

50 `sed` Command Examples

2 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 command 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.

Syntax:

```
sed [options]... [script] [file]
```

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](#)
- 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/'
```

Output:

The word, 'Bash' exists in the text. So the output is 'Perl Scripting Language'.

`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.

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'.

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

Output:

'Sunday' exists in weekday.txt file and this word is replaced by the text, 'Sunday is holiday' after executing the above `sed` command.

[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.

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.

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

Output:

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

[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
```

Output:

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'**.

[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.

lang.txt

Bash Programming Language. Python Programming Language. Perl Programming Language.
Hypertext Markup Language.
Extensible Markup Language.

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

[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.

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

Output:

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

[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.

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

Output:

The following output will appear after running the above commands.

[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'
```

Output:

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.

[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/.*\\/'
```

Output:

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

[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

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’**.

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

Output:

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

[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.

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

Output:

The following output will appear after running the above commands.

[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/'
```

Output:

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

[Go to top](#)

12. Delete matching lines

The following output will appear after running the commands. Here, the word, 'to' is searched in the file, **python.txt** and this word exists in the second line of this file. So, '**to**' with space is added after the matching text.

[Go to top](#)

24. Replace one list of strings with the new string

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

cat list1.txt

```
1001 => Jafar Ali
1023 => Nir Hossain
1067 => John Michel
```

Create a text file named **list2.txt** and add the following content.

\$ cat list2.txt

```
1001  CSE  GPA-3.63
1002  CSE  GPA-3.24
1023  CSE  GPA-3.11
1067  CSE  GPA-3.84
```

The following `sed` command will match the first column of the two text files shown above and replace the matching text with the value of the third column of the file **list1.txt**.

```
$ cat list1.txt
$ cat list2.txt
$ sed `cat list1.txt | awk '{print "-e s/"$1"/"$3"/"}'`<<<` cat list2.txt`
```

Output:

1001, 1023 and 1067 of **list1.txt** file match with the three data of **list2.txt** file and these values are replaced by corresponding names of the third column of **list1.txt**.

[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 newline.

\$ echo "Bash Perl Python Java PHP ASP" | sed 's/Python/Added Text\n/' **Output:** 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**. Here, **-z** option is used to separate the line by NULL character.

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

Output:

The following output will appear after running the above command.

[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 the comma by newline.

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

Output:

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' is used in `sed` command for the case-insensitive match that indicates ignore case. The following `sed` command will search the line that contains the word, **'linux'** and delete the line from **os.txt** file.

```
$ cat os.txt  
$ sed '/linux/I' os.txt
```

Output:

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 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'.

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

Output:

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.


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

Output:

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

'L' is used in 'sed' to convert any text to all lowercase letters. The following 'sed' command will search the word, 'Linux' in the os.txt file and replace the word by all lowercase letters.

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

Output:

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 uppercase characters in the os.txt file and replace the characters by lowercase letters by using 'L'.

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

Output:

The following output will appear after running the above commands.

[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.

items.txt

```
HDD      100
Monitor  80
Mouse    10
```

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

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

Output:

The following output will appear after running the above commands. Here, '\$' symbol is added before the number of each line.

[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 comma after each group of three digits counting from the right. Here, ':a' indicates the label and 'ta' is used to iterate grouping process.

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

Output:

The number 5098673 is given in the `echo` command and the `sed` command generated the number 5,098,673 by adding comma after each group of three digits.

[Go to top](#)

35. Replaces tab character with 4 space characters

The following `sed` command will replace each tab (\t) character by four space characters. '\$' symbol is used in the `sed` command to match the tab character and 'g' is used to replace all tab characters.

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

Output:

The following output will appear after running the above command.

[Go to top](#)

36. Replaces 4 consecutive space characters with tab character

The following command will replace 4 consecutive characters with tab (\t) character.

```
$ echo -e "1    2" | sed 's/    /\t/g'
```

Output:

The following output will appear after running the above command.

[Go to top](#)

37. Truncate all lines to first 80 characters

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

in.txt

PHP is a server-side scripting language.

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

The following `sed` command will truncate each line of **in.txt** file into 80 characters.

```
$ cat in.txt
```

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

Output:

The following output will appear after running the above commands. The second line of in.txt file contains more than 80 characters and this line is truncated in the output.

[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/'
```

Output:

The following output will appear after running the above command.

[Go to top](#)

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

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

```
$ cat input.txt
```

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

Output:

The following output will appear after running the above commands. The searching text, '**PHP**' appears for 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.

[Go to top](#)

40. Running multi-line `sed` scripts from a file

Multiple `sed` scripts can be stored in a file and all the scripts can be executed together by running `sed` command. Create a file named **'sedcmd'** and add the following content. Here, two `sed` scripts are added in the file. One script will replace the text, **'PHP'** by **'ASP'** another script will replace the text, **'independent'** by the text, **'dependent'**.

sedcmd

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

The following `sed` command will replace all 'PHP' and 'independent' text by 'ASP' and 'dependent'. Here, '-f' option is used in the `sed` command to execute `sed` script from the file.

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

Output:

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 the multi-line text, **'Linux\nAndroid'** and if the pattern matches then the matching lines will be replaced by the multi-line text, **'Ubuntu\nAndroid Lollipop'**. Here, P and D are used for multiline processing.

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

Output:

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/'
```

Output:

The following output will appear after running the above command.

[Go to top](#)

43. Execute multiple `sed` commands from the command-line

‘-e’ option is used in `sed` command to run multiple `sed` scripts from the command line. 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/'
```

Output:

The following output will appear after running the above command. Here, ‘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. 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’.

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

Output:

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.

stdlist

```
#ID    #Name
[101]  -Ali
[102]  -Neha
```

‘G’ option is used to insert empty line in a file. The following `sed` command will insert empty lines after each line of **stdlist** file.

```
$ cat stdlist
$ sed G stdlist
```

Output:

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

[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.

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

Output:

The following output will appear after running the above commands.

[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. Here, 'p' is used to print the modified text.

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

Output:

The following output will appear after running the above 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.

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
```

Output:

The following output will appear after switching the pair of words in each line.

[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.

```
$ echo "I like bash programming" | sed 's/^([a-z])\([a-zA-Z0-9]*\)^\u0012/g'
```

Output:

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

``='` symbol is used ``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
```

Output:

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

[Go to top](#)

Conclusion:

Different uses of ``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. After reading this tutorial, hope, any user will get the clear concept about the functions of ``sed`` command.

``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``.

```
$ cat os.txt
```

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

Output:

The following output will appear after running the above commands.

[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
```

Output:

The following output will appear after running the above command.

[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**.

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

Output:

os.txt contains empty lines after each line those are deleted by the above `sed` command.

[Go to top](#)

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

Create a text file named, **input.txt** with the following content and delete those lines of the file that contains the searching pattern two times.

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' text contains two times in the second line of the file, **input.txt**. 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**' and send the output into the second `sed` command as input. The second `sed` command will delete those lines that contain the text, '**dl**'.


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

Output:

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

[Go to top](#)

16. Delete all lines that have only white-space

Select any file that contains empty lines in the content to test this example. **input.txt** file that is created in the previous example, contains two empty lines that can be deleted by using the following ``sed`` command. Here, '^\$' is used to find out the empty lines in the file, **input.txt**.

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

Output:

The following output will appear after running the above commands.

[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$tabWorld"
$ echo "Hello$tabWorld" | sed 's/[^[:print:]]//g'
```

Output:

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.

[Go to top](#)

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

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

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

Output:

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

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

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

Output:

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.

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

Output:

The following output will appear after running the above commands.

[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 does not contain the text, '**Linux**' and append the text, '**Operating System**' at the end of each line. Here, '\$' symbol is used to identify the line where the new text will be appended.

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

Output:

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 and delete lines that do not contain the matching pattern. **web.txt** HTML 5JavaScriptCSSPHPMySQLjQuery The following `sed` command will search and delete those lines that do not contain the text, 'CSS'. \$ cat web.txt\$ sed '/CSS/!d' web.txt **Output:** The following output will appear after running the above commands. There is one line exists in the file that contains the text, 'CSE'. So, the output contains just one line.

[Go to top](#)

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

The following `sed` command will search the word, 'to' in the file, **python.txt** and if the word exists then the same word will be inserted after the search word by adding space. Here, '&' symbol is used to append the duplicate text.

```
$ cat python.txt
```

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

Output:

https://linuxhint.com/50_sed_command_examples/