Theory: Parameters and options

5 minutes 0 / 5 problems solved

We hope that you already know how to open the command-line interpreter and run some basic commands. Now let's take a step further and learn how to expand the functionality of the commands and how to get more information about them.

§1. Commands with parameters

Sometimes using just one command is not enough. Let's take a look at the command `mkdir`, which is used to create a new folder in the current directory. If you try to use it as it is, you will get an error. The terminal needs to know how to name a new folder! That's where parameters come in handy. A **parameter** is some additional information that you give to the command. Simply put, parameters are variables that commands can take.

Now, type the command `mkdir` with a parameter `papers`. We use this command to create a folder named *papers*:

```
C:\users\student> mkdir papers
```

Although the current directory stays the same, if you follow this path, you will see that the new folder *papers* was created in the *student* directory.

> All examples in this topic are for Windows OS, but listed commands are relevant for Linux and macOS too. Note that the path separator in Windows is a backslash but in Linux/macOS it's a usual slash.

Now let's change our location and go to the folder you've just created! Use the `cd` command and the path to the *papers* folder will be a parameter.

```
C:\users\student> cd C:\users\student\papers
C:\users\student\papers>
```

Another useful parameter of the `cd` command is `..` parameter. It allows you to go to the **parent directory**, the directory one level above the current one.

```
C:\users\student\papers> cd ..
C:\users\student>
```

You can also go back to the **root folder**, a top-level directory in the file system. To go back to the root directory, you can use the `/` parameter:

```
C:\users\student> cd /
C:\
```

Thanks to commands and parameters, it seems like we are back to the roots! Actually, without the parameters, most of the commands would be useless.

§2. Options

If you google anything about commands and a terminal, you'll encounter the term "options". Don't be afraid of it! Let's briefly explore what it means.

Options, as the name suggests, are usually optional and are used to somehow change the common behavior of the command. If you use Windows and are already sick and tired of exploring the current drive, you can change it by adding the `/d` option to the `cd`. Don't forget to set the path you want to follow as the parameter, for example, `F:\Codepen snippets`:

```
C:\users\student\Desktop> cd /d F:\Codepen snippets
F:\Codepen snippets>
```

Current topic:

Parameters and options

Topic depends on:

× Command line overview

Topic is required for:

Running programs on your computer

Java Archive

Gradle basics

IntelliJ Run Configurations

Viewing files in shell

os module

Pip

Command line arguments

Local work

The main() function

Starting with Git

Table of contents:

Now you see that with options and parameters you can transform a simple command into something complicated.

To sum up: what are essentially options and parameters? Both of them are just two particular types of arguments. While an **option** changes the behavior of a command, a **parameter** is used to assign information to either a command or one of its options. One of the key differences between them is that the number of possible values in options is limited and locked in the code, while with parameters users have more freedom as they don't have such limitations.

§3. Help Manual

No one can remember all the existing commands, options, and parameters. Don't worry about that. The `help` command is there for you. Type it in Windows, and you will get a list of commands available to you.

> For Linux and macOS, a way to get information about the commands depends on the shell you use. The simplest way for Linux is a `--help` flag. There is also `man` command, short for *manual*. You can use it similar to the help command in Windows: `man mkdir`.

That's not all. The `help` command can take any command as a parameter and return all the available options. Let's try. We will use the simplest command we've learned so far, the `cd` command.

```
C:\users\student> help cd

Displays the name of or changes the current directory.

CD [/D] [drive:][path]
CD [..]

   ..  Specifies that you want to change to the parent directory.


Type CD drive: to display the current directory in the specified drive.

Type CD without parameters to display the current drive and directory.




Use the /D switch to change the current drive in addition to changing the current
directory for a drive.



<...>
```

As you can see, these are all the details you need to know about the `cd` command. We call this description the **help manual**.

Let's discuss what the help manual includes. First, it states what the command is supposed to do. For `cd` command, it reads, *"Displays the name of or changes the current directory".* Then it returns all the combinations of that command along with all possible parameters that you can use. You can also notice that in Windows commands are case insensitive unlike in Linux and macOS. Let's look at the example from the manual:

```
CD [/D] [drive:][path]
```

So, the above command has three parts. `CD` is the command name. `[/D]` is an option, and `[drive:][path]` is a parameter. You might wonder what do these `[]` brackets mean. Well, those are just *notations* that mean that the parameters are optional to the commands. You don't have to add these brackets when you use commands.

You can read this article for Windows or the manual for the cat command on Linux/macOS to learn more about the command-line syntax and look through the examples.

§4. Conclusion

Let's summarize what we've learned so far:

- You can use options and parameters to extend the functionality of commands.
- You can pass different values with the parameters.
- You can get a full list of commands using the `help` and `man` commands.
- You can open a help manual for a command by typing `help [command_name]` or `man [command_name]`. This manual explains how to use a command properly and what options and parameters it has if any.

Although you may feel that using these commands would slow down developer's work and that they are less efficient, we would still urge you to try them out. You have to get used to these commands as early as possible. Once you get accustomed to working with them, you will find that using them is much easier than resorting to the GUI on many occasions.

Report a typo

809 users liked this theory. **21** didn't like it. **What about you?**

Start practicing

Comments (21) Hints (0) Useful links (0) Show discussion