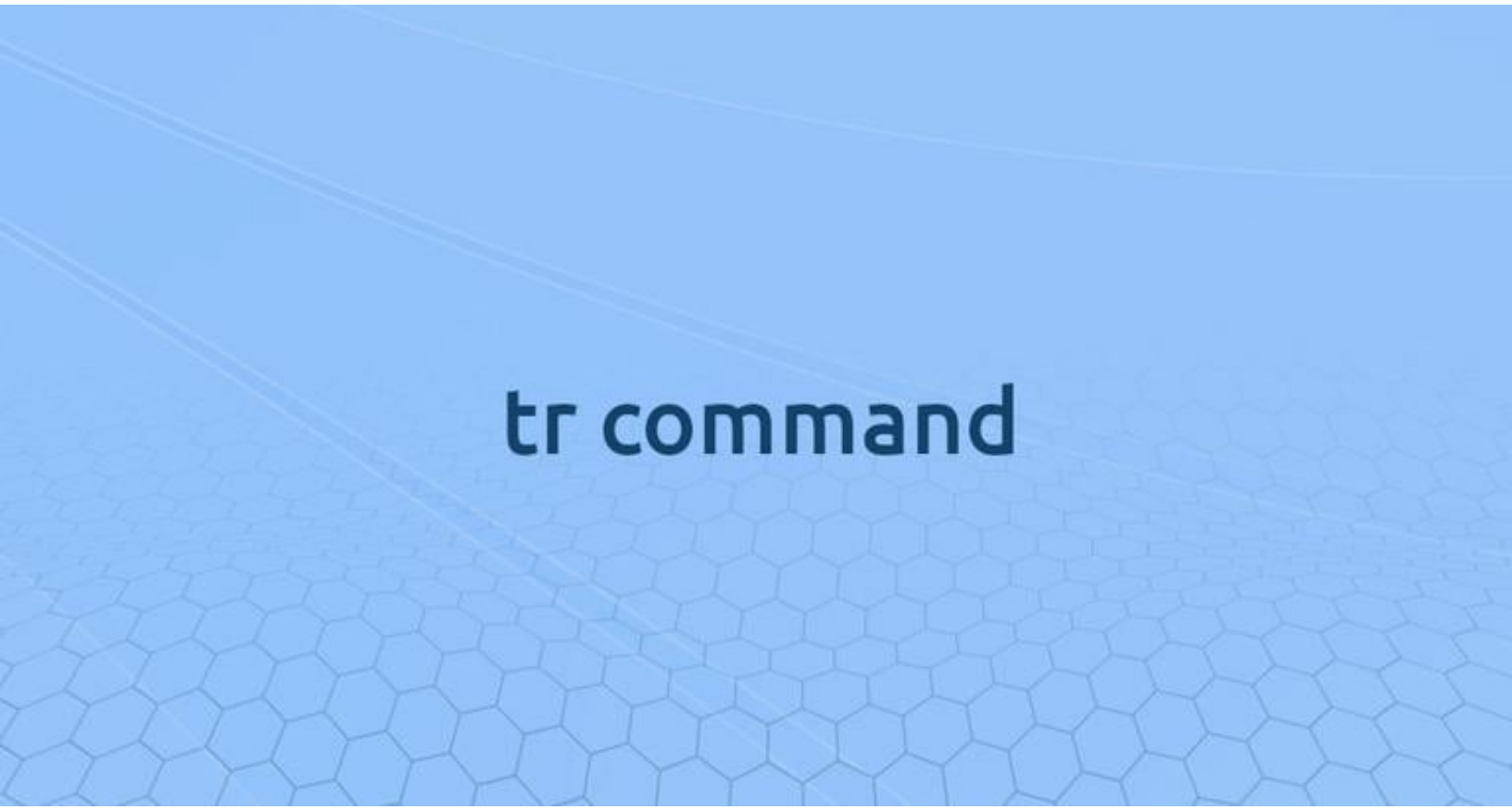
Tr Command in Linux with Examples

Updated Nov 27, 2019 • 5 min read



`tr` is a command-line utility in Linux and Unix systems that translates, deletes, and squeezes characters from the standard input and writes the result to the standard output.

The `tr` command can perform operations like removing repeated characters, converting uppercase to lowercase, and basic character replacing and removing. Typically, it is used in combination with other commands through piping.

In this tutorial, we will show you how to use the `tr` command through practical examples and detailed explanations of the most common options.

The syntax for the `tr` command is as follows:

```
tr OPTION... SET1 [SET2]
```

`tr` accepts two sets of characters, usually with the same length, and replaces the characters of the first sets with the corresponding characters from the second set.

A `SET` is basically a string of characters, including the special backslash-escaped characters.

In the following example, `tr` will replace all characters from the standard input (linuxize), by mapping the characters from the first set with the matching ones from the second set.

```
$ echo 'linuxize' | tr 'lin' 'red'
```

Each occurrence of `l` is replaced with `r`, `i` with `e`, and `n` with `d`:

```
Output
reduxeze
```

The character sets can also be defined using character ranges. For example, instead of

```
$ echo 'linuxize' | tr 'lmno' 'wxyz'
```

you can use:

```
$ echo 'linuxize' | tr 'l-n' 'w-z'
```

When `-c` (`--complement`) option is used, `tr` replaces all characters that are not in SET1.

In the example below all characters except “li” will be replaced with the last character from the second set:

```
$ echo 'linuxize' | tr -c 'li' 'xy'


Output
liyyyiyyy
```

As you may have noticed, the output above has one more visible character than the input. This is because the `echo` command prints an invisible newline character `\n` that is also replaced with `y`. To echo a string without a new line, use the `-n` option.

The `-d` (`--delete`) option tells `tr` to delete characters specified in SET1. When deleting characters without squeezing, specify only one set.

The command below will remove `l`, `i` and `z` characters:

```
$ echo 'Linuxize' | tr -d 'liz'
```

The `L` character is not deleted because the input includes an uppercase `L` while the `l` character in the SET is lowercase.

```
Output
Lnuxe
```

The `-s` (`--squeeze-repeats`) option replaces a sequence of repeated occurrences with the character set in the last SET.

In the following example, `tr` removes the repeated space characters:

```
$ echo "GNU     \    Linux" | tr -s ' '
```

```
Output
GNU \ Linux
```

When SET2 is used the sequence of the character specified in SET1 is replaced with SET2.

```
$ echo "GNU     \    Linux" | tr -s ' ' '_'
```

```
Output
```

The `-t (--truncate-set1)` option forces `tr` to truncate SET1 to the length of SET2 before doing further processing.

By default, if SET1 is larger than SET2 `tr` will reuse the last character of SET2. Here is an example:

```
$ echo 'Linux ize' | tr 'abcde' '12'
```

The output shows that the character `e` from SET1 is matched with the latest character of SET2, which is `2`:

```
Output
Linux iz2
```

Now, use the same command with the `-t` option:

```
$ echo 'Linux ize' | tr -t 'abcde' '12'


Output
Linux ize
```

You can see that the last three characters of the SET1 are removed. SET1 becomes ‘ab’, the same length as SET2, and no replacement is made.

Combining options

The `tr` command also allows you to combine its options. For example, the following command first replaces all characters except `i` with `0` and then squeezes the repeated `0` characters:

```
Output
0i0i0
```

Tr Command Examples

In this section, we'll cover a few examples of common uses of the `tr` command.

Convert lower case to upper case

Converting lower case to upper case or reverse is one of the typical use cases of the `tr` command. `[:lower:]` matches all lower case characters and `[:upper:]` matches all uppercase characters.

```
$ echo 'Linuxize' | tr '[:lower:]' '[:upper:]'
```

```
Output
LINUXIZE
```

Instead of character classes, you can also use ranges:

```
$ echo 'Linuxize' | tr 'a-z' 'A-Z'
```

To convert upper case to lower case, simply switch the places of the sets.

Remove all non-numeric characters

The following command removes all non-numeric characters:

```
$ echo "my phone is 123-456-7890" | tr -cd [:digit:]
```

`[:digit:]` stands for all digit characters, and by using the `-c` option, the command removes all non-digit characters. The output will look like this:

```
Output
1234567890
```

Put each word in a new line

To put each word in a new line, we need to match all non-alphanumerical characters and replace them with a new line:

```
$ echo 'GNU is an operating system' | tr -cs '[:alnum:]' '\n'
```

```
Output
GNU
is
an
operating
system
```

Remove blank lines

To delete the blank lines simply squeeze the repetitive newline characters:

```
$ tr -s '\n' < file.txt > new_file.txt
```

In the command above we are using the redirection symbol `<` to pass the content of the `file.txt` to the `tr` command. The redirection `>` writes the output of the command to `new_file.txt`.

Print `$PATH` directories on a separate line

The `$PATH` environmental variable is a colon-delimited list of directories that tells the shell which directories to search for executable files when you type a command

To print each directory on a separate line we need to match the colon (`:`) and replace it with the new line:

```
$ echo $PATH | tr  ':' '\n'
```

```
Output
/usr/local/sbin
/usr/local/bin
/usr/sbin
/usr/bin
/sbin
/bin
```

Conclusion

`tr` is a command for translating or deleting characters.

Although very useful, `tr` can work only with single characters. For more complex pattern matching and string manipulation, you should use `sed` or `awk` .

If you have any questions or feedback, feel free to leave a comment.

tr terminal
