Transforming Text with SED

SED is a **stream editor**. It takes an input and produces an output from the command line. The reason it is called a **stream** editor is because it streams through the input, line by line. Each line gets sent to the **pattern space**, which is a special place in memory allocated by **SED**. Than some operations are done on that line and it gets sent to the output. There is also another **SED** space, which is called **hold buffer**. This buffer acts like a special variable, the way in which you can store some lines, before they get sent to the ouput. In the video we used them with the **g** and **h** operations to print out all of the quotes from Nikola Tesla.

```
$ sed -n 's/hello/hi/p' file
```

This line will take in all of the lines from a *file* and replace the word *hello* with *hi* for each instance of that word. This replacement operation will be executed on each line while the line is inside of the **pattern space**. The **s** is the replacement operation. **p** operation is a part of that operation, because it is appended after the last slash, without the semicolon. The **p** will print the lines which contain the word *hello*.
n option will prevent lines to be sent from the **pattern space** to the output. Only the lines printed with **p** will explicitly be sent to the output stream.

```
$ sed -n 's/hello/hi/; p' file
```

This will also replace the words, but it will print out all of the lines from a file, because the **p** command is separated with the semicolon. You can separate these operations with semicolon, or you can put each one as an argument to

the **e** option, e.g. **sed -n -e 's/hello/hi/' -e 'p' file**.

If you want to get deeper into **SED**, research the **SED labels**. You can create loops with them and even some logical branching by adding the conditional labels with the **t** operation. So if the **SED** has loops, conditions and hold buffer for the variables, you know that we can do a lot of interesting stuff with it, because it's Turing complete.