

---

bjchenxu 回复于：2002-08-19 15:53:26

再减少几个字吧，呵呵

如果使用 sed 的话
sed 's/...../&@/g' file | tr '@' '\n'
如果想每 m 个字符换行
sed 's/.\{m,m\}/&@/g' file | tr '@' '\n'

---

bjchenxu 回复于：2002-08-19 16:44:12
[这个贴子最后由 bjchenxu 在 2002/08/19 04:50pm 编辑]

本贴为我想出的本主题的答案：不用 tr 也能解决问题

原文件名 test，仅一行
hellohellohello

要求结果：
hello
hello
hello

答案：
我 edit 一个脚本文件
sedfile
内容为
s/hello/\
&/g

运行命令
sed -f sedfile test 即可增加新行

---

menghan 回复于：2002-08-19 16:44:16
真是 vi 高手，我虽然不是高手，但是我喜欢 vi

---

valentine 回复于：2002-08-19 17:24:35
[这个贴子最后由 valentine 在 2002/08/19 05:32pm 编辑]

bjchenxu 精神可嘉啊.
不过这样更符合要求一些:
sedfile
内容改为
s/hello/&\

/g

这样算是彻底的 sed 答案了.
哈哈,俺也学到新东西了.好象&用起来比\(\)的子表达式形式简单啊.不过,不太好理解.
也谢谢 bjchenxu .

---

valentine 回复于：2002-08-20 09:04:40
[这个贴子最后由 valentine 在 2002/08/20 09:30am 编辑]

更准确的答案在这里,(看来俺开始回答的草率了些,不过 tr 的方法也是一种思路)
ksh:
sed 's/hello/&\^J/g' test
or
sed 's/hello/hello\^J/g' test

^J 的输入是 ctrl+V ctrl+J
Bourne shell:

sed 's/hello/&\
/g' test

\后是回车.

---

bjchenxu 回复于：2002-08-20 09:20:28
[这个贴子最后由 bjchenxu 在 2002/08/20 09:24am 编辑]

怪哉
此方法在 solaris 9 的 csh 中不行，但是在 bash 下却可以

另外，在 bash 中，可以直接回车，不用 ctrl_v + ctrl_j,hehe

---

valentine 回复于：2002-08-20 09:32:10
Bourne shell 也是回车.我忘了写了,现在加上了.
csh 俺一直不喜欢.

如何使用 vi or sed ，将文件中的“\n" 替换成“\r\n"

http://www.chinaunix.net 作者:f980215 发表于：2002-07-23 08:11:23

如何使用 vi or sed ，将文件中的“\n" 替换成“\r\n"?

\n ==> 0AH ===> Ctrl+J　　\r ==> ODH ===> Ctrl+M

我用命令 sed 's/^V^J/^J^M/g' file > new_file, 不能成功。

thanks
f980215

【发表回复】【查看 CU 论坛原帖】【关闭】

---

mymm 回复于：2002-07-23 17:18:41
sed -e '%s/\\n/\\r\\n/g'

---

f980215 回复于：2002-07-23 17:43:55
thanks for your reply.

不行!!

f980215

---

jinhg 回复于：2002-07-23 20:12:13
你的 \n 在文本中是一个字符，还是两个字符。
在文本中^M 是显示为一个字符。

---

f980215 回复于：2002-07-24 08:13:43
我用

* LINUX： od -c file 命令， 发现是一个字符（\n）.
* WIN98/DOS： debug file, 发现是一个字符（0A .

由于在 linux 下的软件产生的文件，因 （\n）问题，

我在 WIN98 下 用

* NOTEPAD 打开 file 时，所有行多在同一行显示。
* 用网页方式打开 file 时，也一样内容多在同一行.

THANKS
f980215

---

valentine 回复于：2002-07-24 08:33:52
sed 's/$/^M/g' <unixfile >dosfile #Use CTRL+V then CTRL+M to type in ^M
in vi:
use:
1,$ s/$/\^M/g to and 0xod before 0x0a

请教有关 sed 的用法!!!! 在线等待

http://www.chinaunix.net 作者:黑骏马 发表于：2003-09-26 11:58:48

cat filename
1 11111111111111111
2 222222222222222222
3 333333333333333333
4 444444444444444444

sed -n -e '/^3/{=;x;1!p;g;$!N;p;D;}' -e h filename
3 #匹配行的行号
2 222222222222222222 #上一行
3 333333333333333333 #匹配行
4 444444444444444444 #下一行
请问 D 的具体用法？和上面是怎么执行的，想了一晚上都没想出来？？？？
谢谢!!!!!!!

【发表回复】【查看 CU 论坛原帖】【关闭】

---

fieryfox 回复于：2003-09-27 16:28:18
第一行和第二行时执行的是 h 命令；
匹配到第三行/^3/，此时 hold space 里是第二行的内容，pattern space 里是第三行的内容，执行：
=打印行号，3
x 交换 hold 和 pattern space
1!p，打印 pattern，此时为第二行。这里的 1!其实无用。
g 从 hold 拷贝内容到 pattern，这样 hold 和 pattern 都是第三行了。
$!N 如果不是文件的最后一行则取下一行，添加到 pattern，于是现在 pattern 里现在是 3 和 4 两行了。
p 打印 pattern，输出 3、4 行。
D 删除 pattern 里的第一行，即文件的第三行，并从头开始执行 script。但因为有/^3/限制，所以不会再执行了。

你可以把/^3/去掉看看是什么效果。如果还是不能看清楚，这样：
sed -n -e '{=;x;1!p;g;$!N;p;D;}' -e 'a/++++/' filename
与
sed -n -e '{=;x;1!p;g;$!N;p;}' -e 'a/++++/' filename
比较一下就清楚了。

---

admirer 回复于：2003-09-27 19:18:47
[quote:d7faa18d0f="fieryfox"]第一行和第二行时执行的是 h 命令；
匹配到第三行/^3/，此时 hold space 里是第二行的内容，pattern space 里是第三行的内容，执行：
=打印行号，3
x 交换 hold 和 pattern space
1!p，打印 pattern，此时为第二行。这里的.....[/quote:d7faa18d0f]
说得是！
不过在 SCOUNIX 下
这两行需要稍作修改(可能是笔下之误吧！)
sed -n -e '{=;x;1!p;g;$!N;p;D;}' -e 'a[color=red:d7faa18d0f]\[/color:d7faa18d0f]++++[color=red:d7faa18d0f]\[/color:d7faa18d0f]' filename
与
sed -n -e '{=;x;1!p;g;$!N;p;}' -e 'a[color=red:d7faa18d0f]\[/color:d7faa18d0f]++++[color=red:d7faa18d0f]\[/color:d7faa18d0f]' filename
实际上直接用

[code:1:d7faa18d0f]sed -n -e '{=;x;1!p;g;$!N;p;D;}' filename[/code:1:d7faa18d0f]
就能够说明问题了：
[code:1:d7faa18d0f]sed -n -e '{=;x;1!p;g;$!N;p;D;}' filename
1
1 11111111111111111
2 22222222222222222
2
1 11111111111111111
2 22222222222222222
3 33333333333333333
3
2 22222222222222222
3 33333333333333333
4 444444444444444444
4
3 33333333333333333
4 444444444444444444[/code:1:d7faa18d0f]

---

fieryfox 回复于：2003-09-28 09:05:33
呵呵，不是笔误。用惯了 gsed 了。
另外加上 a 的目的，是为了可以看出 D 的真正作用，此时后边的 a 根本就没有执行。只有没有 D 命令时 a 才会执行。

黑骏马 回复于：2003-09-28 16:00:32
明白了，我本来应该明白的，可能是这几天学这些“学习笔记”，学昏了的缘故吧，很感谢你们，自己还有很多东西要学习，向你们学习!!

[原创]：说说 sed 中引号的用法（抛砖引玉!）

http://www.chinaunix.net 作者:admirer 发表于：2003-08-02 08:43:01

引号（[color=darkblue:c1772557da]"[/color:c1772557da][color=red:c1772557da]'[/color:c1772557da]）在 shell 编程中起着相当重要的脚色，应用得当，则程序易写易读，简洁明快，否则，会让人头痛不已！兵书说“运用之妙，存乎于心!”，下面就自己最近学习中的一点心得，与朋友们交流，希望得到大家的斧正！
其实在 sed 中使用 shell 变量相对来说还是比较简单的，关键是引号的用法：

比如：
借用[b:c1772557da]yoursmile[/b:c1772557da]朋友的数据
[code:1:c1772557da]cat file
1 192.148.99.253 [17/Jun/2003:11:25:44 /sc
2 192.148.99.253 [17/Jun/2003:11:18:21 /si
1 192.148.99.253 [17/Jun/2003:11:20:34 /sp
2 192.148.99.253 [17/Jun/2003:11:18:13 /ap
1 192.148.99.253 [17/Jun/2003:11:17:30 /hou/
1 192.93.108.187 [17/Jun/2003:14:49:14 /sc
3 192.93.108.187 [17/Jun/2003:14:39:11 /si
5 192.68.82.78 [05/Jun/2003:00:05:45 /hou/
9 192.68.82.78 [05/Jun/2003:00:05:45 /ss
4 192.228.210.10 [16/Jun/2003:09:29:30 /hou/[/code:1:c1772557da]d=6
取文件的第六行，则
单引号
[code:1:c1772557da]sed -n ""$d"'/p' file
1 192.93.108.187 [17/Jun/2003:14:49:14 /sc[/code:1:c1772557da]
双引号
[code:1:c1772557da]sed -n "$d"p file or sed -n "${d}p" file
1 192.93.108.187 [17/Jun/2003:14:49:14 /sc[/code:1:c1772557da]
结果完全相同。
又：
取文件的第 1 到第 6 行，则：
单引号：
[code:1:c1772557da]sed -n '1,'"$d"'p' file
1 192.148.99.253 [17/Jun/2003:11:25:44 /sc
2 192.148.99.253 [17/Jun/2003:11:18:21 /si
1 192.148.99.253 [17/Jun/2003:11:20:34 /sp
2 192.148.99.253 [17/Jun/2003:11:18:13 /ap
1 192.148.99.253 [17/Jun/2003:11:17:30 /hou/
1 192.93.108.187 [17/Jun/2003:14:49:14 /sc[/code:1:c1772557da]
双引号：
[code:1:c1772557da]sed -n "1,$d"p file or sed -n "1,${d}p" file
1 192.148.99.253 [17/Jun/2003:11:25:44 /sc
2 192.148.99.253 [17/Jun/2003:11:18:21 /si
1 192.148.99.253 [17/Jun/2003:11:20:34 /sp
2 192.148.99.253 [17/Jun/2003:11:18:13 /ap
1 192.148.99.253 [17/Jun/2003:11:17:30 /hou/
1 192.93.108.187 [17/Jun/2003:14:49:14 /sc[/code:1:c1772557da]
取第 6 行到文件尾
单引号：
[code:1:c1772557da]sed -n ""$d"',$p' file
1 192.93.108.187 [17/Jun/2003:14:49:14 /sc
3 192.93.108.187 [17/Jun/2003:14:39:11 /si

5 192.68.82.78 [05/Jun/2003:00:05:45 /hou/
9 192.68.82.78 [05/Jun/2003:00:05:45 /ss
4 192.228.210.10 [16/Jun/2003:09:29:30 /hou/[/code:1:c1772557da]
双引号：
[code:1:c1772557da]sed -n "$d,\$p" file or sed -n "$d,$"p
1 192.93.108.187 [17/Jun/2003:14:49:14 /sc
3 192.93.108.187 [17/Jun/2003:14:39:11 /si
5 192.68.82.78 [05/Jun/2003:00:05:45 /hou/
9 192.68.82.78 [05/Jun/2003:00:05:45 /ss
4 192.228.210.10 [16/Jun/2003:09:29:30 /hou/[/code:1:c1772557da]
如果上面的 6 保存在一个 line 的文件中(该文件中只此一行时！)，则
单引号：
[code:1:c1772557da]sed -e ""`cat line`"p' file
1 192.93.108.187 [17/Jun/2003:14:49:14 /sc[/code:1:c1772557da]
双引号：
[code:1:c1772557da]sed -n "`cat line`"p file
1 192.93.108.187 [17/Jun/2003:14:49:14 /sc[/code:1:c1772557da]
从上面的例子可以看出，引号的应用，是乎是有讲究的，只要灵活应用，就可使代码简洁明快，看起来舒服，用起来舒心，写起来方便！
简而言之：凡是要引用 shell 变量时，最好使用双引号“[color=red:c1772557da]"[/color:c1772557da]”做 sed 的定界符，这样会更方便些，但要使用双引号一定要注意避免 shell 解释 sed 命令，比如“[color=red:c1772557da]$p[/color:c1772557da]”原意为打印文件末行，但 shell 会解释为取变量[color=red:c1772557da]p[/color:c1772557da]的值，因此在使用中要加以注意，可用“[color=red:c1772557da]\$p[/color:c1772557da]”或“[color=red:c1772557da]"$"p[/color:c1772557da]”等方式来解决此类问题。

---

grljt 回复于：2003-08-02 20:58:59
谢谢！好东东！建议大师用颜色区分一下各引号，便于初学者。

加精吧!!!

---

yoursmile 回复于：2003-08-10 20:59:19
Very Good!

总结的好啊。

---

bjgirl 回复于：2003-08-10 23:33:27
虽说" '这类的符号在 shell 中很不起眼,但是能真正把这些符号运用得自如的却很少!
--[color=red:ac4f4dbcc0]我们要用符号控制我们的电脑![/color:ac4f4dbcc0]

---

yoursmile 回复于：2003-08-11 08:29:05
[quote:d1d61fa6af="bjgirl"]
我们要用符号控制我们的电脑!
[/quote:d1d61fa6af]

有点预言家的味道！~

---

tiansgx 回复于：2003-08-11 09:40:52
收了

阿骁 回复于：2003-08-11 10:26:01
获益非浅 ...

---

xingj_h 回复于：2003-08-11 10:29:07
简直就是符号杀手!

---

admirer 回复于：2003-08-11 12:21:28
[quote:8456fca6ac="xingj_h"]简直就是符号杀手！[/quote:8456fca6ac]

太夸张了吧？！
希望能尽早看见你的[color=red:8456fca6ac]符号[/color:8456fca6ac]精灵！
加油！！

---

fieryfox 回复于：2003-08-11 13:41:53
对引号处理不清楚的，建议读：
Unix? Shell Programming, Third Edition
Chapter 6. Can I Quote You on That?

概括如下：
1、单引号：所有内容都留给应用解释
2、双引号：除 Dollar signs，Back quotes，Backslashes 外，所有内容留给应用解释。
3、Backslash 相当于将跟随字符用单引号处理。

帮楼主明确一下，呵呵。

---

admirer 回复于：2003-08-11 17:37:19
[quote:e185499e3b="fieryfox"]
概括如下：
概括如下：
1、单引号：所有内容都留给应用解释
2、双引号：除 Dollar signs，Back quotes，Backslashes 外，所有内容留给应用解释。
3、Backslash 相当于将跟随字符用单引号处理。
[/quote:e185499e3b]
简明扼要！

---

coolscplayer 回复于：2003-08-11 23:34:11
顶~~
admirer.....
汗.....

---

unixgood 回复于：2003-08-12 13:21:24

真的是 SHELL 高手中的高手，各位观众~~~~~~~~~~~~~~~~~~~
那就是~~~~~~~~~~~~~~~~~~高高手!

---

li2002 回复于：2003-08-13 08:38:29
Unix& Shell Programming, Third Edition 有中文版吗？

---

yuxq 回复于：2003-09-09 14:55:39
wo ding!

光明晓仙 回复于：2003-09-09 16:16:42
[quote:7cfa13c715="bjgirl"][/quote:7cfa13c715]

就是啊，有时搞不清呀.....