**Sed Command**

**What is Sed?**

* Sed is another command on Linux/Unix based systems. Sed is a stream editing tool that can be used to find and edit strings in text files using regular expressions. While Sed can manipulate files, it will not overwrite the files by default unless specified by the user.

To start editing the SedLab file for this example, power on the Ubuntu Server in VirtualBox, and transfer this file to the home directory of the server. If needed, refer to the previous File Share Documentation for assistance.

1. **Command – “sed s/Jo/Josephine/ SedLab”**

A screenshot of a computer

Description automatically generated

* The sed command along with the expression ‘s/Jo/Josephine’ is telling the system to go through every line and find any instances of the text string ‘Jo’ and substitute the first instance, *on each line*, with ‘Josephine’.

1. **Command – “sed ’48,$d’ SedLab”**

A screenshot of a computer

Description automatically generated

* To delete the last five lines using sed, we must first determine how many lines are in the file. The command ‘wc’, abbreviated for Word Count, and flag ‘-l’ will be used to count the number of lines. After running ‘wc -l SedLab’, we get a result of 53 (lines). Subtract 5 from 53 and the result is 48. Then to delete the last five lines, we can use the expression ’48,$d’ which tells the system to delete everything after line 48. As $d is an expression that refers to the last line of a given file.

1. **Command – “sed -n -e ’3,15p’ SedLab”**

A screenshot of a computer

Description automatically generated

* To print lines 3 through 15, we will need to utilize a couple of flags along with an expression. The ‘-n’ flag tells the system to stop the automatic printing of all lines, in this case we want a specific number of lines. And ‘-e’ is used to specify the ensuing expression ‘3,15p’ to run. ‘3,15p’ tells the system to print (p) lines 3 through 15.

1. **Command – “sed ‘/CA/d’ SedLab”**

A screenshot of a computer

Description automatically generated

* The Sed command is useful in finding specific strings in lines, and then deleting them from the file altogether. In this case, the objective is to remove all residents of California from this file. Because the state abbreviations are used to identify states in this file, we are looking to remove any lines containing the string ‘CA’. The expression ‘/CA/d’ tells the system to delete any line containing the string.

1. **Command – “sed -n -r ‘/[0-9]+\/[1-7]\//p’ SedLab”**

A screenshot of a computer

Description automatically generated

* This pattern is very tricky, it was difficult to track down the exact expression and pattern to generate the result I want. With the goal of finding all birthdays within the first week of the month, first use the ‘-n’ flag to avoid printing all lines. Then utilize ‘-r’ to specify the ensuing regular expression. The expression ‘/[0-9]+\/[1-7]\//p’ starts by matching any number with a 0 or 9, as all calendar months have (1-12), and then \/ matches a slash (/) as seen in the date format given. Last, to specify the first week of the year, set the expression to search for any date within the first seven days of the month [1-7] and follow it with another slash (/). /p then finishes the pattern and tells the system to print the results of the command.

1. **Command – “sed /Sir/s/$/ \*\*\*/”**

A screenshot of a computer screen

Description automatically generated

* Another complex pattern, the task here is to append three asterisks (\*\*\*) to the end of all lines that begin with the name “Sir”. Starting with Sed again, specify Sir (/Sir) as the string to identify to the system. Then using /s, same as we did in the first example, tells the system to match and substitute ‘Sir’ with the ensuing pattern. Dollar sign ($) is used to identify the end of the line and append the text asterisks (\*\*\*) to it.

1. **Command – “sed -n ‘/Westley Pirate/c\As You Wish’ SedLab”**

A black screen with white text

Description automatically generated

* To replace the entirety of the line containing the string ‘Westley Pirate’, the pattern above first identifies the string and then changes all lines containing it using ‘c\As You Wish’ to the desired new string. ‘C’ tells the system to change the whole line, different than ‘S’ that substitutes only the given string.

1. **Command – “sed ‘/Minerva/s=[0-9]\+/ [0-9]\+/ [0-9]\+/=10/04/1935=’ SedLab”**

A screenshot of a computer

Description automatically generated

* Knowing that all lines in this file begin with a first name, we can begin the pattern with this to identify birthday to change. Stating “/Minerva/s=” tells the system to find the line beginning with “Minerva” and substitute it “/s” with the pattern specified. Note this time I am putting an equal’s sign (=) after the substitution to call out the date I want to substitute later in the expression. Same concept as before with the date format, use [0-9] to identify any number with a 0 or 9, like all months have. Then using backslash and plus sign (+\) tells the system “And” or “followed by”, a forward slash (/) and another number [0-9], making it so we end up with the standard date format (MM/DD/YYYY). Last, in between two equal’s signs (=) is the new birthdate we want to assign to Minerva. Because we identified “=” earlier in the expression, the system can now identify the new birthday / string to substitute.

1. **Command – “sed ‘/^$/d’ SedLab”**

A screenshot of a computer

Description automatically generated

* To delete all empty lines from a file, start the pattern with “^” to tell the system to start at the beginning of a line, and use “$” to specify the end of the line. With no additional text between this, it identifies the system to look for lines with no text. And finishing the pattern with a delete command (d) we can remove all blank lines from the file.

1. **Command – “sed -f SedScript.sed SedLab > Lab3**

**Script:**

**1i\Great Literary Characters**

**s/\([0-9]\{3\}\)-\([0-9]\{3\}\)-\([0-9]\{4\}\)/1+(\1)-\2-\3/g**

**$ a\Happily Ever after. The End**

A screenshot of a computer screen

Description automatically generated

Use a text editor (in this case I used Notepad++) to start building the script. When finished, save the text file with the extension “.sed”. Then upload it to the Ubuntu server using the file share documentation. The first expression ‘1i\’ states to insert the ensuing string “Great Literary Characters” before the first line on the specified file. The second part of the script is the most complex due to the pattern needed to add +1 to every area code. The pattern begins by searching for a phone number formatted like it is in the text file, example – 123-456-7899. ([0-9]\{3\}\)- is the pattern to search for any 3 digit number followed by a hyphen. Use the same pattern for the second 3 digit number, and then change the last one to ([0-9]\{4\}\)- to search for the last 4 digits of the phone number. The last portion of the expression is how we add +1 to the beginning of the area code. “/1+(\1)-\2-\3/” is telling the system to add +1 and parentheses (\1) around the area code – I chose to add this for formatting purposes. In this sequence, (\1)-\2-\3/g, adds parentheses to the area code, as the ‘1’ in this portion refers to the first digits detected as part of the expression. \2- refers to the second portion of numbers followed by a hyphen (-). And last, \3/g refers to the remaining 4 digits of the phone numbers, along with /g (meaning global) which then applies the substitution to all phone numbers in the file SedLab. At the end of the command, using > Lab3 sends the results to a new file I created called Lab3. I did this to test and confirm the script works as intended.