Jul 6, 2020 • 11 min read

Getting Started With SED Command [Beginner's Guide]

 Sylvain Leroux

Table of Contents

Sed is part of the Unix standard toolbox since the end of the 60s. As any text editor, it will help you to modify text files. However, contrary to the text editors you may have already used, this is a non-interactive one.

That means you specify ahead of time the transformations you want to apply to a file, and then the tool can apply those transformations unsupervised.

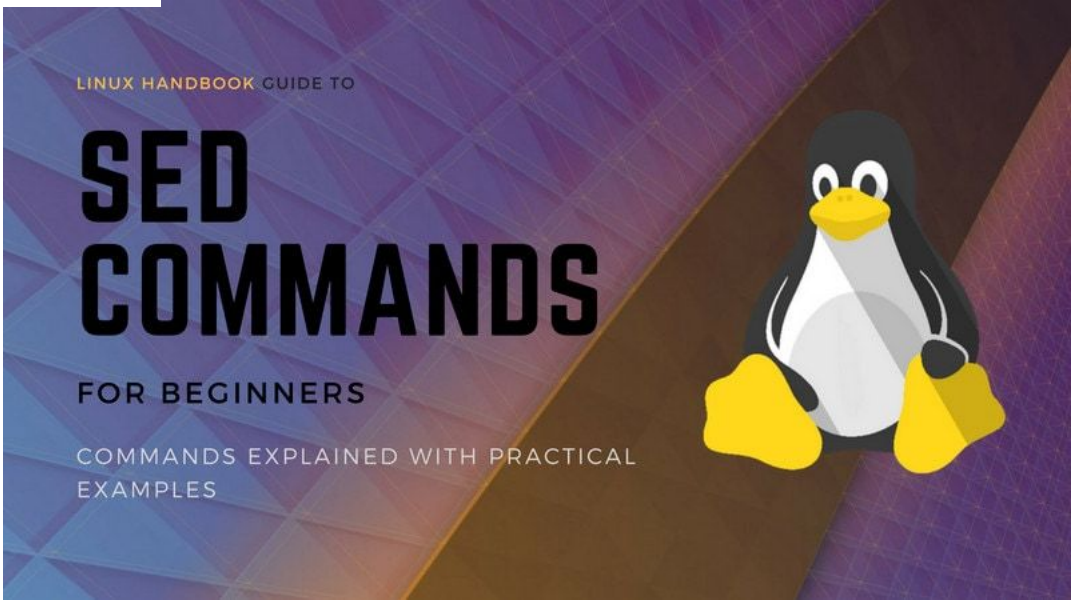
The best description of the tool's design goals comes from Lee E. McMahon, the core developer of the original implementation in his original sed paper:

Sed is a non-interactive context editor that runs on the UNIX operating system. Sed is designed to be especially useful in three cases:

1. To edit files too large for comfortable inter- active editing;
2. To edit any size file when the sequence of editing commands is too complicated to be comfortably typed in interactive mode.
3. To perform multiple `global' editing functions efficiently in one pass through the input.

The goal designs (1) and (3) are probably less relevant with our modern hardware, but the second one remains valid. As a personal addition, I would say sed is particularly well suited for repetitive tasks like when you want to apply the same transformation to a set of files.

Learn basic SED commands with these examples

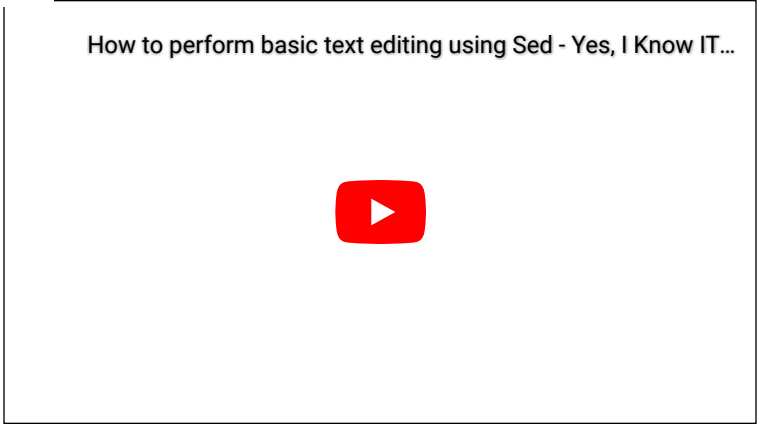


To give you a taste of the power behind sed, I will consider the case of a developer that needs to add a license header on top of each of the source files in her project:

```
linux@handbook:~$ head MIT.LICENSE *.sh
==> MIT.LICENSE <==
-----8<----------------------------------------------------------------
Copyright <YEAR> <COPYRIGHT HOLDER>

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
"Software"), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

==> script1.sh <==
#!/bin/bash

echo Hello, I\'m the first script

==> script2.sh <==
#!/bin/bash

cat << EOF
Hello, I'm the second script
EOF
```

Not only I would like to see the license file on top of each shell script, but I also would like the year and copyright placeholder to be replaced by their actual value. That will be our first use case.

Note: if you want to practice by yourself, you can download the sample files from my website. You may also want to take a look at the video completing this article:



1. Replacing text in SED

In my license file, I would like to replace the <YEAR> and <COPYRIGHT HOLDER> placeholders by their actual value.

This is a job perfectly suited for the sed *substitution* command. Probably the most useful of all the sed commands:

```
linux@handbook:~$ sed -e 's/<YEAR>/2018/' MIT.LICENSE  | head -5
-----8<---------------------------------------------------------------
Copyright 2018 <COPYRIGHT HOLDER>

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
```

Using a pipe (`|`), I forwarded the output of the sed command to the `head` tool to display only the first five lines here. However, for our today's specific topic, the most interesting part is the `s/<YEAR>/2018/` expression.

Sed works by processing the input file one line at a time. On each line, the substitute (`s`) command will replace the first occurrence of the text between the first two slashes (`/<YEAR>/`) by the text between the last two ones (`/2018/`). Think of that like the search-replace feature you have in a GUI text editor.

Worth mentioning here, the original MIT.LICENSE file was not modified. I let you check that by yourself using the following command:

```
head -5 MIT.LICENSE
```

2. Replacing text... again

Great: we have replaced the year placeholder. But there is a second one to replace. If you understood the previous example, you could probably imagine a second sed expression like this one:

```
's/<COPYRIGHT HOLDER>/Sylvain Leroux/'
```

But where to place that? Well, you have several choices. The most obvious if you are already familiar with the concept of redirection is to pipe the output of our first sed command to a second instance of sed:

```
linux@handbook:~$ sed -e 's/<YEAR>/2018/' MIT.LICENSE  |
    sed -e 's/<COPYRIGHT HOLDER>/Sylvain Leroux/' |
    head -5
----8<---------------------------------------------------------------
Copyright 2018 Sylvain Leroux

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
```

But we can do better. Since the `-e` option introduces a sed expression, we can use several of them as part of the same sed invocation, and the result will be the same:

```
# Pay special attention to the \ at the end of the lines
# specifying the *same* command continues on the
# next line:
sh$ sed -e 's/<YEAR>/2018/'  \
                -e 's/<COPYRIGHT HOLDER>/Sylvain Leroux/' \
                MIT.LICENSE |
        head -5
```

Finally, you can also specify several commands in the same sed expression by separating them with a new line. This is particularly useful when you start writing more complex sed programs:

```
# Pay special attention to the single-quotes and
# backslash placement:
sh$ sed -e 's/<YEAR>/2018/
            s/<COPYRIGHT HOLDER>/Sylvain Leroux/' \
        MIT.LICENSE |
    head -5
```

3. Inserting text

Now we have replaced the placeholders by their actual value. But we still have some work to do before being able to insert that license file into the project files. Those later being shell scripts, each line of the license must start with an octothorp (`#`) for the shell to understand it should not try to interpret those lines.

For that, we will use the substitution command again. Something I did not mention previously is, contrary to most search-replace features of GUI editors, the search pattern is not necessarily the literal string to search for. In fact, this is a regular expression (regex). That means, in addition to plain characters that will match verbatim, you can use characters that will have a special meaning. For example, the caret (`^`) represents the start of the line, the dollar sign (`$`) the end of the line, or, as the last example, the dot-star (`.*`) means any sequence of 0, 1 or several characters. There are plenty of other such metacharacters, but for now, this is more than enough.

So to *insert* some text at the start of a line, an option is to *substitute* the start of the line by that text:

```
linux@handbook:~$ sed -e 's/<YEAR>/2018/'  \
        -e 's/<COPYRIGHT HOLDER>/Sylvain Leroux/' \
        -e 's/^/# /' \
        MIT.LICENSE | head -5
# -----8<----------------------------------------------------------------
# Copyright 2018 Sylvain Leroux
#
# Permission is hereby granted, free of charge, to any person obtaining
# a copy of this software and associated documentation files (the
```

4. Clearing selected lines

The substitution command in sed is so versatile that you can express most of the text transformations using it. For example, to remove the dashed lines on top and bottom of the license text, I could write that:

```
linux@handbook:~$ sed -e 's/<YEAR>/2018/'  \
        -e 's/<COPYRIGHT HOLDER>/Sylvain Leroux/' \
        -e 's/^/# /' \
        -e 's/^.*----.*$//' \
        MIT.LICENSE | head -5

# Copyright 2018 Sylvain Leroux
#
# Permission is hereby granted, free of charge, to any person obtaining
# a copy of this software and associated documentation files (the
```

That later substitution has replaced with the empty string all text:

Symbol	Description
^	Starting at the start of the line
.*	Followed by any sequence of 0, 1 or several characters
----	Followed by 4 hyphens
.*	Followed by any sequence of 0, 1 or several characters
$	Followed by the end of the line

In short, this will replace the whole line by the empty string *if* it contains four dashes in a row. But the empty line itself remains in the output and will appear as a blank line.

Depending on your exact needs and tastes, you may also want to consider the alternative solution below. I let you examine that in detail to pinpoint the changes in the command and identify by yourself what were the consequences on the result:

```
linux@handbook:~$ sed -e 's/<YEAR>/2018/'  \
        -e 's/<COPYRIGHT HOLDER>/Sylvain Leroux/' \
        -e 's/^.*----.*$//' \
        -e 's/^/# /' \
        MIT.LICENSE | head -5
```

If you find the regular expression used to clear the line a little bit too complex, we may also take benefit of another sed feature. Almost all commands can take an optional address before the command name. If present, it will limit the scope of the command to the lines *matching* that address:

```
linux@handbook:~$ sed -e 's/<YEAR>/2018/'  \
        -e 's/<COPYRIGHT HOLDER>/Sylvain Leroux/' \
        -e 's/^/# /' \
        -e '/----/s/^.*$//' \
        MIT.LICENSE | head -5
```

Now the latest substitution command will only be applied to lines matching (i.e., “containing”) four dashes in a row. And for each matching line, it will replace everything (`.*`) between the start (`^`) and end (`$`) of the line by the empty string (`//`)

5. Removing selected lines

In the previous section, we tweaked the substitution command to clear some lines of text. But the empty lines remained present. Sometimes this is desirable. Sometimes it is not. In that latter case, you might want to investigate the *delete* command to remove entire lines from the output:

```
# Below, the redirection '> LICENSE' is used to store
# the result of the sed command into the newly
# created LICENSE file:
linux@handbook:~$ sed -e 's/<YEAR>/2018/'  \
        -e 's/<COPYRIGHT HOLDER>/Sylvain Leroux/' \
        -e 's/^/# /' \
        -e '/----/d' \
        MIT.LICENSE > LICENSE
linux@handbook:~$ head -5 LICENSE
# Copyright 2018 Sylvain Leroux
#
# Permission is hereby granted, free of charge, to any person obtaining
# a copy of this software and associated documentation files (the
# "Software"), to deal in the Software without restriction, including
```

The `d` is the *delete* command name. Just like the `s` was the *substitution* command name. Here, we specified an address before the command so only matching lines will be removed (without any address, the `d` command would have deleted every line of the file)

6. Convert to uppercase

Until now, we focussed mostly on the top of the license file. But indeed there are some changes I would like to perform a little bit further into the documents. Let’s see first what I’m talking about:

```
linux@handbook:~$ sed -ne '/The above/,$p' LICENSE
# The above copyright notice and this permission notice shall be
# included in all copies or substantial portions of the Software.
#
# The software is provided "as is", without warranty of any kind,
```

```
# express or implied, including but not limited to the warranties of
# merchantability, fitness for a particular purpose and noninfringement.
# In no event shall the authors or copyright holders be liable for any
# claim, damages or other liability, whether in an action of contract,
# tort or otherwise, arising from, out of or in connection with the
# software or the use or other dealings in the software.
```

In the above command, using the -n option I disabled automatic printing of the pattern space. That means sed will no longer print anything on the output unless I explicitly ask it to do so. This is exactly what I do using the print (p) command. Notice instead of using a single address before the p command, I used a *range* to display the text between the line containing the text "The above" and the end of the document ($).

The print command can be useful when you need to extract some parts of a file. However, for today, I just wanted to display the last two paragraphs to explain what I need now: as it is a tradition with license files, I would like to cover myself by making clear the software is provided "as is." So I would like to put in emphasis the last paragraph (starting with "The software") by rewriting it il all uppercase.

In the replacement part of a substitution command, a & is replaced by the text matching the search pattern. Using the \U GNU extension, we can change the case of the replacement string:

```
linux@handbook:~$ sed -i -e '/The software/,$s/.*/\U&/' LICENSE
linux@handbook:~$ cat LICENSE
```

In plain text `s/.*/\U&/` means "replace any text (`.*`)by the uppercase (`\U`) version of itself (`&`). I let you verify by yourself, the last paragraph should be now written in all uppercase. By the way, you may have noticed because of the `-i` flag, the changes were applied directly to the LICENSE file.

We will see that in more details in the next section. In the meantime, I let you practice and modify those commands at your will. Once you have a license file that is corresponding to your taste, it will be time to see how to include it before each source file of the project.

7. Inserting a text file

If you are expecting some complex command here, you will be disappointed: inserting a file into another one is pretty straightforward:

```
sed -i -e '1r LICENSE' script1.sh
cat script1.sh
```

Two things to see here:

1. the `r LICENSE` expression is the command to *read* and insert an external file into the file currently being processed. It is prefixed here with the number `1` which is an address matching only the line 1 of the input file.
2. the `-i` option allows changing a file *in place*. That means sed will create a temporary file behind the scene to store its output there, and, once the processing has completed, it will replace the original file with the modified one.

An interesting side effect of the '-i' option is you can specify several file names on the command line, and sed will apply the same transformations to each of them *independently*:

```
sed -i -e '1r LICENSE' *.sh
```

8. Back to the future

As our last example of sed command, let's imagine few years have passed and we are now the 1st of January 2024. The copyright notice of all files must be updated. There are several use cases, depending on when the project files were created. So, our Copyright notices should follow one of those two formats:

Current copyright	Description
Copyright 2023	For files created last year
Copyright 2018-2023	For files created before last year

We may capture those two use cases at once using an extended (-E) regular expression. The only "extended" things we will really use here are the parenthesis:

```
sed -i -Ee 's/Copyright (....)(-....)?/Copyright \1-2024/' *.sh
```

I encourage you to manually modify the copyright notice in the *.sh files and then run the command above in different use cases to see how it works.

It might eventually help your understanding though if I say, in the search pattern: Copyright:: is a literal text that will match verbatim; (... .):: defines a capturing group matching four arbitrary characters. Hopefully the four digits of a year; (-... .)?:: defines a capturing group matching a dash followed by four arbitrary characters. The question mark at the end indicates that group is optional. It may, or may not, be present in the input line.

In the replacement string: Copyright:: is a literal text that will be copied verbatim; \1:: is the content of the first capturing group -2024:: is a literal text that will be copied verbatim.

If you took the time to check the command by yourself, it should confirm if I apply those rules to the use cases described in the previous table, I will obtain something like that:

Matching text	\1	\2	Replacement string
Copyright 2023	2023		Copyright 2023-2024
Copyright 2018-2023	2018	-2023	Copyright 2018-2024

To conclude our SED guide

We only have scratched the surface here. The `sed` tool is much more powerful than that. However, even if we have only seen four commands (`s` , `p` , `d` , and `i`) and a

few basic regular expression constructs (`^`, `$`, `.`, `?` and `.*`), you already have enough knowledge to solve *many* days to day problems.

Since I like ending a tutorial with a little challenge, here is what I propose to you: if you have downloaded the support material, you will find in the project's directory a file named `hello.c`. This is the source file of a basic C program:

```
linux@handbook:~$ ls
hello.c  MIT.LICENSE  script1.sh  script2.sh
linux@handbook:~$ gcc hello.c -o hello
linux@handbook:~$ ./hello sylvain
Hello sylvain
linux@handbook:~$ cat hello.c
```

There are already some comments in the source file. By using them as examples of the comment syntax in the C programming language, could you insert the MIT license into the `hello.c` source file using the sed command? You can use one or several sed commands, you can pipe the output of a sed command into another one, you can use temporary files if you want, but you are *not* allowed to use any other command than sed. Of course, the C source file should still compile after you have inserted the license!

I let you now think about that little problem, and I hope you enjoyed that article and its companion video. If you want to know more about sed, let us know it using the comment section!

Commands