Abdullah ÇAY

Bash and Linux - File manipulation

45 minutes Easy

Machine status

Ubuntu Server 18.04 LTS

SSD Volume Type

64-bit x86

Stopped

Connect

Reset Start

DataScientest • com

Introduction to Linux

File manipulation

In this lesson, we are going to see how to manipulate files and folders.

Creating files and folders

The `touch` command is a command that is used to create an empty file. You simply need to specify the name of the file you want to create.

Run this command

```
touch my_file
```

A file called `my_file` has been created in the current directory.

Run the same command and check the resutls with `ls`

We can still see our file. Note that if the file exists, the content of the file is not replaced: its latest modification date is simply update.

Run this command to create a file in the `/home` folder

```
touch /home/my_other_file
```

Check that the file has been created

To create a directory, we can use `mkdir` which stands for `make directory`.

Run this command and check the content of the current working directory

```
mkdir my_directory
```

Create a file `my_file` in the `my_directory` folder

Show / Hide solution

```
# to check
ls -R

```

Removing files and folders

To remove a file, we can use the `rm` command which stands for `remove`. For example, to delete the `my_file` file, we can use `rm my_file`.

> Remove `my_file` which is in the current working directory

This can also be used to remove folders:

> Remove `my_directory` from the current working directory

You should get an error message: `rm: cannot remove 'my_directory/': Is a directory`.

> Use the `--help` argument to find how to remove directories using `rm` and remove `my_directory`

Show / Hide solution

```
# displaying help
rm --help

# deleting my_directory
rm -r my_directory
```

Note that we can create or remove multiple objects by putting multiple names after `touch`, `mkdir` or `rm`: for example, `touch file1 file2` will create two files `file1` and `file2`.

> Run the following commands

```

mkdir folder1 folder2 folder3

touch file1 file2 file3

touch folder1/file1 folder2/file2 folder2/file3

rm folder2/file2 file1

rm -r folder3 folder1

```

> Without looking at the result, what is the structure of the current working directory ? You can check using `ls -R`

Finally, we can delete multiple objects by using `*` as a placeholder for a string.

> Run the following command

```
rm ./f*
```

You get an error message because we are trying to delete everything that starts with `f` in the current directory and `folder2` matches this. But if we check the content of the current directory, we can see that `file2` and `file3` were deleted.

> Delete the last folder

```
rm -r folder2
```

Copy/Paste and Move files

To copy/paste a file or a folder, we can use the `cp` command. This command takes two arguments: first the source file and then the destination file:

```
cp source_file destination_file
```

> Create a file named `my_file`

Show / Hide solution

```
touch my_file
```

> Copy/Paste this file to the parent directory. The new file name should be `my_file_new`

Show / Hide solution

```
cp my_file ../my_file_new
```

Note that we are not forced to give a new name to our file. We can simply specify the directory where we want to paste our file:

```
cp my_file ..
```

Show / Hide solution

To copy/paste a directory, we need to use the `-r` flag.

> Create a new directory `my_new_directory` and two files, `file1` and `file2`, within. Copy/Paste this directory to the parent folder

```
# creating directory
mkdir my_new_directory

# creating files
touch my_new_directory/file1 my_new_directory/file2

# copying/pasting directory
cp -r my_new_directory ..

```

Show / Hide solution

To move a file or a folder, we can use `mv` with the same syntax as `cp`. The only major difference is that we do not need `-r` to move folders.

```
touch file1 file2 file3

mv file* ../my_new_directory

ls .

ls ../my_new_directory

```

We can also use `mv` to rename files:

> Run the following command

```
touch file1

ls

mv file1 file1_but_with_a_new_name

ls

```

In this lesson, we have seen how to create files or folders, how to move them and how to remove them. We do not need a file manager with a graphic interface anymore !

> In this part, we have used files with no extension. You may be used to text files ending in `.txt`, python scripts ending in `.py` or flat data files ending in `.csv`. Actually, those extensions do not serve a great role apart from helping certain tools to open them. For example, you could have csv data in a `data.py` file. Your file manager will open it with a python editor and this editor will show plenty of errors. But you could very well force it to open with Excel, Number (or better: LibreOffice Calc), ... The file extension is therefore only a part of the name and does not affect its content.

Reading the content of a file

To read the content of a file, we have different options but the most used would be `cat` followed by the name of the file.

> Read the content of one of the file we have created in the last step

```
cat ../my_new_directory/file1
```

Show / Hide solution

Nothing is displayed: remember that `touch` only creates an empty file !

> In the `/` and its subfolders, there must be a file that is not empty: find one an print its content

```

# listing content
ls

# going to /etc
cd etc

# listing content
ls

# printing bash.bashrc content
cat bash.bashrc

# returning to home folder
cd

```

Show / Hide solution

You may have encountered a lot of permission denied errors. We will see this in another part. If you want to clean the console, you can use `clear`. Do not forget to get back to the home folder with `cd`.

In some cases, we may want to print only the beginning or the end of a file. To do so we can use `head` or `tail`.

> Find a way to display the 3 first lines and 2 last lines of `/etc/bash.bashrc` by using `head` and `tail` help

```
# 3 first lines
head -n 3 /etc/bash.bashrc
# 2 last lines
tail -n 2 /etc/bash.bashrc
```

Show / Hide solution

Printing data into a file

To print some text directly into the console, we can use `echo`.

> Run this command

```
echo hello world
```

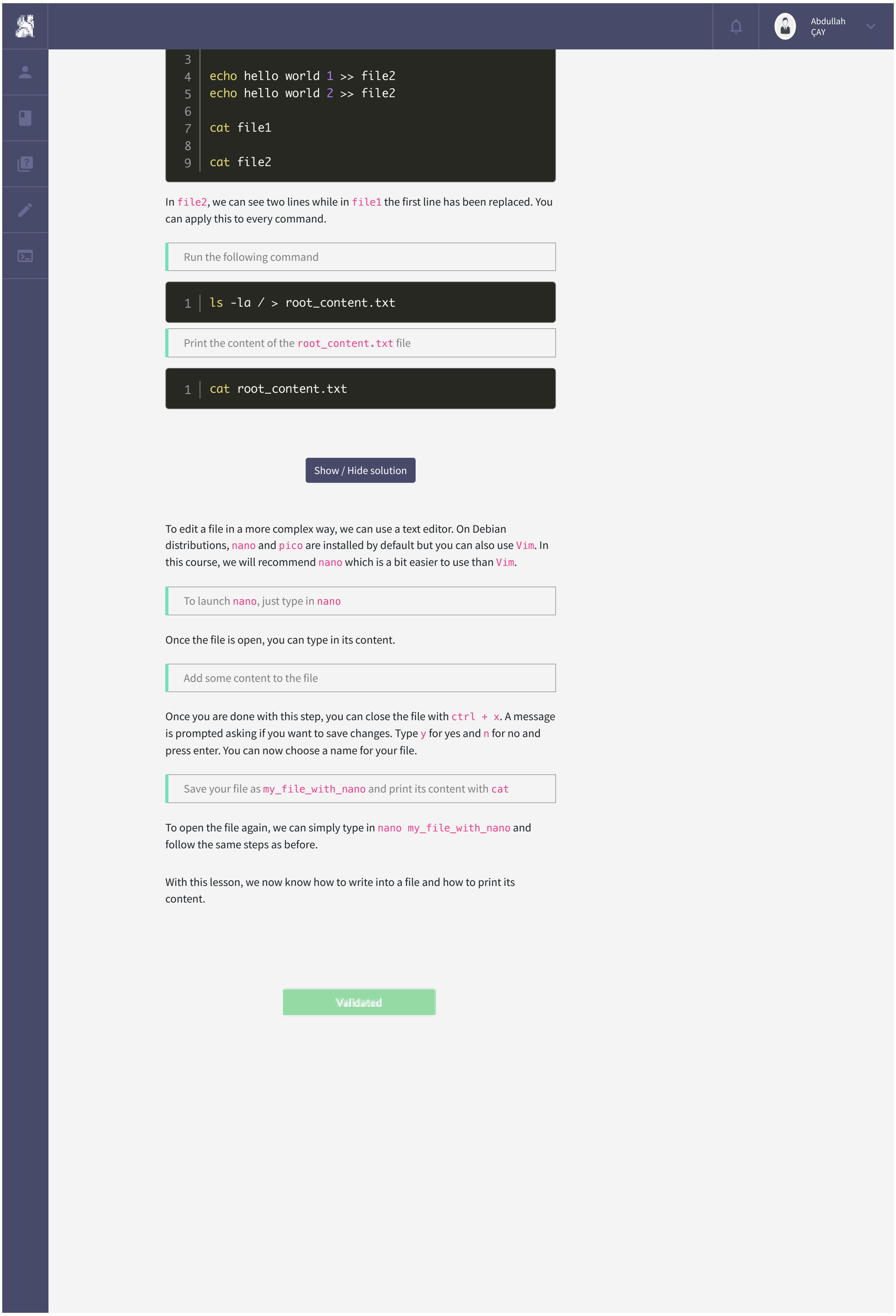
`hello world` is being displayed into the standard output which we will talk about later. To print the content of the standard output into a file, we can use `>` or `>>` followed by the file name.

> Run this command

```
echo hello world > my_file
```

To check the content of `my_file`, we can use `cat my_file`. The difference between `>` and `>>` is that `>` overwrites the content of the file while `>>` appends the results at the end of the file.

> Run the following commands



```

echo hello world 1 >> file2
echo hello world 2 >> file2

cat file1

cat file2
```

In `file2`, we can see two lines while in `file1` the first line has been replaced. You can apply this to every command.

Run the following command

```
ls -la / > root_content.txt
```

Print the content of the `root_content.txt` file

```
cat root_content.txt
```

To edit a file in a more complex way, we can use a text editor. On Debian distributions, `nano` and `pico` are installed by default but you can also use `Vim`. In this course, we will recommend `nano` which is a bit easier to use than `Vim`.

To launch `nano`, just type in `nano`

Once the file is open, you can type in its content.

Add some content to the file

Once you are done with this step, you can close the file with `ctrl + x`. A message is prompted asking if you want to save changes. Type `y` for yes and `n` for no and press enter. You can now choose a name for your file.

Save your file as `my_file_with_nano` and print its content with `cat`

To open the file again, we can simply type in `nano my_file_with_nano` and follow the same steps as before.

With this lesson, we now know how to write into a file and how to print its content.