sed – a stream editor

Alexander Voigt

Technische Universität Dresden
Institut für Kern- und Teilchenphysik

IKTP Computing Kaffee
14 February 2011

Content

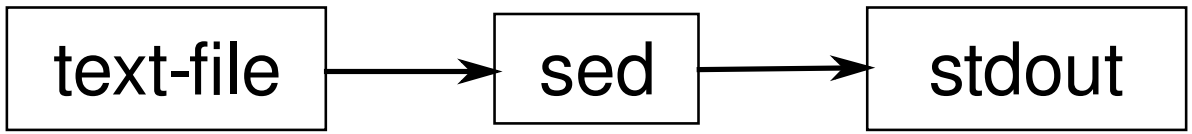
sed – What is it made for?

Intention:

- processing text-based data, either in files or data streams
- automate editing actions (search–and–replace, delete, insert, . . .) on one or more files
- write conversion programs

Design:

- non-interactive editor
- stream-orientated: input flows through the program and is directed to the standard output

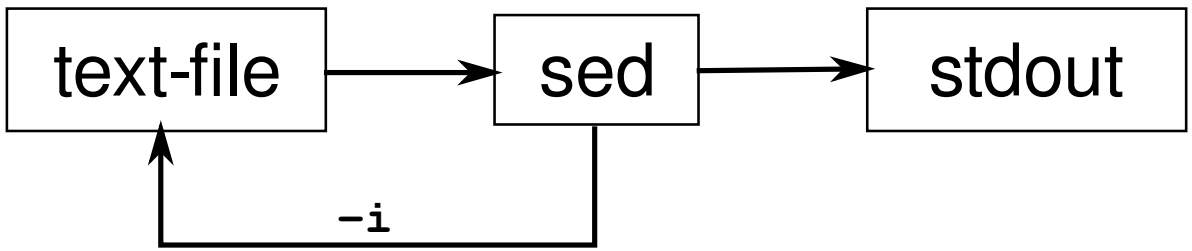


Calling sed from the command line

Syntax to call sed:

```
$ sed [options] -f program-file text-file
$ sed [options] -e 'program-text' text-file
$ ... | sed [options] -f program-file
$ ... | sed [options] -e 'program-text'
```

option	description
`-e`	sed instrucion follows (multiple instructions allowed)
`-f`	file name of sed script follows (multiple files allowed)
`-n`	suppress automatic output of input lines
`-i[suffix]`	edit files in place (make backup if extension supplied)



sed command line examples

Examples using input from text file:

```
$ cat my-program.sed
s/RS/Richard Stallman/
s/EM/Eben Moglen/
$ sed -f my-program.sed text.txt
```

```
$ sed -e 's/IKTP/My institute/' -e 's/TU/My university/'
  ↪ text.txt
```

Example using input from stdin:

```
$ echo "TU Dresden" | sed -e 's/TU/My university in/'
My university in Dresden
```

sed commands

The most important sed commands are:

command	description
`[address]s/a/b/[flags]`	substitute string "a" by string "b" once
`[address]d`	delete line
`[address]p`	print line
`[address]y/ABC/abc/`	transliterate characters
`[address]=`	print line numbers of input file
`[address]N`	search in two sequently lines
`[address]n`	print line and read next line
`[address]q`	quit sed when address is reached
`[address]a\` *text*	insert *text* after the matched line(s)
`[address]i\` *text*	insert *text* before the matched line(s)
`[address]c\` *text*	replace the matched line(s) by *text*

Addresses

The address specifies the (range of) lines on which a sed command shall be applied.

Addresses can be:

address type	syntax	examples
regular expression	`/<re>/`	`/IKTP|iktp/`
line number	*n*	`1` or `$` (the last line)
comma separated address range	`<addr1>,<addr2>`	`1,$` or `1,/^$/`

Note: If an address is followed by a `!`, the command is applied to all lines that do not match the address.

The print command

Syntax:

```
[address]p
```

Examples:

```
$ cat text.txt
line 1
line 2

line 4
$ sed -n -e '1p' text.txt # print first line
line 1
$ sed -n -e '2,$p' text.txt # print lines 2 ... last
line 2

line 4
$ sed -n -e '/^$/!p' text.txt # print non-blank lines
line 1
line 2
line 4
```

The delete command

Syntax:

```
[address]d
```

Simple examples:

```
$ cat text.txt
line 1
line 2

line 4
$ sed -e 'd' text.txt # delete all lines
$ sed -e '1d' text.txt # delete first line
line 2

line 4
$ sed -e '$d' text.txt # delete last line
line 1
line 2

$ sed -e '/^$/d' text.txt # delete blank lines
line 1
line 2
line 4
```

The delete command — examples

Advanced example: Delete all lines except LaTeX align environments

```
$ cat latex.tex
This is Einstein's famous equation:
\begin{align}
  E = m c^2
\end{align}
where the speed of light is
\begin{align}
  c = \lambda f
\end{align}
and this was all I have to say.
```

```
$ sed -e '/\\begin{align}/,/\\end{align}/!d' latex.tex
\begin{align}
  E = m c^2
\end{align}
\begin{align}
  c = \lambda f
\end{align}
```

The subsitution command

Syntax: Replace "replaceme" by "replacement"

```
1 [address]s/replaceme/replacement/[flags]
```

The following flags are allowed:

flags	description
	replace only 1st ocurrence of the pattern
n	a number (1 to 512); replace only *n*th ocurrence of the pattern
g	replace all ocurrences of the pattern
p	print line after substitution was made
w *filename*	write to file after substitution was made
i	case-insensitive matching (GNU extension)

The subsitution command — examples

Most important example: Substituting string "IKTP" in all files "*.txt"

```
$ sed -i~ -e 's/IKTP/My institute/g' *.txt
```

Advanced example: Replace "MSSM" by "E6SSM" only in LaTeX align environments

```
$ sed -i~ -e
  ↪  '/\\begin{align}/,/\\end{align}/s/MSSM/E6SSM/g'
  ↪  thesis.tex
```

Advanced substitution

Syntax: Replace "replaceme" by "replacement"

```
[address]s/replaceme/replacement/[flags]
```

In the replacement section one can use matched (sub-)strings of the pattern section by using the following symbols:

symbol	description
&	the matched pattern
\n	the *n*th substring specified in the pattern using \(and \)

Example: surround matched pattern by *

```
$ echo "The CMSSM is nicer than the MSSM." | sed -e
  ↪  's/C\?MSSM/*&*/g'
The *CMSSM* is nicer than the *MSSM*.
```

Example: switch matched sub-patterns "MSSM" $\leftrightarrow$ "CMSSM"

```
$ echo "The CMSSM is nicer than the MSSM." | sed -e
  ↪  's/\(C\?MSSM\)\(.*\)\(C\?MSSM\)/\3\2\1/g'
The MSSM is nicer than the CMSSM.
```

Grouping commands

Grouping commands:
Execute multiple sed commands on a certain address (range)

Syntax:

```
address{
  command1
  command2
  ...
}
```

Example: do replacements only in LaTeX align environments

```
/\\begin{align}/,/\\end{align}/{
  s/MSSM/\\text{MSSM}/g
  s/E6SSM/\\text{E6SSM}/g
  s/cE6SSM/\\text{cE6SSM}/g
  /^$/d
}
```

Regular expressions

Regular expression = string which describes a set of strings by use of meta symbols and syntactic rules

meta symbol	description	example
`[]`	match single character within [ ]	`[abc]`
`|`	choice operator	`a|b`
`\?`	match zero or one time	`a\?`
`\+`	match one or more times	`a\+`
`*`	match zero or more times	`a*`
`-`	define range within [ ]	`[a-zA-Z]`
`^`	match starting position in string	`^abc`
`$`	match ending position of string	`xyz$`
`.`	match any single character	`ab.defg`
`\`*n*	*n*th marked subexpression	`\1`
`\( \)`	marked subexpression	`\(abc\)`
`\{ \}`	match between m and n times	`a\{1,3\}`