# 最佳实践：

使用sed时使用-r, 启用扩展的正则表达式，修改文件是同时使用-i --follow-symlinks两个选项

其他一些可能会用到的选项：

-n, --quiet, --silent

-e script, --expression=script

-z, --null-data

-i[SUFFIX], --in-place[=SUFFIX] #如果带有suffix,则在修改前会备份，如

[root@bogon dir]# sed -r -i.bak --follow-symlinks 's/(.\*)/\L\1/' a.txt

<http://www.grymoire.com/Unix/Sed.html>

# Sed - An Introduction and Tutorial by Bruce Barnett

man sed

sed --help

info sed #包含example和其他额外更详细的信息

-r, --regexp-extended

use extended regular expressions in the script.

2) 关闭SELINUX

# setenforce 0

# sed -i --follow-symlinks '7s/enforcing/disabled/' /etc/sysconfig/selinux

[root@bogon dir]# cat a.txt

one

two

three

four

five

six

seven

eight

nine

ten

[root@bogon dir]# cat b.txt

111111111

222222222

333333333

444444444

555555555

666666666

777777777

888888888

999999999

000000000

注：sed的一些语法风格与vim中类似的功能语法上是相似的(可能都因为模仿了ed等行编辑器的风格)，所以可以通过联系vim来助忆。

# address！command 地址类型： 在地址后，命令之前插入!，表示对地址取反，类似于grep –v操作

[root@bogon dir]# sed -r -n '1,/four/! p' a.txt

five

six

seven

eight

nine

ten

#未指定address时，默认为整个文档

sed -n 'p' a.txt

# number 地址类型， 即第number行

[root@bogon dir]# sed -n '3p' a.txt

three

# addr1,addr2 地址类型，即区间[addr1,addr2]范围的行（注：addr1总是可被接受的，即使addr2小于addr1）

# addr1和addr2都可以是regexp,当addr2为regexp,则addr2不会(重复)再对addr1所在行进行匹配操作

[root@bogon dir]# sed -n '3,6p' a.txt

three

four

five

six

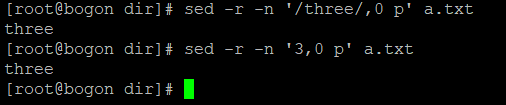
[root@bogon dir]# sed -r -n '/three/,/six/ p' a.txt

three

four

five

six



# 注：当addr2为匹配到时，等价于addr1,$ (注：当addr1,addr2中addr2小于addr1时除外)，

# 如下两条命令返回相同的结果

[root@bogon dir]# sed -r -n '/three/,10000 p' a.txt

[root@bogon dir]# sed -r -n '/three/,/AAAAAAAAAAAA/ p' a.txt

three

four

five

six

seven

eight

nine

ten

# first~step 地址类型，(注:当first为0时，认为是first=step)

[root@bogon dir]# sed -n '1~2p' a.txt

one

three

five

seven

nine

[root@bogon dir]# sed -n '0~2p' a.txt

two

four

six

eight

ten

# $ 地址类型(注：美元符$用于匹配最后一行)

[root@bogon dir]# sed -n '$p' a.txt

ten

[root@bogon dir]# sed -n '8,$p' a.txt

eight

nine

ten

# /regexp/ 地址类型：匹配正则表达式

[root@bogon dir]# sed -r -n '/[sn]/p' a.txt

one

six

seven

nine

ten

# \cregexpc 地址类型：作用和/regexp/完全一样，只是c可以是任意character作为正则表达式的边界

[root@bogon dir]# sed -r -n '\c[sn]c p' a.txt

one

six

seven

nine

ten

#addr1,+N 地址类型： 针对对addr1和后续的N行（注：如果addr1没有匹配到，则没有任何效果）

[root@bogon dir]# sed -r -n '3,+2 p' a.txt

three

four

five

[root@bogon dir]# sed -r -n '/three/,+2 p' a.txt

three

four

five

# addr1,~N 地址类型Will match addr1 and the lines following addr1 until the next line whose input # line number is a multiple of N.

[root@bogon dir]# sed -r -n '1,~4 p' a.txt

one

two

three

four

[root@bogon dir]# sed -r -n '5,~4 p' a.txt

five

six

seven

eight

# 注：如下'9,~4 p'相当于'9,12 p',但是a.txt总共只有10行,所以效果同'9,$ p'

[root@bogon dir]# sed -r -n '9,~4 p' a.txt

nine

ten

[root@bogon dir]# sed -r -n '9,~11 p' a.txt

nine

ten

# 注：'9,~9 p'的效果类似于'9,18 p'

[root@bogon dir]# sed -r -n '9,~9 p' a.txt

nine

ten

# 0,addr2 地址类型（注：这种地址只有在addr2为regexp时才有效，否则会报语法错误）

# 0,addr2 和 1,addr2的区别

# 0,addr2中addr2会从第一行开始匹配

# 1,addr2中addr2会从第二行开始匹配;n,,addr2会从n+1行开始匹配

[root@bogon dir]# sed -r -n '0,/one/ p' a.txt

one

[root@bogon dir]# sed -r -n '1,/one/ p' a.txt

one

two

three

four

five

six

seven

eight

nine

ten

[root@bogon dir]# sed -r -n '0,/four/ p' a.txt

one

two

three

four

[root@bogon dir]# sed -r -n '1,/four/ p' a.txt

one

two

three

four

sed命令：

#【Zero- or One- address commands】

# = 命令： 打印行号

[root@bogon dir]# sed -n '$=' a.txt

10

#【Zero- or One- address commands】

# a \

# text Append text, which has each embedded newline preceded by a backslash.

[root@bogon dir]# sed '2a \AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA' a.txt

one

two

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

three

four

five

six

seven

eight

nine

ten

[root@bogon dir]# sed '2a \AAAAAAAAAA\nBBBBBBBBBB' a.txt

one

two

AAAAAAAAAA

BBBBBBBBBB

three

four

five

six

seven

eight

nine

ten

#【Zero- or One- address commands】

# i \

# text Insert text, which has each embedded newline preceded by a backslash.

[root@bogon dir]# sed '2i \AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA' a.txt

one

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

two

three

four

five

six

seven

eight

nine

ten

[root@bogon dir]# sed '2i \AAAAAAAAAA\nBBBBBBBBBBBBBBBB' a.txt

one

AAAAAAAAAA

BBBBBBBBBBBBBBBB

two

three

four

five

six

seven

eight

nine

ten

#【Zero- or One- address commands】

# 注: r filename和R filename的区别是每次调用r都会读取filename的整个文件内容，而R filename只读取一

行(即还未读取的下一行，如果已经读到文件末尾eof,就不会再产生新的效果了)

# r filename

# Append text read from filename.

# R filename

# Append a line read from filename. Each invocation of the command reads a line from

# the file. This is a GNU extension.

[root@bogon dir]# sed 'r b.txt' a.txt

one

111111111

222222222

333333333

444444444

555555555

666666666

777777777

888888888

999999999

000000000

two

111111111

222222222

333333333

444444444

555555555

666666666

777777777

888888888

999999999

000000000

three

…… 省略

[root@bogon dir]# sed 'R b.txt' a.txt

one

111111111

two

222222222

three

333333333

four

444444444

five

555555555

six

666666666

seven

777777777

eight

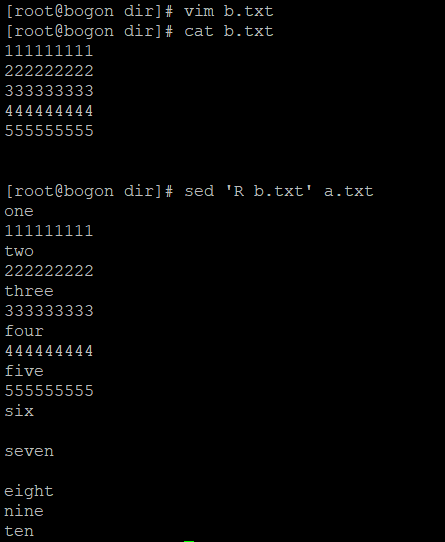
888888888

nine

999999999

ten

000000000



#【Zero- or One- address commands】

# q [exit-code]和Q [exit-code]用于做退出quit操作(即终止sed进程),区别是Q直接退出时不会再做任何的输入处理。而q与之类似，除了当auto-print功能为disabled时，还是仍然会将当前模式空间print出来。

# 另外，exit-code是可选的，在sed命令退出后可以用“$?”变现检测

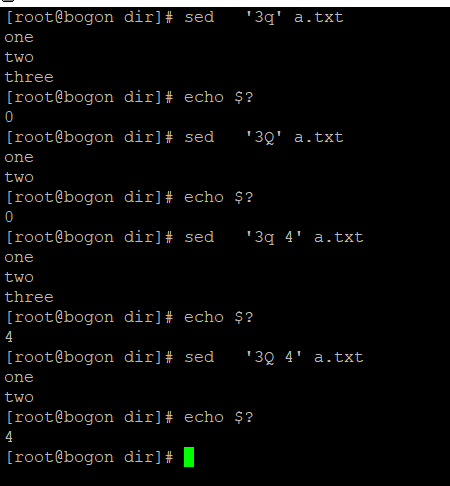
# q [exit-code]

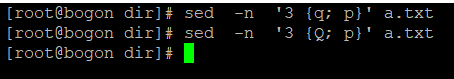
Immediately quit the sed script without processing any more input, except that if auto-print is not disabled the current pattern space will be printed. The exit code

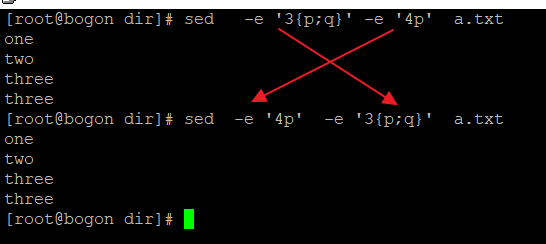
argument is a GNU extension.

# Q [exit-code]

Immediately quit the sed script without processing any more input. This is a GNU extension.







#【Commands which accept address ranges】

# c \

text Replace the selected lines with text, which has each embedded newline preceded by # a backslash.

[root@bogon dir]# sed -e '3c \3333333333' a.txt

one

two

3333333333

four

five

six

seven

eight

nine

ten

[root@bogon dir]# sed -e '3c \3333333333\n444444444444' a.txt

one

two

3333333333

444444444444

four

five

six

seven

eight

nine

ten

#【Commands which accept address ranges】

# n和N的区别

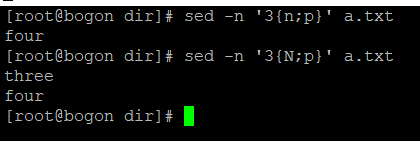
# n是清空模式空间中的内容，然后将下一行读到模式空间中

#（类似：pattern\_space=’one\n’, 执行n后，则pattern\_space=’two\n’）

# N是不清空模式空间中的内容，然后读取下一行，并将该读取的行追加到模式空间内容的最后面。

#（类似：pattern\_space=’one\n’, 执行N后，则pattern\_space=’one\ntwo\n’）

# n N Read/append the next line of input into the pattern space.



#【Commands which accept address ranges】

# p 和 P的区别

# p会print整个模式空间的内容

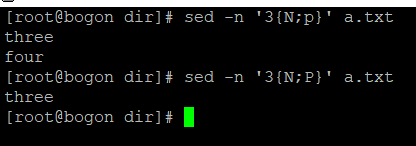
#（类似：pattern\_space=’one\ntwo\n’, 执行p后，则屏幕会输出整个pattern\_space的内容：’one\ntwo\n’）

# P只会print从模式空间开始位置到第一个换行符'\n'之间的内容

#（类似：pattern\_space=’one\ntwo\n’, 执行P后，则屏幕只输出开始到第一个\n之间的内容：’one\n’）

# p Print the current pattern space.

# P Print up to the first embedded newline of the current pattern space.



#【Commands which accept address ranges】

# d和D的区别：

# d 会删除整个pattern\_space内容，并忽略abort针对当前pattern\_space内容的所有后续操作(不管这些操作是# 否在同一个group里)，并开始一个新的周期处理（即读取新的下一行到pattern\_space中，执行相应的操作）

# D 只会删除pattern\_space中从开始位置到第一个’\n’之间的内容，而留下剩余的内容。

# (类似: pattern\_space=’one\ntwo\n’,执行D后，pattern\_space=’two\n’, 即原有的’one\n’被删除了，这种范围选择规则和P命令的范围选定规则是相同的)，

# 同时，与D命令处于相同组的命令会被忽略，当其他组中的命令仍然会被执行(除非pattern\_space中已经不再有内容)，

# （类似： '3{N;D;p};p' 中的p会被忽略，因p与D在同一组中，而p仍会被执行，因p与D不再同一组中。

# d Delete pattern space. Start next cycle.

# D If pattern space contains no newline, start a normal new cycle as if the d command was issued. Otherwise, delete text in the pattern space up to the first newline, and

restart cycle with the resultant pattern space, without reading a new line of input.

[root@bogon dir]# sed -n -e '3{N;d;p};p' a.txt

one

two

five

six

seven

eight

nine

ten

[root@bogon dir]# sed -n -e '3{N;D;p};p' a.txt

one

two

four

five

six

seven

eight

nine

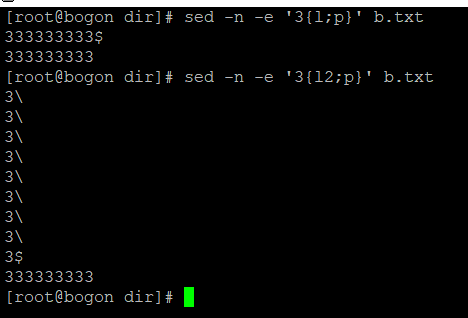
ten

#【Commands which accept address ranges】

# l List out the current line in a ``visually unambiguous'' form.

# l width

# List out the current line in a ``visually unambiguous'' form, breaking it at width # characters. This is a GNU extension.



#【Commands which accept address ranges】

# w 和 W的区别类似于p和P的区别(即只是处理pattern\_space内容是选择范围不同)

# 警告： 使用w和W需要注意： filename后面不要跟空格等任意其他符号，否则空格等任意其他符号也会被认为是filename的一部分

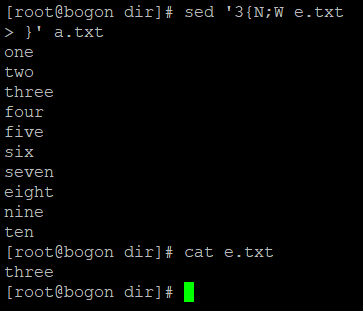
# w filename

# Write the current pattern space to filename.

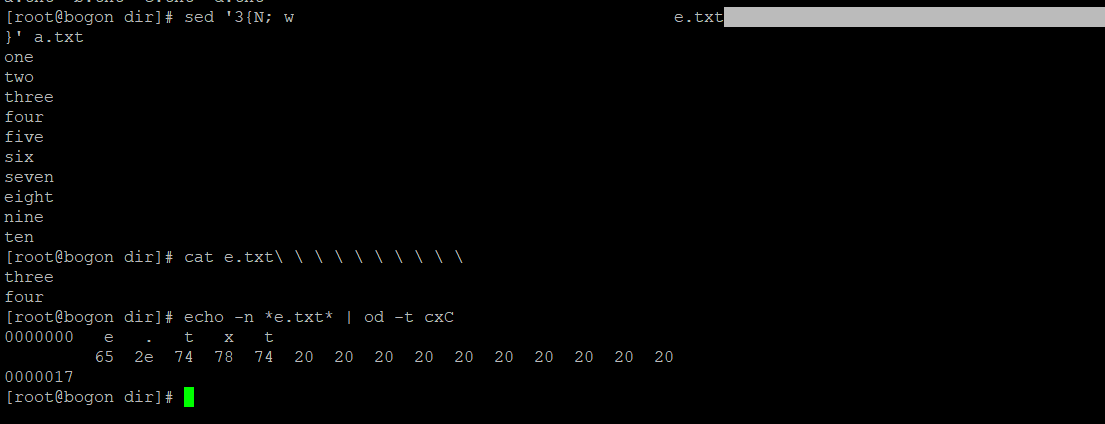
# W filename

# Write the first line of the current pattern space to filename. This is a GNU extension.





#如下就是一个w filename后跟随空格的反面示例



<http://www.grymoire.com/Unix/Sed.html#uh-36>

## [Writing a file with the 'w' command](http://www.grymoire.com/Unix/Sed.html" \l "toc-uh-36)

You may remember that the substitute command can write to a file. Here again is the example that will only write lines that start with an even number (and followed by a space):

sed -n 's/^[0-9]\*[02468] /&/w even' <file

I used the "&" in the replacement part of the substitution command so that the line would not be changed. A simpler example is to use the "w" command, which has the same syntax as the "w" flag in the substitute command:

sed -n '/^[0-9]\*[02468]/ w even' <file

Remember - only one space must follow the command. Anything else will be considered part of the file name. The "w" command also has the same limitation as the "w" flag: only 10 files can be opened in *sed*.

#【Commands which accept address ranges】

# y/source/dest/的功能类似于bash中命令'tr SET1 SET2'的功能，所以y/source/dest/中source和dest的长# 度必须相同（即需要保证source和dest中字符位置的存在一一对应关系）

# y/source/dest/

Transliterate the characters in the pattern space which appear in source to the

# corresponding character in dest.

#一个source和dest不相等的反面例子

[root@bogon dir]# sed '3 y/e/yz/' a.txt

sed: -e expression #1, char 9: strings for `y' command are different lengths

[root@bogon dir]# sed '3 y/e/y/' a.txt

one

two

thryy

four

five

six

seven

eight

nine

ten

[root@bogon dir]#

#【Commands which accept address ranges】

# s/regexp/replacement/

# 类似与vim中对应的替换命令s/regexp/replacement/

# s命令还有很多细节（包括选项g等），更多信息参见'info sed'中的

# 转换为大写，利用到了反向引用

[root@bogon dir]# sed -r 's/(.\*)/\U\1/' a.txt

ONE

TWO

THREE

FOUR

FIVE

SIX

SEVEN

EIGHT

NINE

TEN

# 转换为小写(将上一个命令的大写输出转换为小写)

[root@bogon dir]# sed -r 's/(.\*)/\U\1/' a.txt | sed -r 's/(.\*)/\L\1/'

one

two

three

four

five

six

seven

eight

nine

ten

<https://stackoverflow.com/questions/4569825/sed-one-liner-to-convert-all-uppercase-to-lowercase>

# [sed one-liner to convert all uppercase to lowercase?](https://stackoverflow.com/questions/4569825/sed-one-liner-to-convert-all-uppercase-to-lowercase)

With tr:

# Converts upper to lower case

$ tr '[:upper:]' '[:lower:]' < input.txt > output.txt

# Converts lower to upper case

$ tr '[:lower:]' '[:upper:]' < input.txt > output.txt

Works using GNU sed (BSD sed doesn't support \L \U):

# Converts upper to lower case

$ sed -e 's/\(.\*\)/\L\1/' input.txt > output.txt

# Converts lower to upper case

$ sed -e 's/\(.\*\)/\U\1/' input.txt > output.txt

|  |
| --- |
| 参考资料：  <http://www.grymoire.com/Unix/Sed.html>  <http://sed.sourceforge.net/sed1line.txt>  <https://linuxconfig.org/learning-linux-commands-sed>  <https://www.computerhope.com/unix/used.htm>  <https://www.tecmint.com/linux-sed-command-tips-tricks/>  <http://www.grymoire.com/Unix/Sed.html#uh-51> [Working with Multiple Lines](http://www.grymoire.com/Unix/Sed.html" \l "toc-uh-51) There are three new commands used in multiple-line patterns: "N," "D," and "P." I will explain their relation to the matching "n," "d," and "p" single-line commands.  The "n" command will print out the current pattern space (unless the "-n" flag is used), empty the current pattern space, and read in the next line of input. The "N" command does **not** print out the current pattern space and does **not** empty the pattern space. It reads in the next line, but appends a new line character along with the input line itself to the pattern space.  The "d" command deletes the current pattern space, reads in the next line, puts the new line into the pattern space, and aborts the current command, and starts execution at the first *sed* command. This is called starting a new "cycle." The "D" command deletes the first portion of the pattern space, up to the new line character, leaving the rest of the pattern alone. Like "d," it stops the current command and starts the command cycle over again. However, it will not print the current pattern space. You must print it yourself, a step earlier. If the "D" command is executed with a group of other commands in a curly brace, commands after the "D" command are ignored. The next group of *sed* commands is executed, unless the pattern space is emptied. If this happens, the cycle is started from the top and a new line is read.  The "p" command prints the entire pattern space. The "P" command only prints the first part of the pattern space, up to the NEWLINE character. Neither the "p" nor the "P" command changes the patterns space.  Some examples might demonstrate "N" by itself isn't very useful. the filter  sed -e 'N'  doesn't modify the input stream. Instead, it combines the first and second line, then prints them, combines the third and fourth line, and prints them, etc. It does allow you to use a new "anchor" character: "\n." This matches the new line character that separates multiple lines in the pattern space. If you wanted to search for a line that ended with the character "#," and append the next line to it, you could use  #!/bin/sh  sed '  # look for a "#" at the end of the line  /#$/ {  # Found one - now read in the next line  N  # delete the "#" and the new line character,  s/#\n//  }' file  You could search for two lines containing "ONE" and "TWO" and only print out the two consecutive lines:  #!/bin/sh  sed -n '  /ONE/ {  # found "ONE" - read in next line  N  # look for "TWO" on the second line  # and print if there.  /\n.\*TWO/ p  }' file  The next example would delete everything between "ONE" and "TWO:"  #!/bin/sh  sed '  /ONE/ {  # append a line  N  # search for TWO on the second line  /\n.\*TWO/ {  # found it - now edit making one line  s/ONE.\*\n.\*TWO/ONE TWO/  }  }' file  <https://www.cnblogs.com/theCambrian/p/3606214.html> [sed的工作原理（pattern space 和 hold space）](https://www.cnblogs.com/theCambrian/p/3606214.html)****操作pattern space和hold space的命令：****   $ man sed         d      Delete pattern space.  Start next cycle.               删除pattern space的内容，开始下一个循环.         h H    Copy/append pattern space to hold space.             复制/追加pattern space的内容到hold space.         g G    Copy/append hold space to pattern space.             复制/追加hold space的内容到pattern space.         x      Exchange the contents of the hold and pattern spaces.               交换hold space和pattern space的内容.  --------------------------------------------------下面例子摘自耗子叔的酷壳，(\*^\_\_^\*) 嘻嘻……------------------------------------------------------------  第一个示例：  [复制代码](javascript:void(0);)  $ sed 'H;g' t.txt  one    one  two    one  two  three  [复制代码](javascript:void(0);)  是不是有点没看懂，我作个图你就看懂了。  https://images0.cnblogs.com/i/565494/201403/172140483498208.jpg    第二个示例，反序了一个文件的行：  $ sed '1!G;h;$!d' t.txt  three  two  one  其中的 ’1!G;h;$!d’ 可拆解为三个命令   * 1!G —— 只有第一行不执行G命令，将hold space中的内容append回到pattern space * h —— 第一行都执行h命令，将pattern space中的内容拷贝到hold space中 * $!d —— 除了最后一行不执行d命令，其它行都执行d命令，删除当前行   这个执行序列很难理解，做个图如下大家就明白了：  https://images0.cnblogs.com/i/565494/201403/172142019591054.jpg  就先说这么多吧，希望对大家有用。  （全文完）  Bash shell中引号可以跨多行  <http://www.grymoire.com/Unix/Sed.html#uh-16> [Quoting multiple sed lines in the Bourne shell](http://www.grymoire.com/Unix/Sed.html" \l "toc-uh-19) The Bourne shell makes this easier as a quote can cover several lines:  #!/bin/sh  sed '  s/a/A/g  s/e/E/g  s/i/I/g  s/o/O/g  s/u/U/g' <old >new  编写sed脚本 [A sed interpreter script](http://www.grymoire.com/Unix/Sed.html" \l "toc-uh-20) Another way of executing *sed* is to use an interpreter script. Create a file that contains:   #!/bin/sed -f s/a/A/g s/e/E/g s/i/I/g s/o/O/g s/u/U/g  Click here to get file: [CapVowel.sed](http://www.grymoire.com/Unix/Scripts/CapVowel.sed) If this script was stored in a file with the name "CapVowel" and was executable, you could use it with the simple command:  CapVowel <old >new  关于shebang中带有多个参数的讨论  <https://stackoverflow.com/questions/4303128/how-to-use-multiple-arguments-for-awk-with-a-shebang-i-e>  <https://unix.stackexchange.com/questions/399690/multiple-arguments-in-shebang>  <https://stackoverflow.com/questions/45612626/shebang-multiple-parameters?noredirect=1&lq=1>  [root@bogon dir]# vim ./uppercase.sed    [root@bogon dir]# sed -r -f ./uppercase.sed a.txt  [root@bogon dir]# ./uppercase.sed a.txt |