Learning Linux Commands: sed

28 November 2020 by Admin

Introduction

Welcome to the second part of our series, a part that will focus on sed, the GNU version. As you will see, there are several variants of sed, which is available for quite a few platforms, but we will focus on GNU sed versions 4.x. Many of you have already heard about sed and already used it, mainly as a substitution tool. But that is just a segment of what sed can do, and we will do our best to show you as much as possible of what you can do with it. The name stands for Stream EDitor, and here "stream" can be a file, a pipe or simply stdin. We expect you to have basic Linux knowledge and if you already worked with regular expressions or at least know what a regexp is, the better. We don't have the space for a full tutorial on regular expressions, so instead we will only give you a basic idea and lots of sed examples. There are lots of documents that deal with the subject, and we'll even have some recommendations, as you will see in a minute.

Installation

There's not much to tell here, because chances are you have sed installed already, because it's used in various system scripts and an invaluable tool in the life of a Linux user that wants to be efficient. You can test what version you have by typi

```
$ sed --version
```

On my system, this command tells me I have GNU sed 4.2.1 installed, plus links to the home page and other useful stuff. The package is named simply 'sed' regardless of the distribution, but if Gentoo offers sed implicitly, I believe that means you can rest assured.

Concepts

Before we go further, we feel it's important to point out *what* exactly is it that sed does, because "stream editor" may not ring too many bells. sed takes the input text, does the specified operations on every line (unless otherwise specified) and prints the modified text. The specified operations can be append, insert, delete or substitute. This is not as simple as it may look: be forewarned that there are a lot of options and combinations that can make a sed command rather difficult to digest. So if you want to use sed, we recommend you learn the basics of regexps, and you can catch the rest as you go. Before we start the tutorial, we want to thank Eric Pement and others for inspiration and for what he's done for everyone who wants to learn and use sed.

Regular expressions

As sed commands/scripts tend to become cryptic, we feel that our readers must understand the basic concepts instead of blindly copying and pasting commands they don't know the meaning of. When one wants to understand what a regexp is, the key word is "matching". Or even better, "pattern matching". For example, in a report for your HR department you wrote the name of Nick when referring to the netw architect. But Nick moved on and John came to take his place, so now you have

replace the word Nick with John. If the file is called report.txt, you could do

```
$ cat report.txt | sed 's/Nick/John/g' > report_new.txt
```

By default sed uses stdout, so you may want to use your shell's redirect operator, as in our example below. This is a most simple example, but we illustrated a few points: we match the pattern "Nick" and we substitute all instances with "John". Note that sed is case-sensitive, so be careful and check your output file to see if all the substitutions were made. The above could have been written also like this:

```
$ sed 's/Nick/John/g' report.txt > report_new.txt
```

OK, but where's the regular expressions, you ask? Well, we first wanted to get your feet wet with the concept of matching and here comes the interesting part.

If you aren't sure if you wrote "nick" by mistake instead of "Nick" and want to match that as well, you could use sed 's/Nick|nick/John/g'. The vertical bar has same meaning that you might know if you used C, that is, your expression will match Nick *or* nick. As you will see, the pipe can be used in other ways too, but its' meaning will remain. Other operators widely used in regexps are '?', that match zero or one instance of the preceding element (flavou?r will match flavor and flavour), '*' means zero or more and '+' matches one or more elements. '^' matches the start of the string, while '$' does the opposite. If you're a vi(m) user, some of these things might look familiar. After all, these utilities, together with awk or C have their roots in the early days of Unix. We won't insist anymore on the subject, as things will become simpler by reading examples, but what you should know is that there are various implementations of regexps: POSIX, POSIX Extended, Perl or various implementations of fuzzy regular expressions, guaranteed to give you a headache.

sed examples

Learning Linux sed command with examples	
Linux command syntax	**Linux command description**
`sed 's/Nick/John/g' report.txt`	Replace every occurrence of Nick with John in report.txt
`sed 's/Nick|nick/John/g' report.txt`	Replace every occurrence of Nick or nick with John.
`sed 's/^/ /' file.txt >file_new.txt`	Add 8 spaces to the left of a text for pretty printing.
`sed -n '/Of course/,/attention you \ pay/p' myfile`	Display only one paragraph, starting with “Of course” and ending in “attention you pay”
`sed -n 12,18p file.txt`	Show only lines 12-18 of file.txt

Learning Linux sed command with examples	
`sed 12,18d file.txt`	Show all of file.txt *except* for lines from 12 to 18
`sed G file.txt`	Double-space file.txt
`sed -f script.sed file.txt`	Write all commands in script.sed and execute them
`sed '5!s/ham/cheese/' file.txt`	Replace ham with cheese in file.txt except in the 5th line
`sed '$d' file.txt`	Delete the last line
`sed '/[0-9]\{3\}/p' file.txt`	Print only lines with three consecutive digits
`sed '/boom/!s/aaa/bb/' file.txt`	Unless boom is found replace aaa with bb

Learning Linux sed command with examples	
`sed '17,/disk/d' file.txt`	Delete all lines from line 17 to 'disk'
`echo ONE TWO | sed "s/one/unos/I"`	Replaces one with unos in a case-insensitive manner, so it will print "unos TWO"
`sed 'G;G' file.txt`	Triple-space a file
`sed 's/.$//' file.txt`	A way to replace dos2unix 🙂
`sed 's/^[ ^t]*//' file.txt`	Delete all spaces in front of every line of file.txt
`sed 's/[ ^t]*$//' file.txt`	Delete all spaces at the end of every line of file.txt

Learning Linux sed command with examples	
`sed 's/^[ ^t]*//;s/[ ^]*$//' file.txt`	Delete all spaces in front and at the end of every line of file.txt
`sed 's/foo/bar/' file.txt`	Replace foo with bar only for the first instance in a line.
`sed 's/foo/bar/4' file.txt`	Replace foo with bar only for the 4th instance in a line.
`sed 's/foo/bar/g' file.txt`	Replace foo with bar for all instances in a line.
`sed '/baz/s/foo/bar/g' file.txt`	Only if line contains baz, substitute foo with bar
`sed '/./,/^$/!d' file.txt`	Delete all consecutive blank lines except for EOF

Learning Linux sed command with examples	
`sed '/^$/N;/\n$/D' file.txt`	Delete all consecutive blank lines, but allows only top blank line
`sed '/./,$!d' file.txt`	Delete all leading blank lines
`sed -e :a -e '/^\n*$/{$d;N;};/\n$/ba' \ file.txt`	Delete all trailing blank lines
`sed -e :a -e '/\\$/N; s/\\\n//; ta' \ file.txt`	If a file ends in a backslash, join it with the next (useful for shell scripts)
`sed '/regex/,+5/expr/'`	Match regex plus the next 5 lines
`sed '1~3d' file.txt`	Delete every third line, starting with the first

Learning Linux sed command with examples	
`sed -n '2~5p' file.txt`	Print every 5th line starting with the second
`sed 's/[Nn]ick/John/g' report.txt`	Another way to write some example above. Can you guess which one?
`sed -n '/RE/{p;q;}' file.txt`	Print only the first match of RE (regular expression)
`sed '0,/RE/{//d;}' file.txt`	Delete only the first match
`sed '0,/RE/s//to_that/' file.txt`	Change only the first match
`sed 's/^[^,]*,/9999,/' file.csv`	Change first field to 9999 in a CSV file
`s/^ *\(.*[^ ]\) *$/|\1|/;`	sed script to conver CSV file to bar-

Learning Linux sed command with examples	

`s/" *, */"|/g;`
`: loop`
`s/| *\([^",|][^,|]*\) *, */|\1|/g;`
`s/| *, */|\1|/g;`
`t loop`
`s/ *|/|/g;`
`s/| */|/g;`
`s/^|\(.*\)|$/\1/;`
	separated
	
	(works only on some types of CSV,
	
	with embedded “s and commas)

`sed ':a;s/\(^|[^0-9.]\)\([0-9]\+\)\\` `([0-9]\{3\}\)/\1\2,\3/g;ta' file.txt`	Change numbers from file.txt from 1234.56 form to 1.234.56
`sed -r "s/\<(reg|exp)[a-z]+/\U&/g"`	Convert any word starting with reg or exp to uppercase
`sed '1,20 s/Johnson/White/g' file.txt`	Do replacement of Johnson with White only on lines between 1 and 20
`sed '1,20 !s/Johnson/White/g' file.txt`	The above reversed (match all except lines 1-20)

Learning Linux sed command with examples	
`sed '/from/,/until/ { s/\<red\>/magenta/g; \ s/\<blue\>/cyan/g; }' file.txt`	Replace only between “from” and “until”
`sed '/ENDNOTES:/,$ { s/Schaff/Herzog/g; \ s/Kraft/Ebbing/g; }' file.txt`	Replace only from the word “ENDNOTES:” until EOF
`sed '/./{H;$!d;};x;/regex/!d' file.txt`	Print paragraphs only if they contain regex
`sed -e '/./{H;$!d;}' -e 'x;/RE1/!d;\ /RE2/!d;/RE3/!d' file.txt`	Print paragraphs only if they contain RE1, RE2 *and* RE3
`sed ':a; /\\$/N; s/\\\n//; ta' file.txt`	Join two lines in the first ends in a backslash
`sed 's/14"/fourteen inches/g' file.txt`	This is how you can use double quotes

Learning Linux sed command with examples	
`sed 's/\/some\/UNIX\/path/\/a\/new\\/path/g' file.txt`	Working with Unix paths
`sed 's/[a-g]//g' file.txt`	Remove all characters from a to g from file.txt
`sed 's/\(.*\)foo/\1bar/' file.txt`	Replace only the last match of foo with bar
`sed '1!G;h;$!d'`	A tac replacement
`sed '/\n/!G;s/\(.\)\(.*\n\)/&\2\1\/;//D;s/.//'`	A rev replacement
`sed 10q file.txt`	A head replacement

Learning Linux sed command with examples	
`sed -e :a -e '$q;N;11,$D;ba' \ file.txt`	A tail replacement
`sed '$!N; /^\(.*\)\n\1$/!P; D' \ file.txt`	A uniq replacement
`sed '$!N; s/^\(.*\)\n\1$/\1/;\ t; D' file.txt`	The opposite (or uniq -d equivalent)
`sed '$!N;$!D' file.txt`	Equivalent to tail -n 2
`sed -n '$p' file.txt`	... tail -n 1 (or tail -1)
`sed '/regexp/!d' file.txt`	grep equivalent

Learning Linux sed command with examples	
`sed -n '/regexp/{g;1!p;};h' file.txt`	Print the line before the one matching regexp, but not the one containing the regexp
`sed -n '/regexp/{n;p;}' file.txt`	Print the line after the one matching the regexp, but not the one containing the regexp
`sed '/pattern/d' file.txt`	Delete lines matching pattern
`sed '/./!d' file.txt`	Delete all blank lines from a file
`sed '/^$/N;/\n$/N;//D' file.txt`	Delete all consecutive blank lines except for the first two
`sed -n '/^$/{p;h;};/./{x;/./p;}'\`	Delete the last line each paragraph

Learning Linux sed command with examples	
`file.txt`	
`sed 's/.\x08//g' file`	Remove nroff overstrikes
`sed '/^$/q'`	Get mail header
`sed '1,/^$/d'`	Get mail body
`sed '/^Subject: */!d; s///;q'`	Get mail subject
`sed 's/^/> /'`	Quote mail message by inserting a “> ” in front of every line
`sed 's/^> //'`	The opposite (unquote mail message)

Learning Linux sed command with examples	
`sed -e :a -e 's/<[^>]*>//g;/</N;//ba'`	Remove HTML tags
`sed '/./{H;d;};x;s/\n/={NL}=/g'\ file.txt | sort \ | sed '1s/={NL}=//;s/={NL}=/\n/g'`	Sort paragraphs of file.txt alphabetically
`sed 's@/usr/bin@&/local@g' path.txt`	Replace /usr/bin with /usr/bin/local in path.txt
`sed 's@^.*$@<<<&>>>@g' path.txt`	Try it and see 🙂
`sed 's/\(\/[^:]*\).*/\1/g' path.txt`	Provided path.txt contains $PATH, this will echo only the first path on each line
`sed 's/\([^:]*\).*/\1/' /etc/passwd`	awk replacement – displays only the users from the passwd fil

Learning Linux sed command with examples	
`echo "Welcome To The Geek Stuff" | sed \ 's/\(\b[A-Z]\)/\(\1\)/g' (W)elcome (T)o (T)he (G)eek (S)tuff`	Self-explanatory
`sed -e '/^$/,/^END/s/hills/\ mountains/g' file.txt`	Swap 'hills' for 'mountains', but only on blocks of text beginning with a blank line, and ending with a line beginning with the three characters 'END', inclusive
`sed -e '/^#/d' /etc/services | more`	View the services file without the commented lines
`sed '$s@\([^:]*\):\([^:]*\):\([^:]*\ \)@\3:\2:\1@g' path.txt`	Reverse order of items in the last line of path.txt
`sed -n -e '/regexp/{=;x;1!p;g;$!N;p;D;}'\`	Print 1 line of conte before and after the line

Learning Linux sed command with examples	
`-e h file.txt`	matching, with a line number where the matching occurs
`sed '/regex/{x;p;x;}' file.txt`	Insert a new line above every line matching regex
`sed '/AAA/!d; /BBB/!d; /CCC/!d' file.txt`	Match AAA, BBB and CCC in any order
`sed '/AAA.*BBB.*CCC/!d' file.txt`	Match AAA, BBB and CCC in that order
`sed -n '/^.\{65\}/p' file.txt`	Print lines 65 chars long or more
`sed -n '/^.\{65\}/!p' file.txt`	Print lines 65 chars long or less
`sed '/regex/G' file.txt`	Insert blank line below every line

Learning Linux sed command with examples	
`sed '/regex/{x;p;x;G;}' file.txt`	Insert blank line above and below
`sed = file.txt | sed 'N;s/\n/\t/'`	Number lines in file.txt
`sed -e :a -e 's/^.\{1,78\}$/\`	Align text flush right
`sed -e :a -e 's/^.\{1,77\}$/ &/;ta' -e \`	Align text center

Conclusion

This is only a part of what can be told about sed, but this series is meant as a practical guide, so we hope it helps you discover the power of Unix tools and become more efficient in your work.

Related Linux Tutorials:

- How to Stream Video From VLC
- Listen To Your Favorite Radio Station With A Single Command...
- Things to install on Ubuntu 20.04
- Things to do after installing Ubuntu 20.04 Focal Fossa Linux
- Mint 20: Better Than Ubuntu and Microsoft Windows?
- CentOS vs CentOS Stream
- Bash regexps for beginners with examples

- Vim editor basics in Linux
- Big Data Manipulation for Fun and Profit Part 3
- Hung Linux System? How to Escape to the Command Line and...

System Administration

administration, beginner, commands

Comparison of major Linux package management systems

Understanding Regular Expressions

Comments and Discussions

LINUXCONFIG.ORG FORUM