**Sed**

Justice Escalante

Linux Admin

**Changing the name Jo to Josephine**



Using ***sed -n ‘s/Jo /Josephine /p’ SedLab*** in the terminal will work wonders for this instance because of its properties. In this scenario we are looking to replace Jo to Josephine, Sed takes in strings and not words so you have to be very specific on the expressions/patterns you give it. If we gave sed the pattern as ***/Jo/*** then it would look for any string with Jo in it, so a string like Jolly would also be swapped to Josephinelly if the operand for it to be ran globally was indicated in the command which would be done by adding a g at the end and not the p. The p in this command only print out the line that is being matched/swapped, I have done this in this example for the convenience of the lab.

**Deleting the last 5 lines.**





Text

Description automatically generated

Text

Description automatically generated

For this Example we first need to find how many lines are in the file before we can give ***sed*** a set to follow by when deleting lines. To do this we used ***wc SedLab*** which outputted us a value of 53 lines. Now in this file there are blank lines, so we have to account for that when deleting actual lines of text when using ***sed.*** To delete just the 5 lines at the end including the gaps all we have to do is account for 5 lines including the last one when using the d operand. So with this we would get ***sed ’49,53d’ SedLab*** as there is 53 lines in total and were only going back and deleting 5. Using the d operand you can provide it a range a shown above to delete lines, so in this case we need lines 49-53 deleted as they are the last 5 in this file. Now if were talking about 5 lines of text then we would have to use 44 like shown above in the screenshot as each line of text is followed by a gap/empty line that we have to account for as well. Above you can see Dorothy being the 6th last line being displayed and the following screenshot showing the following 5 lines that would hypothetically be deleted if you go by lines of text and not straight lines.

**Printing lines 3-15.**

Text

Description automatically generated

In this problem we strictly need lines to be printed, and only lines 3-15. To execute this properly we need to provide sed with the correct information to do so. Above you can see the command ***sed -n 3,15p SedLab*** was used to do this. Using ***-n*** to only print specific lines, we can then proceed to give sed the range in which we want to print. So in this case it would be lines 3-15 so we would have to display it as 3,15 and add a p after for proper formatting for printing in the terminal. The screenshot above shows the proper printing when this is executed.

**Deleting lines for people who live in California**

Text

Description automatically generated

Using what we have learned above, this one is relatively simple as we are using the delete operand once again. As we know California is abbreviated in this file as CA, we can search for lines with CA in them and follow up the the delete operand to remove them. The screenshot above shows the command ***sed ‘/CA/d’ SedLab*** being pushed through, in which we see blank/empty spaces left behind following the deletion of those lines.

**Printing all lines where the birthdays are in the first week of the month. Be careful of the dates for birthdays, the format is MM/DD/YY**

Text

Description automatically generated

This example works great as it sort of ties in with another example we did back in grep, where I had explained that you might need an expression for more specific stuff regarding dates. For this one I used the command ***sed -n ‘/[0-1][0-9]\/[1-7]\/[0-9][0-9]/p’*** to properly display what is seen above, In this case we have given sed a regular expression to find birthdays that are only in the first week meaning dates 1-7 of each month. In the regular expression used ‘/[0-1][0-9]\/[1-7]\/[0-9][0-9], each section has its purpose. In bold **‘/[0-1][0-9]\/**[1-7]**\/[0-9][0-9]** is the expression pattern that is responsible for finding a actual month and year, so ***[0-1][0-9]*** is what we have used as no months go above 1 equaling the 0-1 and 0-9 following as the second digit of the month ranging from 1-9 depending if its in the double digits. The most important part in bold that is key is ‘/[0-1][0-9]\/**[1-7]**\/[0-9][0-9], the [1-7] isolates the week of the birthdays till just the 7th. If this was anything like [0-9] or different the results from the search would be inaccurate for this case.

**Appending three asterisks (\*) to the end of lines starting with Sir**



For this example we are doing a simple search and append with a given parameter. In the screenshot above you can see the command ***sed -n “/Sir/ s/$/ \*\*\*/p”*** that provided the above result. To first find Sir we have to make ***sed*** search for it using ***/Sir/,*** now that we have that we need to tell the system to append 3 asterisks at the end of that line. To do so we must use ***$*** which matches the end of the line/goes to it. Since we found sir, then we need to match to the ending using ***$*** and then proceed to use s/ to substitute the ending of the match with three asterisks which would be done by ***s/$/ \*\*\*/.*** Pretty much we need the search part down first, and then we can proceed to place the substitute argument.

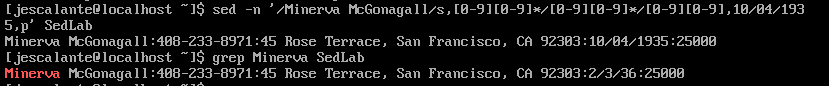
**Replacing the line containing ”Westley Pirate” with the phrase ”As you wish.” Make sure you replace the whole line not just the name**

Text

Description automatically generated

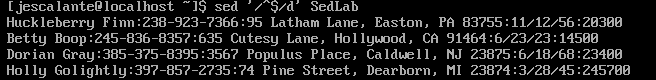
Above you can see the command ***sed ‘s/.\*Westley Pirate.\*/ As you wish/’ SedLab*** which upon execution replaces all lines with Westley Pirate with a clean slate of As you wish. Using ***s/*** once again to substitute we can achieve this by also using ***.\**** to match for the shortest sequence of chars. While using this we give it the parameter ***Westley Pirate*** to match the line for, and then we can proceed by telling the terminal what to swap it for. In this instance it would be the whole line of As you wish, which can be done by adding it following what your substituting. Above you can see the execution of the command followed by the lines which you can see were replaced fully with As you wish.

**Changing Minerva McGonagall’s birthday to 10/04/1935. Assume you don’t know Minerva’s original birthday. Use a regular expression to search for it.**



Without knowing McGonagalls birthday, we can use regular expressions with parameters to find that with sed and replace it with a new birthday. Using the command ***sed -n ‘/Minerva Mcgonagall/s,[0-9][0-9]\*/ ,[0-9][0-9]\*/ ,[0-9][0-9], 10/04/1935,p’ SedLab*** we can achieve this quite simply. First we must find the line containing the individual, which would be Minerva Mcgongall, after this we can run a regex pattern to find the piece of text/string that correlates to what we would perceive as the birthday. Using what we know, ***,[0-9][0-9]\*/ ,[0-9][0-9]\*/ ,[0-9][0-9]*** as a pattern can easily depict a common format in dates. Now with those two combined sed will now find the exact person and the text that contains the birthday, now all we have to do is replace it with the correct or given birthdate. Using s, to substitute the matched date string with the correct as follows in the prompt.

**Deleting all blank lines.**

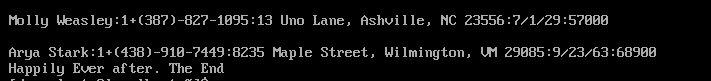


To delete all the lines using sed, this can be done by the command ***sed ‘/^$/d’ SedLab***. Quite simply using ***/d*** to delete the lines, we can give linux the operands ***^*** and ***$*** and that’s it to find the blank lines. This is because ***^*** is the operand for the start of the line and ***$*** is for the end, so if we search for the start to the end with no meat/specifics in the middle of the pattern, then it will simply look for blank from start to end which would be a empty/blank line. Execute this with the delete operand at the end and its gold!

**Writing a sed script that will (actual sed script, NOT just the commands on the command line) (a) Insert above the first line the title - Great Literary Characters -. (b) Print the contents of the file, but instead of the phone number starting with an area code, have it start with a 1+, then include the area code and number. For example, (603)123-1234 would turn into 1+(603)123-1234 (c) Append at the end of the file ”Happily Ever after. The End”**

Text

Description automatically generated



Text

Description automatically generated

For this script its kind of simple but not, think of a sed script as multiple lines being executed at once instead of one after each other or by using pipes. For this we first need to make a file with the .sed extension so that it can be run properly. Now that this is done all we have to include in the contents of the files is just the commands itself and no sed ??? etc. Onto the command portion, the first thing we need to hash out is getting “ – Great Literary Characters – “ Above the first line, to do this we will use the command ***1i\ – Great Literary Characters –.*** Using ***1i\*** we can do this as ***1i\*** inserts the text you give it above the first line. Boom done. Next were on using a regex expression and substitute command with sed to isolate the phone numbers and the give them there proper +1 with the area code. With ***s/\([0-9]\{3\}\)-\([0-9]\{3\}\)-\([0-9]\{4\}\)/1+(\1)-\2-\3/g*** as an expression it will match each set of digits that represents a phone number, and then using substitution it will receive what was caught in the first place and then give it back to you with he +1 and area code surrounding it. s/\([0-9]\{3\}\)-\([0-9]\{3\}\)-\([0-9]\{4\}\)/1+(\***1***)-\***2***-\***3***/g in bold you can see 1,2 and 3 referencing what was caught in the beginning of the expression which was later used for the substitution. Above you can see ***([0-9]\{3\}\)-*** which practically catches a digit from 0-9 followed by ***\{3\}*** which allows for that to be repeated 3 times for 3 digits per part of the phone number. This is also why at the last section you can see \{4\} as some phone number contain 4 as the last section. Lastly ***using*** ***$a*** we can add Happily Ever after. The End as the operand ***$a\*** means to append after the last line. Now to run this sed script we use it in the format of ***sed -f scriptname.sed inputfile,*** which ***-f*** specifies for a script to be ran.

**Bibliography**

[**https://www.2daygeek.com/linux-remove-delete-empty-lines-in-a-file/**](https://www.2daygeek.com/linux-remove-delete-empty-lines-in-a-file/)

[**https://www.baeldung.com/linux/insert-line-with-spaces-sed#:~:text=We%27ve%20learned%20that%20sed,new%20line%20we%27re%20inserting**](https://www.baeldung.com/linux/insert-line-with-spaces-sed#:~:text=We%27ve%20learned%20that%20sed,new%20line%20we%27re%20inserting)**.**

[**https://unix.stackexchange.com/questions/95939/how-exactly-do-i-create-a-sed-script-and-use-it-to-edit-a-file**](https://unix.stackexchange.com/questions/95939/how-exactly-do-i-create-a-sed-script-and-use-it-to-edit-a-file)

[**https://www.youtube.com/watch?v=cx-R8WpQb6A&t=351s**](https://www.youtube.com/watch?v=cx-R8WpQb6A&t=351s)

[**https://www.youtube.com/watch?v=EACe7aiGczw&t=488s**](https://www.youtube.com/watch?v=EACe7aiGczw&t=488s)