1.) The command sed **s/Jo/Josephine/ SedLab** replaces **Jo March** with **Josephine March** as shown in the screenshots below. How it works is that the second argument after the s is the current character you want to replace, then after the slash is the new character and finally is the filename you want to use sed with. To sum up everything stated here, the syntax/arguments for performing this command is as follows **“sed s/original character/ new character/ Filename”.** As such, here is the screenshot of the output of your command.

A screenshot of a computer

Description automatically generated

2.) ***sed ’45,53d’ SedLab*** removes the last 5 lines from the SedLab file. How it works is that the 45 and 53 act as a specified range by which the command is targeting. The ‘d’ character in the command is used for removing lines. This command requires a bit of analysis of your actual file. **What you will notice in the screenshot below is how all the lines after Dorothy gale are not there.**

A screenshot of a computer

Description automatically generated

3.) **Sed -n ‘3,15p’ SedLab** prints the lines 3 to 15 from the SedLab file. The -n character allows for automatic printing and the p is used to print the specified lines. Which in this case was 3 to 15. Note that the comma is the same thing as saying and. As such, here is the screenshot of what the output should look like.

A computer screen with white text

Description automatically generated

4.) **sed ‘/CA/d’ SedLab** This command works by removing the lines where CA was present. The /pattern/d syntax in sed deletes lines matching the pattern specified. Make sure to analyze your data to see what exactly you’re trying to change before you do this**. If you look at the gaps in between the lines, that’s how you know the command worked. For example, the screenshot below shows a successful output.**

A screenshot of a computer screen

Description automatically generated

5.) **sed -n ‘/[0-9]\/[1-7]\//p’ SedLab** How it works is that the patterns matches lines containing a digit from 0 to 9 followed by a slash which is then followed by a digit between 1 and 7.

**-n** hides the extra unwanted data. Without the -n option, all the lines of the file get printed along with the pattern matches which are duplicated. What you will notice in the screenshot below is that not only are blank lines not present, only the pattern matches are present.

**/[0-9]** acts as search criteria for each month. Note that the forward slashes(/) in this command tell sed to search for the specified patterns which in this case is 0-9.

The backslashes (\) act as delimiters which tell sed that you’re searching for a literal slash (**\/**). This is needed because the date format is MM/DD/YYYY

**[1-7]** tells sed to search for dates for the first week which in this case is days 1 through 7. This search pattern signifies the day option of birthdays and the [1-7] signifies week 1.

**\//p** tells sed to again search for a literal forward slash and then the /p signifies the end of the search and tells sed to print the output.

A screenshot of a computer program

Description automatically generated

6.) **sed ‘/^Sir/s/$/\*\*\*/’ SedLab** replaces the end of the lines with 3 asterisks.

Each slash represents a different argument of the command. In other words, it’s a separator for parts of your command.

The ^ in **^Sir** tells sed to look for line(s) that starts with sir. Not just any line(s) that have sir anywhere on a given line. The **s** signifies a substitution. The $ is an ending character for a line and then the 3 asterisks are what’s being substituted for the end of the line.

A screenshot of a computer

Description automatically generated

7.) **sed ‘/Westley Pirate/c As you wish.’ SedLab.** This command searches for the lines containing Westley Pirate in SedLab and replaces the whole line only saying As you Wish.

The forward slashes act as separators as parts of your command.

The first argument is the original name which is **Westley Pirate** which tells sed that this is the character that you want to modify.

The c tells sed to replace the whole line where Westley Pirate is present.

The **As you wish** tells sed to replace with the whole line with just that. As such, the screenshot below shows the output.A computer screen with a blue line

Description automatically generated

8.) **sed ‘/Minerva McGonagall/s,[0-9]\+/ [0-9]\+/ [0-9]\+,10/04/1935,’ SedLab** is the command used.

This command is used for changing someone’s bday to October 4th, 1935. Here’s how it works:

How it works is that other than the sed command used to invoke the program itself, **‘/Minerva McGonagall/** serves as a search target which is enclosed with forward slashes which matches any line that contains Minerva McGonagall.

**/s,** is a substitution command which is an indicator that you want to replace one pattern with another.

**[0-9]\+/ [0-9]\+/ [0-9]\+,** matches a date pattern that is to be replaced.

**[0-9]** matches any single digit from 0 to 9.

**\+** matches one or more occurrences of the previous element which in this case is the digits. The forward slash is a part of the pattern for the birthdays.

**10/04/1935,’** is the replacement pattern for changing the birthday of Minerva. The commas are used as delimiters. The slashes were used as a separator for the date format and the commas were used because a new way was needed to separate the patterns.

The **‘** is used as a closing single quote for the regex that started at the beginning of the command.

A screenshot of a computer

Description automatically generated

9.) **sed ‘/^$/d’ SedLab** shows the file without blank lines. Here’s how it works.

**‘/^$/** is a regex pattern that is enclosed within slashes. The **^** indicates the beginning of a line, **$** is an indicator for the end of a line. In conclusion, these 2 characters combined with enclosed slashes represent empty lines with no characters between the beginning and end of a line.

**d** is the command for deletion when the **‘/^$/** matches a line in which case sed will delete it. Basically, the d is what deletes all the whitespace that you would otherwise see in every other screenshot before the below screenshot. Finally, you have your filename argument as always which at this point, is standard for working with sed.

A screenshot of a computer

Description automatically generated

10.) This is my script: A screenshot of a computer

Description automatically generated

**#!/bin/sed -f** is the shebang line that specifies to the interpreter that sed is used to execute the script.

Note that the **#insert title** is a mere comment.

**1li\-Great Literary Characters** is the line that inserts “Great Literary Characters” at the beginning of input text before the phone numbers. The 1 is the specified line number and the i\ is an indicator that the text following it to be inserted before the line specified.

**s/\([0-9]\{3\}\)-\([0-9]\{3\}\)-\([0-9]\{4\}\)/1+\1-\2-\3/g** Is a substitution command using regex. Patterns are to match the phone numbers in the format 000-000-0000 where # represents a single number and is replaced with 1+000-000-0000. The majority of the characters/numbers are used for grouping parts of the regex and the \1, \2 and \3 refer to their respective groups of the phone number. The g you see at the end stands for global which means it replaces all occurrences on each line.

**$a/Happily Ever after. The End** appends the text “Happily ever after. The End” at the end of input text. The $ refers to the last line and the \a indicates that text following it should be appended after the specified line.

Here is the script for the phone numbers and the title and ending character:

10b. Here is the command used for executing the script. I redirected the output to another file because the output results are not saved. The syntax is, **sed**, **-f** which is used to read a script file from the specified file other than the command line which brings me to my next point. sed.script.sed is the script file. The .sed at the end serves as a file type extension. It specifies that the script is being executed using sed. Sedlab is the file that the script is to be ran on. The > and then SedLab.Update redirects the output and script results to another file. 

10c. The screenshot below shows the script results:A screenshot of a computer

Description automatically generated

**A computer screen shot of a black screen

Description automatically generated**

**References**

<https://www.2daygeek.com/linux-remove-delete-lines-in-file-sed-command/>