Linux Sed Commands

**Overview**

**Sed** or **Stream Editor** is a type of command in Linux that is used to modify file outputs. **Sed** can be used to modify the contents of the file without changing the file itself but rather getting an edited version of the file or you can modify the file proper with -i flag. We will simply be editing file outputs but not the file itself. A sed command is formatted by starting with **sed** then any flag then quotes with what you want to modify and how you want to modify it then after the quotes the file name or location of the file if you are not in the directory you want to edit your file.

**Question 1 Changing Jo to Josephine**

The command I used to change Jo to Josephine was **sed -n “s/Jo /Josephine /p” SedLab** as show below:



**Explanation:** I start every **sed** command with **sed** then I put the **-n** flag to suppress automatic printing without this flag it would print every line that is being processed this stops that process. A sed text substation is formatted as follows **“s/text\_to\_modify/modified\_text/p”** the **s** command states that we are substituting text then in the next dash I then input the pattern of characters I want modified which is **Jo** with a space it is important to be specific and have your pattern of characters be as unique to what you want modify so you don’t accidentally modify something you didn’t intend to modify then in the next slash I put **Josephine** since that is what I want to replace **Jo** with then the last slash before the closing quote I use the **p** command which states to only print the modified lines. After the quotes I put the name or location of the file I want to modify which is **SedLab**.

**Question 2 Deleting the Last 5 Lines**

The command I used to delete the last 5 lines from SedLab was **tac SedLab | sed “1,5 d” > rmfive && tail -5 rmfive** as shown below.

A screenshot of a computer screen

Description automatically generated

**Explanation**: Before explaining how this command works first let’s look at what last five lines of the file looks before an edited output using **tail -5 SedLab**

A black screen with white text

Description automatically generated

The edited version of this output should not be showing these lines when trying to get the bottom 5 lines again here’s how the command works now. The **tac** command can be used to reverse the order of the file this is important as whenever using the **d** to delete certain range of lines to the end is usually formatted as a range to delete like **“n, $ d”** where **n** is the starting point of the range by line numberthe **$**  symbol states that it must be found from the end of the line or if it being used in range like the current example must be to the end of the file the **d** then states to delete anything in the range basically it’s saying from **n** to the end **$** delete **d**. The issue with this is you need find the exact line number to delete from to the end to delete the last n lines. This can be done through an arithmetic expression, or we can use **tac** so we can use the start from line 1 as the starting point to how many lines we wanted deleted as the end point to delete then use **tac** again to reset the reverse to get the proper output this is formatted as follows. **tac** to reverse file then filename then a **|** then a **sed** range to go from **“1, 5”**  then a **d** to delete then a **|** then **tac** again to reverse order again so the **sed** part is saying from line 1 to 5 delete I then use the **> rmfive** to append the output to a file called **rmfive** which will be created if it does not exist I then do a **&&** to do multiple commands and the next command is **tail -5 rmfive** which uses the **tail** command to get an output from the bottom of the file by **-n** amounts or **-5** which states the last 5 files the output should now be showing different users.

**Question 3 Printing Lines 3 – 15**

The command I used to print lines 3 – 15 in my file is **sed -n “3, 15p” SedLab** as shown below.

A screenshot of a computer

Description automatically generated

**Explanation:** The **-n** after the **sed** flag stops the process of printing every line processed regardless of if it meets the criteria this stops that process. A sed command to print a range of lines is formatted as follows in the quotes **“n,mp”** the **n** is the starting line to where to print and the **m** is where to stop printing since I am using the **-n** flag I will use the **p** command which only prints lines that meet the criteria so this type of command is saying line **n** to line **m** print every line (**p**) since I want to print lines 3 to 15 I simply say **“3,15p”** which is saying from line 3 to 15 print those lines after the quotes state the file you want to modify which in my case is **SedLab**

**Question 4 Deleting Users from California**

The command to delete lines of users from California is **sed “/CA/d” SedLab > noCal”** as shown below.

A black screen with white text

Description automatically generated

**Explanation:** Before going any further in the explanation lets first see what users in the file are from California we can use **grep “CA” SedLab** for this

A screenshot of a computer

Description automatically generated

Since I am not modifying the file but getting a modified version of the output, I am appending the output of the command to the file **noCal**. I should get nothing when doing the same **grep** command for the file **noCal** but how did I do this? In the quotes I use I have **“/CA/d”** the **/** is used as delimiter separating the text to be modified from how to modify it the **/CA/** specifies the pattern of characters to look for in between the slashes the **d** states to delete any line that contains what in the slashes before it so **CA** the modified output then gets appended using the **>** to a file called **noCal** which will be created if it does not exist. Doing the same grep command I did earlier with the **noCal** file **grep “CA” noCal** shows nothing as the output deleted any user from California and that output was appended to the **noCal** file.

**Question 5 Users who’s birthday is first week**

The command I used to get users whose birthday is the first of the month is **sed -n -E ‘/:[0-9]{1,2}\/[1-7]\//p’ SedLab** as show below

A screenshot of a computer screen

Description automatically generated

**Explanation**: the **-n** flag after the **sed** stops automatically printing every line processed in the file **SedLab** regardless of if it meets the criteria requested by the command. The **-E** flag allows use to use extended regular expressions without it we’d have to format the command differently and we could not use the **{n,m}** command. In the quotes I start with a **/** used as a delimiter to separate what I want modified and how I want it modified in the next **/** each birthday is bordered by a colon so put that first as a unique identifier of the pattern of characters I want to modify I then use the **[1-9]** expression to get any digit with in the range of the bracket since expression in itself only covers 1 digit and the months can have two digits I use the **{1,2}** command to have that **[1-9]** pattern be found 1 or two times in that position of the pattern so it is saying to find **[1-9][1-9]** or **[1-9]** for each digit. In sed a **/** can be used a delimiter this can be a problem for any a pattern of characters we want to modify that have a **/** and it is important to have the pattern of characters be as unique to what you want to modify and to be able to identify it and this a problem for our pattern since birth dates in our code is formatted as **MM/DD/YY** which have a slash thankfully we could use the **\/** as a substitute for **/** which would treat **/** as a part of the pattern to look for instead of the delimiter so put a **\/** after the **[1-9]{1,2}** then put a **[1-7]** which would find a digit within the range of the bracket which in this case is the first seven days put another **\/** for the next slash in the date and then put another **/** for a delimiter and put a **p** to state to only print lines that the meet the criteria in the quotes. After the quotes put the name or location of the file to search which in this case is **SedLab**

**Question 6 Appending \*\*\* to users with Sir on their Name**

The command I used to append \*\*\* at the end of each line where the user has Sir in their name is **sed “/^Sir/s/$/\*\*\*/” SedLab > ast && grep “Sir” ast** as shown below



**Explanation**: Every **sed** command starts with **sed** then in the quotes how sed gets a modified output each **/** in the command is used as delimiter separating either what text to modify or how to modify between each **/** in the first **/** after the opening quote is what I want to find which is **Sir** I put a **^** anchor to specify that the pattern of **Sir** I am looking for should be looked for from the start of the line since each user’s name is the first thing in each line I put a **/** for how I want to modify it I put the **s** command to show I am doing substitution (sort of) I am replacing the end of the line but nothing from the contents of the line to get the end of the line I put another **/** for the next command which is **$** states to go to the end of the line so far this command is saying to find **Sir** at the start of the line using the **^**  anchor of then substitute at the end of the line using **s** and **$** perspective for that in the next **/** I put in there what I want to modify with which is **\*\*\*** I put the closing **/** then quote I then put the name of the file I want to get an edited output from which is **SedLab** I then use the **>** command to output to a file callsed **ast** which will be created if it does not exist I then use the **&&** command to do another command which is **grep “Sir” ast** which uses **grep** to find the **Sir** pattern in the file **ast** this will display lines with Sir and looking at the picture we can see the three \*\*\* at the end of the line.

**Question 7 Replacing every Westley Pirate Line with as you wish**

The command I used to replace every line with Westly Pirate with as you wish is **sed -n “/^Westley Pirate:/c\As you Wish” SedLab** as shown below.

A black screen with white text

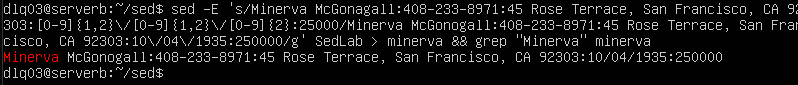
Description automatically generated

**Explanation:** the **-n** flag after the **sed** stops automatic printing which would print every processed line of the file I’m looking for regardless of if it meets the **sed** condition. The **/** is used as delimiter to separate what to modify from how to modify it the first part after the first **/** after the opening quotes shows the pattern of characters, I want find which is **Westley Pirate:** the **^** anchor to state the pattern of characters I’m looking should be started to be looked for at the start of the line the I then go to the next **/** and there I put the **c\As you Wish** which will replace the whole line with **As you Wish** the **c\** function prints the modified lines I do not need the **p** command which does this I put the closing quote then the file I want to get the modified output from which is **SedLab**

**Question 8 Changing Minerva’s Birthday**

The Command I used to change Minerva’s Birthday assuming I do not know it is

**sed - E's/Minerva McGonagall: 408-233-8971:45 Rose Terrace, San Francisco, CA 92303: [0-9]{1,2}\/[0-9]{1,2}\/ [0-9]{2}:25000/Minerva McGonagall: 408-233-8971:45 Rose Terrace, San Francisco, CA 92303:10\/04\/1935:25000/g' SedLab > minerva && grep "Minerva" minerva** as shown below



**Explanation:** Boy is this a long one but simple but I first start with the **-E** flag after the **sed** to use extended regular expressions with this flag I can use the **{n,m}** and **[0-9]** expression without putting slashes in between them. Beyond that it is simple **sed** substitution which is formatted as **“s/text to modify/modified text/command”** in the quotes I first put the **s** command which means to substitute text when it comes to **grep** and **sed** commands those come down to finding the pattern characters and it must be as unique to what we are looking for to not only making it easier to but also so you don’t modify something you did not intent to.

If you look at the picture each user is formatted by full name, phone number, address, city, state, zip code, birth date and end number this questions assume I don’t know Minerva’s Birthday but I can use the rest of the information by putting in the part of the command to show what text to modify to put the exact line as is to find Minerva until I get to the birthday. Assuming I do not know the birthday I can use a regular expression to find I and that is **[0-9]{1,2}\/[0-9]{1,2}\/ [0-9]{2}.** I use the **[0-9]** expression to get a digit within the range of the square brackets I use the **{1,2}** expression to get 1 or 2 of the **[1-9]** expression prior to get or 2 digits since birthdates in this file have or two digits for the month and day the **\/** portion of the expression is used to look for **/** as a character since that character is regularly used as a delimiter to separate portions of the command the **\/** simply makes it look for the **/** command as a character I do the same for the birth day part of the birth day and again for birth year expect I put a **{2}** instead of a **{1,2}** since the birth year will always be two digits. In the second **/** I put what I put on the previous portion but on the birth date part where I put the date I want to replace the birth date with which is **10\/04\/1935** after that I put the closing slash for the section of the code and put the **g** command to do this for every instance.

After the quotes I put the name of the file I want to modify which is **SedLab** then use the **> minerva**  to append the output to a file named **minerva** which will be created if it does not exist I use the **&&** to do multiple commands at once and find the Minerva user using a grep command on the file **minerva** using **grep “Minerva” minerva** doing so shows the birth date is changed.

**Question 9 Removing the spaces**

The command I used to remove any blank space in the **SedLab** file is as shown below.



**Explanation**: I start with **sed** then in the quotes put a **/** to separate the first portion of the code and in that **/** I put **^$** the **^** anchor specifies to look from the start of a line and the **$** anchor tells it look at the end of a line since I previously did not specify what I wanted look for from start or end it will be matching for blank from start to end then in the next **/** I put the **d** command to state to delete this command is saying from start **^** to end **$** delete nothing **d**. I then use the  **> nospace** portion of the code to append to a file named **nospace** which will be created if it does not exist I then open the file using **nano** a text editor in Linux using **nano nospace** as shown in the picture doing so shows this

A screenshot of a computer

Description automatically generated

As you can see in blank space between each line is now gone for comparison here is how the original **SedLab** file looks like

A screenshot of a computer

Description automatically generated

**Question 10 Sed Script to add header, footer and modify phone number**

The script I used to add a header, footer and a 1+ at the start of each number is **sed -E -f sedScript.sed SedLab > output** with **sedScript.sed** containing

**1i\**

**Great Literacy Characters –**

**s/[0-9]{3}-[0-9]{3}-[0-9]{4}/1+&/g**

**$a\**

**Happily Ever after. The End**

as shown below



A screen shot of a computer

Description automatically generated

**Explanation:** This command is for a sed Script which allows to do multiple commands to use a script in sed I first created this script by typing **vi sedScript.sed** where **vi** is the text editor used in linux then the name of script **sedScript** and most importantly the **.sed** at the end to make it a sedscript file. Sed Scripts have the same concepts as sed commands but they formatted a bit differently. Each portion is started by command then a **\** then below or next to it the next portion of the command the **1i\** is a command that would put the pattern below before the first line I put **1** for the first line to specify for it to be before the 1st line and below I put what I want there which is **Great Literacy Characters**

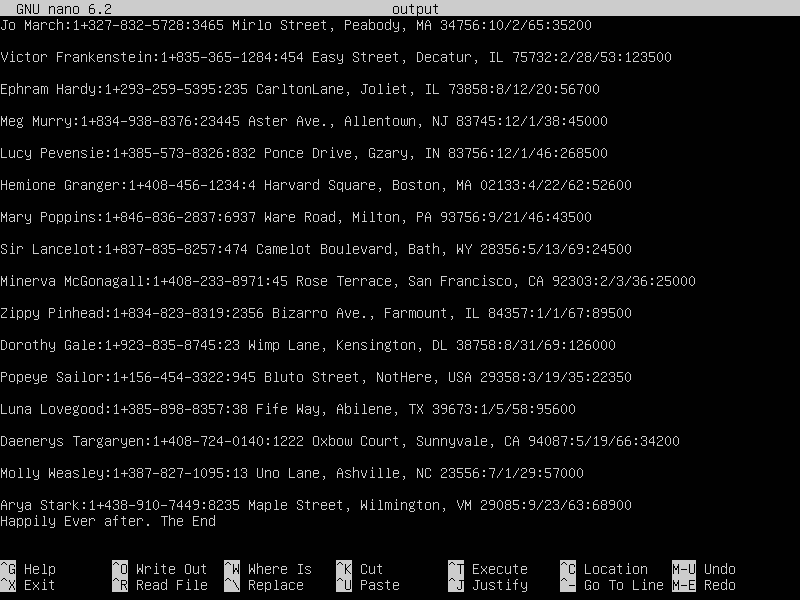
To do a substitution I put below the previous segment a portion to substitute **s/** then the the pattern of characters to find I do **[0-9]** expression to find digits in the brackets range and **{3}** to find simultaneous instances in that position of the pattern along with a dash as a string to find to identify the unique portion of the phone number part I do the same for the second portion of the number and the third except I do a **{4}** expression since the last portion has 4 digits instead of 3 In the next portion of the slashes I put a **1+** to replace the matched pattern and the **&** to represent the matched pattern so it is replace each match with a **1+** then the matched number.

Next I put **$a\** for the command and the slash to border the segment of the command where **$a** puts the pattern of characters below on the bottom of the file and below what I want to put at the bottom which is **Happily Ever After. The End**. This command in the script will do all 3 sed edits to the same edited output. Exit script by pressing **esc** then type **:wq** to save and quit and type **sed -E -f sedScript.sed SedLab > output** make sure to enable extended regular expressions with the **-E** flag otherwise the second portion of the code would not work which uses extended regular expressions to find its pattern the **-f** flag is used to execute the file. After that put the file you want to use the command on which in this case is **SedLab** then the **> output** appends the output of the command to the file called **output** the following pictures show the edited output in the **output** file using **nano** with the command **nano output**

A screenshot of a computer

Description automatically generated

(Header at top with changed phone numbers)



(Footer)

**Sources Used**

<https://www.digitalocean.com/community/tutorials/the-basics-of-using-the-sed-stream-editor-to-manipulate-text-in-linux#basic-usage>

<https://www.baeldung.com/linux/remove-last-n-lines-of-file>

<https://stackoverflow.com/questions/11245144/replace-whole-line-containing-a-string-using-sed>

<https://unix.stackexchange.com/questions/268640/make-multiple-edits-with-a-single-call-to-sed>

<https://unix.stackexchange.com/questions/99350/how-to-insert-text-before-the-first-line-of-a-file>

<https://unix.stackexchange.com/questions/20573/sed-insert-text-after-the-last-line>

<https://unix.stackexchange.com/questions/296705/using-sed-with-ampersand#:~:text=1%20Answer,replacement%20text%2C%20use%20%5C%26%20>.

<https://unix.stackexchange.com/questions/95939/how-exactly-do-i-create-a-sed-script-and-use-it-to-edit-a-file>