1.sed -n '2'p filename   
打印文件的第二行。   
2.sed -n '1,3'p filename   
打印文件的1到3行   
3. sed -n '/Neave/'p filename   
打印匹配Neave的行(模糊匹配)   
4.

5. sed -n '1,$'p filename   
打印整个文件，$表示最后一行。   
6. sed -n '/.\*ing/'p filename   
匹配任意字母，并以ing结尾的单词(点号不能少)，打印该行  
7 .sed -n / -e '/music/'= filename   
打印匹配行的行号，**-e** 会打印两次相同的行，-n只打印一次。   
8.

9.sed '/company/'a‘Then suddenly it happened’ filename （a前后都无空格，a表示append,在行后）  
选择含有company的行，将后面的内容"Then suddenly it happend"加入下一行。注意：它并不改变文件，所有操作在缓冲区，如果要保存输出，重定向到一个文件。

10. sed '/company/'i‘Then suddenly it happened’ filename (i前后都无空格，i表示insert，在行前)   
选择含有company的行，将"Then suddenly it happend"插入前一行。

11.sed '/company/'c"Then suddenly it happend" filename（c前后都无空格，c表示copy）   
用"Then suddenly it happend"替换匹配company的整行的内容。

12.

Sed ‘1’d filename删除第一行

Sed ‘1,3’d filename删除一到三行

Sed ‘$’d filename 删除最后一行

Sed ‘/helixiao’d filename删除匹配helixiao的所有行

13.在3~5行中，在第一次出现helixiao的前面加’hello ’

Sed ‘3,5’s’/helixiao/hello &/’ filename

注意：s前后没有空格，s表示替换命令

如果要删除某行中所有的helixiao可以这样

Sed ‘3,5’s’helixiao//g’ 注意，在replace-pattern域内为空！替换为空也就等价于删除了嘛！

如果是所有，需要加/g，否则，默认只会替换掉匹配行中的第一次匹配

**总的结构如下：**

**[address]s’/ pattern-to-find /replacement-pattern/[g p w n]’   
s选项通知s e d这是一个替换操作，并查询pattern-to-find，成功后用replacement-pattern替换它。   
替换选项如下：   
g 缺省情况下只替换第一次出现模式，使用g选项替换全局所有出现模式。   
p 缺省s e d将所有被替换行写入标准输出，加p选项将使- n选项无效。- n选项不打印输出结果。   
w 文件名使用此选项将输出定向到一个文件。(注意只将匹配替换的行写入文件，而不是整个内容)**

14.sed s'/nurse/hello &/' filename   
将'hello '增加到'nurse' 的前面。   
15. sed '/company/r append.txt' filename   
在匹配company的行的下一行开始加入文件append.txt的内容。   
16. sed '/company/'q filename   
首次匹配company后就退出sed程序

只所以看sed命令，是因为我遇到了这个一个问题。   
网上有很多教程，他们发表了很多程序代码，但是作者为了解释方便，都对程序作了行号编码，就像下面这样：   
**代码::**  
1:#!/bin/bash   
2:#rename file extesions   
3:#   
4:#     rfe old\_extensions new\_extension  
假设这个文件名是tmp，那么我们可以使用下面的命令来去掉这个行号和冒号(：)   
**代码::**  
sed -e  s'/^[0-9]\{1,\}://g' tmp  
不过上面的命令的命令有一个缺点，那就是如果这个行号不是数字开头，而是有空格的话，那就需要修改匹配规则，规则应该修改为匹配第一个非空白字符是数字开始，后面接一个冒号的配对。命令如下：   
**代码::**  
sed -e  s'/^[^0-9a-zA-Z]\*[0-9]\{1,\}://g' tmp  
这令我很兴奋，于是想看看sed到底有多厉害，看了以后，明白的是不是sed有多厉害，就像awk一样，他们只是把正规表达式用到了极致。  
以 Redhat6.0 为测试环境   
事实上在solaris下的sed命令要比linux强，但因为没有测试环境，我这里只给在linux下经过测试的用法。   
★ 命令行参数简介   
★ 首先假设我们有这样一个文本文件 sedtest.txt   
★ 输出指定范围的行 p   
★ 在每一行前面增加一个制表符(^I)   
★ 在每一行后面增加--end   
★ 显示指定模式匹配行的行号 [/pattern/]=   
★ 在匹配行后面增加文本 [/pattern/]a\ 或者 [address]a\   
★ 删除匹配行 [/pattern/]d 或者 [address1][,address2]d   
★ 替换匹配行 [/pattern/]c\ 或者 [address1][,address2]c\   
★ 在匹配行前面插入文本 [/pattern/]i\ 或者 [address]i\   
★ 替换匹配串(注意不再是匹配行) [addr1][,addr2]s/old/new/g   
★ 限定范围后的模式匹配   
★ 指定替换每一行中匹配的第几次出现   
★ &代表最后匹配   
★ 利用sed修改PATH环境变量   
★ 测试并提高sed命令运行效率   
★ 指定输出文件 [address1][,address2]w outputfile   
★ 指定输入文件 [address]r inputfile   
★ 替换相应字符 [address1][,address2]y/old/new/   
★ !号的使用   
★ \c正则表达式c 的使用   
★ sed命令中正则表达式的复杂性   
★ 转换man手册成普通文本格式(新)   
★ sed的man手册(用的就是上面的方法)   
★ 命令行参数简介   
sed   
-e script 指定sed编辑命令   
-f scriptfile 指定的文件中是sed编辑命令   
-n 寂静模式，抑制来自sed命令执行过程中的冗余输出信息，比如只   
显示那些被改变的行。   
不明白？不要紧，把这些肮脏丢到一边，跟我往下走，不过下面的介绍里   
不包括正则表达式的解释，如果你不明白，可能有点麻烦。   
首先假设我们有这样一个文本文件 sedtest.txt   
  
★ 输出指定范围的行 p other types of editors.   
sed –n ‘1,4’p sedtest.txt   
sed -n ‘/from/’p sedtest.txt   
sed -n ‘1,/from/’p sedtest.txt

★ 在每一行前面增加一个制表符(^I)   
sed s’/^/^I/g’ sedtest.txt   
注意^I的输入方法是ctrl-v ctrl-i   
单个^表示行首

★ 在每一行后面增加--end   
sed s’/$/--end/’ sedtest.txt   
单个$表示行尾

★ 显示指定模式匹配行的行号 [/pattern/]=   
sed -e '/is/=' sedtest.txt   
  
★ 在匹配’helixiao’模式行后面增加文本‘+++++++++’   
sed ‘/helixiao/’a‘+++++++++++++++++’

在文件最后增加一些内容（可能有好几行）

Sed ‘$’a‘\

>What is your name?\

>My name is helixiao’ aaa

在文件最后增加 ’sedd’ = false 这条字段

Sed ‘$’a“\

>‘sedd’ = false” aaa

注意，此处的单引号不用转义，还有对于a命令的文本域，不支持单引号，只能用双引号。

★ 删除匹配行

删除6-10行的内容，包括6和10   
sed '6,10'd sedtest.txt

删除从第一行到第一个包含from字符串的行之间的所有内容，包括第一个包含from字符串的行。

sed ‘1,/from/’d sedtest.txt

给所有is字符串外增加()

sed s‘/is/(&)/g’ sedtest.txt   
  
★ 利用sed修改PATH环境变量   
先查看PATH环境变量   
> echo $PATH   
/usr/bin:/usr/bin:/bin:/usr/local/bin:/sbin:/usr/sbin:/usr/X11R6/bin:.

去掉尾部的{ :/usr/X11R6/bin:. }   
> echo $PATH | sed "s/^\(.\*\):\/usr[/]X11R6\/bin:[.]$/\1/"

去掉中间的{ :/bin: }   
> echo $PATH | sed "s/^\(.\*\):\/bin:\(.\*\)$/\1\2/"

去掉尾部的:，然后增加新的路径   
PATH=`echo $PATH | sed 's/\(.\*\):$/\1/'`:$HOME/src   
注意反引号`和单引号'的区别。

[/]表示/失去特殊意义   
\/同样表示/失去意义   
\1表示子匹配的第一次出现   
\2表示子匹配的第二次出现   
\(.\*\)表示子匹配：在正规表达式中使用 **\(** 和 **\)** 符号括起正规表达式，即可在后面使用**\1**、**\2** 等变量来访问 **\(** 和 **\)** 中的内容。

==============================

路径"../syslog/tmp/"替换为"/data/syslog/tmp”  
  
首先要搞清楚，[转义](http://zhidao.baidu.com/search?word=%E8%BD%AC%E4%B9%89&fr=qb_search_exp&ie=utf8)符\的作用是消除有特殊含义字符的特殊意义，使其还原为普通字符。  
sed -i 's/\.\.\/syslog\/tmp\//\/data\/syslog\/tmp/' file.ini  
[点号](http://zhidao.baidu.com/search?word=%E7%82%B9%E5%8F%B7&fr=qb_search_exp&ie=utf8).在[正则表达式](http://zhidao.baidu.com/search?word=%E6%AD%A3%E5%88%99%E8%A1%A8%E8%BE%BE%E5%BC%8F&fr=qb_search_exp&ie=utf8)中有其特殊含义（表示任意一个字符），所以要表示[点号](http://zhidao.baidu.com/search?word=%E7%82%B9%E5%8F%B7&fr=qb_search_exp&ie=utf8)本身，需要使用[转义](http://zhidao.baidu.com/search?word=%E8%BD%AC%E4%B9%89&fr=qb_search_exp&ie=utf8)符。  
sed替换命令的格式一般为 s/.../.../ ，所以如果里面的内容也包含 / ，比如你这里的路径，为了区分，也需要[转义](http://zhidao.baidu.com/search?word=%E8%BD%AC%E4%B9%89&fr=qb_search_exp&ie=utf8)。  
  
最后说明一下，这样看起来很混乱，比较容易搞错。  
其实sed的替换命令格式不一定要是 s/.../.../，下面这样也都可以：  
s#...#...#  
s\_...\_...\_   
即命令s后可以跟任意字符，只要跟搜索替换内容不重复即可。这样，路径里的/就不再需要转义了。

=================================

★ 指定输出文件 [address1][,address2]w outputfile   
sed "1,10w sed.out" sedtest.txt -n   
将sedtest.txt中1-10行的内容写到sed.out文件中。

★ 指定输入文件 [address]r inputfile   
sed "1r sedappend.txt" sedtest.txt   
将sedappend.txt中的内容附加到sedtest.txt文件的第一行之后   
★ 替换相应字符 [address1][,address2]y/old/new/   
sed "y/abcdef/ABCDEF/" sedtest.txt   
将sedtest.txt中所有的abcdef小写字母替换成ABCDEF大写字母。   
★ !号的使用   
sed -e '3,7!d' sedtest.txt   
删除3-7行之外的所有行   
sed -e '1,/from/!d' sedtest.txt   
找到包含from字符串的行，删除其后的所有行   
★ \c正则表达式c 的使用   
sed -e "\:from:d" sedtest.txt   
等价于 sed -e "/from/d" sedtest.txt   
★ sed命令中正则表达式的复杂性   
cat > sedtest.txt   
^\/[}]{.\*}[\(]$\)   
^D   
如何才能把该行替换成   
\(]$\)\/[}]{.\*}^[   
★ 转换man手册成普通文本格式(新)   
man sed | col -b > sed.txt   
sed -e "s/^H//g" -e "/^$/d" -e "s/^^I/ /g" -e "s/^I/ /g" sed.txt > sedman   
txt   
删除所有退格键、空行，把行首的制表符替换成8个空格，其余制表符替换成一个空格。   
★ sed的man手册(用的就是上面的方法)   
NAME   
sed - a Stream EDitor   
SYNOPSIS   
sed [-n] [-V] [--quiet] [--silent] [--version] [--help]   
[-e script] [--expression=script]   
[-f script-file] [--file=script-file]   
[script-if-no-other-script]   
[file...]   
DESCRIPTION   
Sed is a stream editor. A stream editor is used to per-   
form basic text transformations on an input stream (a file   
or input from a pipeline). While in some ways similar to   
an editor which permits scripted edits (such as ed), sed   
works by making only one pass over the input(s), and is   
consequently more efficient. But it is sed's ability to   
filter text in a pipeline which particularly distinguishes   
it from other types of editors.   
OPTIONS   
Sed may be invoked with the following command-line   
options:   
-V   
--version   
Print out the version of sed that is being run and   
a copyright notice, then exit.   
-h   
--help Print a usage message briefly summarizing these   
command-line options and the bug-reporting address,   
then exit.   
-n   
--quiet   
--silent   
By default, sed will print out the pattern space at   
the end of each cycle through the script. These   
options disable this automatic printing, and sed   
will only produce output when explicitly told to   
via the p command.   
-e script   
--expression=script   
Add the commands in script to the set of commands   
to be run while processing the input.   
-f script-file   
--file=script-file   
Add the commands contained in the file script-file   
to the set of commands to be run while processing   
the input.   
If no -e,-f,--expression, or --file options are given on   
the command-line, then the first non-option argument on   
the command line is taken to be the script to be executed.   
If any command-line parameters remain after processing the   
above, these parameters are interpreted as the names of   
input files to be processed. A file name of - refers to   
the standard input stream. The standard input will pro-   
cessed if no file names are specified.   
Command Synopsis   
This is just a brief synopsis of sed commands to serve as   
a reminder to those who already know sed; other documenta-   
tion (such as the texinfo document) must be consulted for   
fuller descriptions.   
Zero-address ``commands''   
: label   
Label for b and t commands.   
#comment   
The comment extends until the next newline (or the   
end of a -e script fragment).   
} The closing bracket of a { } block.   
Zero- or One- address commands   
= Print the current line number.   
a \   
text Append text, which has each embedded newline pre-   
ceeded by a backslash.   
i \   
text Insert text, which has each embedded newline pre-   
ceeded by a backslash.   
q Immediately quit the sed script without processing   
any more input, except that if auto-print is not   
diabled the current pattern space will be printed.   
r filename   
Append text read from filename.   
Commands which accept address ranges   
{ Begin a block of commands (end with a }).   
b label   
Branch to label; if label is omitted, branch to end   
of script.   
t label   
If a s/// has done a successful substitution since   
the last input line was read and since the last t   
command, then branch to label; if label is omitted,   
branch to end of script.   
c \   
text Replace the selected lines with text, which has   
each embedded newline preceeded by a backslash.   
d Delete pattern space. Start next cycle.   
D Delete up to the first embedded newline in the pat-   
tern space. Start next cycle, but skip reading   
from the input if there is still data in the pat-   
tern space.   
h H Copy/append pattern space to hold space.   
g G Copy/append hold space to pattern space.   
x Exchange the contents of the hold and pattern   
spaces.   
l List out the current line in a ``visually unambigu-   
ous'' form.   
n N Read/append the next line of input into the pattern   
space.   
p Print the current pattern space.   
P Print up to the first embedded newline of the cur-   
rent pattern space.   
s/regexp/replacement/   
Attempt to match regexp against the pattern space.   
If successful, replace that portion matched with   
replacement. The replacement may contain the spe-   
cial character & to refer to that portion of the   
pattern space which matched, and the special   
escapes \1 through \9 to refer to the corresponding   
matching sub-expressions in the regexp.   
w filename Write the current pattern space to file-   
name.   
y/source/dest/   
Transliterate the characters in the pattern space   
which appear in source to the corresponding charac-   
ter in dest.   
Addresses   
Sed commands can be given with no addresses, in which case   
the command will be executed for all input lines; with one   
address, in which case the command will only be executed   
for input lines which match that address; or with two   
addresses, in which case the command will be executed for   
all input lines which match the inclusive range of lines   
starting from the first address and continuing to the sec-   
ond address. Three things to note about address ranges:   
the syntax is addr1,addr2 (i.e., the addresses are sepa-   
rated by a comma); the line which addr1 matched will   
always be accepted, even if addr2 selects an earlier line;   
and if addr2 is a regexp, it will not be tested against   
the line that addr1 matched.   
After the address (or address-range), and before the com-   
mand, a ! may be inserted, which specifies that the com-   
mand shall only be executed if the address (or address-   
range) does not match.   
The following address types are supported:   
number Match only the specified line number.   
first~step   
Match every step'th line starting with line first.   
For example, ``sed -n 1~2p'' will print all the   
odd-numbered lines in the input stream, and the   
address 2~5 will match every fifth line, starting   
with the second. (This is a GNU extension.)   
$ Match the last line.   
/regexp/   
Match lines matching the regular expression regexp.   
\cregexpc   
Match lines matching the regular expression regexp.   
The c may be any character.   
Regular expressions   
POSIX.2 BREs should be supported, but they aren't com-   
pletely yet. The \n sequence in a regular expression   
matches the newline character. There are also some GNU   
extensions. [XXX FIXME: more needs to be said. At the   
very least, a reference to another document which   
describes what is supported should be given.]   
Miscellaneous notes   
This version of sed supports a \ sequence in all   
regular expressions, the replacement part of a substitute   
(s) command, and in the source and dest parts of a   
transliterate (y) command. The \ is stripped, and the   
newline is kept.   
SEE ALSO   
awk(1), ed(1), expr(1), emacs(1), perl(1), tr(1), vi(1),   
regex(5) [well, one ought to be written... XXX], sed.info,   
any of various books on sed, the sed FAQ   
(http://www.wollery.demon.co.uk/sedtut10.txt,   
http://www.ptug.org/sed/sedfaq.htm).   
BUGS   
E-mail bug reports to bug-gnu-utils@gnu.org. Be sure to   
include the word ``sed'' somewhere in the ``Subject:''   
field.  
Sed学习笔记  
作者：Jims of   
[肥肥世家](http://www.ringkee.com/)  
  
**Table of Contents**  
[1. Sed简介](http://www.ringkee.com/jims/technic_folder/linux/#id2875136)  
  
[2. 定址](http://www.ringkee.com/jims/technic_folder/linux/#id2875115)  
  
[3. Sed命令](http://www.ringkee.com/jims/technic_folder/linux/#id2874783)  
  
[4. 选项](http://www.ringkee.com/jims/technic_folder/linux/#id2810312)  
  
[5. 元字符集](http://www.ringkee.com/jims/technic_folder/linux/#id2810415)  
  
[6. 实例](http://www.ringkee.com/jims/technic_folder/linux/#id2861474)  
  
[7. 脚本](http://www.ringkee.com/jims/technic_folder/linux/#id2808386)  
1. Sed简介  
sed是一种在线编辑器，它一次处理一行内容。处理时，把当前处理的行存储在临时缓冲区中，称为“模式空间”（pattern space），接着用sed命令处理缓冲区中的内容，处理完成后，把缓冲区的内容送往屏幕。接着处理下一行，这样不断重复，直到文件末尾。文件内容并没有改变，除非你使用重定向存储输出。Sed主要用来自动编辑一个或多个文件；简化对文件的反复操作；编写转换程序等。以下介绍的是Gnu版本的Sed 3.02。  
2. 定址  
可以通过定址来定位你所希望编辑的行，该地址用数字构成，用逗号分隔的两个行数表示以这两行为起止的行的范围（包括行数表示的那两行）。如1，3表示1，2，3行，美元符号($)表示最后一行。范围可以通过数据，正则表达式或者二者结合的方式确定 。  
3. Sed命令  
调用sed命令有两种形式：   
sed [options] 'command' file(s)  
sed [options] -f scriptfile file(s)  
a\   
在当前行后面加入一行文本。  
b lable   
分支到脚本中带有标记的地方，如果分支不存在则分支到脚本的末尾。  
c\   
用新的文本改变本行的文本。  
d   
从模板块（Pattern space）位置删除行。  
D   
删除模板块的第一行。  
i\   
在当前行上面插入文本。  
h   
拷贝模板块的内容到内存中的缓冲区。  
H   
追加模板块的内容到内存中的缓冲区  
g   
获得内存缓冲区的内容，并替代当前模板块中的文本。  
G   
获得内存缓冲区的内容，并追加到当前模板块文本的后面。  
l   
列表不能打印字符的清单。  
n   
读取下一个输入行，用下一个命令处理新的行而不是用第一个命令。  
N   
追加下一个输入行到模板块后面并在二者间嵌入一个新行，改变当前行号码。  
p   
打印模板块的行。  
P（大写）   
打印模板块的第一行。  
q   
退出Sed。  
r file   
从file中读行。  
t label   
if分支，从最后一行开始，条件一旦满足或者T，t命令，将导致分支到带有标号的命令处，或者到脚本的末尾。  
T label   
错误分支，从最后一行开始，一旦发生错误或者T，t命令，将导致分支到带有标号的命令处，或者到脚本的末尾。  
w file   
写并追加模板块到file末尾。  
W file   
写并追加模板块的第一行到file末尾。  
!   
表示后面的命令对所有没有被选定的行发生作用。  
s/re/string   
用string替换正则表达式re。  
=   
打印当前行号码。  
#   
把注释扩展到下一个换行符以前。  
以下的是替换标记   
g表示行内全面替换。  
p表示打印行。  
w表示把行写入一个文件。  
x表示互换模板块中的文本和缓冲区中的文本。  
y表示把一个字符翻译为另外的字符（但是不用于正则表达式）  
4. 选项  
-e command, --expression=command   
允许多台编辑。  
-h, --help   
打印帮助，并显示bug列表的地址。  
-n, --quiet, --silent   
取消默认输出。  
-f, --filer=script-file   
引导sed脚本文件名。  
-V, --version   
打印版本和版权信息。  
5. 元字符集  
^ 锚定行的开始 如：/^sed/匹配所有以sed开头的行。   
$ 锚定行的结束 如：/sed$/匹配所有以sed结尾的行。   
. 匹配一个非换行符的字符 如：/s.d/匹配s后接一个任意字符，然后是d。   
\* 匹配零或多个字符 如：/\*sed/匹配所有模板是一个或多个空格后紧跟sed的行。   
[] 匹配一个指定范围内的字符，如/[Ss]ed/匹配sed和Sed。   
[^] 匹配一个不在指定范围内的字符，如：/[^A-RT-Z]ed/匹配不包含A-R和T-Z的一个字母开头，紧跟ed的行。   
\(..\) 保存匹配的字符，如s/\(love\)able/\1rs，loveable被替换成lovers。   
& 保存搜索字符用来替换其他字符，如s/love/\*\*&\*\*/，love这成\*\*love\*\*。   
\ 锚定单词的开始，如:/\匹配包含以love开头的单词的行。   
\> 锚定单词的结束，如/love\>/匹配包含以love结尾的单词的行。   
x\{m\}重复字符x，m次，如：/0\{5\}/匹配包含5个o的行。   
x\{m,\} 重复字符x,至少m次，如：/o\{5,\}/匹配至少有5个o的行。   
x\{m,n\}重复字符x，至少m次，不多于n次，如：/o\{5,10\}/匹配5--10个o的行。  
6. 实例  
删除：d命令   
$ sed '2d' example-----删除example文件的第二行。  
$ sed '2,$d' example-----删除example文件的第二行到末尾所有行。  
$ sed '$d' example-----删除example文件的最后一行。  
$ sed '/test/'d example-----删除example文件所有包含test的行。  
替换：s命令   
$ sed 's/test/mytest/g' example-----在整行范围内把test替换为mytest。如果没有g标记，则只有每行第一个匹配的test被替换成mytest。  
$ sed -n 's/^test/mytest/p' example-----(-n)选项和p标志一起使用表示只打印那些发生替换的行。也就是说，如果某一行开头的test被替换成mytest，就打印它。  
$ sed 's/^192.168.0.1/&localhost/' example-----&符号表示替换换字符串中被找到的部份。所有以192.168.0.1开头的行都会被替换成它自已加localhost，变成192.168.0.1localhost。  
$ sed -n 's/\(love\)able/\1rs/p' example-----love被标记为1，所有loveable会被替换成lovers，而且替换的行会被打印出来。  
$ sed 's#10#100#g' example-----不论什么字符，紧跟着s命令的都被认为是新的分隔符，所以，“#”在这里是分隔符，代替了默认的“/”分隔符。表示把所有10替换成100。  
选定行的范围：逗号   
$ sed -n '/test/,/check/p' example-----所有在模板test和check所确定的范围内的行都被打印。  
$ sed -n '5,/^test/p' example-----打印从第五行开始到第一个包含以test开始的行之间的所有行。  
$ sed '/test/,/check/s/$/sed test/' example-----对于模板test和west之间的行，每行的末尾用字符串sed test替换。  
多点编辑：e命令   
$ sed -e '1,5d' -e 's/test/check/' example-----(-e)选项允许在同一行里执行多条命令。如例子所示，第一条命令删除1至5行，第二条命令用check替换test。命令的执行顺序对结果有影响。如果两个命令都是替换命令，那么第一个替换命令将影响第二个替换命令的结果。  
$ sed --expression='s/test/check/' --expression='/love/d' example-----一个比-e更好的命令是--expression。它能给sed表达式赋值。  
从文件读入：r命令   
$ sed '/test/r file' example-----file里的内容被读进来，显示在与test匹配的行后面，如果匹配多行，则file的内容将显示在所有匹配行的下面。  
写入文件：w命令   
$ sed -n '/test/w file' example-----在example中所有包含test的行都被写入file里。  
追加命令：a命令   
$ sed '/^test/a\\--->this is a example' example被追加到以test开头的行后面，sed要求命令a后面有一个反斜杠。  
插入：i命令   
$ sed '/test/i\\  
new line  
-------------------------' example  
如果test被匹配，则把反斜杠后面的文本插入到匹配行的前面。  
下一个：n命令   
$ sed '/test/{ n; s/aa/bb/; }' example-----如果test被匹配，则移动到匹配行的下一行，替换这一行的aa，变为bb，并打印该行，然后继续。  
变形：y命令   
$ sed '1,10y/abcde/ABCDE/' example-----把1--10行内所有abcde转变为大写，注意，正则表达式元字符不能使用这个命令。  
退出：q命令   
$ sed '10q' example-----打印完第10行后，退出sed。  
保持和获取：h命令和G命令   
$ sed -e '/test/h' -e '$G example-----在sed处理文件的时候，每一行都被保存在一个叫模式空间的临时缓冲区中，除非行被删除或者输出被取消，否则所有被处理的行都将打印在屏幕上。接着模式空间被清空，并存入新的一行等待处理。在这个例子里，匹配test的行被找到后，将存入模式空间，h命令将其复制并存入一个称为保持缓存区的特殊缓冲区内。第二条语句的意思是，当到达最后一行后，G命令取出保持缓冲区的行，然后把它放回模式空间中，且追加到现在已经存在于模式空间中的行的末尾。在这个例子中就是追加到最后一行。简单来说，任何包含test的行都被复制并追加到该文件的末尾。  
保持和互换：h命令和x命令   
$ sed -e '/test/h' -e '/check/x' example -----互换模式空间和保持缓冲区的内容。也就是把包含test与check的行互换。  
7. 脚本  
Sed脚本是一个sed的命令清单，启动Sed时以-f选项引导脚本文件名。Sed对于脚本中输入的命令非常挑剔，在命令的末尾不能有任何空白或文本，如果在一行中有多个命令，要用分号分隔。以#开头的行为注释行，且不能跨行。