How to use the Tee command in Linux

If you've ever used pipe and redirection in Linux shell, sometimes you'll need to use the tee utility.

If you've ever used pipe and redirection in Linux shell, sometimes you'll need to use the `tee` utility.

What does Tee hold?

A command like `ls` will display the contents of the current directory. In other words, it displays these content into a stdout (standard output), usually your screen, or more accurately a virtual terminal screen.

A command like `ls > file123` will not display anything on the screen. That's because the `>` sign redirects all output to a file instead of displaying it as stdout. Now **file123** will be filled with the contents shown earlier on the screen.

To display the contents of the folder on the screen and write it to a file, you must use two commands. But with a `tee` , you can do both at the same time.

```
ls | tee file123
```

Why should Tee be used when you can run a similar command twice?

In the above example, it is clear that you don't need a `tee` if you can execute `ls` normally, then redo and redirect the output to a file. However, you will encounter situations in which the output is unique. Imagine the situation you are trying to diagnose a problem. You run:

```
diagnose | tee error.log
```

The errors you get may be unique. You want them to be displayed on the screen so you can see what's going on when checking everything. But you also want those errors to be saved in a file, to review later or paste the output into a discussion forum and consult people about it.

Another common situation that you may need to `tee` is when you want to take the output of the command to a location that only root users can read or write. The following command does not work:

```
/sbin/blkid > /root/somefile
```

Then you might think, just use `sudo` ! But you will be surprised when this command is not working either:

```
sudo blkid > /root/somefile
```

That's because after `sudo blkid` executes, you are still logged in as not a root user. And your shell (usually bash) tries to write to **/ root / somefile** with your regular user information. To solve this, you can use the `tee` :

```
/sbin/blkid | sudo tee /root/somefile
```

Append text and redirect errors

`Tee` is a simple but useful command. A `command | tee somefile` basic `command | tee somefile` usually sufficient for most situations. However, there are 2 cases where you will need these tips.

The first thing to know is a `tee` , by default, always overwrites a file. If you run:

```
ls | tee somefile
```

Then run:

```
ls /tmp | tee somefile
```

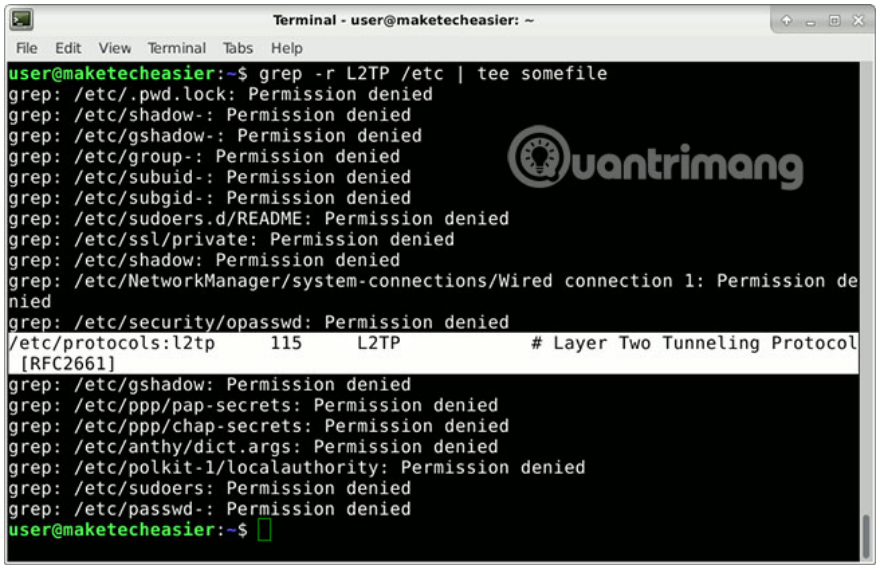
The second command will overwrite the content of **somefile** and you will only see the contents of the last command executed. To change this behavior, you can create text that links `tee` instead of overwriting. To do so, simply use the **-a** switch command.

```
ls | tee -a somefile
```

The second thing to know is that not all outputs are the same. Error messages are handled differently and although they appear on the screen, they are not considered stdout, but are considered stderr and not handled by `tee` . The following is an example of `grep` .

```
grep -r L2TP /etc | tee somefile
```

The results shown will look like the following image:



Notice **Permission denied** is written into stderr. The only thing written into stdout is the highlighted text. That's why you notice that the content of '**somefile**' is what is shown in the image below:

In this case, `grep` is used to search for text and is useful when error messages are not redirected to the file. Error messages only fill the file with unnecessary information. You just want to see the results found. But when you need an error message, use `2>&1` , to redirect stderr to stdout.

```
grep -r L2TP /etc 2>&1 | tee somefile
```

With this command, you will notice that **somefile** now also contains error messages.

Hopefully, this guide includes everything you need to make the most of the `tee` . But if you encounter some difficult situation with a `tee` , leave comments in the comment section below for everyone to share.

Wish you use the successful `tee` command!

You finished reading the article "**How to use the Tee command in Linux**" edited by the **TipsMake** team. We hope this article has provided you with many useful tech tips and tricks. You can search for similar articles on tips and guides. Thank you for reading and for following us regularly.
