About sed Command

Introduction

If you are a developer, or system administrator, or database administrator, or IT manager, or just someone who spends a significant amount of time on UNIX / Linux, you should become proficient in Sed and Awk.

Sed and Awk are two great utilities that can solve a lot of complex tasks quickly with only a few lines of code--in most cases, with just a single line of code.

A note on the examples: Most examples are identified in the following way.

Example Description:-

Lines of code for you to type, with the result you will see on screen.

- Any additional clarification or discussion will appear below the code section in plain text.
- Also please note that commands should be typed on one line. If you copy and paste, be sure that command is pasted as a single line.

Chapter 1: Sed Syntax and Basic Commands

For all sed examples, we'll be using the following employee.txt file. Please create this text file to try out the commands given in this book.

```
$ vi employee.txt
101,John Doe,CEO
102,Jason Smith,IT Manager
103,Raj Reddy,Sysadmin
104,Anand Ram,Developer
105,Jane Miller,Sales Manager
```

The above employee database contains the following fields for every record:

- Employee Id
- Employee Name
- Title

- **Sed stands for Stream Editor.** It is very powerful tool to manipulate, filter, and transform text. Sed can take input from a file, or from a pipe.

- You might even have several sed one line commands in your bash startup file that you use for various scenarios without exactly understanding the sed scripts.

- For beginners, sed script might look cryptic. Once you understand the sed commands in detail, you'll be able to solve a lot of complex text manipulation problems by writing a quick sed script.

- In this book, I've explained all sed commands and provided easy-to understand

1.Sed Command Syntax

The purpose of this section is to get you familiarized with sed syntax and command structure. This is not meant to explain the individual sed commands, which are covered in detail later.

Basic sed syntax:

sed [options] {sed-commands} {input-file}

- Sed reads one line at a time from the {input-file} and executes the{sed-commands} on that particular line.

- It reads the 1st line from the {input-file} and executes the {sedcommands} on the 1st line. Then it reads the 2nd line from the {input-file} and executes the {sed-commands} on the 2nd line. Sed repeats this process until it reaches the end of the {input-file}.

- There are also a few optional command line options that can be passed to sed as indicated by [options].

The following example demonstrates the basic sed syntax:-

sed example:- prints all the lines from the /etc/passwd file

```
sed -n 'p' /etc/passwd
```

The main focus here is on the {sed-commands}, which can be either a single sed command or multiple sed commands. You can also combine multiple sed-commands in a file and call the sed script file using the -f option as shown below

Basic sed syntax for use with sed-command file:-

```
sed [options] -f {sed-commands-in-a-file} {input-file}
```

Example :-prints lines beginning with root and nobody from the /etc/passwd file.

```
$ vi test-script.sed
/^root/ p
/^nobody/ p
$ sed -n -f test-script.sed /etc/passwd
```

While executing multiple sed commands, you can also directly specify them in the command line using -e as shown below:-

Basic sed syntax using -e:

sed [options] -e {sed-command-1} -e {sed-command-2}{input-file}

The following example demonstrates the basic syntax:-

Example:-This prints lines beginning with root and nobody from /etc/passwd file:

```
sed -n -e '/^root/ p' -e '/^nobody/ p' /etc/passwd
```

If you are executing a lot of commands in a single line using several -e arguments, you can split them into multiple lines using a back slash as shown below:-

```
sed -n \
-e '/^root/ p' \
-e '/^nobody/ p' \
/etc/passwd
```

You can also execute multiple sed commands in the command line by grouping them together using { }

Basic sed syntax using { }:-

```
sed [options] '{
sed-command-1
sed-command-2
}' input-file
```

The following example demonstrates this version of the basic syntax:-

Example:-This also prints lines beginning with root and nobody from/etc/passwd file.

```
 sed -n '{
/^root/ p
/^nobody/ p
}' /etc/passwd
```

Note: Sed never modifies the original file. It always prints the output to stdout. If you want to save the changes, you should redirect the output to a file by explicitly specifying **> filename.txt.**

2. Sed Scripting Flow

Sed scripting follows the easily remembered sequence Read,Execute, Print, Repeat. Use the simple REPR acronym to remember sed execution flow.

We look at the steps in this sequence. Sed will:-

• **Read** a line into the pattern space (an internal temporary sed buffer, where it places the line it reads from the input file).

• **Execute** the sed command on the line in the sed pattern space. If there are more than one sed commands available,either via a sed script, -e options, or { }, it executes all the sed commands one by one in sequence on the line that is currently in the pattern space.

• **Print** the line from the pattern space. After printing this line,the sed pattern space will be empty.

• **Repeat** this again until the end of the input file is reached.

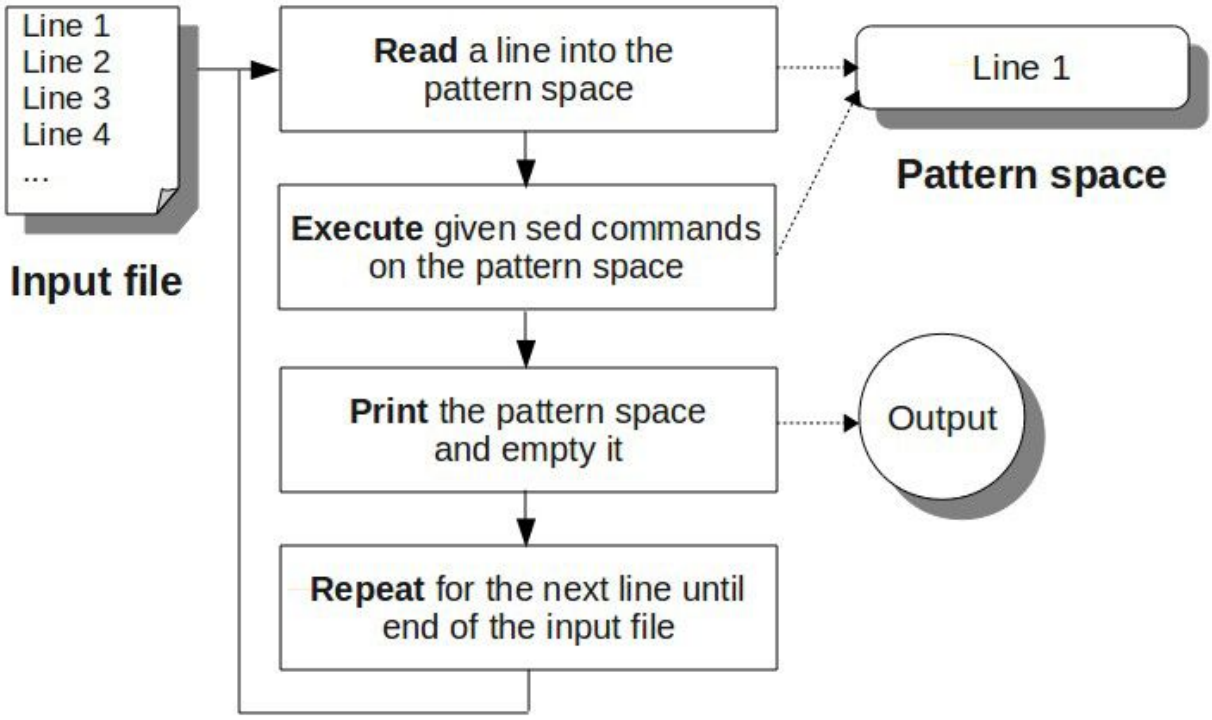

Fig: Illustration of SED execution flow

3. Print Pattern Space (p command)

Using the sed p command, you can print the current pattern space.You may wonder why you would need the p command, since bydefault sed prints the pattern buffer after executing its commands.

There are reasons, as you will see; the command allows you to specifically control what is printed to stdout. Usually when p is used you will use the -n option to suppress the the default printing that happens as part of the standard sed flow. Otherwise, when execute p (print) as one of the commands, the line will be printed twice.

Example:- prints every line of employee.txt twice

```
$ sed 'p' employee.txt
101,John Doe,CEO
101,John Doe,CEO
102,Jason Smith,IT Manager
102,Jason Smith,IT Manager
103,Raj Reddy,Sysadmin
103,Raj Reddy,Sysadmin
104,Anand Ram,Developer
104,Anand Ram,Developer
105,Jane Miller,Sales Manager
105,Jane Miller,Sales Manager
```

Example:-Print each line once (functionally the same as 'cat employee.txt')

```
$ sed -n 'p' employee.txt
101,John Doe,CEO
102,Jason Smith,IT Manager
103,Raj Reddy,Sysadmin
104,Anand Ram,Developer
105,Jane Miller,Sales Manager
```

Specifying an Address Range:-

If you don't specify an address range before the sed command, by default it matches all the lines.

The following are some examples of specifying an address range before the sed command.Print only the 2nd line

```
$ sed -n '2 p' employee.txt
102,Jason Smith,IT Manager
```

Example:-Print from line 1 through line 4

```
$ sed -n '1,4 p' employee.txt
101,John Doe,CEO
102,Jason Smith,IT Manager
103,Raj Reddy,Sysadmin
104,Anand Ram,Developer
```

Example:-Print from line 2 through the last line ($ represents the last line)

```
$ sed -n '2,$ p' employee.txt
102,Jason Smith,IT Manager
103,Raj Reddy,Sysadmin
104,Anand Ram,Developer
105,Jane Miller,Sales Manager
```

Modify Address Range

You can modify address range using comma, plus, and tilde.In the examples above, we saw the use of the comma (,) as part of an address range specification. Its meaning is clear: n,m indicates n through m.

The plus (+) may be used in conjunction with the comma, to specify a number of lines instead of an absolute line number. For example, n, +m means the m lines starting with n.

The tilde (~) may also be used in an address range. Its special meaning is to skip lines between commands. For example, address range n~m indicates that sed should start at the nth line and pick up every mth line from there.

- 1~2 matches 1,3,5,7, etc.
- 2~2 matches 2,4,6,8, etc.
- 1~3 matches 1,4,7,10, etc.
- 2~3 matches 2,5,8,11, etc.

Example:-Print only odd numbered lines

```
$ sed -n '1~2 p' employee.txt
101,John Doe,CEO
103,Raj Reddy,Sysadmin
105,Jane Miller,Sales Manager
```

Pattern Matching

Just as you can specify a numbered address (or address range), you can also specify a pattern (or pattern range) to match, as shown in the next few examples.

Example:-Print lines matching the pattern “Jane”

```
$ sed -n '/Jane/ p' employee.txt
105,Jane Miller,Sales Manager
```

Example:-Print lines starting from the 1st match of "Jason" until the 4th line

```
$ sed -n '/Jason/,4 p' employee.txt
102,Jason Smith,IT Manager
103,Raj Reddy,Sysadmin
104,Anand Ram,Developer
```

Note: If there were no matches for "Jason" in the 1st 4 lines, this command would print the lines that match "Jason" after the 4th line.

Example:-Print lines starting from the 1st match of "Raj" until the last line:

```
$ sed -n '/Raj/,$ p' employee.txt
103,Raj Reddy,Sysadmin
104,Anand Ram,Developer
105,Jane Miller,Sales Manager
```

Example:-Print lines starting from the line matching "Raj" until the line matching "Jane"

```
$ sed -n '/Raj/,/Jane/ p' employee.txt
103,Raj Reddy,Sysadmin
104,Anand Ram,Developer
105,Jane Miller,Sales Manager
```

Example:-Print the line matching "Jason" and 2 lines immediately after that

```
$ sed -n '/Jason/,+2 p' employee.txt
102,Jason Smith,IT Manager
103,Raj Reddy,Sysadmin
104,Anand Ram,Developer
```

4. Delete Lines (d command)

Using the sed d command, you can delete lines. Please note that the lines are only deleted from the output stream. Just like any other sed command, the d command doesn't modify the content of the original input file.

By default if you don't specify any address range before the sed command, it matches all the lines. So, the following example will not print anything, as it matches all the lines in the employee.txt and deletes them.

```
sed 'd' employee.txt
```

It's more useful to specify an address range to be deleted. The following are some examples:-

Example:-Delete only the 2nd line

```
$ sed '2 d' employee.txt
101,John Doe,CEO
103,Raj Reddy,Sysadmin
104,Anand Ram,Developer
105,Jane Miller,Sales Manager
```

Example:-Delete from line 1 through 4

```
$ sed '1,4 d' employee.txt
105,Jane Miller,Sales Manager
```

Example:-Delete from line 2 through the last line

```
$ sed '2,$ d' employee.txt
101,John Doe,CEO
```

Example:-Delete only odd number of lines

```
$ sed '1~2 d' employee.txt
102,Jason Smith,IT Manager
104,Anand Ram,Developer
```

Example:-Delete lines matching the pattern "Manager"

```
$ sed '/Manager/ d' employee.txt
101,John Doe,CEO
103,Raj Reddy,Sysadmin
104,Anand Ram,Developer
```

Example:-Delete lines starting from the 1st match of "Jason" until the 4th line

```
$ sed '/Jason/,4 d' employee.txt
101,John Doe,CEO
105,Jane Miller,Sales Manager
```

If there are no matches for "Jason" in the 1st 4 lines, this command deletes only the lines that match "Jason" after the 4th line.

Example:-Delete lines starting from the 1st match of "Raj" until the last line

```
$ sed '/Raj/,$ d' employee.txt
101,John Doe,CEO
102,Jason Smith,IT Manager
```

Example:-Delete lines starting from the line matching "Raj" until the line matching "Jane"

```
$ sed '/Raj/,/Jane/ d' employee.txt
101,John Doe,CEO
102,Jason Smith,IT Manager
```

Example:-Delete lines starting from the line matching "Jason" and 2 lines immediately after that

```
$ sed '/Jason/,+2 d' employee.txt
101,John Doe,CEO
105,Jane Miller,Sales Manager
```

Useful Delete Examples

The following examples are very helpful in actual day-to-day operations.

Example:-Delete all the empty lines from a file

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

Example:-Delete all comment lines (assuming the comment starts with#)

```
sed '/^#/ d' employee.txt
```

5. Write Pattern Space to File (w command)

Using the sed w command, you can write the current pattern space to a file. By default as per the sed standard flow, the pattern space will be printed to stdout, so if you want output to file but not screen you should also use the sed option -n.

The following are some examples:-

Example:-Write the content of employee.txt file to file output.txt (and display on screen)

```
$ sed 'w output.txt' employee.txt
101,John Doe,CEO
102,Jason Smith,IT Manager
103,Raj Reddy,Sysadmin
104,Anand Ram,Developer
105,Jane Miller,Sales Manager

$ cat output.txt
101,John Doe,CEO
102,Jason Smith,IT Manager
103,Raj Reddy,Sysadmin
104,Anand Ram,Developer
105,Jane Miller,Sales Manager
```

Example:-Write the content of employee.txt file to output.txt file but not to screen

```
$ sed -n 'w output.txt' employee.txt
$ cat output.txt
101,John Doe,CEO
102,Jason Smith,IT Manager
103,Raj Reddy,Sysadmin
104,Anand Ram,Developer
105,Jane Miller,Sales Manager
```

Example:-Write only the 2nd line

```
$ sed -n '2 w output.txt' employee.txt
$ cat output.txt
102,Jason Smith,IT Manager
```

Example:-Write lines 1 through 4

```
$ sed -n '1,4 w output.txt' employee.txt
$ cat output.txt
101,John Doe,CEO
102,Jason Smith,IT Manager
103,Raj Reddy,Sysadmin
104,Anand Ram,Developer
```

Example:-Write from line 2 through the last line

```
$ sed -n '2,$ w output.txt' employee.txt
$ cat output.txt
102,Jason Smith,IT Manager
103,Raj Reddy,Sysadmin
104,Anand Ram,Developer
105,Jane Miller,Sales Manager
```

Example:-Write only odd numbered lines

```
$ sed -n '1~2 w output.txt' employee.txt
$ cat output.txt
101,John Doe,CEO
103,Raj Reddy,Sysadmin
105,Jane Miller,Sales Manager
```

Example:-Write lines matching the pattern "Jane"

```
$ sed -n '/Jane/ w output.txt' employee.txt
$ cat output.txt
105,Jane Miller,Sales Manager
```

Example:-Write lines starting from the 1st match of "Jason" until the 4thline

```
$ sed -n '/Jason/,4 w output.txt' employee.txt
$ cat output.txt
102,Jason Smith,IT Manager
103,Raj Reddy,Sysadmin
104,Anand Ram,Developer
```

If there are no matches for "Jason" in the 1st 4 lines, this commandwrites only the lines that match "Jason" after the 4th line.

Example:-Write lines starting from the 1st match of "Raj" until the last line

```
$ sed -n '/Raj/,$ w output.txt' employee.txt
$ cat output.txt
103,Raj Reddy,Sysadmin
104,Anand Ram,Developer
105,Jane Miller,Sales Manager
```

Example:-Write lines starting from the line matching "Raj" until the line matching "Jane"

```
$ sed -n '/Raj/,/Jane/ w output.txt' employee.txt
$ cat output.txt
103,Raj Reddy,Sysadmin
104,Anand Ram,Developer
105,Jane Miller,Sales Manager
```

Example:-Write the line matching "Jason" and the next 2 lines immediately after that

```
$ sed -n '/Jason/,+2 w output.txt' employee.txt
$ cat output.txt
102,Jason Smith,IT Manager
103,Raj Reddy,Sysadmin
104,Anand Ram,Developer
```

Note: You might not use the w command frequently. Most people use UNIX output redirection, instead, to store the output of sed to a file.

For example: sed 'p' employee.txt > output.txt

Chapter 2. Sed Substitute Command

The most powerful command in the stream editor is **s**ubstitute. It has such power and so many options that we give it a whole chapter.

6. Sed Substitute Command Syntax

sed '[address-range|pattern-range] s/originalstring/replacement-string/[substitute-flags]' inputfile

In the above sed substitute command syntax:

• address-range or pattern-range is optional. If you don't specify one, sed will execute the substitute command on all lines.

• s – tells Sed to execute the substitute command

• original-string – this is the string to be searched for in the input file. The original-string can also be a regular expression.

• replacement-string – Sed will replace original-string with this string.

• substitute-flags are optional. More on this in the next section.

Remember that the original file is not changed; the substitution takes place in the pattern space buffer which is then printed to stdout.

The following are couple of simple sed substitute examples (changes shown in bold):-

Example:-Replace all occurrences of Manager with Director

```
$ sed 's/Manager/Director/' employee.txt
101,John Doe,CEO
102,Jason Smith,IT Director
103,Raj Reddy,Sysadmin
104,Anand Ram,Developer
105,Jane Miller,Sales Director
```

Example:-Replace Manager with Director only on lines that contain the keyword 'Sales'

```
$ sed '/Sales/s/Manager/Director/' employee.txt
101,John Doe,CEO
102,Jason Smith,IT Manager
103,Raj Reddy,Sysadmin
104,Anand Ram,Developer
105,Jane Miller,Sales Director
```

Note: That the use of the address range caused just one change rather than the two shown in the previous example.

7. Global Flag (g flag)

Sed substitute flag g stands for global. By default sed substitute command will replace only the 1st occurrence of the {original-string}on each line.

If you want to change all the occurrences of the {original-string} in the line to the {replacement-string}, you should use the **global flag g.**

Example:-Replace the 1st occurrence of lower case a with upper case A

```
$ sed 's/a/A/' employee.txt
101,John Doe,CEO
102,JAson Smith,IT Manager
103,RAj Reddy,Sysadmin
104,AnAnd Ram,Developer
105,JAne Miller,Sales Manager
```

Example:-Replace all occurrences of lower case a with upper case A

```
$ sed 's/a/A/g' employee.txt
101,John Doe,CEO
102,JAson Smith,IT MAnAger
103,RAj Reddy,SysAdmin
104,AnAnd RAm,Developer
105,JAne Miller,SAles MAnAger
```

Note: these examples were applied to the entire file because no address range was specified.

8. Number Flag (1,2,3.. flag)

Use the number flag to specify a specific occurrence of the originalstring.Only the n-th instance of original-string will trigger the substitution. Counting starts over on each line, and n can be anything from 1 to 512.

For example, /11 will replace only the 11th occurrence of the originalstring in a line.The following are few examples.

Example:-Replace the 2nd occurrence of lower case a to upper case A

```
$ sed 's/a/A/2' employee.txt
101,John Doe,CEO
102,Jason Smith,IT MAnager
103,Raj Reddy,SysAdmin
104,Anand RAm,Developer
105,Jane Miller,SAles Manager
```

Example:-For this example, create the following file with three lines

```
$ vi substitute-locate.txt
locate command is used to locate files
locate command uses database to locate files
locate command can also use regex for searching
```

Example:-In the file you just created, change only the 2nd occurrence of locate to find

```
$ sed 's/locate/find/2' substitute-locate.txt
locate command is used to find files
locate command uses database to find files
locate command can also use regex for searching
```

Note: On line 3 in the above example, there is only one "locate" in the original substitute-locate.txt file. So, nothing is changed on line 3.

9. Print Flag (p flag)

The sed substitute flag p stands for print. When the substitution is successful, it prints the changed line. Like most print commands in sed, it is most useful when combined with the -n option to suppress default printing of all lines.

Example:-Print only the line that was changed by the substitute command

```
$ sed -n 's/John/Johnny/p' employee.txt
101,Johnny Doe,CEO
```

In our number flag example, we used /2 to change the 2nd occurrence of "locate" to "find". On line 3 of locate.txt there is no 2nd occurrence and substitution never happened on that line. Adding the p flag to the command we used before will print the two lines that did change.

Example:-Change the 2nd instance of “locate” to “find” and print the result

```
$ sed -n 's/locate/find/2p' substitute-locate.txt
locate command is used to find files
locate command uses database to find files
```

10. Write Flag (w flag)

The sed substitute flag w stands for write. When the substitution is successful, it writes the changed line to a file. Most people use the p flag instead, and redirect the output to a file. We include this command for completeness.

Example:-Write only the line that was changed by the substitute command to output.txt

```
$ sed -n 's/John/Johnny/w output.txt' employee.txt
$ cat output.txt
101,Johnny Doe,CEO
```

Just as we showed for the p command, adding w to our example with the substitute-locate.txt file will send the two lines that were changed to the output file.

Example:-Change the 2nd instance of "locate" to "find",write the result to a file, print all lines

```
$ sed 's/locate/find/2w output.txt' substitutelocate.txt
locate command is used to find files
locate command uses database to find files
locate command can also use regex for searching
$ cat output.txt
locate command is used to find files
    locate command uses database to find files
```

11. Ignore Case Flag (i flag)

The sed substitute flag i stands for ignore case. You can use the i flag to match the original-string in a case-insensitive manner. This is available only in GNU Sed.

In the following example Sed will not replace "John" with "Johnny", as the original-string was given in lower case "john".

Example:-Replace “john” with Johnny

```
$ sed 's/john/Johnny/' employee.txt
101,John Doe,CEO
102,Jason Smith,IT Manager
103,Raj Reddy,Sysadmin
104,Anand Ram,Developer
105,Jane Miller,Sales Manager
```

Example:Replace “john” or “John” with Johnny

```
$ sed 's/john/Johnny/i' employee.txt
101,Johnny Doe,CEO
102,Jason Smith,IT Manager
103,Raj Reddy,Sysadmin
104,Anand Ram,Developer
105,Jane Miller,Sales Manager
```

12. Execute Flag (e flag)

The sed substitute flag e stands for execute. Using the sed e flag, you can execute whatever is available in the pattern space as a shell command, and the output will be returned to the pattern space. This is available only in the GNU sed.

The following are few examples:-

Example:-For these examples create the following files.txt that contains a list of filenames with their full path

```
$ cat files.txt
/etc/passwd
/etc/group
```

Example:-Add the text "ls -l " in front of every line in the files.txt and print the output

```
$ sed 's/^/ls -l /' files.txt
ls -l /etc/passwd
ls -l /etc/group
```

Example:-Add the text "ls -l " in front of every line in the files.txt and execute the output

```
$ sed 's/^/ls -l /e' files.txt
-rw-r--r-- 1 root root 1547 Oct 27 08:11 /etc/passwd
-rw-r--r-- 1 root root 651 Oct 27 08:11 /etc/group
```

13. Combine Sed Substitution Flags

You can combine one or more substitute flags as required. The following example will replace all occurrences of "Manager" or "manager" to "Director". This will also print only the line that was changed by the substitute command to the screen, and write the same information to the output.txt file.

Example:-Combine g,i,p and w flags

```
$ sed -n 's/Manager/Director/gipw output.txt'
 employee.txt
 102,Jason Smith,IT Director
 105,Jane Miller,Sales Director
$ cat output.txt
 102,Jason Smith,IT Director
 105,Jane Miller,Sales Director
```

14. Sed Substitution Delimiter

In all the above sed examples, we used the default sed delimiter /. i.e. s/original-string/replacement-string/ When there is a slash / in the original-string or the replacement-string, we need to escape it using \.For this example create a path.txt file which contains a directory path as shown below.

```
$ vi path.txt
reading /usr/local/bin directory
```

Now, let us change /usr/local/bin to /usr/bin using the sed substitute command. In this sed substitution example, the delimiter path delimiter '/' was escaped using back slash '\' in the original-string and the replacement-string.

```
$ sed 's/\/usr\/local\/bin/\/usr\/bin/' path.txt
reading /usr/bin directory
```

Ugly isn't it? When you are trying to replace a long path name, it might be very confusing to use all those escape characters '\'.Fortunately, you can use any character as substitution delimiter.

For example, | or ^ or @ or !.

All of the following are valid and easy to read. I have not shown the output of the commands since they all produce exactly the same result. I prefer to use @ (or !) symbol when replacing a directory path but it is your personal choice.

```
sed 's|/usr/local/bin|/usr/bin|' path.txt
sed 's^/usr/local/bin^/usr/bin^' path.txt
sed 's@/usr/local/bin@/usr/bin@' path.txt
sed 's!/usr/local/bin!/usr/bin!' path.txt
```

15. Multiple Substitute Commands Affecting the Same Line

As we discussed earlier, the sed execution flow is Read, Execute, Print, Repeat. The Execute portion, as we mentioned, may consist of multiple sed commands, which sed will execute one-by-one.

For example, if you have two sed commands, sed will execute command-1 on the pattern space, then execute command-2 on thepattern space. If command-1 changed something in the patternspace, command-2 will be executed on the newly changed patternspace (and not the original line that was Read).

The following example demonstrates the execution of two sed substitute commands on the pattern space

Example:-Change Developer to IT Manager, then change Manager to Director

```
$ sed '{
s/Developer/IT Manager/
s/Manager/Director/
}' employee.txt
101,John Doe,CEO
102,Jason Smith,IT Director
103,Raj Reddy,Sysadmin
104,Anand Ram,IT Director
105,Jane Miller,Sales Director
```

Let us analyze the sed execution flow for line 4 in the example.

1. Read: At this stage, Sed reads the line and puts it in the pattern space. So, the following is the content of the pattern space.

104,Anand Ram,Developer

2. Execute: Sed executes the 1st sed command on the pattern space, which is s/Developer/IT Manager/. So, after this command, the following is the content of the pattern space.

104,Anand Ram,IT Manager

Now, sed executes the 2nd sed command on the pattern space, which is

s/Manager/Director/.

After this command, the following is the content of the pattern space.

104,Anand Ram,IT Director

Remember: Sed executes the 2nd command on the content of the current pattern space after execution of the first command.

3. Print: It prints the content of the current pattern space, which is the following.

104,Anand Ram,IT Director

4. Repeat: It moves on to the next line and repeats from step#1.

16. Power of & - Get Matched Pattern

When & is used in the replacement-string, it replaces it with whatever text matched the original-string or the regular-expression. This is very powerful and useful.

The following are few examples:-

Example:-Enclose the employee id (the 1st three numbers) between [and], i.e. 101 becomes [101], 102 becomes [102], etc.

```
$ sed 's/^[0-9][0-9][0-9]/[&]/g' employee.txt
[101],John Doe,CEO
[102],Jason Smith,IT Manager
[103],Raj Reddy,Sysadmin
[104],Anand Ram,Developer
[105],Jane Miller,Sales Manager
```

Example:-Enclose the whole input line between < and >

```
$ sed 's/^.*/<&>/' employee.txt
<101,John Doe,CEO>
<102,Jason Smith,IT Manager>
<103,Raj Reddy,Sysadmin>
<104,Anand Ram,Developer>
<105,Jane Miller,Sales Manager>
```

16. Substitution Grouping (Single Group)

Grouping can be used in sed just like in a normal regular expression.A group is opened with "\(" and closed with "\)". Grouping can be used in combination with back-referencing.

A back-reference is the re-use of a part of a regular expression selected by grouping. Back-references in sed can be used in both a regular expression and in the replacement part of the substitute command.

Single grouping:

```
$ sed 's/\([^,]*\).*/\1/g' employee.txt
101
102
103
104
105
```

In the above example:-

• Regular expression \([^,]*\) matches the string up to the 1st comma.

• \1 in the replacement-string replaces the first matched group.

• g is the global substitute flag.

Example:-This sed example displays only the first field from the /etc/passwd file, i.e. it displays only the username

```
sed 's/\([^:]*\).*/\1/' /etc/passwd
```

Example:-The following example encloses the 1st letter in every word inside (), if the 1st character is upper case.

```
$ echo "The Geek Stuff" | sed 's/\(\b[A-Z]\)/\(\1\)/g'
(T)he (G)eek (S)tuff
```

Example:-For the next example create a numbers.txt sample file as shown below.

```
$ vi numbers.txt
   1
   12
   123
       1234
   12345
   123456
```

Commify numbers, i.e. insert commas to make them more readable:

```
$ sed 's/\(^\|[^0-9.]\)\([0-9]\+\)\([0-9]\
{3\}\)/\1\2,\3/g' numbers.txt
   1
   12
   123
   1,234
   12,345
   123,456
```

The above command should be executed in a single line as shown below.

```
sed 's/\(^\|[^0-9.]\)\([0-9]\+\)\([0-9]\{3\}\)/\1\2,\3/g' numbers.txt
```

17. Substitution Grouping (Multiple Group)

In multi grouping, you can have multiple groups enclosed in multiple "\(" and "\)". When you have multiple groups in the substitute regular expression, you can use \n to specify the nth group in the sed replacement string. An example is shown below.

Example:-Get only the 1st column (employee id) and the 3rd column(title):

```
$ sed 's/\([^,]*\),\([^,]*\),\([^,]*\).*/\1,\3/g' employee.txt
101,CEO
102,IT Manager
103,Sysadmin
104,Developer
105,Sales Manager
```

The above command should be executed in a single line as shown below.

Sed and Awk 101 Hacks

```
sed 's/\([^,]*\),\([^,]*\),\([^,]*\).*/\1,\3/g' employee.txt
```

In the above example, you can see three groups mentioned in the original-string (reg-ex). These three groups are separated bycommas.

- ([^,]*\) is group 1 that matches the employee id
- , is the field separator after group 1
- ([^,]*\) is group 2 that matches the employee name

• , is the field separator after group 2

• ([^,]*\) is group 3 that matches the employee title

• , is the field separator after group 3 The replacement-string section of the above example indicates how these groups should be used.

• \1 is to print group 1 (employee id)

• , is to print a comma after printing group 1

• \3 is to print group 1 (title)

Note: Sed can hold a maximum of 9 groups referenced using \1 through \9

Example:-Swap field 1 (employee id) with field 2 (employee name); print the employee.txt file

```
$ sed 's/\([^,]*\),\([^,]*\),\(.*\).*/\2,\1,\3/g employee.txt
John Doe,101,CEO
Jason Smith,102,IT Manager
Raj Reddy,103,Sysadmin
Anand Ram,104,Developer
Jane Miller,105,Sales Manager
```

The above command should be executed in a single line as shown below.

```
sed 's/\([^,]*\),\([^,]*\),\(.*\).*/\2,\1,\3/g' employee.txt
```

18. Gnu Sed Only Replacement String Flags

These flags are available only in GNU version of sed. They can be used in the replacement-string part of the sed substitute command.

\l replacement string flag

When you specify \l in the replacement-string part, it treats the character that immediately follows \l as lower case. You already know the following simple example will change John to JOHNNY.

```
sed 's/John/JOHNNY/' employee.txt
```

The following example contains \l before H in the replacement-string (i.e. JO\lHNNY). This will change only the character h in JOHNNY to lower case.

Example:-Change John to JOhNNY:

```
$ sed -n 's/John/JO\lHNNY/p' employee.txt
101,JOhNNY Doe,CEO
```

\L replacement string flag

When you specify \L in the replacement-string part, it treats the rest of the characters as lower case.

The following example contains \L before H in the replacement-string (i.e. JO\lHNNY). This will change the rest of the characters from h to lower case.

Change Johnny to JOhnny:

```
$ sed -n 's/John/JO\LHNNY/p' employee.txt
101,JOhnny Doe,CEO
```

\u replacement string flag

Just like \l, but for upper case. When you specify \l in the replacement-string part, it treats the character that immediately follows \u as upper case.

The following example contains \u before h in the replacement-string (i.e. jo\uhnny). This will change only the character h in johnny to upper case.

Change John to joHnny:

```
$ sed -n 's/John/jo\uhnny/p' employee.txt
101,joHnny Doe,CEO
```

\U replacement string flag

When you specify \U in the replacement-string part, it treats the rest of the characters as upper case. The following example contains \U before h in the replacement-string (i.e. jo\Uhnny). This will change the rest of the characters from h in johnny to upper case.

Change John to joHNNY:

```
$ sed -n 's/John/jo\Uhnny/p' employee.txt
101,joHNNY Doe,CEO
```

\E replacement string flag

This should be used in conjunction with either \L or \U. This stops the conversion initiated by either \L or \U. The following example prints the whole replacement string "Johnny Boy" in upper case, as we have

\U at the beginning of the replacement-string.

Change John to JOHNNY BOY:

```
$ sed -n 's/John/\UJohnny Boy/p' employee.txt
101,JOHNNY BOY Doe,CEO
```

Change John to JOHNNY Boy:

```
$ sed -n 's/John/\UJohnny\E Boy/p' employee.txt
101,JOHNNY Boy Doe,CEO
```

The above example prints only "Johnny" in the upper case, as we have \E immediately after "Johnny" in the replacement-string.

Replacement String Flag Usages

The above static examples are shown only to understand how these switches works. However, the flags don't have much value when used with static values, as you can just type the static values in the exact case needed.

The flags are quite useful when combined with grouping. In the previous example we learned how to swap field 1 with field 3 using grouping. You can convert a whole grouping to either upper or lower case using these switches.

Employee name in all upper case, and title in all lower case:

```
$ sed 's/\([^,]*\),\([^,]*\),\(.*\).*/\U\2\E,\1,\L\3/g' employee.txt
JOHN DOE,101,ceo
JASON SMITH,102,it manager
RAJ REDDY,103,sysadmin
ANAND RAM,104,developer
JANE MILLER,105,sales manager
```

The above command should be executed in a single line as shown below.

```
sed 's/\([^,]*\),\([^,]*\),\(.*\).*/\U\2\E,\1,\L\3/g' employee.txt
```

In the above example, in the replacement-string, we have the following:

• \U\2\E - This indicates that this field, which is the 2nd group (employee name), should be converted to upper case. \U start the upper case conversion, and \E stops it.

• \L\3 - This indicates that this field, which is 3rd group (title), should be converted to lower case. \L starts the lower caseconversion for rest of the characters.

Chapter 3. Regular Expressions

20. Regular Expression Fundamentals

Regular expressions (or regex) are used in many *nix commands,including sed.

Beginning of line (^): The Caret Symbol ^ matches at the start of a line.

Example:-Display lines which start with 103:

```
$ sed -n '/^103/ p' employee.txt
103,Raj Reddy,Sysadmin
```

Note:- That ^ matches the expression at the beginning of a line, only if it is the first character in a regular expression. In this example, ^N matches all the lines that begins with N.

End of line ($):-The dollar symbol $ matches the end of a line.

Example:Display lines which end with the letter r:

```
$ sed -n '/r$/ p' employee.txt
102,Jason Smith,IT Manager
104,Anand Ram,Developer
```

```
105,Jane Miller,Sales Manager
```

Single Character (.)

The special meta-character “.” (dot) matches any character exceptthe end of the line character.

- . matches single character
- .. matches two characters
- ... matches three characters
- etc.

In the following example, the pattern "J followed by three charactersand a space" will be replaced with "Jason followed by a space". So, "J... " matches both "John " and "Jane " from employee.txt, and these two lines are replaced accordingly as shown below

```
$ sed -n 's/J... /Jason /p' employee.txt
101,Jason Doe,CEO
105,Jason Miller,Sales Manager
```

Zero or more Occurrences (*):-The special character “*” (star) matches zero or more occurrences of the previous character. For example, the pattern ’1*’ matches zero or more ’1'.

For this example create the following log.txt file:

$ vi log.txt

log: Input Validated

log:

log: testing resumed

log:

log:output created

Suppose you would like to view only the lines that contain "log:" followed by a message. The message might immediately follow the log: or might have some spaces. You don't want to view the lines that contain "log:" without anything.

Example:-Display all the lines that contain "log:" followed by one or more spaces followed by a character:

```
$ sed -n '/log: *./ p' log.txt
log: Input Validated
log: testing resumed
log:output created
```

Note: In the above example the dot at the end is necessary. If not included, sed will also print all the lines containing "log:" only.

One or more Occurrence (\+):-The special character "\+" matches one or more occurrence of the previous character. For example, a space before "\+", i.e " \+" matches at least one or more space character.

Let us use the same log.txt as an example file.

Example:-Display all the lines that contain "log:" followed by one or more spaces:

```
$ sed -n '/log: \+/ p' log.txt
log: Input Validated
log: testing resumed
```

Note: In addition to not matching the "log:" only lines, the aboveexample also didn't match the line "log:output created", as there is no space after "log:" in this line.

Zero or one Occurrence (\?):-The special character "?" matches zero or one occurrences of the previous character as shown below.

```
$ sed -n '/log: \?/ p' log.txt
log: Input Validated
log:
log: testing resumed
log:
log:output created
```

Escaping the Special Character (\):-If you want to search for special characters (for example: * , dot) in the content you have to escape the special character in the regularexpression.

```
$ sed -n '/127\.0\.0\.1/ p' /etc/hosts
127.0.0.1 localhost.localdomain localhost
```

Character Class ([0-9]):-The character class is nothing but a list of characters mentioned within a square bracket; this is used to match only one out of several characters.

Match any line that contains 2 or 3 or 4:

```
$ sed -n '/[234]/ p' employee.txt
```

```
102,Jason Smith,IT Manager
103,Raj Reddy,Sysadmin
104,Anand Ram,Developer
```

Within the square bracket, you can use a hyphen you can specify a range of characters. For example, [0123456789] can be represented by [0-9], and alphabetic ranges can be specified such as [a-z],[A-Z] etc.

Match any line that contains 2 or 3 or 4 (alternate form):

```
$ sed -n '/[2-4]/ p' employee.txt
102,Jason Smith,IT Manager
103,Raj Reddy,Sysadmin
104,Anand Ram,Developer
```

21. Additional Regular Expressions

OR Operation (|):-The pipe character (|) is used to specify that either of two whole subexpressions could occur in a position. “subexpression1|subexpression2” matches either subexpression1 or subexpression2.

Example:-Print lines containing either 101 or 102:

```
$ sed -n '/101\|102/ p' employee.txt
101,John Doe,CEO
102,Jason Smith,IT Manager
```

Please note that the | symbol is escaped with a /.

Example:-Print lines that contain a character from 2 to 3 or that contain the string 105:

```
$ sed -n '/[2-3]\|105/ p' employee.txt
102,Jason Smith,IT Manager
103,Raj Reddy,Sysadmin
105,Jane Miller,Sales Manager
```

Exactly M Occurrences ({m})

A Regular expression followed by {m} matches exactly m occurrences of the preceding expression.

For this example create the following numbers.txt file.

```
$ vi numbers.txt
1
12
123
1234
12345
123456
```

Example:-Print lines that contain any digit (will print all lines):

```
$ sed -n '/[0-9]/ p' numbers.txt
1
12
123
1234
12345
123456
```

Example:-Print lines consisting of exactly 5 digits:

```
$ sed -n '/^[0-9]\{5\}$/ p' numbers.txt
12345
```

M to N Occurrences ({m,n}):-A regular expression followed by {m,n} indicates that the preceding item must match at least m times, but not more than n times. The values of m and n must be non-negative and smaller than 255.

Example:-Print lines consisting of at least 3 but not more than 5 digits:

```
$ sed -n '/^[0-9]\{3,5\}$/ p' numbers.txt
123
1234
12345
```

A Regular expression followed by {m,} is a special case that matches m or more occurrences of the preceding expression.

Word Boundary (\b):-\b is used to match a word boundary. \b matches any character(s) at the beginning (\bxx) and/or end (xx\b) of a word, thus \bthe\b will find the but not they. \bthe will find the or they.

Create the following sample file for testing.

```
$ cat words.txt
word matching using: the
word matching using: thethe
word matching using: they
```

Example:-Match lines containing the whole word "the":

```
$ sed -n '/\bthe\b/ p' words.txt
word matching using: the
```

Please note that if you don't specify the \b at the end, it will match all lines.

Match lines containing words that start with “the”:

```
$ sed -n '/\bthe/ p' words.txt
word matching using: the
word matching using: thethe
word matching using: they
```

Back References (\n):-Back references let you group expressions for further use.

Match only the line that has the word "the" repeated twice:

```
$ sed -n '/\(the\)\1/ p' words.txt
```

Using the same logic, the regular expression "\([0-9]\)\1" matches two digit number in which both the digits are same number—like 11,22,33

22. Sed Substitution Using Regular Expression

The following are few sed substitution examples that uses regular expressions.

Example:-Replace the last two characters in every line of employee.txt with ",Not Defined":

```
$ sed 's/..$/,Not Defined/' employee.txt
101,John Doe,C,Not Defined
102,Jason Smith,IT Manag,Not Defined
```

103,Raj Reddy,Sysadm**,Not Defined**

104,Anand Ram,Develop**,Not Defined**

105,Jane Miller,Sales Manag**,Not Defined**

Example:Delete the rest of the line starting from “Manager”:

```
$ sed 's/Manager.*//' employee.txt
101,John Doe,CEO
102,Jason Smith,IT
103,Raj Reddy,Sysadmin
104,Anand Ram,Developer
105,Jane Miller,Sales
```

Delete all lines that start with "#" :

```
sed -e 's/#.*// ; /^$/ d' employee.txt
```

Create the following test.html for the next example:

```
$ vi test.html
<html><body><h1>Hello World!</h1></body></html>
```

Strip all html tags from test.html:

```
$ sed -e 's/<[^>]*>//g' test.html
Hello World!
```

Remove all comments and blank lines:

```
sed -e 's/#.*//' -e '/^$/ d' /etc/profile
```

Remove only the comments. Leave the blank lines:

```
sed -e '/^#.*/ d' /etc/profile
```

You can convert DOS newlines (CR/LF) to Unix format Using sed. When you copy the DOS file to Unix, you could find \r\n in the end of each line.

Convert the DOS file format to Unix file format using sed:

```
sed 's/.$//' filename
```

Chapter 4. Sed Execution

23. Multiple Sed Commands in Command Line

As we showed in Chapter 1, there are several methods to execute multiple sed commands from the command line.

1. Use multiple -e option in the command line

Use multiple sed commands using -e sed command line option as shown below:

```
sed -e 'command1' -e 'command2' -e 'command3'
```

Search for root, or nobody, or mail in the /etc/passwd file:

```
sed -n -e '/^root/ p' -e '/^nobody/ p' -e '/^mail/p' /etc/passwd
```

The above command should be executed in a single line as shown below:-

```
sed -n -e '/^root/ p' -e '/^nobody/ p' -e '/^mail/ p' /etc/passwd
```

**2. Break-up several sed commands using **

When you have a very long command, such as when executing several sed commands in the command line using -e, you can break it up using \

```
sed -n -e '/^root/ p' \
-e '/^nobody/ p' \
-e '/^mail/ p' \
/etc/passwd
```

3. Group multiple commands using { }: When you have a lot of sed commands to be executed, you can group them together using { } as shown below.

```
sed -n '{
/^root/ p
/ ^nobody/ p
/^mail/ p
}' /etc/passwd
```

24. Sed Script Files

If you want to reuse a set of sed commands, create a sed script file with all the sed commands and execute it using -f command line option as shown below.

First, create a file that contains all the sed commands as shown below. You already know what these individual sed commands do, as we explained it in the previous sections.

```
$ vi mycommands.sed
s/\([^,]*\),\([^,]*\),\(.*\).*/\2,\1,\3/g
s/^.*/<&>/
s/Developer/IT Manager/
s/Manager/Director/
```

Next, execute this sed command file on the input file.

```
$ sed -f mycommands.sed employee.txt
<John Doe,101,CEO>
<Jason Smith,102,IT Director>
<Raj Reddy,103,Sysadmin>
<Anand Ram,104,IT Director>
```

<Jane Miller,105,Sales Director>

25. Sed Comments

Sed comments start with a #. We all understand that sed uses very cryptic language. The sed commands that you write today might look unfamiliar if you view them after a long time. So, it is recommended to document what you mean inside the sed script file using sed comments, as shown below.

```
$ vi mycommands.sed
# Swap field 1 (employee id) with field 2 (employeename)
s/\([^,]*\),\([^,]*\),\(.*\).*/\2,\1,\3/g
# Enclose the whole line within < and >
s/^.*/<&>/
# Replace Developer with IT Manager
s/Developer/IT Manager/
# Replace Manager with Director
s/Manager/Director/
```

Note: If the 1st 2 characters of the 1st line in the *.sed script are #n, sed will automatically use the -n (don't print the pattern buffer) option.

26. Sed as an Interpreter

Just as you write shell scripts and execute them from the command line just by calling the file name, you can set up sed scripts for execution from the command line, i.e. Sed can be involved as an interpreter. To do this, add "#!/bin/sed -f" as the 1st line to your sedscript.

sh file as shown below.

```
$ vi myscript.sed
#!/bin/sed -f
# Swap field 1 (employee id) with field 2 (employeename)
s/\([^,]*\),\([^,]*\),\(.*\).*/\2,\1,\3/g
# Enclose the whole line within < and >
s/^.*/<&>/
# Replace Developer with IT Manager
s/Developer/IT Manager/
# Replace Manager with Director
s/Manager/Director/
```

Now, execute the sed script directly by invoking it from the command line.

```
chmod u+x myscript.sed
./myscript.sed employee.txt
```

You can also specify -n in the 1st line of the sed script to suppress output.

```
$ vi testscript.sed
#!/bin/sed -nf
/root/ p
/nobody/ p
```

Now, execute the above test script as shown below.

```
chmod u+x testscript.sed
./testscript.sed /etc/passwd
```

Just for testing purposes, remove the -n from the 1st line of testscript.sed and execute it again to see how it works.

Important note: you must use -nf (and not -fn). If you specify -fn, you'll get the following error message when you execute the sed script.

```
$ ./testscript.sed /etc/passwd
/bin/sed: couldn't open file n: No such file or directory
```

27. Modifying the Input File Directly

As you know already, sed doesn't modify the input files by default. Sed writes the output to standard output. When you want to store that in a file, you redirect it to a file (or use the w command.Before we continue with this example, take a backup of employee.txt:

```
cp employee.txt employee.txt.orig
```

To make a modification directly on the input-file, you typically redirect the output to a temporary file, and then rename the temporary file to a new file.

```
sed 's/John/Johnny/' employee.txt > new-employee.txt
mv new-employee.txt employee.txt
```

Instead, you can use the sed command line option -i, which lets sed

directly modify the input file.

Replace John with Johnny in the original employee.txt file itself:

```
$ sed -i 's/John/Johnny/' employee.txt
$ cat employee.txt
101,Johnny Doe,CEO
102,Jason Smith,IT Manager
103,Raj Reddy,Sysadmin
104,Anand Ram,Developer
105,Jane Miller,Sales Manager
```

Again, please pay attention that **-i modifies the input-file**. Probably you will want to do this sometimes, but be very careful. One thing you can do to protect yourself is to add a file extension whenever you use -i. Sed will make a backup of the original file before writing thenew content.

Replace John with Johnny in the original employee.txt file but save a backup copy:

```
$ sed -ibak 's/John/Johnny/' employee.txt
```

This takes the backup of the original file as shown below:-

```
$ cat employee.txtbak
101,John Doe,CEO
102,Jason Smith,IT Manager
103,Raj Reddy,Sysadmin
104,Anand Ram,Developer
105,Jane Miller,Sales Manager
```

The original input file was modified by the above sed command.

```
$ cat employee.txt
101,Johnny Doe,CEO
102,Jason Smith,IT Manager
103,Raj Reddy,Sysadmin
104,Anand Ram,Developer
105,Jane Miller,Sales Manager
```

Instead of -i, you can also use the longer form, --in-place. Both of the following commands are the same.

```
sed -ibak 's/John/Johnny/' employee.txt
sed --in-place=bak 's/John/Johnny/' employee.txt
```

Finally, restore the original employee.txt file, as we need that for the rest of our examples:

```
cp employee.txt.orig employee.txt
```

Chapter 5. Additional Sed Commands

28. Append Line After (a command)

You can insert a new line after a specific location by using the sed append command (a).

Syntax:

```
$ sed '[address] a the-line-to-append' input-file
```

Example:-Add a new record to the employee.txt file after line number:

```
$ sed '2 a 203,Jack Johnson,Engineer' employee.txt
101,John Doe,CEO
102,Jason Smith,IT Manager
203,Jack Johnson,Engineer
103,Raj Reddy,Sysadmin
104,Anand Ram,Developer
105,Jane Miller,Sales Manager
```

Example:-Add a new record to the end of the employee.txt file:

```
$ sed '$ a 106,Jack Johnson,Engineer' employee.txt
101,John Doe,CEO
102,Jason Smith,IT Manager
103,Raj Reddy,Sysadmin
104,Anand Ram,Developer
105,Jane Miller,Sales Manager
```

106,Jack Johnson,Engineer

You can also append multiple lines using the sed a command.

Example:-Add two lines after the line that matches 'Jason':

```
$ sed '/Jason/a\
203,Jack Johnson,Engineer\
204,Mark Smith,Sales Engineer' employee.txt
101,John Doe,CEO
102,Jason Smith,IT Manager
203,Jack Johnson,Engineer
204,Mark Smith,Sales Engineer
103,Raj Reddy,Sysadmin
104,Anand Ram,Developer
105,Jane Miller,Sales Manager
```

29. Insert Line Before (i command)

The sed insert command (i) works just like the append command except that it inserts a line **before** a specific location instead of after the location.

Syntax:

```
$ sed '[address] i the-line-to-insert' input-file
```

Example:-Insert a new record before line number 2 of the employee.txt file:

```
$ sed '2 i 203,Jack Johnson,Engineer' employee.txt
101,John Doe,CEO
203,Jack Johnson,Engineer
102,Jason Smith,IT Manager
103,Raj Reddy,Sysadmin
```

104,Anand Ram,Developr

105,Jane Miller,Sales Manager

Example:-Insert a new record before the last line of the employee.txt file:

$ sed '$ i 108,Jack Johnson,Engineer' employee.txt

101,John Doe,CEO

102,Jason Smith,IT Manager

103,Raj Reddy,Sysadmin

104,Anand Ram,Developer

108,Jack Johnson,Engineer

105,Jane Miller,Sales Manager

You can also insert multiple lines using the sed i command.

Example:-Insert two lines before the line that matches 'Jason':

$ sed '/Jason/i\

203,Jack Johnson,Engineer\

204,Mark Smith,Sales Engineer' employee.txt

101,John Doe,CEO

203,Jack Johnson,Engineer

204,Mark Smith,Sales Engineer

102,Jason Smith,IT Manager

103,Raj Reddy,Sysadmin

104,Anand Ram,Developer

105,Jane Miller,Sales Manager

30. Change Line (c command)

The sed change command (c) lets you replace an existing line with new text.

Syntax:

```
$ sed '[address] c the-line-to-insert' input-file
```

Example:-Delete the record at line number 2 and replace it with a new record:

```
$ sed '2 c 202,Jack Johnson,Engineer' employee.txt
101,John Doe,CEO
202,Jack Johnson,Engineer
103,Raj Reddy,Sysadmin
104,Anand Ram,Developer
105,Jane Miller,Sales Manager
```

You can also replace a single line with multiple lines.

Example:-Delete the line that matches 'Raj' and replaces it with two new lines:

```
$ sed '/Raj/c\
203,Jack Johnson,Engineer\
204,Mark Smith,Sales Engineer' employee.txt
101,John Doe,CEO
102,Jason Smith,IT Manager
203,Jack Johnson,Engineer
204,Mark Smith,Sales Engineer
104,Anand Ram,Developer
```

105,Jane Miller,Sales Manager

31. Combine a, i, and c Commands

You can also combine the a, i, and c commands. the following sed example does all these three things:

• a - Append 'Jack Johnson' after 'Jason'

• i - Insert 'Mark Smith' before 'Jason'

• c - Change 'Jason' to 'Joe Mason'

```
$ sed '/Jason/ {
   a\
204,Jack Johnson,Engineer
   i\
202,Mark Smith,Sales Engineer
   c\
203,Joe Mason,Sysadmin
}' employee.txt
101,John Doe,CEO
202,Mark Smith,Sales Engineer
203,Joe Mason,Sysadmin
204,Jack Johnson,Engineer
103,Raj Reddy,Sysadmin
104,Anand Ram,Developer
105,Jane Miller,Sales Manager
```

32. Print Hidden Characters (l command)

The sed l command prints the hidden characters, for example, \t for tab, and $ for end of the line.

For testing, create a test file with the following content. Make sure to use the tab key between the fields in this file.

```
$ cat tabfile.txt
fname First Name
lname Last Name
mname Middle Name
```

Executing the sed l command will display \t for tab, and $ for EOL:

```
$ sed -n l tabfile.txt
fname\tFirst Name$
lname\tLast Name$
mname\tMiddle Name$
```

When you specify a number followed by the l command, the output line is wrapped at the nth number using a non printable character as shown in the example below. This works only on GNU sed.

```
$ sed -n 'l 20' employee.txt
101,John Doe,CEO$
102,Jason Smith,IT \Manager$
103,Raj Reddy,Sysad\min$
104,Anand Ram,Devel\oper$
```

105,Jane Miller,Sal\es Manager$

33. Print Line Numbers (= command)

The sed = command prints line numbers followed by the line content from the input-file.

Example:-Print all line numbers:

```
$ sed = employee.txt
1
101,John Doe,CEO
2
102,Jason Smith,IT Manager
3
103,Raj Reddy,Sysadmin
4
104,Anand Ram,Developer
5
105,Jane Miller,Sales Manager
```

Note: You can print the line number and the line content in the same line by combining = command with N command (more on this later).

Example:-Print line numbers only for lines 1,2 and 3:

```
 $ sed '1,3 =' employee.txt
1
101,John Doe,CEO
2
58
```



```
102,Jason Smith,IT Manager
3
103,Raj Reddy,Sysadmin
104,Anand Ram,Developer
105,Jane Miller,Sales Manager
```

Print the line number only for those lines that contain the keyword Jane. This still prints the original line content from the intput-file:

```
$ sed '/Jane/ =' employee.txt
101,John Doe,CEO
102,Jason Smith,IT Manager
103,Raj Reddy,Sysadmin
104,Anand Ram,Developer
5
105,Jane Miller,Sales Manager
```

If you want to know only the line numbers of lines that contains the keyword (i.e. without printing the original lines from the file), use –n option along with = as shown below.

```
$ sed -n '/Raj/ =' employee.txt
3
```

Example:-Print the total number of lines in a file:

```
$ sed -n '$ =' employee.txt
 5
```

34. Change Case (using the y 'transform'command)

The sed y command transforms characters by position. A convenient use for this is to convert upper case to lower case and vice versa.

Example:-In this example character "a" will be transformed to A, b to B, c to C, etc.:

```
 $ sed 'y/abcde/ABCDE/' employee.txt
101,John DoE,CEO
102,JAson Smith,IT MAnAgEr
103,RAj REDDy,SysADmin
104,AnAnD RAm,DEvElopEr
105,JAnE MillEr,SAlEs MAnAgEr
```

Transform all lower-case letters to upper-case:

```
$ sed
'y/abcdefghijklmnopqrstuvwxyz/ABCDEFGHIJKLMNOPQRSTUVWXY
Z/' employee.txt
101,JOHN DOE,CEO
102,JASON SMITH,IT MANAGER
103,RAJ REDDY,SYSADMIN
104,ANAND RAM,DEVELOPER
105,JANE MILLER,SALES MANAGER
```

The above command should be executed in a single line as shown below.

```
sed    'y/abcdefghijklmnopqrstuvwxyz/ABCDEFGHIJKLMNOPQRSTUVWXYZ/'
employee.txt
```

35. Multiple Files in Command Line

In all our previous sed examples, we passed only one input file. You can also pass multiple input files as shown below.

Example:-The following example searches for root in the /etc/passwd file and prints it:

```
$ sed -n '/root/ p' /etc/passwd
 root:x:0:0:root:/root:/bin/bash
```

The following example searches for root in the /etc/group and prints it:

```
$ sed -n '/root/ p' /etc/group
```

```
root:x:0:
```

Search for root in both the /etc/passwd and /etc/group file:

```
$ sed -n '/root/ p' /etc/passwd /etc/group
root:x:0:0:root:/root:/bin/bash
root:x:0:
```

36. Quit Sed (q command)

The sed q command causes sed to quit executing commands.As we discussed earlier, the normal sed execution flow is Read,Execute, Print, Repeat.

When sed executes the q command, it simply quits without executing the rest of the sed commands, and without repeating the rest of the lines from the input-file.

Quit after printing the 1st line:

```
$ sed 'q' employee.txt
101,John Doe,CEO
```

Quit after the 5th line. So, this prints the 1st 5 lines:

```
$ sed '5 q' employee.txt
101,John Doe,CEO
102,Jason Smith,IT Manager
103,Raj Reddy,Sysadmin
104,Anand Ram,Developer
105,Jane Miller,Sales Manager
```

Example:-Print all the lines until the 1st line that contains the keyword 'Manager':

```
$ sed '/Manager/q' employee.txt
101,John Doe,CEO
102,Jason Smith,IT Manager
```

Note: q command doesn't take range of address. It works only on a single address. (or a single pattern)

37. Read from File (r command)

The sed r command will read the content of another file and print it at a specified location while processing the input-file. The following example will read the content of log.txt file and print it after printing the last line of employee.txt. Basically this combines both employee.txt and log.txt and prints the result.

```
$ sed '$ r log.txt' employee.txt
```

You can also specify a pattern with the r command. The following example will read the content of log.txt and print it after the line that matches 'Raj' in the employee.txt.

Example:-Insert the log.txt file after the 'Raj' keyword in the employee.txt file:

```
$ sed '/Raj/ r log.txt' employee.txt
```

38. Simulating Unix commands in sed (cat, grep, head)

We have already seen examples that worked very much like other standard UNIX commands. Using sed you can simulate many commands. Do this just to learn how sed works.

Cat in sed

```
cat employee.txt
```

Each of the following sed commands produces the same output as the cat command above"-

```
sed 's/JUNK/&/p' employee.txt
sed -n 'p' employee.txt
sed 'n' employee.txt
sed 'N' employee.txt
```

Grep in sed

Simple grep:

```
grep Jane employee.txt
```

Each of the following sed commands produces the same output as the grep command above:-

```
sed -n 's/Jane/&/p' employee.txt
sed -n '/Jane/ p' employee.txt
```

grep -v (print non-matching lines):

```
grep -v Jane employee.txt
```

The following sed command is equivalent to the above "grep -v" command:-

```
sed -n '/Jane/ !p' employee.txt
```

Head in sed

```
head -10 /etc/passwd
```

Each of the following sed commands produces the same output as the head command above.

```
sed '11,$ d' /etc/passwd
sed -n '1,10 p' /etc/passwd
sed '10 q' /etc/passwd
```

39. Sed Command Line Options

-n option

We already discussed this option and we have used it in many examples. The sed option -n suppresses the default printing that happens as part of the standard sed flow.

You can also use --quiet, or –-silent instead of -n. They are identical in function.

All of the following commands are the same:

```
sed -n 'p' employee.txt
sed --quiet 'p' employee.txt
sed --silent 'p' employee.txt
```

-f option

You can also combine multiple sed-commands in a file and call the sed script file using the -f option. We demonstrated this earlier. You can also use -–file.

All of the following commands are the same:

```
sed -n -f test-script.sed /etc/passwd
sed -n --file=test-script.sed /etc/passwd
```

-e option

Use -e to execute a sed command script from the command line. You can use multiple -e options from the command line. You can also use–-expression.

All of the following commands are the same:

```
sed -n -e '/root/ p' /etc/passwd
sed -n --expression '/root/ p' /etc/passwd
```

-i option

As we already discussed sed doesn't touch the input file. It always prints to standard output, Or you can use the w command to write the output to a different file. We also showed how sed can use the –I option to modify the input file directly.

Replace John with Johnny in the original employee.txt file:

```
sed -i 's/John/Johnny/' employee.txt
```

Perform the same command but take a backup by passing an extension to -i.

```
sed -ibak 's/John/Johnny/' employee.txt
```

Instead of -i, you can also use –-in-place.

Both of the following commands are the same:

```
sed -ibak 's/John/Johnny/' employee.txt
sed --in-place=bak 's/John/Johnny/' employee.txt
```

-c option

This should be used in conjunction with sed option -i. Sed option –i typically uses a temporary file to create the changes and renames it to the original input-file when the operation is completed. This might cause file ownership to change. When you use -c along with -i, the input file ownership will not change. You can also use copy.

Both of the following commands are the same:

```
sed -ibak -c 's/John/Johnny/' employee.txt
sed --in-place=bak --copy 's/John/Johnny/' employee.txt
```

-l option

Specify the line length. This needs to be used in conjunction with the sed l command. The value you specify in the -l option will be used as the line size. You can also use –-line-length.

All the following commands are the same.

```
sed -n -l 20 'l' employee.txt
sed -n --line-length=20 employee.txt
```

Please note that you can also achieve the same output without specifying -n option as shown below.

```
sed -n 'l 20' employee.txt --posix option
```

40. Print Pattern Space (n command)

The sed n command prints the current pattern space and fetches the next line from the input-file. This happens in the middle of command execution, and so it can change the normal flow if it occurs between other commands.

Print the pattern space for each line:

```
$ sed n employee.txt
101,John Doe,CEO
102,Jason Smith,IT Manager
103,Raj Reddy,Sysadmin
104,Anand Ram,Developer
105,Jane Miller,Sales Manager
```

If you specify -n flag when you are using the n command, sed will not print anything.

```
$ sed -n n employee.txt
```

As we discussed earlier, normal sed execution flow is Read, Execute (all available sed commands), Print, Repeat. The sed n command lets you change that flow. The sed n command will print the current pattern space, clear the current pattern space, read the next line from the input-file, and continue the command flow.

Let us assume that you have 2 sed commands before and 2 after the n command as shown below:-

```
sed-command-1
sed-command-2
 n
 sed-command-3
 sed-command-4
```

In this case, sed-command-1 and sed-command-2 will be applied to the current line in the pattern space; when sed encounters the n command, it will clear the current line from the pattern space, read the next line from the input-file, and apply sed-command-3 and sedcommand- 4 to this newly read line in the sed pattern space.

Note: The sed n command by itself is relatively useless as you see in the above examples. However, it is extremely powerful when combined with the sed hold pattern commands that are discussed in the following hacks.

Chapter 6. Sed Hold and Pattern Space Commands

Sed has two types of internal storage space:

• **Pattern space:** You already know about pattern space, which is used as part of the typical sed execution flow. Pattern space is the internal sed buffer where sed places, and modifies, the line it reads from the input file.

• **Hold space:** This is an additional buffer available where sed can hold temporary data. Sed allows you to move data back and forth between pattern space and hold space, but you cannot execute the typical sed commands on the hold space.

As we already discussed, pattern space gets deleted at the end of every cycle in a typical sed execution flow. However, the content of the hold space will is retained from one cycle to the next; it is not deleted between cycles.

Please create a new text file to be used for the sed hold space examples:

```
$ vi empnametitle.txt
John Doe
 CEO
 Jason Smith
 IT Manager
  Raj Reddy
  Sysadmin
  Anand Ram
```

Developer

Jane Miller

Sales Manager

As you can see, for each employee this file contains name and title on two consecutive lines.

41. Swap Pattern Space with Hold Space (x command)

The sed Exchange (x) command swaps pattern space with hold space. This command in itself is not that helpful, unless it is combined with other sed commands; however, in conjunction with other commands, it is quite powerful.

Suppose that pattern space contains "line 1" and hold space contains "line 2". After the x command is executed, pattern space will have "line 2", and hold space will have "line 1".

The following example prints the names of the managers. It looks for the keyword 'Manager' and prints the previous line.

Print manager names from empnametitle.txt:

```
$ sed -n -e 'x;n' -e '/Manager/{x;p}' empnametitle.txt
Jason Smith
Jane Miller
```

In the above example:

• **{x;n}** - x swaps pattern space to the hold space; n reads the next line into the pattern space. So, this command saves the current line in hold space and reads the next line into pattern space. For the example file, it is saving employee name to hold space and fetching employee title into pattern space.

• **/Manager/{x;p}** - If the content of the pattern space contains the keyword 'Manager', this command swaps pattern space with hold space and then prints pattern space. This means that if the employee title contains 'Manager' the employee name will be printed.

You can also save this in a sed script file and execute it as shown below:-

```
$ vi x.sed
#!/bin/sed -nf
x;n
/Manager/{x;p}
$ chmod u+x empnametitle.txt
$ ./x.sed empnametitle.txt
Jason Smith
Jane Miller
```

42. Copy Pattern Space to Hold Space (h command)

The hold command (h) copies pattern space to hold space. Unlike the x command, the h command does not change the content of pattern space. The previous content of the hold space is overwritten with the content from the pattern space.

Suppose pattern space contains "line 1" and hold space contains "line 2"; after the h command is executed, pattern space is not changed and will still have "line 1", but hold space will also have "line 1".

Print the names of the managers:

```
$ sed -n -e '/Manager/!h' -e '/Manager/{x;p}'
empnametitle.txt
 Jason Smith
 Jane Miller
```

The above command should be executed in a single line as shown below.

```
sed -n -e '/Manager/!h' -e '/Manager/{x;p}' empnametitle.txt
```

In the above example:-

• **/Manager/!h** - If the content of the pattern space doesn't contain Manager (the ! after the pattern means "not equal to" the pattern), copy the content of the pattern space to the hold space. (In this case, this might be employee name (or) a title that is not "Manager".)

Note that, unlike the previous example, this one does not use the 'n' command to get the next line; instead, the next line is fetched via normal execution flow.

• **/Manager/{x;p}** - If the content of the pattern space contains the keyword 'Manager', this command swaps pattern space with hold space and prints.

This is identical to the command we used for printing in the example for the x command.

You can also save this in a sed script file and execute it as shown below:-

```
$ vi h.sed
#!/bin/sed -nf
/Manager/!h
/Manager/{x;p}
$ chmod u+x empnametitle.txt
$ ./h.sed empnametitle.txt
Jason Smith
Jane Miller
```

43. Append Pattern Space to Hold Space (H command)

Capital H is the command to append pattern space to hold space with a new line. The previous content of hold space is not overwritten; instead the content of pattern space is appended to the existing content of hold space by adding a new line at the end.

Suppose pattern space contains "line 1" and hold space contains "line 2"; after the H command is executed, pattern space is not changed and will still have "line 1", but hold space will have "line 2\nline 1".

Print the name and title (in separate lines) of the managers:

```
$ sed -n -e '/Manager/!h' -e '/Manager/{H;x;p}'
empnametitle.txt
Jason Smith
IT Manager
Jane Miller
```

```
Sales Manager
```

The above command should be executed in a single line as shown below:-

```
sed -n -e '/Manager/!h' -e '/Manager/{H;x;p}' empnametitle.txt
```

In the above example:

• **/Manager/!h** - If the content of the pattern space doesn't contain Manager (the ! after the pattern means "not equal to" the pattern), copy the content of the pattern space to the hold space. (In this case, this might employee name (or) a title that is not "Manager".) This is the same command we used in the h command example.

• **/Manager/{H;x;p}** - If the content of the pattern space contains the keyword 'Manager', the H command appends pattern space (which is Manager) to hold space with a new line. So, the hold space at this stage will have "Employee Name\nTitle" (which contains the keyword manager). The x command swaps hold space back into pattern space, and p prints the pattern space.

You can also save this in a sed script file and execute it as shown below:-

```
$ vi H-upper.sed
 #!/bin/sed -nf
/Manager/!h
/Manager/{H;x;p}
$ chmod u+x H-upper.sed
$ ./H-upper.sed empnametitle.txt
Jason Smith
 IT Manager
Jane Miller
Sales Manager
```

The above example can be slightly modified, if you want the employee name and title to be printed on the same line with colon : as a delimiter:

```
$ sed -n -e '/Manager/!h' -e '/Manager/{H;x;s/\n/:/;p}'
empnametitle.txt
 Smith:IT Manager
 Jane Miller:Sales Manager
```

The above command should be executed in a single line as shown below:-

```
sed -n -e '/Manager/!h' -e '/Manager/{H;x;s/\n/:/;p}' empnametitle.txt
```

In the second example everything is same as the previous example except for the substitute command added to the 2nd -e option. The H, x, and p commands do the same thing as before; the s commandreplaces \n with : after swapping but before printing. Therefore the name and title are printed on one line, separated by a colon.

You can also save this in a sed script file and execute it as shown below:-

```
$ vi H1-upper.sed
 #!/bin/sed -nf
 /Manager/!h
 /Manager/{H;x;s/\n/:/;p}
$ chmod u+x H1-upper.sed
$ ./H1-upper.sed empnametitle.txt
Jason Smith:IT Manager
Jane Miller:Sales Manager
```

44. Copy Hold Space to Pattern Space (g command)

The sed get (g) command copies the content of hold space to pattern space.
Think of it this way: h command "holds" it in the hold space, g command "gets" it from the hold space.

Suppose pattern space contains "line 1" and hold space contains "line 2"; after the g command is executed, pattern space is changed and now contains "line 2", while hold space is not changed and still contains "line 2".

Print the names of the managers:

```
$ sed -n -e '/Manager/!h' -e '/Manager/{g;p}'
empnametitle.txt
Jason Smith
Jane Miller
```

The above command should be executed in a single line as shown below:-

```
sed -n -e '/Manager/!h' -e '/Manager/{g;p}' empnametitle.txt
```

In the above example:

• **/Manager/!h** – we've been using this one for the last few examples. If the content of the pattern space doesn't contain Manager, copy the content of pattern space to hold space.

• **/Manager/{g;p}** – g gets the line from hold space and puts it in pattern space, then prints it.

You can also save this in a sed script file and execute it as shown below:-

```
$ vi g.sed
#!/bin/sed -nf
/Manager/!h
/Manager/{g;p}
$ chmod u+x g.sed
 $ ./g.sed empnametitle.txt
 Jason Smith
 Jane Miller
```

45. Append Hold Space to Pattern Space (G command)

Upper case G appends the content of hold space to pattern space with a new line. The previous content in the pattern space is not overwritten; instead the content from hold space is appended to the existing content in pattern space by adding a new line at the end. G and g are related in the same way as H and h; the lower case version replaces the content while the upper case one appends to it.

Suppose pattern space contains "line 1" and hold space contains "line 2"; after the G command is executed, pattern space is changed to contain "line 1\nline 2" while hold space is not changed and still contains "line 2".

Prints the employee name and title of the managers separated by colon.

```
$ sed -n -e '/Manager/!h' -e '/Manager/{x;G;s/\n/:/;p}'
empnametitle.txt
Jason Smith:IT Manager
Jane Miller:Sales Manager
```

The above command should be executed in a single line as shown below:-

```
sed -n -e '/Manager/!h' -e '/Manager/{x;G;s/\n/:/;p}' empnametitle.txt
```

In the above example:

• **/Manager/!h** – As in previous examples, if the content of pattern space doesn't contain Manager, copy pattern space to hold space.

• **/Manager/{x;G;s/\n/:/;p}** - If the content of the pattern space contains Manager, do the following:

• **x** - Swap the content of pattern space with hold space. So, the employee name stored in hold space will now be in pattern space, while the title will be in hold space.

• **G** - Appends the content of hold space (title) to pattern space (employee name). So, the pattern space at this stage will have "Employee Name\nTitle"

• **s/\n/:/** This replaces the \n that separates the "Employee Name\nTitle" with a colon :

• **p** prints the result (i.e. the content of pattern space).

Note that if we left out the x command, i.e. if we used /Manager/{G;s/\n/:/;p}, we would print the title:name instead of name:title for each manager.

You can also save this in a sed script file and execute it as shown below:-

```
$ vi G-upper.sed
#!/bin/sed -nf
/Manager/!h
/Manager/{x;G;s/\n/:/;p}
 $ chmod u+x G-upper.sed
$ ./G-upper.sed empnametitle.txt
Jason Smith:IT Manager
Jane Miller:Sales Manager
```

Chapter 7. Sed Multi-LineCommands and loops

Sed by default always handles one line at a time, unless we use the H, G, or N command to create multiple lines separated by new line.

This chapter will describe sed commands applicable to such multi-line buffers.

Note: When we have multiple lines, please keep in mind that ^ matches only the 1st character of the buffer, i.e. of all the multiple lines combined together, and $ matches only the last character in the buffer, i.e. the newline of the last line.

46. Append Next Line to Pattern Space (N command)

Just as upper case H and G append rather than replacing, the N command appends the next line from input-file to the pattern buffer, rather than replacing the current line.

As we discussed earlier the lower case n command prints the current pattern space, clears the pattern space, reads the next line from the input-file into pattern space and resumes command execution where it left off.

The upper case N command does not print the current pattern space and does not clear the pattern space. Instead, it adds a newline (\n) at the end of the current pattern space, appends the next line from the input-file to the current pattern space, and continues with the sed standard flow by executing the rest of the sed commands.

Print employee names and titles separated by colon:

```
$ sed -e '{N;s/\n/:/}' empnametitle.txt
John Doe:CEO
Jason Smith:IT Manager
Raj Reddy:Sysadmin
Anand Ram:Developer
Jane Miller:Sales Manager
```

In the above example:

• **N** appends new line to current pattern space (which has employee name) and appends the next line from input-file to the current pattern space. So, the pattern space will contain (employee name\ntitle).

• **s/\n/:/** This replaces the \n that separates the "Employee Name\nTitle" with a colon :

Fig: Illustration of the above example

The following example demonstrates the use of the N command to print the line number on the same line as the text, while printing each line from employee.txt.

Example:-Print line numbers:

```
$ sed -e '=' employee.txt | sed -e '{N;s/\n/ /}'
1 101,John Doe,CEO
2 102,Jason Smith,IT Manager
3 103,Raj Reddy,Sysadmin
4 104,Anand Ram,Developer
5 105,Jane Miller,Sales Manager
```

As we saw in our previous examples, the sed = command prints the line number first, and the original line next.

In this example, the N command adds \n to the current pattern space (which contains the line number), then reads the next line and appends it. So, the pattern space will contain "line-number\nOriginalline- content". Then we execute s/\n/ / to change the newline (\n) to a space.

47. Print 1st Line in MultiLine (P command)

We have seen three upper case commands so far, each of which appended to rather than replacing the content of a buffer. We will now see that upper case P and D operate in a fashion similar to their lower case equivalents, but that they also do something special related to MultiLine buffers.

As we discussed earlier the lower case p command prints the pattern space. Upper case P command also prints the pattern space, but only until it encounters a new line (\n). The following example prints all the managers names from the empnametitle.txt file

```
$ sed -n -e 'N' -e '/Manager/P' empnametitle.txt
Jason Smith
Jane Miller
```

48. Delete 1st Line in MultiLine (D command)

As we discussed earlier the lower case d command deletes the current pattern space, reads the next line from the input-file to the pattern space, aborts the rest of the sed commands and starts the loop again.

The upper case D command does not read the next line to the pattern space after deleting it, nor does it completely clear the pattern buffer (unless it only has one line). Instead, it does the following:

• Deletes part of the pattern space until it encounters new line (\n).

• Aborts the rest of the sed commands and starts command execution from the beginning on the remaining content in the pattern buffer.

Consider the following file, which has comments enclosed between @ and @ for every title. Note that this comment also spans across the lines in some cases. For example @Information Technology officer@ spans across two rows.

Create the following sample file:-

```
$ vi empnametitle-with-comment.txt
John Doe
 CEO @Chief Executive Officer@
Jason Smith
 IT Manager @Information Technology
  Officer@
  Raj Reddy
  Sysadmin @System Administrator@
  Anand Ram
  Developer @Senior
  Programmer@
  Jane Miller
  Sales Manager @Sales
  Manager@
```

Our goal is to remove these comments from this file. This can be done as shown below.

```
$ sed -e '/@/{N;/@.*@/{s/@.*@//;P;D}}' empnametitlewith-comment.txt
John Doe
CEO
Jason Smith
IT Manager
Raj Reddy
Sysadmin
Anand Ram
Developer
Jane Miller
Sales Manager
```

The above command should be executed in a single line as shown below:-

```
sed -e '/@/{N;/@.*@/{s/@.*@//;P;D}}' empnametitle-with-comment.txt
```

You can also save this in a sed script file and execute it as shown below:-

```
$ vi D-upper.sed
#!/bin/sed -f
/@/ {
 N
/@.*@/ {s/@.*@//;P;D }
 }
```

```
$ chmod u+x D-upper.sed
$ ./D-upper.sed empnametitle-with-comment.txt
```

In the above example:

• **/@/ {** - This is the outer loop. Sed looks for any line that contains @ symbol. If it finds one, it executes the rest of the logic. If not, it reads the next line. For example, let us take line 4, which is "@Information Technology" (the comment spans to multiple column and goes to line 5 also).

There is an @ symbol on line 4, so the rest of the commands are executed.

• **N** - Get the next line from the input file and append it to the pattern space. For example, this will read line 5 "Officer@", and append it to pattern space. So, pattern space will contain "@Information Technology\nOfficer@".

• **/@.*@/** - Searches whether pattern space has the pattern "@.*@", which means anything enclosed between @ and @.

The expression is true for the current pattern space, so, it goes to the next step.

• **s/@.*@//;P;D** - This substitutes the whole text "@Information Technology\nOfficer@" with nothing (basically it deletes the text). P prints the 1st portion of the line. D deletes the rest of the content of pattern space. And the logic continues from the top again.

49. Loop and Branch (b command and :label)

You can change the execution flow of the sed commands by using label and branch (b command).

• **:label** defines the label.

• **b label** branches the execution flow to the label. Sed jumps to the line marked by the label and continues executing the rest of the commands from there.

Note: You can also execute just the b command (without any label name). In this case, sed jumps to the end of the sed script file.

The following example combines the employee name and title (from the empnametitle.txt file) to a single line separated by : between the fields, and also adds a "*" in front of the employee name, when that employee's title contains the keyword "Manager".

```
$ vi label.sed
#!/bin/sed -nf
 h;n;H;x
 s/\n/:/
/Manager/!b end
 s/^/*/
 :end
   p
```

In the above example, you already know what "h;n;H;x" and "s/\n/:/" does, as we discussed those in our previous examples.

Following are the branching related lines in this file:-

• /Manager/!b end - If the lines doesn't contain the keyword "Manager", it goes to the label called "end". Please note that the name of the label can be anything you want. So, this executes "s/^/*/" (add a * in the front), only for the Managers.

• :end - This is the label.

Execute the above label.sed script:

$ chmod u+x label.sed

$ ./label.sed empnametitle.txt

John Doe:CEO

*Jason Smith:IT Manager

Raj Reddy:Sysadmin

Anand Ram:Developer

*Jane Miller:Sales Manager

50. Loop Using t command

The sed command t label branches the execution flow to the label only if the previous substitute command was successful. That is, when the previous substitution was successful, sed jumps to the line marked by the label and continues executing the rest of the commands from there, otherwise it continues normal execution flow.

The following example combines the employee name and title (from the empnametitle.txt file) to a single line separated by : between the fields, and also adds three "*" in front of the employee name, when that employee's title contains the keyword "Manager".

Note: We could've just changed the substitute command in the previous example to "s/^/***/" (instead of s/^/*/) to achieve the same result. This example is given only to explain how the sed t command works.

```
$ vi label-t.sed
#!/bin/sed -nf
h;n;H;x
s/\n/:/
:repeat
/Manager/s/^/*/
/\*\*\*/!t repeat
p
$ chmod u+x label-t.sed
$ ./label-t.sed empnametitle.txt
John Doe:CEO
***Jason Smith:IT Manager
Raj Reddy:Sysadmin
Anand Ram:Developer
***Jane Miller:Sales Manager
```

In the above example:•The following code snippet does the looping:-

```
:repeat
/Manager/s/^/*/
/\*\*\*/!t repeat
```

- **/Manager/s/**^/*/ - If it is Manager, it adds a single * in front of the line.

- **/***/!t repeat** - If the line doesn't contain three *s (represented by /***/!), and if the previous substitute command is successful by adding a single star in front of the line, sed jumps to the label called repeat (this is represented by t repeat)

- **:repeat** - This is just the label