Sed

- Sed

Sed

Quite often when working with text files you'll need to find and replace strings of text in one or more files.

sed is a stream editor. It can perform basic text manipulation on files and input streams such as pipelines. With sed you can search, find and replace, insert, and delete words and lines. It supports basic and extended regular expressions that allow you to match complex patterns.

In this article, we'll talk about how to find and replace strings with sed. We'll also show you how to perform a recursive search and replace.

Find and Replace String with sed

There are several versions of sed, with some functional differences between them. macOS uses the BSD version and most Linux distributions come with GNU sed pre-installed by default. We'll use the GNU version.

The general form of searching and replacing text using sed takes the following form:

```
sed -i 's/SEARCH_REGEX/REPLACEMENT/g' INPUTFILE
```

- -i - By default sed writes its output to the standard output. This option tells sed to edit files in place. If an extension is supplied (ex -i.bak) a backup of the original file will be created.
- S - The substitute command, probably the most used command in sed.
- / / / - Delimiter character. It can be any character but usually the slash (/) character is used.
- SEARCH_REGEX - Normal string or a regular expression to search for.
- REPLACEMENT - The replacement string.
- g - Global replacement flag. By default, sed reads the file line by line and changes only the first occurrence of the - SEARCH_REGEX on a line. When the replacement flag is provided, all occurrences will be replaced.
- INPUTFILE - The name of the file on which you want to run the command.

It is a good practice to put quotes around the argument so the shell meta-characters won't expand. Let's see examples of how to use the sed command to search and replace text in files with some of

its most commonly used options and flags.

For demonstration purposes, we will be using the following file:

```
123 Foo foo foo
foo /bin/bash Ubuntu foobar 456
```

If you omit the g flag only the first instance of the search string in each line will be replaced:

```
sed -i 's/foo/linux/' file.txt
```

```
# output
123 Foo linux foo
linux /bin/bash Ubuntu foobar 456
```

With the global replacement flag sed replaces all occurrences of the search pattern:

```
sed -i 's/foo/linux/g' file.txt
```

```
# output
123 Foo linux linux
linux /bin/bash Ubuntu linuxbar 456
```

As you might have noticed, in the previous example the substring foo inside the foobar string is also replaced. If this is not the wanted behavior, use the word-boundery expression (\b) at both ends of the search string. This ensures the partial words are not matched.

```
sed -i 's/\bfoo\b/linux/g' file.txt
```

```
# output
123 Foo linux linux
linux /bin/bash Ubuntu foobar 456
```

To make the pattern match case insensitive, use the I flag. In the example below we are using both the g and I flags:

```
sed -i 's/foo/linux/gI' file.txt
```

```
123 linux linux linux
linux /bin/bash Ubuntu linuxbar 456
```

If you want to find and replace a string that contains the delimiter character (/) you'll need to use the backslash () to escape the slash. For example to replace /bin/bash with /usr/bin/zsh you would use

```
sed -i 's/\/bin\/bash/\/usr\/bin\/zsh/g' file.txt
```

The easier and much more readable option is to use another delimiter character. Most people use the vertical bar (|) or colon (:) but you can use any other character:

```
sed -i 's|/bin/bash|/usr/bin/zsh|g' file.txt
```

```
# output
123 Foo foo foo
foo /usr/bin/zsh Ubuntu foobar 456
```

You can also use regular expressions. For example to search all 3 digit numbers and replace them with the string number you would use:

```
sed -i 's/\b[0-9]\{3\}\b/number/g' file.txt
```

```
# output
number Foo foo foo
foo /bin/bash demo foobar number
```

Another useful feature of `sed` is that you can use the ampersand character `&` which corresponds to the matched pattern. The character can be used multiple times.

For example, if you want to add curly braces `{}` around each 3 digit number, type:

```
sed -i 's/\b[0-9]\{3\}\b/{&}/g' file.txt
```

```
# output
{123} Foo foo foo
foo /bin/bash demo foobar {456}
```

Last but not least, it is always a good idea to make a backup when editing a file with `sed`. To do that just provide an extension to the `-i` option. For example, to edit the `file.txt` and save the original file as `file.txt.bak` you would use:

```
sed -i.bak 's/foo/linux/g' file.txt
```

If you want to make sure that the backup is created list the files with the `ls` command:

```
ls
```

```
# output
file.txt file.txt.bak
```

Recursive Find and Replace

Sometimes you want to recursively search directories for files containing a string and replace the string in all files. This can be done by using commands such as find or grep to recursively find files in the directory and piping the file names to sed.

The following command will recursively search for files in the current working directory and pass the file names to sed.

```
find . -type f -exec sed -i 's/foo/bar/g' {} +
```

To avoid issues with files containing space in their names use the -print0 option which tells find to print the file name, followed by a null character and pipe the output to sed using xargs -0 :

```
find . -type f -print0 | xargs -0 sed -i 's/foo/bar/g'
```

If you want to search and replace text only on files with specific extension you would use:

```
find . -type f -name "*.md" -print0 | xargs -0 sed -i 's/foo/bar/g'
```