45 - Text bearbeiten mit sed (Stream Editor)

a\	append
Text	hängt den Text an die angegebene Cursor Position
d	delete - löscht die komplette Zeile
i\	insert
Text	der Text wird vor dem Cursor Position ausgegeben
q	quit
r Datei	read - Der Inhalt der Datei wird hinter dem Cursor angehängt
s/*suchen/ersetzen/flag*	
	flag: g = alle ersetzetnflag 1 = nur einmal ersetzen

z.B. `head -20 /etc/services | sed -e '1,14d'` (löscht die ersten 14 Zeilen)

```
find . -type d -maxdepth 1 | sort | sed -e 's/.\/*//'
find . -type d -maxdepth 1 | sort | sed -e '1d; s/.\///'
```

---

1. Sed syntax:

sed [-n] [-e *commands***] [-f** *scriptname***]** *textfile* **[>** *filename***]**

-n	Prevents sed from sending the changed file to standard output, except if instructed by p (print) flag if it is present.
-e	option alerts sed that commands are immediately following
-f	option names the file containing the sed commands(scripts)
textfile	is the the file to be processed
> *filename*	is the name of the output (processed) file to save onto disk.

When sed runs, sed does the following:

- sed reads one line of input text from the textfile
- it then reads the first command (in command line or sed script)
- if the command applies to this line it executes it
- it reads the next command and so on till end of commands
- when all commands are processed then sed reads the next line from the textfile and processes it as well.

Sed instructions are formatted as follows:

[*address1* **[,***address2***]]** *instruction* **[***argument-list***]**

- Addresses are optional. They can be line number(s) or a pattern (regex.) They specify the line to which the instruction apply
- If no addresses are given than sed processes all the lines of the textfile
- 2 addresses define the range of line numbers to which the processing will occur These 2 addresses are shown as **[***address1* **[,***address2***]**
- By default each line read from the textfile (whether processed or not) is sent to the standard output.
- Each time the term /pattern/x is seen in the following examples, it also means the possibility to enter an address range in the format /pattern1,*pattern2*/x where x is the command.
- Sed never changes the original. All output can be redirected to a new file using > *file*
- **Note:** For all examples given below it is recommended to pipe the result into less program to see the results. (**| less**)

2. Displaying a range of lines (/.../p)
sed can be used to do the same function as grep with the following syntax:

```
sed -n /textToSeacrh/p filename
```
eg. **sed -n /ftp/p /etc/services**
Shows all the lines having at least once the word 'ftp' in them
(-n option prevents all the input lines to be displayed as well)

3. Displaying a range of lines (p)
sed can also be used to display a range of lines depending on their line numbers:

```
sed -n 'firstLine,lastLine p'  filename
```
eg. **sed -n '10,100 p' /etc/services**

4. Inserting a line before the text in file (i\)

```
sed 'i\Text to insert' /etc/motd
```
eg. `sed 'i\This is the inserted text' /etc/motd`

5. Appending a line after a text (a\)

```
sed 'a\Text to append' /etc/motd
```
eg. `sed 'a\This is the appended text' /etc/motd`

6. Find and quit printing when found (/q)

```
sed '/text to find/q' /etc/services
```
eg. `sed '/ftp/q' /etc/services`

7. Quit printing the text after *x* lines (like head -n*x*)

```
sed '4q' /file/to/edit
```
eg. `nl -ba /etc/fstab | sed '4q'` (shows only the first 4 lines)

8. Substitutions (s)
sed can search for a pattern and each time replace it with another

```
sed 's/pattern/replacementText/flag' filename
```
eg1. **sed 's/ftp/FTP/g' /etc/services**
eg2. **sed -n 's/ftp/FTP/gpw result.txt' /etc/services**

Notes: - The flag 'g' is different than the command g (paste: given after a search pattern only)
and is given for globally changing all occurences of the pattern.
Without the 'g' , sed stops processing the line after the first substitution.
- The flag 'p' provokes the extra printing of changed lines
- The flag '`w file`' writes the line to a *file*

Important for the w flag:
You must leave one and only one space between the **w** and the ***file***

More substitutions Examples:

's3/*pattern1/replacement/***'**	Replace only the THIRD occurence of the pattern
'1s/*pattern1/replacement/***'**	Process only the first line for substitution if applies
'$s/*pattern1/replacement/***'**	Process only the last line for substitution if applies
'1,3s/*pattern1/replacement/***'**	Process only the line 1 to 3 for substitution "
-n '3,7s/*pattern1/replacement***/p'**	Print only the modified lines between line 3 and 7
-n '3,7!s/*pattern1/replacement***/p'**	Print only the modified lines of the whole file except between line 3 and 7.

`'/word/s/pattern1/replacement/'` Process all lines that contains the `word` for substitution if applies.(*)

`'/Word1/,/Word2/s/pattern1/replacement/'`
Process all lines between the first line containing the **Word1** and the line containing **word2**. After this process sed resumes with the rest of the input text and processes it the same way. (*)

Note: (*) These above last 2 substitutions can be surrounded by quotes `" ... "` instead of `' ... '`. Useful when we need to use variables in the command.

9. Addind a string after the specified line (a\)

```
sed -e '/pattern/a\
string' filename
```

This example(in 2 lines) appends a line containing `hallo` after each line containing `echo`

```
sed -e '/echo/a\
hallo ' /etc/services
```

Note: - The command MUST be given in minimum 2 lines
- If the *string* contains more than one line each line must be terminated with a '\'

eg.
```
sed -e '/echo/a\
hallo this is the first line to append \
the second line terminated also with a backslash \
and the last one with the quote and filename ' /etc/services
```

10. Addind a string before the specified line (i\)

Same as above except the command is **i** instead of **a**

11. Deleting lines (d)

`sed -e '/pattern/d'` Deletes all lines containing the pattern

or `sed -e '/pattern/!d'` Deletes all lines NOT containing this pattern

eg. `sed -e '/echo/d' /etc/services`
Deletes all lines containing the pattern 'echo'

12. Replacing a whole line with a text (c\)

```
sed -e '/pattern/c\
newtext ' filename
```

eg.
```
sed -e '/echo/c\
This line had an echo keyword in it....\
now its replaced with this one' /etc/services
```

Replaces all the lines containing the pattern 'echo' with the text 'This line.....

13. Copy and Replace (h ....g)

Copy a line(s) into the hold space and paste it (replace original) somewhere further.

```
sed -e '/pattern1/h' -e '/pattern2/g' filename
```

eg: `sed -e '/echo/h' -e '/quote/g' /etc/services`

This example finds all the lines containing 'echo' and copies(**h**) it into the hold space.
Note: Only the last found line or last group of lines containing 'echo' will be remaining in the hold space. And each line containing 'quote' will be replaced(**g**) with the content of the hold space.
Attention: If the line containing 'quote' is found before the one containing 'echo' then the line(s) containing 'quote' will be replaced with 2 blank lines. Not quite the same as delete.

14 . Exclusive Cut and paste (h d.....G)

```
    sed -e '/pattern1/h' -e '/pattern1/d' -e '/pattern2/G' filename
eg. sed -e '/echo/h' -e '/echo/d' -e '/quote/G' /etc/services
```

The line containing *pattern1* is copied into the hold space(**h**) and then deleted(**d**) to simulate the cut function. Then the hold space's content line(s) will be appended(**G**) after each line(s) containing the *pattern2*.
Note: although each line containing the *pattern1* will be deleted, only the last line will be retained in the hold space with the command **h**. To accumulate all the 'cutted' lines into the hold space before pasting them, use the command below **Cumulative cut and Paste.**

15. Cumulative cut and Paste (H d .....G)

```
    sed -e '/pattern1/H' -e '/pattern1/d' -e '/pattern2/G' filename
eg. sed -e '/echo/H' -e '/echo/d' -e '/quote/G' /etc/services
```

This function above is almost the same as the 'Exclusive Cut and Paste' except that it will accumulate the cut lines into the hold space before it will be pasted.
NOTE: The **H** command will add an extra blank line in the hold space before it saves the first line containing the *pattern1*. The result is that there will be a blank line between the line containing the *pattern2* and the pasted lines (from the hold space).

16. Exchanging a line containing a pattern with the hold space (x)

```
    sed -e '/pattern/x' filename
eg. sed -e '/^fsp/h' -e '/23\/udp/x' -e '/31\/tcp/g' /etc/services
```

This example copies the line where **fsp** starts the line(^) into the hold space
This content is then exchanged for the line containing the pattern **23/udp**
The content being the line of the last exchange is then replacing the line containing the pattern **31/tcp**

17. Character translation(y)

```
    sed -e '/pattern/y/abc/xyz/' filename
```

Transtes all **a** into **x**, **b** into **y**, and **c** into **z** in all lines containing the *pattern*

```
eg. sed -e '/tcp/y/\#t/x§T/' /etc/services
```

Translates all # into **§** and **t** into **T** in lines containing the **tcp** pattern.

Note: Since the # is a regex metacharacter the \ is used to escape it (take it litteral) but since both fields (field of chars to be replaced and the replacement chars field) MUST be of the same length for sed to accept it (in this case 3 chars) the **x** is used as a placeholder only and is not taken as anything else than that.

eg2. sed -e '/tcp/y/\//-/' /etc/services
This time the / as a character to be replaced being a command delimiter for sed is simply escaped with a \ but doesn't need to be equalized like the example above (number of characters per fields)...probably because sed gets rid of the \ immediately which is not the case for escaping regular expression's metacharacters.

18. Quitting the processing of the input file prematurely (q)
It is possible to give sed the order to stop the input processing at a certain point depending on a line number or a pattern.

```
    sed -e '/pattern/q' filename
eg. sed -e '/tcp/y/\#/x§/' -e '/telnet/q' /etc/services
```

In this above example all lines containing **tcp** are having their # translated into **§** until a line containning **telnet** occurs which provokes the premature stop of the file editing.

19. External file reading and insertion

sed -e '*/pattern/***r** *extfilename*' *filename*
eg. **sed -e '/echo/r /etc/exports' /etc/services**

Important:
You must leave one and only one space between the **r** and the *extfilename*

20. Replacement using the full search patern recall

We can use the full content of the patern found again in the replacement string using the char '&'

eg. **sed /hallo[0-9][0-9]/&new/**
Substitutes all the **hallo00** to **hallo00new**
Substitutes all the **hallo01** to **hallo01new**
etc.

21. Replacement using the full search patern recall (Cut and Paste)

We can save a part or whole of the search pattern into a buffer and recall it in the replacement string by enclosing the part to save into parenteses **\(.....\)**
Maximum 9 strings can be saved in this manner and recalled via **\1** to **\9**

eg.
sed 's/hallo\([0-9][0-9]\)/allo\1/'
Substitues the words **hallo00** to **allo00**
Substitues the words **hallo01** to **allo01**

sed commands summary (used with `-e 'command'`)

Basic editing

a\<cr>*string* Adds a string after the specified line Note: <cr> is a Carriage Return
c\<cr>*string* Replaces the specified line with the following string (range of lines allowed)
i\<cr>*string* Adds a string before the specified line
d *lineNr* Deletes the specified line (range of lines allowed)
y Translates characters like UNIX command **tr**
s/*string1*/*string2*/ Substitute the specified string1 for the string2

Line Information

r Inserts an file at the specified line(the file will be inserted after the line)
p Print line to standard output
q Quit after the search has succeeded

Input/Output Processing

n Skip current line and go to line below
r Read another file's content into the output stream
w Write input lines to another file
q Quit the sed script (no further output)

Yanking and Putting (Cut and Paste)

h Copy into hold space; wipe out what's there
H Copy into hold space; append to what's there
g Get the hold space back; whipe out the pattern space
G Get the host space back; append to the pattern space
x Exchange contents of the hold and pattern space

Branching commands

b Branch to label or to end of script
t Same as **b** but branch only after substitution
:*label* Label branched to by **t** or **b**

Multiple Input Processing

N Read another line of input (creates embedded newline)
D Delete up to the embedded newline
P Print up to the embedded newline

Misc

Comments
= Writes line numbers to standard output

sed

Stream Editor

$sed [**-n**] [**-e** '*script* ' [**-f** *file*] [*files*]

Description:

The **sed** command copies *files* to standard output and edits them according to the given editing commands.

Options:

-e *script*	execute commands in *script*
-f *file*	get commands from *file*
-n	suppress default output
files	read standard input if no *files* given

Command Format:

[*address* [,*address*]] *commands* [*arguments*]

execute *commands* for each input line that matches *address* or range of addresses. If *commands* is preceded by "!", input lines that do not match the address are used.

Addresses:

If no address is given, all input lines are matched.

Two addresses indicate a range.

.	current line
$	last line
n	*n*th line
/regularexpression/	regular expression
\n	newline

Commands:

The maximum number of addresses listed in parentheses.

(1)**a**	append the following text to the output *text*
(2)**b** *label*	branch to the **:***label* command. If *label* is empty, go to the end of the script.
(2)**c**	change line *text*
(2)**d**	delete lines
(2)**D**	delete first line of input only
(2)**g**	replace input lines with buffer contents
(2)**G**	append buffer contents to input lines
(2)**h**	replace buffer contents with input lines
(2)**H**	append input lines to buffer contents
(1)**i**	insert the following text *text*
(2)**l**	display input lines
(2)**n**	display input line; read next input line

(2)**N**	append next input line to current line
(2)**p**	display input lines
(2)**P**	display first line of input line only
(1)**q**	quit
(2)**r** *file*	display contents of *file*
(2) **s**/*RE*/*s*/*flags*	substitute *s* for the regular expression *RE* that matches input lines according to *flags*. The *flags* can consist of zero or more of: **n** substitute for just the *nth* occurrence of the regular expression *RE* (**1-512**) **g** substitute for all occurrences of *RE* **p** display input line if substitution was made **w** *file* append input line to *file* if substitution made
(2)**t** *label*	branch to **:***label* command if substitution was made. If label is empty, go to end of script.
(2)**w** *file*	append input line to *file*
(2)**x**	exchange input line with buffer contents
(2)**y**/*s1*/*s2*	replace characters in *s1* with characters in *s2*. The lengths of *s1* and *s2* must be equal.
(2)**!***cmd*	execute command for the input lines not selected
(1) =	display current line number
(2) **{**	treat commands up to closing } as a group
(0)#	interpret rest of input line as comments
(0)#**n**	interpret rest of input line as comments and ignore