Lesson 1: list files

What is ls?

The first command we learn is ls. This is the list command. It allows us... list the content of the directory.

ls is the command we can encounter in all Linux and Unix systems. We can see it as one of the 'core' commands in the system.

Ok, let's play with it a little.

Run ls

What we see on the screen is a list of the content of the current directory.

What is visible immediately, that we have different colors. We can imagine what it means, but let's forget it for a second. What many waves of "classic" sys admins saw in their days:

ls --color=no

So, this leads us to the first output, the colorized list of content. In order to have it, we need to run

ls --color=yes

or, alternatively for now

ls --color=auto .

For now, it will be enough to say, this colorized option is added as default in one place in the system, we will go there in one of the future lessons.

NEXT

# How to get more details

All right. We know ho to get list of items from directory.

But this is less than informant. Yes, we have a view. But without details.

Before we learn how to get more details, there is a command we will run now, to clear the screen.

clear

What does it do? Well, it clears the screen. How it works, doesn't matter for us, really.

The same you can achieve by running /bin/echo -e "\x1b\x5b\x48\x1b\x5b\x32\x4a\c" , but we can agree, that 'clear' is easier to remember!

Ok, let's finally learn more about ls. The command has multiple arguments we can add in order to change the output of the command.

Now, important to understand. What we pass after the command - these are **arguments**. Not switches, parameters, etc. We can hear these names, but to learn how to work with Linux properly, we should name the elements with proper convention.

So, our first argument will give us more details. Please execute

ls -l .

- indicates that we will pass arguments. We have two ways here

* - one dash informs the system that we will pass one letter argument, like 'l'
* -- two dashes means that argument will contain more than one letter. Most commonly it will be an english word.

l means long listing format.

What we have here?

The format contains a few columns. Let's try to understand them:

* permissions. Here we can see what kind of permission has the object. We will work with permissions... soon :)
* number of hard links. By default every object has 1 hard link. What are links? We will learn it in 18th lesson.
* Owner. The owner of the object. It doesn't mean who created it, but who owns it at this moment.
* Group. Owner belongs to the group (of users). This inforation is here too. This means that our 'groupmates' have specific access to the file.
* Size. File size in bytes.
* date and time of **last modification** of the object.
* file name.

Owner and group. For now it will be enough to know that we see on the listing above the names of the users and groups. The system keeps and translates them from numerical representation. These are UID for user identifier and GID for group identifier. We can list this information in numeric way, by entering the

ls -n command. This works like ls -l, but it changes the user-friendly names to UIDs and GIDs.

Right now we listed all files in alphabetical order.

But... All files?

List all files

Maybe some of you know, but there are many files and directories which are hidden. This generally means that these object are not listed by using *standard* listing commands (or by standard view in GUI). In Linux these files and directories are present too. There are a lot already in your directory.

Linux uses . on the beginning of the object's name. We call these files the dotfiles. Ok, let's print the files again.

ls

Hm, there are no 'hidden' files.

Yes, we have to find a proper argument. this time it will be -a.

ls -a

This command listed much more files than before. A lot of dotfiles! No matter what are they, at least, not now. But two of them needs to be explained.

* .
* ..

The first, . simply means the current directory of the user.

The second, .. means parent directory.

Simple like that :)

So... Let's try something.

ls .

will show exactly the same output like simple ls. And

ls .. shows... yes, the parent directory structure!

Final argument for this section is -A (capital a). While a means all, A means almost all. In this case the command will show all files, except the . and ...

Let's try: ls -A .

Ok, last thing for now. We can combine multiple arguments in the command. Try run

ls -al . What can you see?

# Different formats

Ok, we are able to sort the list. Now it is time to format it a little.

The first option will be applied to simple ls.

Let's recall, how the simple ls looks like. Now, let's use it with our new argument.

ls -1

Now, please think, why it is obvious thet we do not need to use -1 with -l argument?

--format can be usable when scripting and ls is used for input for other parts of the script.

ls --format=commas will print the files separated by commas. We can use shorter syntax and write

ls -m

Surprise! -l is also the --format option. If you wish to use it in full, use ls --format=long

ls -lQ prints the filenames in quotes

--time-style changes the way how the date is formated in long format. Let's experiment:

ls -l  
ls -l --time-style=locale  
ls -l --time-style=iso  
ls -l --time-style=full-iso

BACKNEXT

# A few arguments more

To finalize some usefull arguments, please take a look on these:

ls -al --author prints the username of the creator of the file.

ls -ald prints directories only. Very useful in some circumstances.

ls -ali prints inodes (there will be a lesson about inodes).

ls -alR recursively prints all subdirectories.

ls -alr prints list in the reversed order. So,

ls -alSr is printing... what? :)

## **Finale**

Two last commands in this scenario.

ls --version prints the version of the binary.

All commands which we used here are available in help. How to get the help?

ls --help

# A few arguments more

To finalize some usefull arguments, please take a look on these:

ls -al --author prints the username of the creator of the file.

ls -ald prints directories only. Very useful in some circumstances.

ls -ali prints inodes (there will be a lesson about inodes).

ls -alR recursively prints all subdirectories.

ls -alr prints list in the reversed order. So,

ls -alSr is printing... what? :)

## **Finale**

Two last commands in this scenario.

ls --version prints the version of the binary.

All commands which we used here are available in help. How to get the help?

ls --help