How to Find and Replace Text in a File

Last modified: March 11, 2022

by baeldung (https://www.baeldung.com/linux/author/baeldung)

File Editing (https://www.baeldung.com/linux/category/files/editing)

Files (https://www.baeldung.com/linux/category/files)

awk (https://www.baeldung.com/linux/tag/awk)

sed (https://www.baeldung.com/linux/tag/sed)

In this tutorial, **we're going to take a look at how we can harness the power of built-in Linux commands to search for and replace a string of characters quickly and easily**. This is a very handy technique whenever we need to update all occurrences of a particular string in a large number of files.

For example, one typical scenario could be when we want to update the copyright notice in a collection of static HTML files.

2. Search and Replace With *sed*

The first command that we're going to look at is *sed (https://linux.die.net/man/1/sed)*, **a powerful stream editor that is useful for performing basic transformations on an input stream**. Take a look at our Guide to Stream Redirections in Linux (/linux/stream-redirections) for a refresher on what streams are.

Using *sed*, we're able to quickly and easily find and replace a set of characters within a file.

Let's begin by creating a test file to use in our examples. We'll use a here document (https://www.tldp.org/LDP/abs/html/here-docs.html) to create this test file quickly:

```
$ cat <<-EOF > test.txt
This is a sample file created in 2019 to demonstrate character
substitution.
It will be used in 2019.
Linux has a wide array of tools for us to use to achieve this.
2019 is a leap year.
EOF
```

Let's confirm that we created the test file successfully:

```
$ cat test.txt
This is a sample file created in 2019 to demonstrate character
substitution.
It will be used in 2019.
Linux has a wide array of tools for us to use to achieve this.
2019 is a leap year.
```

Now everything is as it should be, so let's begin using *sed*.

2.1. Find and Replace the First Occurrence

We're going to use *sed* to find and replace the first occurrence of a term. In our case, we'll update the year to the current year. Let's see the syntax to accomplish this:

```
$ sed -i 's/{OLD_TERM}/{NEW_TERM}/' {file}
```

Now let's apply this command to our example:

```
$ sed -i 's/2019/2020/' test.txt
```

And now, we print the contents of our file to confirm that we only replaced the first occurrence of "2019":

```
$ cat test.txt
This is a sample file created in 2020 to demonstrate character
substitution.
It will be used in 2019.
Linux has a wide array of tools for us to use to achieve this.
2019 is a leap year.
```

Let's break down our statement to get a deeper understanding of how this works. First, we pass the *-i* option to instruct *sed* to make the changes inside our *test.txt* file. By default, *sed* prints the changes to the terminal. Then the regular expression specifies what we want to change :

- We start the expression with the letter **"s"** so that *sed* knows we're performing a substitution
- Next up, we have the string "2019" which is the value for the *OLD_TERM* placeholder. This is the section of text that we want to replace
- Following that is the value of the *NEW_TERM* placeholder and that's "2020"
- We separated these parameters using forward slashes *(/)*

As we can see, *sed* is a quick and easy way to perform search and replace operations.

2.2. Find and Replace All Occurrences

By default, *sed* only replaces the first occurrence that it finds. We

can easily override this behavior to replace all instances within a file. Let's see the expression that we can use to achieve this:

```
$ sed -i 's/{OLD_TERM}/{NEW_TERM}/g' {file}
```

We've added "g" to the end of our search expression. **This instructs *sed* to replace all occurrences globally**.

So let's apply this to our text file:

```
$ sed -i 's/2019/2020/g' test.txt
```

We can also add more files to our command to replace all occurrences globally in multiple files:

```
$ sed -i 's/2019/2020/g' test.txt test2.txt test3.txt
```

Now let's print out the contents of the *test.txt* again. This time we've replaced all occurrences of "2019" with "2020":

```
$ cat test.txt
This is a sample file created in 2020 to demonstrate character
substitution.
It will be used in 2020.
Linux has a wide array of tools for us to use to achieve this.
2020 is a leap year.
```

2.3. Find and Replace All Occurences Across Multiple Files Using *find*

The Linux *find (https://man7.org/linux/man-pages/man1/find.1.html)* command is one the most important and commonly used command-line utility in Unix-based systems. We can use it to search and locate a list of files or directories based on the conditions we specify. Let's combine the *sed* and *find* commands to search and replace occurrences across multiple files.

```
$ find . -name *.txt -exec sed -i 's/2020/2070/g' {} \;
```

This statement finds and replaces all instances of "2020" to "2070" across all files with a .txt extension

Let's break down this statement:

- We used **.** to target the current working directory
- *-name* argument lets us restrict our results to files that match the given pattern, in our case it's **.txt* which gets all text files
- We used *-exec* in conjunction with *find* to execute a command after matching files are found
- We used *{}* to temporarily store the matching files for execution with *sed* command
- *\;* at the end of our statement marks the end of the *sed* command

3. Search and Replace With *awk*

In this section, we'll take a look at *awk (https://linux.die.net/man/1/awk)*. A powerful scripting language that is designed for text processing and is often used for data extraction and reporting purposes. As is common with other Linux utilities, *awk* can perform operations on both streams and files.

awk has two functions; *sub* and *gsub* that we can use to perform substitutions.

sub and *gsub* are mostly identical for the most part, but ***sub* will only replace the first occurrence of a string. On the other hand,**

***gsub* will replace all occurrences.**

Let's take a closer look at how we can make substitutions using *awk.*

3.1. Using *awk* With *sub*

Let's look at how *awk* performs find-and-replace operations on our *test.txt* sample file:

```
$ awk '{sub(/{OLD_TERM}/,{NEW_TERM}); print}' {file}
```

In this statement, we're invoking *awk* and sending in a list of tasks for *awk* to perform.

Let's break this down:

- The first task in our list is a substitute, which we write as *sub*
- Sub directs *awk* to find all occurrences of *OLD_TERM* and replace them with *NEW_TERM*
- Our next instruction directs *awk* to print the output to the standard output stream; the console
- Finally, we have the name of the file that *awk* will be working on

A notable difference with *sed* is that *awk* will not perform an in-place substitution; meaning that the updates will not be made inside the file.

We'll tackle that shortly but for now, let's see this in action:

```
$ awk '{sub(/2019/,2020); print}' test.txt
```

Our resulting output confirms that the replacement has been carried out correctly:

```
This is a sample file created in 2020 to demonstrate character
substitution.
It will be used in 2020.
Linux has a wide array of tools for us to use to achieve this.
```

As expected, *awk* has replaced all instances of “2019” with “2020”.

However, our output is directed to the console instead of updating our original file. We have a trick for taking care of that.

We'll use our knowledge of streams and stream redirection to update our command so that changes made are saved to a new specified file:

```
$ awk '{sub(/2019/,2020); print . "text.txt" }' > test.txt
```

This time nothing is printed to the console but let's dump the contents of *test.txt* to see what's happened:

```
$ cat test.txt
This is a sample file created in 2020 to demonstrate character
substitution.
It will be used in 2020.
Linux has a wide array of tools for us to use to achieve this.
```

Here we can see that the output of the *awk* command was correctly redirected to our input file thereby updating it for us.

3.2. Using *awk* With *gsub*

Let's look at the syntax:

```
$ awk '{gsub(/{OLD_TERM}/,{NEW_TERM}); print}' {file}
```

***gsub* stands for global substitution. Therefore, we can use it to replace all occurrences of a string or regex with a given string**. Let's look at how *awk* performs find-and-replace operations on our sample file using the *gsub* command:

```
$ awk '{gsub(/i/,"a"); print}' test.txt
```

Here is our resulting output:

```
Thas as a sample fale created an 2020 to demonstrate character
substatutaon.
It wall be used an 2020.
Lanux has a wade array of tools for us to use to achaeve thas.
```

As expected *gsub* replaced all instances of "i" with "a".

However this only prints the result on our terminal, to save the changes we could modify our command like this:

```
$ awk '{gsub(/i/,"a"); print}' test.txt > test2.txt
```

The modification at the end of our command instructs *awk* to create a new file called test2.txt and save the changes to it.

4. Conclusion

In this tutorial, we looked at how we could use two of the most common Linux utilities to find and replace a string of characters in a single file or a set of files without manually having to edit each individual file.

These utilities are extremely powerful and come in handy for a variety of day-to-day tasks whilst working on the Linux command line. Be sure to have a look at the documentation of these commands to learn more about them.

If you have a few years of experience in the Linux ecosystem, and you're interested in sharing that experience with the community, have a look at our **Contribution Guidelines** (/linux/contribution-guidelines).

ADMINISTRATION (/LINUX/CATEGORY/ADMINISTRATION)
FILES (/LINUX/CATEGORY/FILES)

SERIES

LINUX FILES (/LINUX/LINUX-FILES-SERIES)
LINUX SCRIPTING (/LINUX/LINUX-SCRIPTING-SERIES)

ABOUT

ABOUT BAELDUNG (/ABOUT)
THE FULL ARCHIVE (HTTPS://WWW.BAELDUNG.COM/LINUX/FULL_ARCHIVE)
WRITE FOR BAELDUNG (/LINUX/CONTRIBUTION-GUIDELINES)
EDITORS (HTTPS://WWW.BAELDUNG.COM/EDITORS)

TERMS OF SERVICE (HTTPS://WWW.BAELDUNG.COM/TERMS-OF-SERVICE)
PRIVACY POLICY (HTTPS://WWW.BAELDUNG.COM/PRIVACY-POLICY)
COMPANY INFO (HTTPS://WWW.BAELDUNG.COM/BAELDUNG-COMPANY-INFO)
CONTACT (/CONTACT)