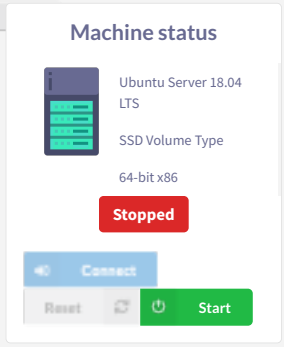
Bash and Linux - Introduction (EN)

30 minutes Easy



DataScientest • com

Introduction to Linux

In this course, we are going to introduce basic concept on `Linux` systems and `Bash`, a Linux scripting language. In this lesson, we will also learn to use a terminal.

Some history

In the 80s, MIT researcher Richard Stallman, created an open and free operating system named GNU based on Unix. Around the same time, Linus Torwald, a Finnish student created an operating systeme called Linux. Eventually, the two projects merged under the new name **GNU/Linux**, better known as simply **Linux**. On this basis, some distributions (also named "distros") of GNU/Linux were created:

- Debian
- Red Hat
- CentOS
- ...

You can think of them as flavors of Linux. On top of that, people have created distributions of distributions: One of the most used is **Ubuntu** which is based on **Debian**. It is certainly the most user friendly distribution.

Thanks to a large community of users and contributors, Ubuntu is frequently updated and very well documented. It is certainly the easiest to use coming from a different OS (Operating System). In this training, we are going to use `Ubuntu 18.04`.

Ubuntu file structure

In a Ubuntu system, files are stored in a tree structure: the following example shows two sub-folders of one folder:

```
folder1

- subfolder1
- subfolder2
```

If the folder `subfolder1` contains two files, `file1` and `file2`, then it will look something like that.

```
folder1

- subfolder1
- - file1
- - file2
- subfolder2
```

In the / (simply called root) folder, we can find the following folders :

- /bin : essential binary files (Bash shell, files manipulation, ...)
- /boot : files required to start the system
- /cdrom : temporary folder for CD-ROM
- /dev : special files for a set of tools (console, stdout, ...)
- /etc : filesystem configuration files
- /home : user data
- /lib : library files to run binary files in /bin and /sbin
- /lost+found : corrupted files after a system crash
- /media : external temporary folders (USB key, ...)
- /mnt : mounted directories
- /opt : files that do not respect normal system structure
- /proc : files that contain system information
- /root : home folder of the root user
- /run : temporary files that should not be deleted
- /sbin : binary files used by root user
- /srv : files used by service providers (web app, ...)
- /tmp : temporary files that can be deleted
- /usr : user application (divided in /bin, /lib, ...)
- /var : log files.

First commands

You may be used to Windows or MacOs that are packaged with a graphic interface. In Linux, it is not always the case but we can use the terminal to perform some queries.

> Open a terminal window and connect to your virtual machine using the information in the Connect menu

You should see the following information:

- username : your current user name
- @ : a separator
- machine_name : the name of the computer/virtual machine
- : : a separator
- ~ : your current working directory
- $: a separator

Note that the $ corresponds to a normal user. If you are a root user (admin user), it is replaced with #.

By default, a terminal opens in the /home/username directory, which is aliased ~ when connected as username. To check where you are at, you can use pwd which stands for print working directory.

> Type this command and type enter to run this command

```
1 | pwd
```

On your virtual machine, this should return:

```
1 | /home/ubuntu
```

To list the content of a directory, we can use ls:

> Run this command

```
1 | ls
```

Right now the folder is empty. We are going to print the content of an another folder:

> Run this command

```
1 | ls /
```

This command prints the content of the / folder. We can pass arguments to a command by separating adding a space and the argument.

> Run a command to print the content of the /usr/bin folder

Show / Hide solution

```
ls /usr/bin
```

In those two examples, we have used the **absolute path**, meaning the path from the root directory `/` to the file/diretory we are interested in. `ls` can also take some predefined arguments:

- `-a` : to list every files even the hidden ones
- `-l` : to display files and folder informations
- `-R` : to list files recursively
- `-h` : to display the size of the files in a human readable fashion

> Try the following command to list folder information and hidden files in the `~` directory

```
ls -l -a ~
```

Remember that `~` stands for `/home/ubuntu` when you are connected as `ubuntu` user.

Note that we could have also used:

```
ls -la ~
```

To display the complete information on `ls` command, we can use `--help` flag:

> Run this command

```
ls --help
```

A lot of tools contain this `--help` option. Feel free to use it whenever you want to get more information on a command. We can also use `man` to display the help. Then you can scroll through the help and quit it by typing `q`.

> Try this out on `ls`

```
man ls
```

When we ran `ls -la ~`, we can see two things:

```
.
..
```

Those two things are not really folders nor files. They are used to refer to the current folder (`.`) and the parent folder `..`. Those are really useful: they can be used to create **relative paths**. Relative paths are used to designate an object from the current directory instead of the absolute path that starts from the `/` root directory.

For example, to access the folder `/usr/bin` from the `/home/ubuntu` directory, we can use the absolute path `/usr/bin` or the relative path `../../usr/bin` because the filesystem structure is:

```
/

- home
- - ubuntu
- usr
- - bin
- ...
```

> Run a command to display the content of the parent folder of the current working directory

```
ls ..
```

Of course, as with a file explorer, we can change the current working directory: to do so, we can use `cd` which stands for `change directory`, followed by the path to the directory we want to go to:

Run this command to get to the `/` folder

```
cd /
```

Check where you are by using `pwd`

Navigate to the `/usr/bin` folder

Show / Hide solution

```
# first way - relative
cd /
cd usr
cd bin

# second way - relative
cd /home/ubuntu
cd ..
cd ..
cd usr
cd bin

# third way - absolute
cd /usr/bin
```

When used alone, `cd` gets you back to your home folder.

Run `cd` and check the current directory with `pwd`

In this lesson, we have seen how to navigate through Linux filesystem. Note that MacOS is based on Unix and the commands used in this lesson can also be used in MacOS terminal. Windows has its own functions but you can use Powershell or install `gitbash` to have a console with Linux commands.

Validated