

BASH Programming

50 sed Command Examples

4 years ago • by Fahmida Yesmin

sed is a useful text processing feature of GNU/Linux. The full form of **sed** is Stream Editor. Many types of simple and complicated text processing tasks can be done very easily by using **sed** command. Any particular string in a text or a file can be searched, replaced and deleted by using regular expression with **sed** command. But this commands performs all types of modification temporarily and the original file content is not changed by default. The user can store the modified content into another file if needs. The basic uses of `sed` command are explained in this tutorial by using 50 unique examples. Before starting this tutorial you must check the installed version of `sed` in your operating system by running the following command. The tutorial is designed based on GNU sed. So this version of `sed` will be required to practice the examples shown in this tutorial.

```
sed --version
```

The following output shows that GNU Sed of version 4.4 is installed in the system.

```
fahmida@fahmida-virtualbox:~$ sed --version
sed (GNU sed) 4.4
Copyright (C) 2017 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Written by Jay Fenlason, Tom Lord, Ken Pizzini,
and Paolo Bonzini.
GNU sed home page: <http://www.gnu.org/software/sed/>.
General help using GNU software: <http://www.gnu.org/gethelp/>.
E-mail bug reports to: <bug-sed@gnu.org>.
fahmida@fahmida-virtualbox:~$
```

Syntax:

If no filename is provided with `sed` command, then the script will work on standard input data. The **sed** script can be executed without any option.

Content:

1. Basic text substitution using 'sed'
2. Replace all instances of a text in a particular line of a file by using 'g' option
3. Replace the second occurrence only of a match on each line
4. Replace the last occurrence only of a match on each line
5. Replace the first match in a file with new text
6. Replace the last match in a file with new text
7. Escaping backslash in replace commands to manage search and replace of file paths
8. Replace all files full path with just the filename no directory
9. Substitute text but only if some other text is found in the string
10. Substitute text but only if some other text is not found in the string
11. Add string before after the matching pattern using '\1'
12. Delete matching lines
13. Delete matching line and 2 lines after matching line
14. Delete all spaces at end of the line of text
15. Delete all lines that have a match two times on the line
16. Delete all lines that have the only whitespace
17. Delete all non-printable characters
18. If there is a match in line append something to end of line
19. If there is a match in the line insert line before the match
20. If there is a match in the line insert line after the match
21. If there is not a match append something to the end of line
22. If there is not a match delete the line
23. Duplicate matched text after adding a space after the text
24. Replace one of a list of strings with the new string from other list
25. Replace a matched string with a string that contains newlines
26. Remove newlines from file and insert a comma at end of each line
27. Remove commas and add newlines to split the text into multiple lines
28. Find case insensitive match and delete line
29. Find case insensitive match and replace with new text
30. Find case insensitive match and replace with all uppercase of the same text

31. Find case insensitive match and replace with all lowercase of same text
32. Replace all uppercase characters in text with lowercase characters
33. Search for the number in line and append currency symbol after the number
34. Add commas to numbers that have more than 3 digits
35. Replace tab characters with 4 space characters
36. Replace 4 consecutive space characters with a tab character
37. Truncate all lines to first 80 characters
38. Search for a string regex and append some standard text after it
39. Search for a string regex and a second copy of found string after it
40. Running multi-line `sed` scripts from a file
41. Match a multi-line pattern and replace with new multi-line text
42. Replace order of two words that match a pattern
43. Use multiple sed commands from the command-line
44. Combine sed with other commands
45. Insert an empty line in a file
46. Delete all alpha-numeric characters from each line of a file.
47. Use `&` to match string
48. Switch pair of words
49. Capitalize the first character of each word
50. Print line numbers of the file

1. Basic text substitution using sed

Any particular part of a text can be searched and replaced by using searching and replacing pattern by using **sed** command. In the following example, 's' indicates the search and replace task. The word 'Bash' will be searched in the text, 'Bash Scripting Language' and if the word exists in the text then it will be replaced by the word 'Perl'.

```
echo "Bash Scripting Language" | sed 's/Bash/Perl/'
```

In the output of this execution, the word, 'Bash' exists in the text. So the output is 'Perl Scripting Language'.

```
fahmida@fahmida-virtualbox:~$ echo "Bash Scripting Language" | sed 's/Bash/Perl/'
Perl Scripting Language
fahmida@fahmida-virtualbox:~$
```

sed command can be used to substitution any part of a file content also. Create a text file named **weekday.txt** with the following content:

```
$ cat weekday.txt
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Sunday
```

The following command will search and replace the text 'Sunday', by the text 'Sunday is holiday'.

```
sed 's/Sunday/Sunday is holiday/' weekday.txt
```

In the output of this execution, 'Sunday' exists in weekday.txt file and this word is replaced by the text, 'Sunday is holiday' after executing the above **sed** command.

```
fahmida@fahmida-virtualbox:~$ cat weekday.txt
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Sunday
fahmida@fahmida-virtualbox:~$ sed 's/Sunday/Sunday is holiday/' weekday.txt
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Sunday is holiday
fahmida@fahmida-virtualbox:~$
```

[Go to top](#)

2. Replace all instances of a text in a particular line of a file using 'g' option

'g' option is used in **sed** command to replace all occurrences of matching pattern. Create a text file named **python.txt** with the following content to know the use of 'g' option. This file contains the word 'Python' multiple times.

```
$ cat python.txt
Python is a very popular language.
Python is easy to use. Python is easy to learn.
Python is a cross-platform language
```

The following command will replace all occurrences of 'Python' in the second line of the file **python.txt**. Here, 'Python' occurs two times in the second line.

```
sed '2 s/Python/perl/g' python.txt
```

The following output will appear after running the script. Here, All occurrence of 'Python' in the second line is replaced by 'Perl'.

```
fahmida@fahmida-virtualbox:~$ cat python.txt
Python is a very popular language.
Python is easy to use. Python is easy to learn.
Python is a cross-platform language
fahmida@fahmida-virtualbox:~$ sed '2 s/Python/perl/g' python.txt
Python is a very popular language.
perl is easy to use. perl is easy to learn.
Python is a cross-platform language
fahmida@fahmida-virtualbox:~$
```

[Go to top](#)

3. Replace the second occurrence only of a match on each line

If any word appears multiple times in a file then the particular occurrence of the word in each line can be replaced by using **sed** command with the occurrence number. The following **sed** command will replace the second occurrence of the searching pattern in each line of the file **python.txt**.

```
sed 's/Python/perl/g2' python.txt
```

The following output will appear after running the above command. Here, the searching text, 'Python' appears two times in the second line only and it is replaced by the text 'Perl':

```
linuxhint@u22: ~
linuxhint@u22:~$ vim python.txt
linuxhint@u22:~$ cat python.txt
Python is a very popular language.
Python is easy to use. Python is easy to learn.
Python is a cross-platform language
linuxhint@u22:~$ sed 's/Python/perl/g2' python.txt
Python is a very popular language.
Python is easy to use. perl is easy to learn.
Python is a cross-platform language
linuxhint@u22:~$
```

[Go to top](#)

4. Replace the last occurrence only of a match on each line

Create a text file named 'lang.txt' with the following content.

```
$ cat lang.txt
Bash Programming Language. Python Programming Language. Perl Programming Language.
Hypertext Markup Language.
Extensible Markup Language.
```

In order to replace only the last instance of 'Programming' on a line with 'Scripting', run the sed command as follows:

```
sed 's/\(.*\)Programming/\1Scripting/' lang.txt
```

To understand this command, note that the pattern match in this sed command: `\(.*\)Programming`, will create a capture group with all the text matching the line in a greedy fashion until the last instance of programming. This capture group is stored as '`\1`' and referenced in the replace part of sed: `\1Scripting`. Therefore the entire line match up to and until last instance of 'Programming' will be included again in the replacement line with the 'Scripting' word to end the line. This is the technique to make it work as shown in the output:

```
fahmida@fahmida-virtualbox:~$ sed 's/\(.*\)Programming/\1Scripting/g' lang.txt
Bash Programming Language.Python Programming Language.Perm Scripting Language
Hypertext Markup Language
Extensible Markup Language
fahmida@fahmida-virtualbox:~$
```

[Go to top](#)

5. Replace the first match in a file with new text

The following command will replace only the first match of the searching pattern, 'Python' by the text 'perl'. Here, '`1`' is used to match the first occurrence of the pattern.

```
sed '1 s/Python/perl/' python.txt
```

The following output will appear after running the above commands. Here. the first occurrence of 'Python' in the first line is replaced by 'perl'.


```
fahmida@fahmida-virtualbox:~$ cat python.txt
Python is a very popular language.
Python is easy to use. Python is easy to learn.
Python is a cross-platform language
fahmida@fahmida-virtualbox:~$ sed '1 s/Python/perl/' python.txt
perl is a very popular language.
Python is easy to use. Python is easy to learn.
Python is a cross-platform language
fahmida@fahmida-virtualbox:~$
```

[Go to top](#)

6. Replace the last match in a file with new text

The following command will replace the last occurrence of the searching pattern 'Python' by the text 'Bash'. Here, '\$' symbol is used to match the last occurrence of the pattern.

```
sed -e '$s/Python/Bash/' python.txt
```

The following output will appear after running the above commands:

```
fahmida@fahmida-virtualbox:~$ cat python.txt
Python is a very popular language.
Python is easy to use. Python is easy to learn.
Python is a cross-platform language
fahmida@fahmida-virtualbox:~$ sed -e '$s/Python/Bash/' python.txt
Python is a very popular language.
Python is easy to use. Python is easy to learn.
Bash is a cross-platform language
fahmida@fahmida-virtualbox:~$
```

[Go to top](#)

7. Escaping backslash in replace commands to manage search and replace of file paths

It is necessary to escape the backslash in the file path for searching and replacing. The following command of **sed** will add backslash (\) in the file path.

```
echo /home/ubuntu/code/perl/add.pl | sed 's;/;\//g'
```

The file path '/home/ubuntu/code/perl/add.pl' is provided as input in the **sed** command and the following output will appear after running the above command:

```
fahmida@fahmida-virtualbox:~$ echo /home/ubuntu/code/perl/add.pl | sed 's;/;\|/;g'
/home/ubuntu/code/perl/add.pl
fahmida@fahmida-virtualbox:~$
```

[Go to top](#)

8. Replace all files full path with just the filename no directory

The filename can be retrieved from the file path very easily by using **basename** command. **sed** command can also be used to retrieve the filename from the file path. The following command will retrieve the filename only from the file path provided by **echo** command.

```
echo "/home/ubuntu/temp/myfile.txt" | sed 's/.*\|/'
```

The following output will appear after running the above command. Here, the filename, 'myfile.txt' is printed as output.

```
fahmida@fahmida-virtualbox:~$ echo "/home/ubuntu/temp/myfile.txt" | sed 's/.*\|/'
myfile.txt
fahmida@fahmida-virtualbox:~$
```

[Go to top](#)

9. Substitute text but only if some other text is found in the string

Create a file named 'dept.txt' with the following content to replace any text based on other text:

dept.txt

```
$ cat dept.txt
List of Total Students:

CSE - Count
EEE - Count
Civil - Count
```

Two replace commands are used in the following **sed** command. Here, the text, 'Count' will be replaced by '100' in the line that contains the text 'CSE' and the text 'Count' will be replaced by '70' in the line that contains the searching pattern, 'EEE'.

```
sed -e '/CSE/ s/Count/100/; /EEE/ s/Count/70/;' dept.txt
```

The following output will appear after the running the above commands:


```
fahmida@fahmida-virtualbox:~$ cat dept.txt
List of Total Students:

CSE - Count
EEE - Count
Civil - Count
fahmida@fahmida-virtualbox:~$ sed -e '/CSE/ s/Count/100/; /EEE/ s/Count/70/;' dept.t
xt
List of Total Students:

CSE - 100
EEE - 70
Civil - Count
fahmida@fahmida-virtualbox:~$
```

[Go to top](#)

10. Substitute text but only if some other text is not found in the string

The following **sed** command will replace the 'Count' value in the line that does not contain the text 'CSE'. **dept.txt** file contains two lines that do not contain the text 'CSE'. So the 'Count' text will be replaced by 80 in two lines.

```
sed -i -e '/CSE/! s/Count/80/;' dept.txt
```

Output:

The following output will appear after running the above commands:

```
fahmida@fahmida-virtualbox:~$ cat dept.txt
List of Total Students:

CSE - Count
EEE - Count
Civil - Count
fahmida@fahmida-virtualbox:~$ sed -i -e '/CSE/! s/Count/80/;' dept.txt
fahmida@fahmida-virtualbox:~$ cat dept.txt
List of Total Students:

CSE - Count
EEE - 80
Civil - 80
fahmida@fahmida-virtualbox:~$
```

[Go to top](#)

11. Add string before and after the matching pattern using '\1'

The sequence of matching patterns of `sed` command is denoted by '\1', '\2' and so on. The following **sed** command will search the pattern 'Bash' and if the pattern matches then it will be accessed by '\1' in the part of replacing text. Here, the text 'Bash' is searched in the input text and one text is added before and another text is added after '\1'.

```
echo "Bash language" | sed 's/\(Bash\) /Learn \1 programming/'
```

The following output will appear after running the above command. Here 'Learn' text is added before 'Bash' and 'programming' text is added after 'Bash'.

```
fahmida@fahmida-virtualbox:~$ echo "Bash language" | sed 's/\(Bash\) /Learn \1 programming/'
Learn Bash programming language
fahmida@fahmida-virtualbox:~$
```

[Go to top](#)

12. Delete matching lines

'd' option is used in **sed** command to delete any line from the file. Create a file named 'os.txt' and add the following content to test the function of 'd' option.

```
$ cat os.txt
Windows

Linux

Android

OS
```

The following **sed** command will delete those lines from 'os.txt' file that contains the text 'OS':

```
sed '/OS/d' os.txt
```

The following output will appear after running the above commands:

```
fahmida@fahmida-virtualbox:~$ cat os.txt
Windows

Linux

Android

OS
fahmida@fahmida-virtualbox:~$ sed '/OS/d' os.txt
Windows

Linux

Android

fahmida@fahmida-virtualbox:~$
```

[Go to top](#)

13. Delete matching line and 2 lines after matching line

The following command will delete three lines from the file **os.txt** if the pattern, '**Linux**' is found. **os.txt** contains the text, '**Linux**' in the second line. So, this line and the next two lines will be deleted.

```
sed '/Linux/,+2d' os.txt
```

The following output will appear after running the above command:

```
linuxhint@u22: ~
linuxhint@u22:~$ cat os.txt
Windows

Linux

Android

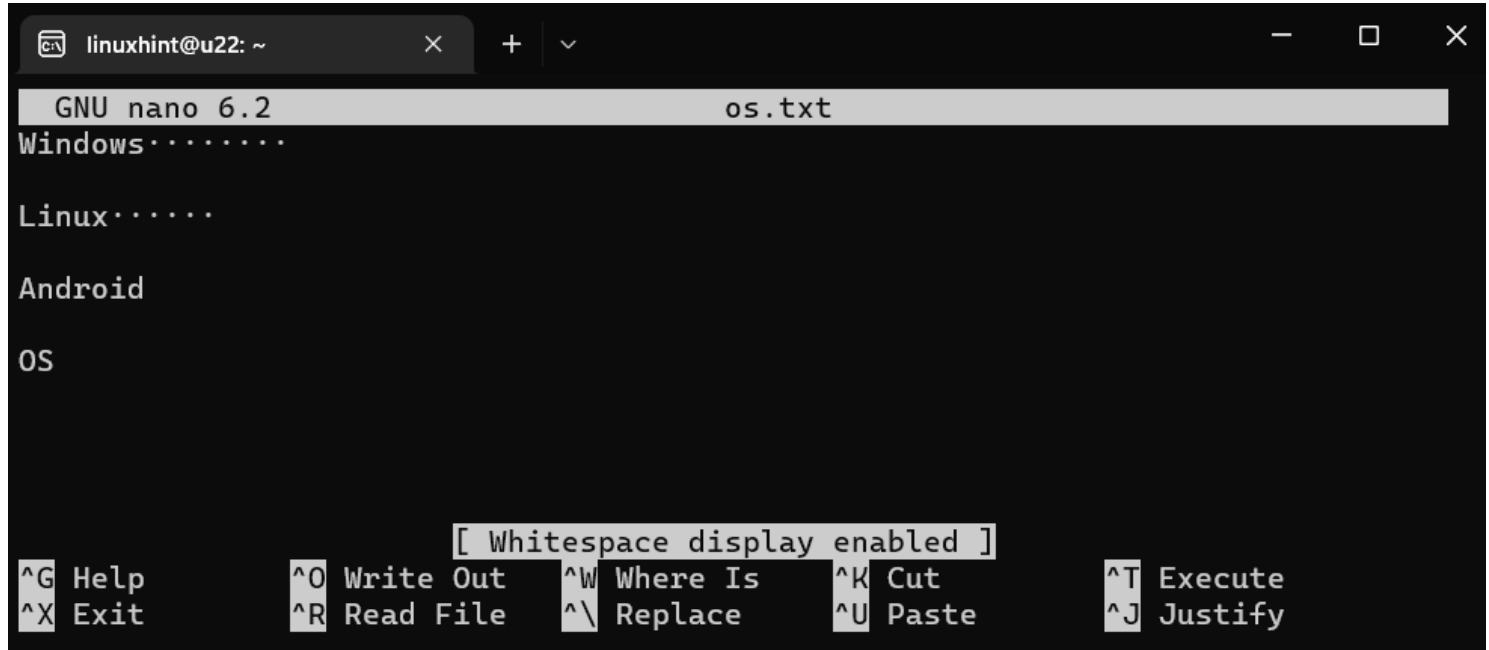
OS
linuxhint@u22:~$ sed '/Linux/,+2d' os.txt
Windows

OS
linuxhint@u22:~$
```

[Go to top](#)

14. Delete all spaces at end of the line of text

Using `[:blank:]` class can be used to remove spaces and tabs from the text or the content of any file. The following command will remove the spaces at the end of each line of the file 'os.txt'. Since this example is with empty white space we will first look at the screen shot of the file using Nano with **Alt-P** to display white space:



```
linuxhint@u22: ~  
GNU nano 6.2 os.txt  
Windows.....  
  
Linux.....  
  
Android  
  
OS  
  
[ Whitespace display enabled ]  
^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute  
^X Exit      ^R Read File  ^\ Replace    ^U Paste      ^J Justify
```

Now we can run the example sed command below to remove white space from the end of the line of each line. We will use `-i` command line to do an in place file edit/update for this example to permanently make the changes:

```
sed -i 's/[[:blank:]]*$//' os.txt
```

Now we can view the file again with nano Alt-p option to see the white space from end of line removed:

```
linuxhint@u22: ~  
GNU nano 6.2 os.txt  
Windows  
  
Linux  
  
Android  
  
OS  
  
[ Whitespace display enabled ]  
^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute  
^X Exit      ^R Read File  ^\ Replace    ^U Paste      ^J Justify
```

[Go to top](#)

15. Delete all lines that have a match two times on the line

In this example we first create a text file named 'input.txt' with the following content. This example code will delete those lines of the file that contains the searching pattern two times.

```
$ cat input.txt  
PHP is a server-side scripting language.  
PHP is an open-source language and PHP is case-sensitive.  
PHP is platform-independent. PHP is faster.
```

Two **sed** commands are used in this example to remove those lines that contain the pattern 'php' two times. The first **sed** command will replace the second occurrence of 'php' in each line by 'dl'. Then we pipe the first **sed** output into the second **sed** command as input. The second **sed** command will delete those lines that contain the special signal text 'dl'. 'dl' is used in this example as a signal for 'delete line' but the text may have 'dl' in it also, so this is a bit dangerous. For production system consider using a code like 'aNh4JPrSf3EXmdkKDpyz8K5u' instead of 'dl' to be more random or a more fancy trick you create to avoid accidents.

```
sed 's/php/dl/i2;t' input.txt | sed '/dl/d'
```

For this example output, the file 'input.txt' has two lines that contain the pattern, '**php**' two times. So, the following output will appear after running the above commands:


```
fahmida@fahmida-virtualbox:~$ cat input.txt
PHP is a server-side scripting language.
PHP is a open-source language and PHP is case-sensitive.
PHP is platform independent. PHP is faster.
fahmida@fahmida-virtualbox:~$ sed 's/php/dl/i2;t' input.txt | sed '/dl/d'
PHP is a server-side scripting language.
fahmida@fahmida-virtualbox:~$
```

[Go to top](#)

16. Delete all lines that have only white-space

For this example we add some empty lines in the content to test this example in the file 'input.txt'. Here, '^\$' is used to find out the empty lines in the file 'input.txt'.

```
$ cat input.txt
PHP is a server-side scripting language.

PHP is an open-source language and PHP is case-sensitive.

PHP is platform-independent. PHP is faster.
```

Now we run the sed command to delete the empty lines:

```
sed '/^$/d' input.txt
```

The following output will appear after running the above commands:

```
fahmida@fahmida-virtualbox:~$ cat input.txt
PHP is a server-side scripting language.

PHP is a open-source language and PHP is case-sensitive.

PHP is platform independent. PHP is faster.
fahmida@fahmida-virtualbox:~$ sed '/^$/d' input.txt
PHP is a server-side scripting language.
PHP is a open-source language and PHP is case-sensitive.
PHP is platform independent. PHP is faster.
fahmida@fahmida-virtualbox:~$
```

[Go to top](#)

17. Delete all non-printable characters

Non-printable characters can be deleted from any text by replacing non-printable characters by none. [:print:] class is used in this example to find out the non-printable characters. '\t' is a non-printable character and it can't be parsed directly by the **echo** command. For this, '\t' character is assigned in a variable, **\$tab** that is used in an **echo** command. The output of the

echo command is sent in the **sed** command that will remove the character, '\t' from the output.

```
$ tab=$'\t'
$ echo Hello"$tab"World
$ echo Hello"$tab"World | sed 's/^[[:print:]]//g'
```

The following output will appear after running the above commands. The first **echo** command will print the output with tab space and the **sed** command will print the output after removing the tab space.

```
fahmida@fahmida-virtualbox:~$ tab=$'\t'
fahmida@fahmida-virtualbox:~$ echo Hello"$tab"World
Hello  World
fahmida@fahmida-virtualbox:~$ echo Hello"$tab"World | LANG=c sed 's/^[[:print:]]//g'
HelloWorld
fahmida@fahmida-virtualbox:~$
```

[Go to top](#)

18. If there is a match in line append something to end of line

Ensure you have a file 'os.txt' with this content:

```
$ cat os.txt
Windows
Linux
Android
OS
```

The following command will append '10' at the end of the line that contains the text 'Windows' in the 'os.txt' file:

```
sed '/Windows/ s/$/ 10/' os.txt
```

The following output will appear after running the command:

[Go to top](#)

19. If there is a match in the line insert a line before the text

Ensure you have 'input.txt' for this example:

```
$ cat input.txt
PHP is a server-side scripting language.
PHP is an open-source language and PHP is case-sensitive.
PHP is platform-independent. PHP is faster.
```

The following **sed** command will search for the text 'PHP is platform-independent' in the 'input.txt' file. If the file contains this text, 'PHP is an interpreted language' will be inserted before that line.

```
sed '/PHP is platform-independent/ s/^/PHP is an interpreted language.\n/' input.txt
```

The following output will appear after running the above commands:

[Go to top](#)

20. If there is a match in the line insert a line after that line

The following **sed** command will search the text, 'Linux' in the file 'os.txt' and if the text exists in any line then a new text, 'Ubuntu', will be inserted after that line. First create the 'os.txt' file:

```
$ cat os.txt
Windows
Linux
Android
OS
```

```
sed 's/Linux/&\nUbuntu/' os.txt
```

The following output will appear after running the above commands with the added newline inserted with new text and keeping all the old text from the '&' special variable which includes the entire match:

[Go to top](#)

21. If there is not a match append something to the end of line

The following **sed** command will search those lines in 'os.txt' that do not contain the text 'Linux' and append the text, 'Operating System' at the end of those lines. Here, '\$' symbol is used to identify the end of the line which is where the new text will be substituted in. Make sure you have a 'os.txt' file:

```
$ cat os.txt
Windows
Linux
Android
OS
```

And then run the command:

```
sed '/Linux/! s/$/ Operating System/' os.txt
```

The following output will appear after running the above commands. Three lines exist in the file os.txt that does not contain the text 'Linux' and the new text is added at the end of these lines.

[Go to top](#)

22. If there is not a match delete the line

Create a file named 'web.txt' and add the following content.

```
$ cat web.txt
HTML 5
Javascript
CSS
PHP
MySQL
jQuery
```

This sed command will delete lines that do not contain the matching pattern.

```
sed '/CSS/!d' web.txt
```

The following output will appear after running the above commands. There is one line exists in the file that contains the text, 'CSS'. So, the output contains just one line. If you change the pattern from 'CSS' to 'J' you will see 2 lines of output.

```
sed '/J/!d' web.txt
```

[Go to top](#)

23. Duplicate matched text after adding a space after the text

The following **sed** command will search for a matching word, 'very ' in the file, 'python.txt' and if the word exists then the same word will be inserted again as a duplicate . Here, '&' symbol represents the match text and we add it twice for duplication. Ensure you have 'python.txt':

```
$ cat python.txt
Python is a very popular language.
Python is easy to use. Python is easy to learn.
Python is a cross-platform language
```


Here is the code:

```
sed -e 's/very /&&/g' python.txt
```

The following output will appear after running the commands. You can see we duplicated the 'very ' to make it 'very very':

[Go to top](#)

24. Replace one list of strings with the new string from other list

You have to create two list files for testing this example. Create two text files named 'list1.txt' and 'list2' and add the following content.

```
$ cat list1.txt
1001:Jafar Ali
1023:Nir Hossain
1067:John Michel
```

```
$ cat list2.txt
1001   CSE      GPA-3.63
1002   CSE      GPA-3.24
1023   CSE      GPA-3.11
1067   CSE      GPA-3.84
```

We will make this process simple from complex by breaking it down into small steps. **sed** can take a list of substitution commands as a file input, so our approach is to first create a file with the list of the relevant sed syntax substitution commands with the 's' substitution operator.

Here is the simple code to create the command file using another linux command called **awk** with low complexity. Note this is a cat of original file, piped into awk and the output redirected to 'command.txt':

```
cat list1.txt | awk -F: '{printf("s/%s/%s/\n", $1, $2)}' > command.txt
```

With the 'command.txt' file created we can run this list of substitutions easily on 'list2.txt' with a basic **sed** command:

```
sed -f command.txt list2.txt
```

The output below shows the process in 2 steps to replace the matching keys and values and the output:

[Go to top](#)

25. Replace the matched string with a string that contains newlines

The following command will take input from the **echo** command and search the word, 'Python', in the text. If the word exists in the text then a new text, 'Added Text' will be inserted with a newline in the new text. Here is the example code:

```
echo "Bash Perl Python Java PHP ASP" | sed 's/Python/Added Text\n/'
```

Both the new text and the newline are inserted. The following output will appear after running the above command:

[Go to top](#)

26. Remove newlines from file and insert a comma at end of each line

The following **sed** command will replace each newline by a comma in the file 'os.txt'.

```
$ cat os.txt
Windows
Linux
Android
OS
```

Here, '-z' option is used to instruct **sed** to interpret the input with each line separated NULL character instead of newline, which allows for further manipulation of the newlines in the text without interference. We then do a simple substitution of all newlines '\n' with commas:

```
sed -z 's/\n/,/g' os.txt
```

The following output will appear after running the above command, note the lack of terminating newline which may or may not be what your use case needs:

[Go to top](#)

27. Remove commas and add newline to split the text into multiple lines

The following **sed** command will take the comma-separated line from the **echo** command as input and replace all the commas with newlines.

```
echo "Kaniz Fatema,30th,batch" | sed "s/,/\n/g"
```

The following output will appear after running the above command. The input text contains three comma-separated data that are replaced by newline and printed in three lines:

[Go to top](#)

28. Find case insensitive match and delete line

'I' indicator is used in **sed** command for the case-insensitive match that indicates ignore case. The following **sed** command will search for lines that contains the word, 'linux', with case insensitive search, and delete the line from the output. Setup your 'os.txt' firstly noting the capitalization:

```
$ cat os.txt
Windows
Linux
Android
OS
```

Then run the **sed** command:

```
sed '/linux/I'd' os.txt
```

The following output will appear after running the above command. 'os.txt' contains the word 'Linux' that matched with the pattern, 'linux' for case-insensitive search and is deleted:

[Go to top](#)

29. Find case insensitive match and replace with new text

The following **sed** command will take the input from the **echo** command and replace the word, 'bash' by the word, 'PHP' using the case insensitive indicator at the end of the command: 'i'.

```
echo "I like bash programming " | sed 's/Bash/PHP/i'
```

The following output will appear after running the above command. Here, the word, 'Bash' matched with the word, 'bash' for case-insensitive search and replaced by the word, 'PHP'.

[Go to top](#)

30. Find case insensitive match and replace with all uppercase of the same text

'**U**' is used in **sed** to convert any text to all uppercase letter. The following **sed** command will search the word 'linux' in the 'os.txt' file and if the word exists then it will replace the word with all uppercase letters. First create the 'os.txt' file.

```
$ cat os.txt
Windows
Linux
Android
OS
```

Now here is the command to run:

```
sed 's/\(linux\)/\U\1/Ig' os.txt
```

This command uses **sed** capture groups. **\(linux\)** is captured in the **\1** for reuse in the replacement. In our example '**U**' is used with '**\1**' to make the captured match convert to uppercase. Also note '**Ig**' commands at the end of the instruction which enforce case insensitive and global matching. The following output will appear after running the above commands. The word, 'Linux' of 'os.txt file' is replaced by the word, 'LINUX':

[Go to top](#)

31. Find case insensitive match and replace with all lowercase of same text

Similar to the previous example for upper case, we will do a case insensitive search and replace matching text in all lowercase.

```
$ cat os.txt
Windows
Linux
Android
OS
```

'\L' is used in sed to convert any text to all lowercase letters, and again we use capture groups to find the matching text of `\(linux\)` and store as '`\1`' to be used for lower casing. Here is the command:

```
sed 's/\(linux\)/\L\1/Ig' os.txt
```

The following output will appear after running the above commands. The word, 'Linux' is replaced by the word, 'linux' here:

[Go to top](#)

32. Replace all uppercase characters of the text with lowercase characters

The following **sed** command will search all characters in the file 'os.txt' and replace the characters by lowercase letters by using '`\L`'.

Below is the data file:

```
$ cat os.txt
Windows
Linux
```

And here is the **sed** command to run:

```
sed 's/\(.*\) /\L\1/' os.txt
```

The following output will appear after running the above commands, note this works because we create a capture group `'1'` and use the Lower case command on it: `'\L'`.

[Go to top](#)

33. Search for number in line and append any currency symbol before the number

Create a file named 'items.txt' with the following content:

```
$ cat items.txt
HDD          100
Monitor      80
Mouse        10
```

The following **sed** command will search for the first matching number in each line of 'items.txt' file and append the currency symbol, '\$' before each number.

```
$ cat items.txt
$ sed 's/\([0-9]\)/$\1/' items.txt
```

The following output will appear after running the above commands. Here, '\$' symbol is added before the number of each line. Note we don't use '**g**' for global we only match the first number, if the example needed to accommodate multiple multi-digit numbers per line something more complex for a solution is needed.

[Go to top](#)

34. Add commas to numbers that have more than 3 digits

The following **sed** command will take a number as input from **echo** command and add a comma after each group of three digits counting from the right. Here, **‘a’** indicates the label and **‘ta’** is used to iterate conditionally the grouping process on label **‘a’** when the match is true.

```
echo "5098673" | sed -e :a -e 's/\([.*[0-9]\)\([0-9]\{3\}\)/\1,\2/;ta'
```

The number 5098673 is given in the **echo** command. Its a substitute command with a conditional label **‘a’** referenced with **‘ta’**. The search pattern is any digit followed by exactly 3 digits. The replacement pattern when found is capture group **‘\1’** separated by a **‘,’** and capture group **‘\2’**. Hence the loop is done through the entire string producing the output:

[Go to top](#)

35. Replaces tab character with 4 space characters

The following **sed** command will replace each tab character by four space characters. **‘t’** symbol is used in the **sed** command to match the tab character and **‘g’** is used to replace globally all matches. Here is the code:

```
echo -e "1\t2\t3" | sed $'s/\t/    /g'
```

The following output will appear after running the above command:

Let's see a second example of tab replacement using Nano with Alt-P to show white space as visible before and after.

File Before:

sed command:

```
sed 's/\t/ /g' input.txt > output.txt
```

File After Replacement:

[Go to top](#)

36. Replaces 4 consecutive space characters with tab character

Exactly the reverse of example 35 above, we will show a text file before replacement with consecutive spaces, and then sed to substitute tab characters using '\t' ascii code.

Before and After file shown below, notice the punctuation marks visible with Nano's 'Alt-P' to show white space indicators:

Whitespaces shown by little dots

Tabs indicated by little marks in color

Here is the command used in this example to replace 4 spaces with tabs:

```
sed -e 's/    /\t/g' input.txt > output.txt
```

[Go to top](#)

37. Truncate all lines to first 80 characters

Create a text file named **input.txt** that contains the lines more than 80 characters to test this example.

```
linuxhint@u20:~$ cat input.txt
PHP is a server-side scripting language.
PHP is an open-source language and PHP is case-sensitive.
PHP is platform-independent. PHP is faster. I like to talk a lot and hate when i get cut
off :)
```

```
sed 's/\(^.\{1,80\}\).*$/\1/' input.txt
```

The following output will appear after running the above commands, we cut off the long lines:

```
linuxhint@u20:~$ sed 's/\(^.\{1,80\}\).*$/\1/' input.txt
PHP is a server-side scripting language.
PHP is an open-source language and PHP is case-sensitive.
PHP is platform-independent. PHP is faster. I like to talk a lot and hate when i
linuxhint@u20:~$
```


[Go to top](#)

38. Search for a string regex and append some standard text after it

The following sed command will search the text, 'hello' in the input text and append the text 'John' after that text.

```
echo "hello, how are you?" | sed 's/\(hello\)/\1 John/'
```

This code works using a capture group '**\(hello\)**' referenced as '**\1**' in the replacement string. The following output will appear after running the above command:

```
linuxhint@u22:~$ echo "hello, how are you?" | sed 's/\(hello\)/\1 John/'
hello John, how are you?
linuxhint@u22:~$
```

[Go to top](#)

39. Search for string regex and append some text after the second match in each line

You may want to replace a string only on the second (or other number) match on each line. Let's create a data file 'input.txt':

```
linuxhint@u22:~$ cat input.txt
PHP is a server-side scripting language.
PHP is an open-source language and PHP is case-sensitive.
PHP is platform-independent. PHP is faster.
```

The following **sed** command will search the text for 'PHP' in each line of 'input.txt' and replace the **second match** only in each line with the text, 'New Text Added'.

```
sed 's/\(PHP\)/\1 (New Text added)/2' input.txt
```

The following output will appear after running the above commands. The searching text, 'PHP' appears two times in the second and third lines of 'input.txt' file. So, the text, 'New Text added' is inserted in the second and third lines after the second match:

[Go to top](#)

40. Running multi-line sed scripts from a file

Multiple sed commands can be stored in a file and all the commands can be executed together by running **sed** command. To demonstrate this ensure you have 'input.txt':

```
linuxhint@u22:~$ cat input.txt
PHP is a server-side scripting language.
PHP is an open-source language and PHP is case-sensitive.
PHP is platform-independent. PHP is faster.
```

Create a file named 'sedcmd' and add the following content:

```
s/PHP/ASP/
s/independent/dependent/
```

Above, two **sed** commands are added in the file. One command will replace the text 'PHP' by 'ASP' another command will replace the text 'independent' by the text 'dependent'. The '**-f**' option is used in the **sed** command to execute all the commands from the file.

```
sed -f sedcmd input.txt
```

The following output will appear after running above commands:

[Go to top](#)

41. Match a multi-line pattern and replace with new multi-line text

The following **sed** command will search for the multi-line text 'Linux\nAndroid' and if the pattern matches then the matching lines will be replaced by the new multi-line text 'Ubuntu\nAndroid Lollipop'. 'os.txt' data file is here:

```
$ cat os.txt
Windows
Linux
Android
OS
```

The sed command is here:

```
sed '$!N;s/Linux\nAndroid/Ubuntu\nAndroid Lollipop/;P;D' os.txt
```

Here, P and D are used for multiline processing. The following output will appear after running the above commands:

[Go to top](#)

42. Replace order of two words in a text that match a pattern

The following **sed** command will take the input of two words from **echo** command and replace the order of these words.

```
echo "perl python" | sed -e 's/\([^ ]*\) *\([^ ]*\)/\2 \1/'
```

This code uses 2 capture groups to get the first and second word separated by space and then in the output reverses the order of the capture groups: **'\2' '\1'**. Here is the output:

```
linuxhint@u22:~$ echo "perl python" | sed -e 's/\([^ ]*\) *\([^ ]*\)/\2 \1/'
python perl
linuxhint@u22:~$
```

[Go to top](#)

43. Execute multiple sed commands from the command-line

'-e' when combined with multiple **sed** commands separated by semi-colon can be combined together. The following **sed** command will take a text as input from **echo** command and

replaces 'Ubuntu' by 'Kubuntu' and 'Centos' by 'Fedora'.

```
$ echo "Ubuntu Centos Debian" | sed -e 's/Ubuntu/Kubuntu/; s/Centos/Fedora/'
```

Note due to the separating semicolon multiple 's/' commands are combined with one sed. The following output will appear after running the above command and 'Ubuntu' and 'Centos' are replaced by 'Kubuntu' and 'Fedora':

[Go to top](#)

44. Combine sed with other commands

The following command will combine the **sed** command with **cat** command. In this example the **cat** command is used to generate output to stdout. The first **sed** command will take input from 'os.txt' file and send the output of the command to second **sed** command after replacing the text 'Linux' by 'Fedora'. The second **sed** command will replace the text 'Windows' by 'Windows 10'. These 3 commands are combined with linux pipes.

Here is the 'OS.txt' file:

```
$ cat os.txt
Windows
Linux
Android
OS
```

And here is the command to run:

```
cat os.txt | sed 's/Linux/Fedora/' | sed 's/windows/Windows 10/i'
```

The following output will appear after running the above command:

[Go to top](#)

45. Insert empty line in a file

Create a file named 'stdlist' with the following content:

```
$ cat stdlist
#ID      #Name
[101]    -Ali
[102]    -Neha
```

sed's 'G' option is used to insert empty lines in a file after each existing line. You can think of this as converting a single spaced file to a double spaced file with a blank line between each existing line. Here is the example command:

```
sed G stdlist
```

The following output will appear after running the above commands. An empty line is inserted after each line of the file:

```
linuxhint@u22:~$ sed G stdlist
#ID      #Name
[101]    -Ali
[102]    -Neha
linuxhint@u22:~$
```

[Go to top](#)

46. Replace all alpha-numeric characters by space in each line of a file.

The following command will replace all alpha-numeric characters by space in the 'stdlist' file whose content is shown here:

```
$ cat stdlist
#ID      #Name
```

```
[101] -Ali  
[102] -Neha
```

This **sed** command works by matching all alphabetical characters in the full range of upper case and lower case as well as integers and uses **sed 's'** substitution command with '**g**' global modifier:

```
sed 's/[A-Za-z0-9]//g' stdlist
```

The following output will appear after running the above commands:

```
linuxhint@u22:~$ sed 's/[A-Za-z0-9]//g' stdlist  
#      #  
[]     -  
[]     -  
linuxhint@u22:~$
```

[Go to top](#)

47. Use '&' to print matched string

The following command will search the word starting with 'L' and replace the text by appending 'Matched String is -' with the matched word by using '&' symbol. In this example '**p**' is used to print the modified text. The 'os.txt' file is used with this content:

```
$ cat os.txt  
Windows  
Linux  
Android  
OS
```

Here is the command, note the '^' used to signify start of line:

```
sed -n 's/^L/Matched String is - &/p' os.txt
```

Note in the output only matched lines are printed when using '**p**' command:

[Go to top](#)

48. Switch pair of words in a file

Create a text file named 'course.txt' with the following content that contains the pair of words in each line:

```
$ cat course.txt
PHP      ASP
MySQL    Oracle
CodeIgniter  Laravel
```

The following command will switch the pair of words in each line of the file 'course.txt'

```
sed 's/\([^ ]*\) *\([^ ]*\)/\2 \1/' course.txt
```

The following output will appear after switching the pair of words in each line. This is a set of 2 capture groups which are reversed in the output of the 's' substitution:

[Go to top](#)

49. Capitalize the first character of each word

The following **sed** command will take input text from the **echo** command and convert the first character of each word to a capital letter. The command features 2 capture groups and then capitalizes the first capture group only:

```
echo "I like bash programming" | sed 's/\([a-z]\)\([a-zA-Z0-9]*\)/\u\1\2/g'
```

The following output will appear after running the above command. The input text, "I like bash programming" is printed as "I Like Bash Programming" after capitalizing the first word:

[Go to top](#)

50. Print line numbers of the file

Assume we want to print line numbers in this sample file shown below:

```
$ cat os.txt
Windows
Linux
Android
OS
```

'=' symbol is used in **sed** command to print the line number before each line of a file. The following command will print the content of 'os.txt' file with line number:

```
sed '=' os.txt
```

The following output will appear after running the above command. There are four lines in 'os.txt' file. The line number is printed before each line of the file:

[Go to top](#)

Conclusion

Different uses of the **sed** command are explained in this tutorial by using very simple examples. The output of all **sed** scripts mentioned here are generated temporary and the content of the original file remained unchanged. But if you want you can modify the original file by using **-i** or **-in-place** option of **sed** command. If you are a new Linux user and want to learn the basic uses of **sed** command to perform various types of string manipulation tasks, then this tutorial will help you.

Frequently Asked Questions

What is the sed command used for?

The **sed** command has a number of different uses. That being said, the main usage is for substituting words in a file, or finding and replacing. The great thing about **sed** is that you can search for a word in a file and replace it, but you never even have to open the file – **sed** just

does it all for you! As well as this, it can be used for deletion. All you need to do is type the word you want to find, replace or delete into sed, and it brings it up for you – you can then choose to replace that word or delete all traces of the word from your file. sed is a fantastic tool to be able to replace things like IP addresses and anything that is highly sensitive that you would not otherwise want to put in a file. sed is a must-know for any software engineer, system admin or dev ops engineer!

What is s and g in sed command?

In its most simple terms, the ‘**s**’ function that can be used in sed simply means ‘substitute’. After typing the ‘s’ you can replace or substitute anything you wish – just typing ‘**s**’ will only replace the first occurrence of the word on a line. Specifying ‘**g**’ modifier at the end of the sed command will do a global replacement (that is what the G stands for). With this in mind, if you specify ‘**g**’ it will replace every occurrence of the word you have chosen rather than just the first occurrence that is the default behavior with ‘**s**’.

How do I run a sed script?

You can run a sed script in a number of ways but the most common is on the command line. Here you can just specify sed and the file you want to use the command on. This allows you to use sed on that file, allowing you to find, delete and substitute as needed. You can also use it in a shell script, and this way you can pass whatever you want to the script, and it will run the find and replace command for you. This is useful for not wanting to specify highly sensitive data inside a script, so instead, you can pass it in as a variable.

What is a capture group in sed

Capture groups allow the programmer to create regular expressions to find matching text and then use those matches as assigned variables to do operations on the in the replaced output, like changing the order of the output, modifying with a capitalization command or any other sed modification. Capture groups allow dynamic text matching a patten to be captured and reused in complex ways.

ABOUT THE AUTHOR

Fahmida Yesmin



I am a trainer of web programming courses. I like to write article or tutorial on various IT topics. I have a YouTube channel where many types of tutorials based on Ubuntu, Windows, Word, Excel, WordPress, Magento, Laravel etc. are published: [Tutorials4u Help](#).

[View all posts](#)

RELATED LINUX HINT POSTS

How To Use readarray Command To Read 2D Array in Bash

How to Use awk Command in Bash

How to Timeout a Command in Bash Without Unnecessary Delay

How to Check if a Bash Array Contains a Value

Regex Matching in a Bash if Statement

How to Get Arguments with Flags in Bash

How To Check if String Is Neither Empty nor Space in Shell Script

Linux Hint LLC, editor@linuxhint.com
1309 S Mary Ave Suite 210, Sunnyvale, CA
94087

[Privacy Policy](#) and [Terms of Use](#)