
SED单行脚本快速参考（Unix 流编辑器） 2005年12月29日

英文标题：USEFUL ONE-LINE SCRIPTS FOR SED (Unix stream editor)
原标题：HANDY ONE-LINERS FOR SED (Unix stream editor)

整理：Eric Pement - 电邮：pemente[at]northpark[dot]edu 版本5.5
译者：Joe Hong - 电邮：hq00e[at]126[dot]com

在以下地址可找到本文档的最新（英文）版本：
http://sed.sourceforge.net/sed1line.txt
http://www.pement.org/sed/sed1line.txt

其他语言版本：
中文 - http://sed.sourceforge.net/sed1line_zh-CN.html
捷克语 - http://sed.sourceforge.net/sed1line_cz.html
荷语 - http://sed.sourceforge.net/sed1line_nl.html
法语 - http://sed.sourceforge.net/sed1line_fr.html
德语 - http://sed.sourceforge.net/sed1line_de.html

葡语 - http://sed.sourceforge.net/sed1line_pt-BR.html

文本间隔：

```
# 在每一行后面增加一空行
sed G

# 将原来的所有空行删除并在每一行后面增加一空行。
# 这样在输出的文本中每一行后面将有且只有一空行。
sed '/^$/d;G'

# 在每一行后面增加两行空行
sed 'G;G'

# 将第一个脚本所产生的所有空行删除（即删除所有偶数行）
sed 'n;d'

# 在匹配式样“regex”的行之前插入一空行
sed '/regex/{x;p;x;}'

# 在匹配式样“regex”的行之后插入一空行
sed '/regex/G'

# 在匹配式样“regex”的行之前和之后各插入一空行
sed '/regex/{x;p;x;G;}'
```

编号：

```
# 为文件中的每一行进行编号（简单的左对齐方式）。这里使用了“制表符”
#  （tab，见本文末尾关于'\t'的用法的描述）而不是空格来对齐边缘。
sed = filename | sed 'N;s/\n/\t/'

# 对文件中的所有行编号（行号在左，文字右端对齐）。
sed = filename | sed 'N; s/^/     /; s/ *\(.\{6,\}\)\n/\1  /'

# 对文件中的所有行编号，但只显示非空白行的行号。
```

```
sed '/./=' filename | sed '/./N; s/\n/ /'

# 计算行数 (模拟 "wc -l")
sed -n '$='
```

文本转换和替代:

```
# Unix环境: 转换DOS的新行符 (CR/LF) 为Unix格式。
sed 's/.$//'                 # 假设所有行以CR/LF结束
sed 's/^M$//'                # 在bash/tcsh中, 将按Ctrl-M改为按Ctrl-V
sed 's/\x0D$//'              # ssed、gsed 3.02.80, 及更高版本

# Unix环境: 转换Unix的新行符 (LF) 为DOS格式。
sed "s/$/`echo -e \\\r`/"        # 在ksh下所使用的命令
sed 's/$"/`echo \\\r`/"          # 在bash下所使用的命令
sed "s/$/`echo \\\r`/"           # 在zsh下所使用的命令
sed 's/$/\r/'                    # gsed 3.02.80 及更高版本

# DOS环境: 转换Unix新行符 (LF) 为DOS格式。
sed "s/$//"                      # 方法 1
sed -n p                         # 方法 2

# DOS环境: 转换DOS新行符 (CR/LF) 为Unix格式。
# 下面的脚本只对UnxUtils sed 4.0.7 及更高版本有效。要识别UnxUtils版本的
#  sed可以通过其特有的 "--text" 选项。你可以使用帮助选项 ( "--help" ) 看
# 其中有无一个 "--text" 项以此来判断所使用的是否是UnxUtils版本。其它DOS
# 版本的的sed则无法进行这一转换。但可以用 "tr" 来实现这一转换。
sed "s/\r//" infile >outfile     # UnxUtils sed v4.0.7 或更高版本
tr -d \r <infile >outfile        # GNU tr 1.22 或更高版本

# 将每一行前导的 "空白字符" (空格, 制表符) 删除
# 使之左对齐
sed 's/^[ \t]*//'                # 见本文末尾关于'\t'用法的描述

# 将每一行拖尾的 "空白字符" (空格, 制表符) 删除
sed 's/[ \t]*$//'                # 见本文末尾关于'\t'用法的描述

# 将每一行中的前导和拖尾的空白字符删除
sed 's/^[ \t]*//;s/[ \t]*$//'

# 在每一行开头处插入5个空格 (使全文向右移动5个字符的位置)
sed 's/^/     /'

# 以79个字符为宽度, 将所有文本右对齐
sed -e :a -e 's/^.\{1,78\}$/ &/;ta'  # 78个字符外加最后的一个空格

# 以79个字符为宽度, 使所有文本居中。在方法1中, 为了让文本居中每一行的前
# 头和后头都填充了空格。 在方法2中, 在居中文本的过程中只在文本的前面填充
# 空格, 并且最终这些空格将有一半会被删除。此外每一行的后头并未填充空格。
sed  -e :a -e 's/^.\{1,77\}$/ & /;ta'                     # 方法1
sed  -e :a -e 's/^.\{1,77\}$/ &/;ta' -e 's/\( *\)\1/\1/'  # 方法2

# 在每一行中查找字串 "foo", 并将找到的 "foo" 替换为 "bar"
sed 's/foo/bar/'                 # 只替换每一行中的第一个 "foo" 字串
sed 's/foo/bar/4'                # 只替换每一行中的第四个 "foo" 字串
sed 's/foo/bar/g'                # 将每一行中的所有 "foo" 都换成 "bar"
sed 's/\(.*\)foo\(.*foo\)/\1bar\2/' # 替换倒数第二个 "foo"
sed 's/\(.*\)foo/\1bar/'             # 替换最后一个 "foo"
```

```
# 只在行中出现字串“baz”的情况下将“foo”替换成“bar”
sed '/baz/s/foo/bar/g'

# 将“foo”替换成“bar”，并且只在行中未出现字串“baz”的情况下替换
sed '/baz/!s/foo/bar/g'

# 不管是“scarlet”“ruby”还是“puce”，一律换成“red”
sed 's/scarlet/red/g;s/ruby/red/g;s/puce/red/g'   #对多数的sed都有效
gsed 's/scarlet\|ruby\|puce/red/g'                # 只对GNU sed有效

# 倒置所有行，第一行成为最后一行，依次类推（模拟“tac”）。
# 由于某些原因，使用下面命令时HHsed v1.5会将文件中的空行删除
sed '1!G;h;$!d'               # 方法1
sed -n '1!G;h;$p'             # 方法2

# 将行中的字符逆序排列，第一个字成为最后一字，……（模拟“rev”）
sed '/\n/!G;s/\(.\)\(.*\n\)/&\2\1/;//D;s/.//'

# 将每两行连接成一行（类似“paste”）
sed '$!N;s/\n/ /'

# 如果当前行以反斜杠“\”结束，则将下一行并到当前行末尾
# 并去掉原来行尾的反斜杠
sed -e :a -e '/\\$/N; s/\\\n//; ta'

# 如果当前行以等号开头，将当前行并到上一行末尾
# 并以单个空格代替原来行头的“=”
sed -e :a -e '$!N;s/\n=/ /;ta' -e 'P;D'

# 为数字字串增加逗号分隔符号，将“1234567”改为“1,234,567”
gsed ':a;s/\B[0-9]\{3\}\>/,&/;ta'                     # GNU sed
sed -e :a -e 's/\(.*[0-9]\)\([0-9]\{3\}\)/\1,\2/;ta'  # 其他sed

# 为带有小数点和负号的数值增加逗号分隔符（GNU sed）
gsed -r ':a;s/(^|[^0-9.])([0-9]+)([0-9]{3})/\1\2,\3/g;ta'

# 在每5行后增加一空白行 （在第5，10，15，20，等行后增加一空白行）
gsed '0~5G'                   # 只对GNU sed有效
sed 'n;n;n;n;G;'              # 其他sed
```

选择性地显示特定行：

```
# 显示文件中的前10行 （模拟“head”的行为）
sed 10q

# 显示文件中的第一行 （模拟“head -1”命令）
sed q

# 显示文件中的最后10行 （模拟“tail”）
sed -e :a -e '$q;N;11,$D;ba'

# 显示文件中的最后2行（模拟“tail -2”命令）
sed '$!N;$!D'

# 显示文件中的最后一行（模拟“tail -1”）
sed '$!d'                     # 方法1
sed -n '$p'                   # 方法2

# 显示文件中的倒数第二行
```

```
sed -e '$!{h;d;}' -e x                  # 当文件中只有一行时，输入空行
sed -e '1{$q;}' -e '$!{h;d;}' -e x  # 当文件中只有一行时，显示该行
sed -e '1{$d;}' -e '$!{h;d;}' -e x  # 当文件中只有一行时，不输出

# 只显示匹配正则表达式的行（模拟“grep”）
sed -n '/regexp/p'                  # 方法1
sed '/regexp/!d'                    # 方法2

# 只显示“不”匹配正则表达式的行（模拟“grep -v”）
sed -n '/regexp/!p'                 # 方法1，与前面的命令相对应
sed '/regexp/d'                     # 方法2，类似的语法

# 查找“regexp”并将匹配行的上一行显示出来，但并不显示匹配行
sed -n '/regexp/{g;1!p;};h'

# 查找“regexp”并将匹配行的下一行显示出来，但并不显示匹配行
sed -n '/regexp/{n;p;}'

# 显示包含“regexp”的行及其前后行，并在第一行之前加上“regexp”所
# 在行的行号 （类似“grep -A1 -B1”）
sed -n -e '/regexp/{=;x;1!p;g;$!N;p;D;}' -e h

# 显示包含“AAA”、“BBB”或“CCC”的行（任意次序）
sed '/AAA/!d; /BBB/!d; /CCC/!d'  # 字串的次序不影响结果

# 显示包含“AAA”、“BBB”和“CCC”的行（固定次序）
sed '/AAA.*BBB.*CCC/!d'

# 显示包含“AAA” “BBB”或“CCC”的行 （模拟“egrep”）
sed -e '/AAA/b' -e '/BBB/b' -e '/CCC/b' -e d     # 多数sed
gsed '/AAA\|BBB\|CCC/!d'                         # 对GNU sed有效

# 显示包含“AAA”的段落 （段落间以空行分隔）
# HHsed v1.5 必须在“x;”后加入“G;”，接下来的3个脚本都是这样
sed -e '/./{H;$!d;}' -e 'x;/AAA/!d;'

# 显示包含“AAA” “BBB”和“CCC”三个字串的段落 （任意次序）
sed -e '/./{H;$!d;}' -e 'x;/AAA/!d;/BBB/!d;/CCC/!d'

# 显示包含“AAA”、“BBB”、“CCC”三者中任一字串的段落 （任意次序）
sed -e '/./{H;$!d;}' -e 'x;/AAA/b' -e '/BBB/b' -e '/CCC/b' -e d
gsed '/./{H;$!d;};x;/AAA\|BBB\|CCC/b;d'          # 只对GNU sed有效

# 显示包含65个或以上字符的行
sed -n '/^.\{65\}/p'

# 显示包含65个以下字符的行
sed -n '/^.\{65\}/!p'               # 方法1，与上面的脚本相对应
sed '/^.\{65\}/d'                   # 方法2，更简便一点的方法

# 显示部分文本——从包含正则表达式的行开始到最后一行结束
sed -n '/regexp/,$p'

# 显示部分文本——指定行号范围（从第8至第12行，含8和12行）
sed -n '8,12p'                      # 方法1
sed '8,12!d'                        # 方法2

# 显示第52行
sed -n '52p'                        # 方法1
sed '52!d'                          # 方法2
```

```
sed '52q;d'                             # 方法3, 处理大文件时更有效率

# 从第3行开始, 每7行显示一次
gsed -n '3~7p'                          # 只对GNU sed有效
sed -n '3,${p;n;n;n;n;n;n;}'      # 其他sed

# 显示两个正则表达式之间的文本（包含）
sed -n '/Iowa/,/Montana/p'          # 区分大小写方式
```

选择性地删除特定行:

```
# 显示通篇文档, 除了两个正则表达式之间的内容
sed '/Iowa/,/Montana/d'

# 删除文件中相邻的重复行（模拟“uniq”）
# 只保留重复行中的第一行, 其他行删除
sed '$!N; /^\(.*\)\n\1$/!P; D'

# 删除文件中的重复行, 不管有无相邻。注意hold space所能支持的缓存
# 大小, 或者使用GNU sed。
sed -n 'G; s/\n/&&/; /^\([ -~]*\n\).*\n\1/d; s/\n//; h; P'

# 删除除重复行外的所有行（模拟“uniq -d”）
sed '$!N; s/^\(.*\)\n\1$/\1/; t; D'

# 删除文件中开头的10行
sed '1,10d'

# 删除文件中的最后一行
sed '$d'

# 删除文件中的最后两行
sed 'N;$!P;$!D;$d'

# 删除文件中的最后10行
sed -e :a -e '$d;N;2,10ba' -e 'P;D'   # 方法1
sed -n -e :a -e '1,10!{P;N;D;};N;ba'  # 方法2

# 删除8的倍数行
gsed '0~8d'                               # 只对GNU sed有效
sed 'n;n;n;n;n;n;n;d;'                  # 其他sed

# 删除匹配式样的行
sed '/pattern/d'                          # 删除含pattern的行。当然pattern
                                          # 可以换成任何有效的正则表达式

# 删除文件中的所有空行（与“grep '.' ”效果相同）
sed '/^$/d'                               # 方法1
sed '/./!d'                               # 方法2

# 只保留多个相邻空行的第一行。并且删除文件顶部和尾部的空行。
# （模拟“cat -s”）
sed '/./,/^$/!d'          #方法1, 删除文件顶部的空行, 允许尾部保留一空行
sed '/^$/N;/\n$/D'        #方法2, 允许顶部保留一空行, 尾部不留空行

# 只保留多个相邻空行的前两行。
sed '/^$/N;/\n$/N;//D'

# 删除文件顶部的所有空行
```

```
sed '/./,$!d'

# 删除文件尾部的所有空行
sed -e :a -e '/^\n*$/{$d;N;ba' -e '}'  # 对所有sed有效
sed -e :a -e '/^\n*$/N;/\n$/ba'          # 同上，但只对 gsed 3.02.*有效

# 删除每个段落的最后一行
sed -n '/^$/{p;h;};/./{x;/./p;}'
```

特殊应用：

```
# 移除手册页（man page）中的nroff标记。在Unix System V或bash shell下使
# 用'echo'命令时可能需要加上 -e 选项。
sed "s/.`echo \\\b`//g"    # 外层的双括号是必须的（Unix环境）
sed 's/.^H//g'             # 在bash或tcsh中，按 Ctrl-V 再按 Ctrl-H
sed 's/.\x08//g'           # sed 1.5，GNU sed，ssed所使用的十六进制的表示方法

# 提取新闻组或 e-mail 的邮件头
sed '/^$/q'                   # 删除第一行空行后的所有内容

# 提取新闻组或 e-mail 的正文部分
sed '1,/^$/d'                 # 删除第一行空行之前的所有内容

# 从邮件头提取“Subject”（标题栏字段），并移除开头的“Subject:”字样
sed '/^Subject: */!d; s///;q'

# 从邮件头获得回复地址
sed '/^Reply-To:/q; /^From:/h; /./d;g;q'

# 获取邮件地址。在上一个脚本所产生的那一行邮件头的基础上进一步的将非电邮
# 地址的部分剃除。（见上一脚本）
sed 's/ *(.*)//; s/>.*//; s/.*[:<] *//'

# 在每一行开头加上一个尖括号和空格（引用信息）
sed 's/^/> /'

# 将每一行开头处的尖括号和空格删除（解除引用）
sed 's/^> //'

# 移除大部分的HTML标签（包括跨行标签）
sed -e :a -e 's/<[^>]*>//g;/</N;//ba'

# 将分成多卷的uuencode文件解码。移除文件头信息，只保留uuencode编码部分。
# 文件必须以特定顺序传给sed。下面第一种版本的脚本可以直接在命令行下输入；
# 第二种版本则可以放入一个带执行权限的shell脚本中。（由Rahul Dhesi的一
# 个脚本修改而来。）
sed '/^end/,/^begin/d' file1 file2 ... fileX | uudecode   # vers. 1
sed '/^end/,/^begin/d' "$@" | uudecode                    # vers. 2

# 将文件中的段落以字母顺序排序。段落间以（一行或多行）空行分隔。GNU sed使用
# 字元“\v”来表示垂直制表符，这里用它来作为换行符的占位符——当然你也可以
# 用其他未在文件中使用的字符来代替它。
sed '/./{H;d;};x;s/\n/={NL}=/g' file | sort | sed '1s/={NL}=//;s/={NL}=/\n/g'
gsed '/./{H;d};x;y/\n/\v/' file | sort | sed '1s/\v//;y/\v/\n/'

# 分别压缩每个.TXT文件，压缩后删除原来的文件并将压缩后的.ZIP文件
# 命名为与原来相同的名字（只是扩展名不同）。（DOS环境：“dir /b”
# 显示不带路径的文件名）。
echo @echo off >zipup.bat
```

```
dir /b *.txt | sed "s/^\(.*\)\.TXT/pkzip -mo \1 \1.TXT/" >>zipup.bat
```

使用SED: Sed接受一个或多个编辑命令，并且每读入一行后就依次应用这些命令。当读入第一行输入后，sed对其应用所有的命令，然后将结果输出。接着再读入第二行输入，对其应用所有的命令……并重复这个过程。上一个例子中sed由标准输入设备（即命令解释器，通常是以管道输入的形式）获得输入。在命令行给出一个或多个文件名作为参数时，这些文件取代标准输入设备成为sed的输入。sed的输出将被送到标准输出（显示器）。因此:

```
cat filename | sed '10q'          # 使用管道输入
sed '10q' filename                 # 同样效果，但不使用管道输入
sed '10q' filename > newfile       # 将输出转移（重定向）到磁盘上
```

要了解sed命令的使用说明，包括如何通过脚本文件（而非从命令行）来使用这些命令，请参阅《sed & awk》第二版，作者Dale Dougherty和Arnold Robbins（O'Reilly，1997; http://www.ora.com），《UNIX Text Processing》，作者Dale Dougherty和Tim O'Reilly（Hayden Books，1987）或者是Mike Arst写的教程——压缩包的名称是“U-SEDIT2.ZIP”（在许多站点上都找得到）。要发掘sed的潜力，则必须对“正则表达式”有足够的理解。正则表达式的资料可以看《Mastering Regular Expressions》作者Jeffrey Friedl（O'reilly 1997）。Unix系统所提供的手册页（“man”）也会有所帮助（试一下这些命令“man sed”、“man regexp”，或者看“man ed”中关于正则表达式的部分），但手册提供的信息比较“抽象”——这也是它一直为人所诟病的。不过，它本来就不是用来教初学者如何使用sed或正则表达式的教材，而只是为那些熟悉这些工具的人提供的一些文本参考。

括号语法: 前面的例子对sed命令基本上都使用单引号（'...'）而非双引号（"..."）这是因为sed通常是在Unix平台上使用。单引号下，Unix的shell（命令解释器）不会对美元符（$）和后引号（`...`）进行解释和执行。而在双引号下美元符会被展开为变量或参数的值，后引号中的命令被执行并以输出的结果代替后引号中的内容。而在“csh”及其衍生的shell中使用感叹号（!）时需要在其前面加上转义用的反斜杠（就像这样: \!）以保证上面所使用的例子能正常运行（包括使用单引号的情况下）。DOS版本的Sed则一律使用双引号（"..."）而不是引号来圈起命令。

'\t'的用法: 为了使本文保持行文简洁，我们在脚本中使用'\t'来表示一个制表符。但是现在大部分版本的sed还不能识别'\t'的简写方式，因此当在命令行中为脚本输入制表符时，你应该直接按TAB键来输入制表符而不是输入'\t'。下列的工具软件都支持'\t'做为一个正则表达式的字元来表示制表符: awk、per1、HHsed、sedmod以及GNU sed v3.02.80。

不同版本的SED: 不同的版本间的sed会有些不同之处，可以想象它们之间在语法上会有差异。具体而言，它们中大部分不支持在编辑命令中间使用标签（:name）或分支命令（b,t），除非是放在那些的末尾。这篇文档中我们尽量选用了可移植性较高的语法，以使大多数版本的sed的用户都能使用这些脚本。不过GNU版本的sed允许使用更简洁的语法。想像一下当读者看到一个很长的命令时的心情:

```
sed -e '/AAA/b' -e '/BBB/b' -e '/CCC/b' -e d
```

好消息是GNU sed能让命令更紧凑:

```
sed '/AAA/b;/BBB/b;/CCC/b;d'        # 甚至可以写成
sed '/AAA\|BBB\|CCC/b;d'
```

此外，请注意虽然许多版本的sed接受象“/one/ s/RE1/RE2/”这种在's'前带有空格的命令，但这些版本中有些却不接受这样的命令:“/one/! s/RE1/RE2/”。这时只需要把中间的空格去掉就行了。

速度优化: 当由于某种原因（比如输入文件较大、处理器或硬盘较慢等）需要提高

命令执行速度时，可以考虑在替换命令（“s/…/…/”）前面加上地址表达式来提高速度。举例来说：

```
sed 's/foo/bar/g' filename          # 标准替换命令
sed '/foo/ s/foo/bar/g' filename    # 速度更快
sed '/foo/ s//bar/g' filename       # 简写形式
```

当只需要显示文件的前面的部分或需要删除后面的内容时，可以在脚本中使用“q”命令（退出命令）。在处理大的文件时，这会节省大量时间。因此：

```
sed -n '45,50p' filename            # 显示第45到50行
sed -n '51q;45,50p' filename        # 一样，但快得多
```

如果你有其他的单行脚本想与大家分享或者你发现了本文档中错误的地方，请发电子邮件给本文档的作者（Eric Pement）。邮件中请记得提供你所使用的sed版本、该sed所运行的操作系统及对问题的适当描述。本文所指的单行脚本指命令行的长度在65个字符或65个以下的sed脚本〔译注1〕。本文档的各种脚本是由以下所列作者所写或提供：

```
A1 Aab                              # 建立了“seders”邮件列表
Edgar Allen                         # 许多方面
Yiorgos Adamopoulos                 # 许多方面
Dale Dougherty                      # 《sed & awk》作者
Carlos Duarte                       # 《do it with sed》作者
Eric Pement                         # 本文档的作者
Ken Pizzini                         # GNU sed v3.02 的作者
S.G. Ravenhall                      # 去html标签脚本
Greg Ubben                          # 有诸多贡献并提供了许多帮助
```

译注1：大部分情况下，sed脚本无论多长都能写成单行的形式（通过`-e'选项和`;'号）——只要命令解释器支持，所以这里说的单行脚本除了能写成一行还对长度有所限制。因为这些单行脚本的意义不在于它们是以单行的形式出现。而是让用户能方便地在命令行中使用这些紧凑的脚本才是其意义所在。