Very often in cyber security, analyst has to perform so called “non-interactive” stream-oriented editing. Usually this comes from necessity to process a large ASCII files that regular editor would be able to perform at all and resulted in GUI handing of the program. Even so efficient command line Linux based editor as “vim” would failed to do this job. Sometime simple well-known tools as “cut”, “grep”, “tr” as sufficient but in other case non-interative editors as “sed” and “awk” must be utilized.

Out of the two, sed is more minimalistic and it is not a bad approach of using appropriate tool for the appropriate job and avoid overkill. We’ll concentrate on it in this part1. And part2 will be dedicated to more advanced editor called awk that particularly appropriate for parsing “zeek” output files.

*Substitution*

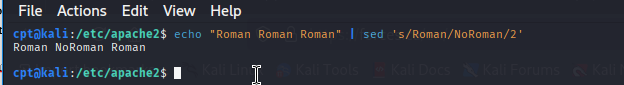
Syntax:

**[*address*]s/*pattern*/*replacement*/*flags***

where the *flags* that modify the substitution are:

**n** A number (1 to 512) indicating that a replacement should be made for only

the *n* th occurrence of the *patter n* (IN THE LINE)



**g** Make changes globally on all occurrences in the pattern space. Normally only

the first occurrence is replaced.

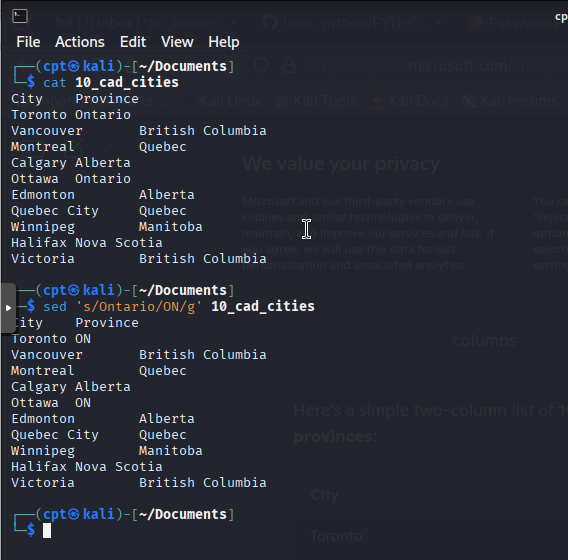
**p** Print the contents of the pattern space.

**w** *file*

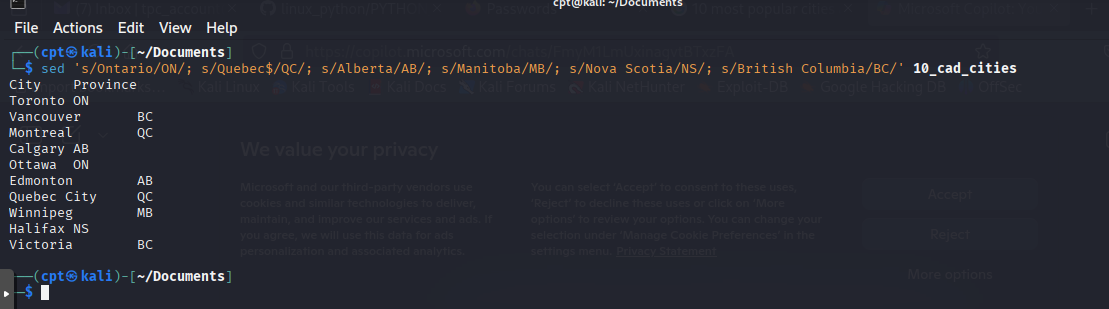
Write the contents of the pattern space to *file*.

The substitute command is applied to the lines matching the address. If no address is specified, it is applied to all lines that match the patter n, a regular expression.

The simplest example1 of using sed is what one word/pattern must be found and replaced on another. In the example below we replace “Ontario” to “ON”.



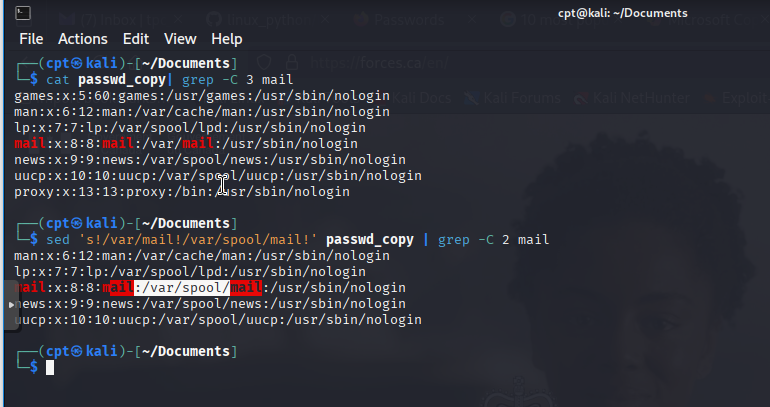
If there is a need to perform a few replacement on the same document, it’s also not an issue.



(Note: try to understand why for Quebec province replacement I had to use character $).

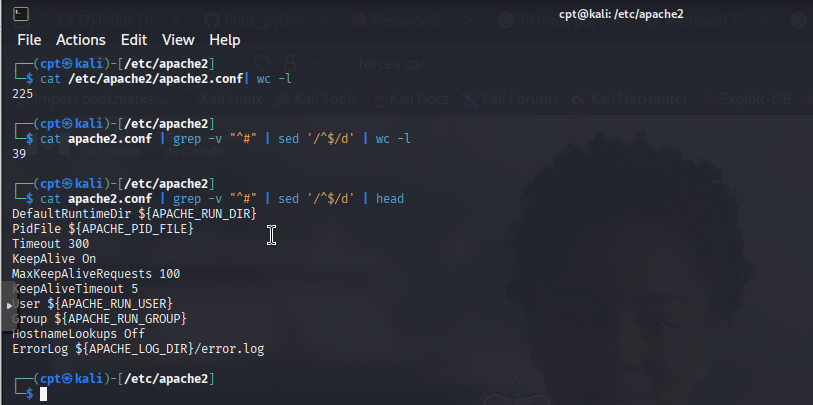
Unlike addresses, which requir e a slash (/) as a delimiter, the regular expression can be delimited by any character except a newline. Thus, if the pattern contained slashes, you could choose another character, such as an exclamation mark, as the delimiter. Below is the command that changes path for mail spool directory in the copy of passwd file.

**s!/usr/mail!/usr/spool/mail!**



*Delete*

One of the classic and often performed operation done in linux command line is deleting empty lines in configuration files. Here how this can be easely done with sed:



*Append, Insert, and Change*

The append (a), insert (i), and change (c) commands provide editing functions that are commonly performed with an interactive editor, such as vi. You may find it strange to use these same commands to “enter” text using a noninteractive editor. The syntax of these commands is unusual for sed because they must be specified over multiple lines. The syntax follows:

**append [*line-address*]a\**

***text***

**inser t [*line-address*]i\**

***text***

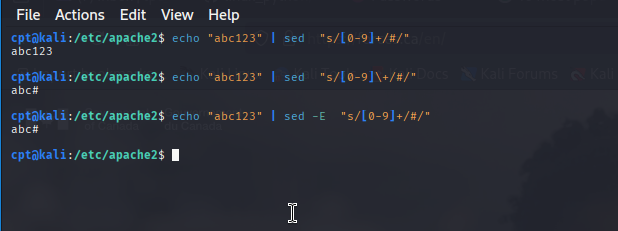
**change [*address*]c\**

***text***

The insert command places the supplied text before the current line in the pattern space.

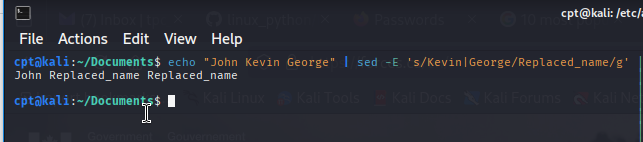


In **sed**, the -E option enables **extended regular expressions (ERE)**, allowing more advanced pattern matching compared to basic regular expressions (BRE). Here’s a comparison:



*Alternative Operations*

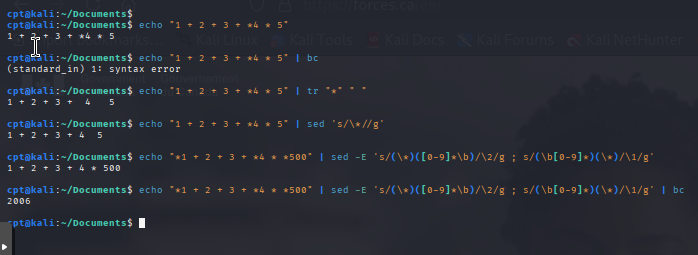
The vertical bar (|) metacharacter, part of the extended set of metacharacters,allows you to specify a union of regular expressions. A line will match the pattern if it matches one of the regular expressions.



*Grouping Operations*

Parentheses, ( ), are used to group regular expressions and establish precedence. They are part of the extended set of metacharacters.

Let’s image we have a large stream of data (math expression in our case) that was corrupted randomly by character “\*”. We must get rid of them. The task is a bit complicated by the fact that in the original stream there were legit “\*” character playing role multiplication operation. This complication can be easely solved by grouping. Essentially we are grouping character “\*” and any numbers in close adjacent to it as 2 different group and perform replace operation by displaying “numbers” group only. Marking groups is done with backspaces “\1” – group one, “\2” – group 2.



Another quick example. Let’s say we must found a word that is following another word (Canada in our case) and double it (without even knowing the exact word that is following word “Canada”).

