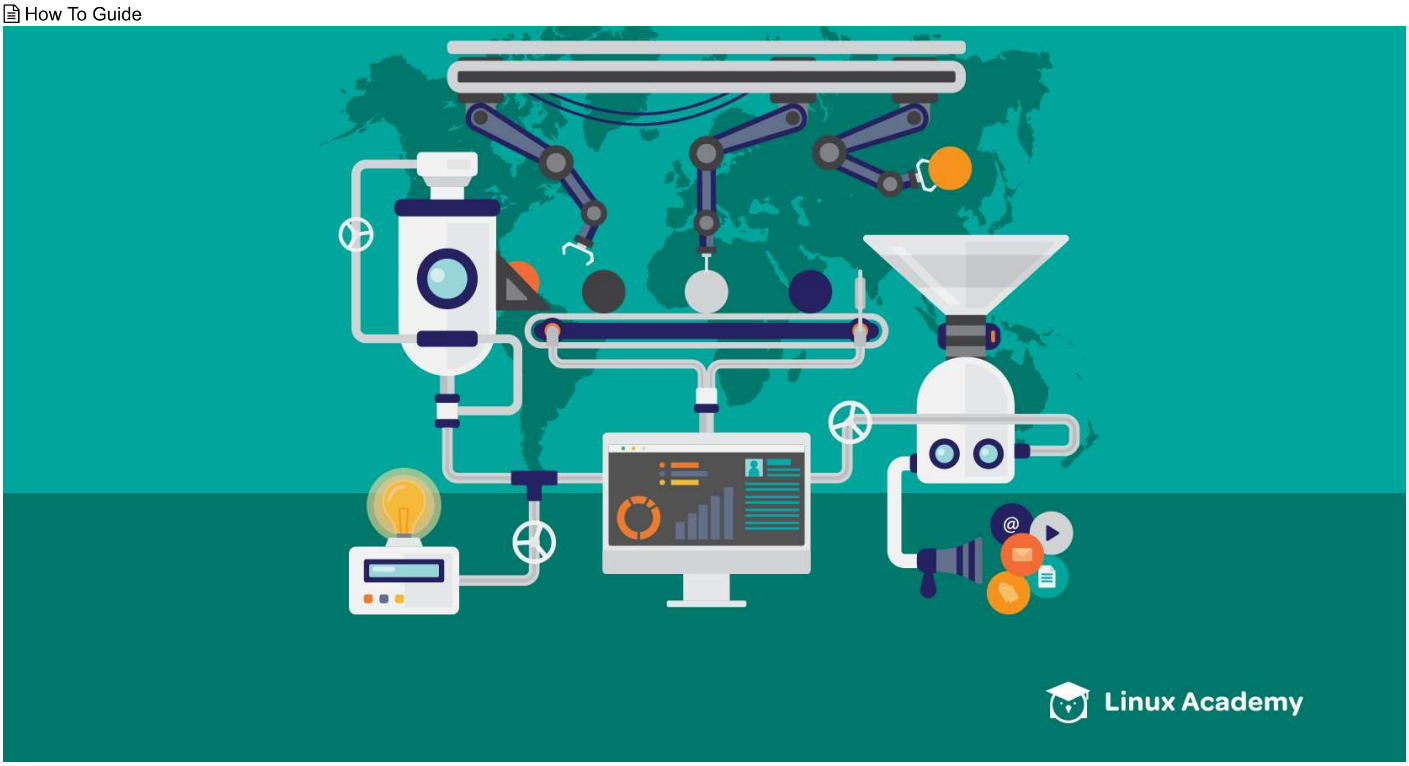
- Spam!

How To Guide



Using Sed to Manipulate Files in Linux
2/24/2017

Instructor Approved

Table of Contents

Introduction

As a systems administrator you'd find yourself editing and looking through a lot of configuration files. Most of the times your favorite text editor would suffice but there are times when you'd need the power of sed.

Sed, short for stream editor, is a command line tool to manipulate text and files.

This guide is an introduction to sed. It contains examples that will help you get familiar with sed.

Requirements

To follow this guide, you'll need access to a Linux or Mac machine. You'll need some familiarity

with using the terminal to execute commands.

If you aren't familiar with using the terminal, feel free to take advantage of the resources

available at Linux Academy to get up to speed. These resources are listed at the end of this guide.

Getting Started

Before we begin, there's a word of caution for Mac users. sed that ships with Mac is BSD sed whereas sed that ships with Linux distros is GNU sed. Both these versions of sed behave differently. We'll focus on using GNU sed. You can install GNU sed via homebrew as:

```
brew install gnu-sed
```

Once you've installed GNU sed on Mac, you'd use the command gsed instead of sed in the rest of the tutorial.

We'll start with simpler examples and move to more involved ones. Before we get started, let's make a file that we'll edit throughout this guide.

This is line one and this has a few words in it; 16 words to be precise.

```
Line 2 starts here.
3rd line here.
And this is the 4th.
```

Save the above in a file named text.txt in your home directory. We're all set to get started with sed.

The way to use sed is by specifying commands and flags that tell sed what to do along with the file on which you want to operate. That is,

```
sed [flags] commands /path/to/file
```

The path may be relative or absolute.

Printing content

Let's start with something very simple - printing the contents to standard out. The way we do this is by passing an empty string as command.

```
me@home:~$ sed "" text.txt
```

This will cause the entire content of the file to be printed in your console.

We can get more control over printing the contents by using the print command explicitly. You specify a p as the command for sed.

```
me@home:~$ sed "p" text.txt
```

This will cause all the lines to be printed twice. The reason is that sed is processing the file line-by-line and since we've told it to print the contents, we see the lines twice.

Let's stop the repetition of the lines by using the -n flag.

```
me@home:~$ sed -n "p" text.txt
```

By default, sed prints outputs every line of input it processes. The -n flag will suppress the output sed generates as it processes each line.

We can decide how many lines we want to print by specifying the line numbers with the p command. Here's how:

```
me@home:~$ sed -n "1,2p" text.txt
```

Here we are printing lines 1 through 2. If you were to specify just one number, you'd end up printing that specific line.

We can print the next few lines starting from a particular line using the + symbol. Here's how:

```
me@home:~$ sed -n "1,+2p" text.txt
```

The above command prints the first line and the next two lines.

We can also print in specific intervals by using a tilde (~). You can print all the odd-numbered lines as follows:

```
me@home:~$ sed -n "1~2p" text.txt
```

Deleting content

We can delete content from the file using the d command. We specify the lines we want to delete in the same way as we do for printing but instead of p we use d. For example, here's how we would delete the first line of the file.

```
me@home:~$ sed "1d" text.txt
```

This will remove the first line of text from the file and print the rest in the terminal. A thing to note is that this is a non-destructive operation and the file is unchanged. We can make the changes apply to the file directly by specifying the flag for in-place editing as -i or -- in-place.

```
me@home:~$ sed -i "1d" text.txt
```

We can delete lines containing specific words as follows:

```
me@home:~$ sed "/3rd/d" text.txt
```

The above example will delete lines containing the word "3rd". You're not limited to just words, you can bring the full power of regular expressions to your advantage.

Let's delete the lines that contain numbers at the beginning of the line.

```
me@home:~$ sed "/^[0-9]/d" text.txt
```

This will delete the 3rd line since that is the only line with a number at the beginning.

Replace content

In this section we'll look at how to replace content. Let's begin by replacing all the numbers in the file with underscores.

```
me@home:~$ sed -r "s/[0-9]{1,}/_/g" text.txt
```

The -r flag tells sed to use extended regular regular expressions. The curly braces {1,} tell sed to replace one or more numbers with an underscore. We do this to ensure that numbers like 16 are replaced by a single underscore instead of two. The s at the beginning is the substitute command and the g, which stands for global, causes all numbers to be replaced with underscores. Without this, only the first occurrence of a number on the line would be replaced.

The general form of the command for replacing text is this as follows:

```
s/what_to_replace/how_to_replace/g
```

Compare this to the command we've entered for substitution.

Now let's replace the words This, with a capital T, and this, with a small t with the word that.

Here's how:

```
me@home:~$ sed "s/This|this/that/g" text.txt
```

The pipe symbol | means OR and so This OR this will be replaced with that.

Next we'll surround every number with parentheses. Here's how that's done:

```
me@home:~$ sed "s/[0-9]{1,}/(&)/g" text.txt
```

You'd notice that we've used an ampersand and surrounded it with parentheses and the end result is that the numbers in our file are surrounded by parentheses. With our regex, we are matching numbers. Ampersand represents the matched number.

Let's do what we've done above in a slightly different way.

```
me@home:~$ sed -r "s/[[:digit:]]+/(&)/g" text.txt
```

Here [[:digit:]] is a character class that is equivalent to [0-9] and the + sign after it is the one-or-more quantifier that is the same as {1,}.

This brings us to the end of the guide on sed.

Miscellaneous

The following sections contains some more information on sed.

Address range

Address range specifies the lines sed should operate on. By default, sed operates on all the

lines. However, sometimes we want to limit that, say the first 10 lines of the file. We do this by specifying an address range.

We've seen this before when we printed content with p. The line numbers are the address range. This can take a number of forms.

```
me@home:~$ sed -n "1p" text.txt
me@home:~$ sed -n "1,2p" text.txt
me@home:~$ sed -n "1,+2p" text.txt
me@home:~$ sed -n "1 ! p" text.txt
me@home:~$ sed "1,2 ! p" text.txt
```

When an address range contains just one number, sed operates on that line. This is shown in the first example. In the second example, the address range has both a start and an end so sed will operate on all the lines that fall in that range. The third example is similar to the second one in that it has both a start and an end but the end is relative to the start. In the fourth example,we are printing everything except the first line. In the last example, we are printing everything except the first two lines.

Regular expression

Regular expressions are, simply put, strings that represent patterns to match other strings. For example, we've seen how to match numbers using [0-9]. There's a lot more to regex and lies beyond the scope of this guide.

Character class

A character class represents a group of characters. For example, [0-9] represents numbers, [a-z] represents all lowercase alphabets, and [A-Z] represents all uppercase alphabets. They provide a succinct way to write your regular expressions.

There is also another way to write character classes as [[:digit:]] as we saw before. This way makes your script a lot more readable and also more portable. Consider [[:alpha:]] which represents [a-zA- Z] for US locale. This might represent something different for Arabic or some other language.

Here's a list of character classes in their second form:

```
[[:alpha:]] - represents alphabets. Translates to [a-zA- Z].
[[:alnum:]] - represents alphabets and numbers. Translates to [a-zA- Z0-9].
[[:blank:]] - represents blank spaces like tabs and spaces.
[[:cntrl:]] - represents control characters like new line, carriage return, etc.
[[:digit:]] - represents digits. Translates to [0-9].
[[:graph:]] - represents printable and visible characters.
[[:lower:]] - represents lowercase characters. Translates to [a-z].
[[:print:]] - represents printable characters.
[[:punct:]] - represents punctuation marks. Translates to [!-/:- @[-`{- ~].
[[:space:]] - represents all whitespaces. Translates to [ \t\v\f].
[[:upper:]] - represents uppercase characters. Translates to [A-Z].
[[:xdigit:]] - represents hexadecimal characters. Translates to [0-9a- fA-F]
```

Additional Resources

If you are new to the Linux operating system, take a look at the Linux Essentials course available at Linux Academy. The course will give you a basic understanding of Linux and give you a gentle introduction to the command line.

https://linuxacademy.com/cp/modules/view/id/38 (https://linuxacademy.com/cp/modules/view/id/38)

If you'd like to master the terminal, have a look at Mastering Linux Command Line:

https://linuxacademy.com/cp/modules/view/id/10 (https://linuxacademy.com/cp/modules/view/id/10)

To take your skills to the expert level, take a look at Linux by Example from Novice to Pros:

https://linuxacademy.com/cp/modules/view/id/19 (https://linuxacademy.com/cp/modules/view/id/19)

Linux Academy provides a large library of in-depth online Linux training, and they also give

access to many other online courses in topics like AWS (https://linuxacademy.com/amazon-web-services/courses), DevOps (https://linuxacademy.com/devops/courses), Azure (https://linuxacademy.com/azure/courses), OpenStack (https://linuxacademy.com/openstack/courses), and Big Data (https://linuxacademy.com/bigdata/courses).

7 13

... Show Previous Comments

Hector F. Jimenez Saldarriaga (/cp/socialize/profile/user/h3ct0rjs) 7/13/2017
Nice guide

0

AJ NOURI (/cp/socialize/profile/user/AJNOURI) 7/31/2017
Thanks for the great guide Fasih, short and concise!

0

Reply