Introduction to GNU Linux CLI

Dmytro Strunin, MOBA BioInf

What is CLI?

CLI stands for **C**ommand **L**ine **I**nterface, usually shortened to *command line* or, by historical reasons, into *terminal*. The former dates back to the days when **UNIX** ran on large mainframes accessed via remote terminals.

What is shell?

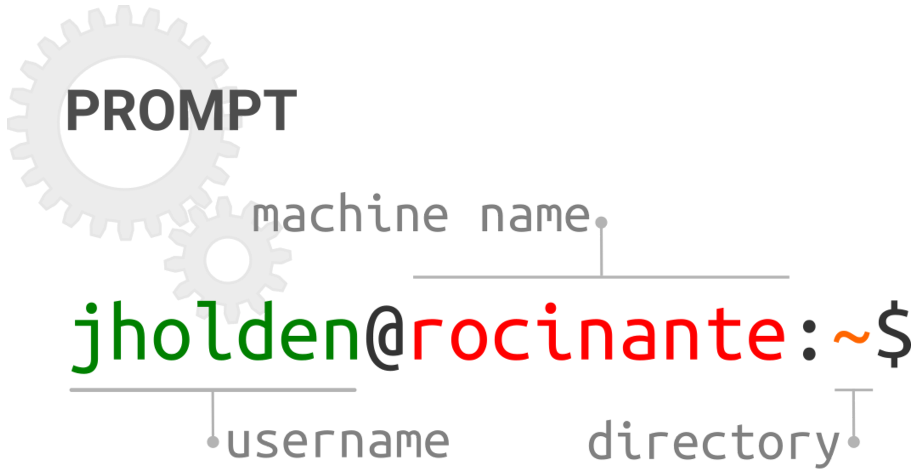
Shell is a program which serves as an interface between you and the **o**perating **s**ystem. It runs in your **terminal emulator** and allows you to interact with the *kernel* of your system.

There are *many* such programms, which can be qualified as *shells*: **GUI**, **sh**, **csh**, **ksh**, **zsh**, **c**, **bash**.

The **bash** shell is the most popular shell used across most of the Linux distributions.

What is prompt?

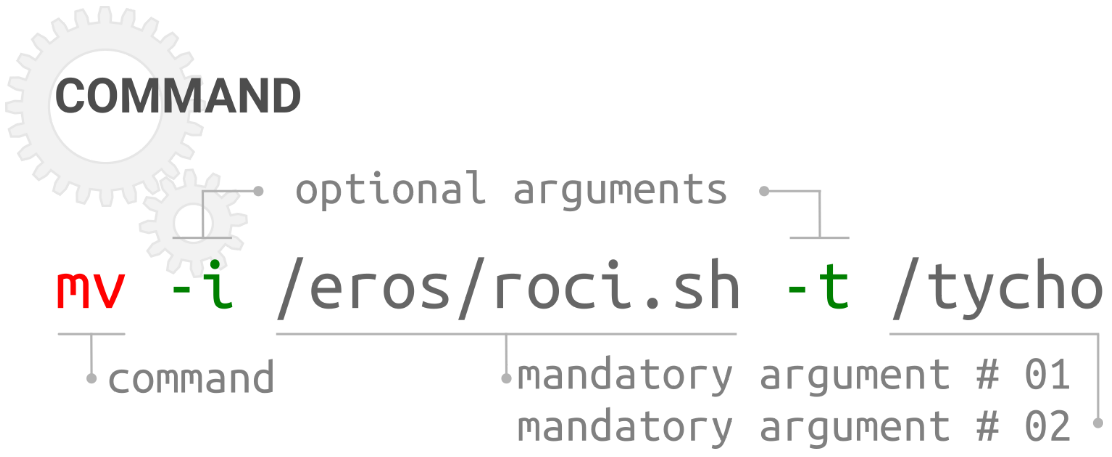
The *prompt* is what you see after you've logged into the server. The prompt is called *prompt* because it *prompts* you for your *command*:



*~ is a shorthand for the user's **home** directory*

What is command?

Command is a single-purpose *tool*, a *program* which is designed to do a *single task*, and do it well.



Command accepts zero or more *mandatory* and zero or more *optional arguments* which allows you to *get the desired result*, and *fine-tune the output* of the command.

Echo

Intro

`echo` simply prints into the terminal window anything you give it as an *argument*:

```
echo Hello, bash!
echo echo
```

In the example above `Hello, bash!` and second `echo` are *arguments* the command `echo` *evaluates*.

Hands-on

Print a string *O Romeo, Romeo! Wherefore art thou Romeo?* into your screen.

Brackets expansion

Intro

A *brackets expansion* is a nice feauture of the bash which you might find very interesting. To use brackets expansion, separate the arguments you are going to *expand* by the comma, and enclose them to the curly brackets `{}`.

```
echo file_{a,b,c}.txt
echo file_{1..5}.txt
echo file_{1..5}{a,b,c}.txt
echo {a..z..2}
```

Hands-on

Find the way to print all odd numbers within the interval [1, 50]

Print working directory

Intro

`pwd` stands for **p**rint **w**orking **d**irectory. When you enter this command **bash** will print your *current or working directory*.

```
pwd
# /home/jholden/
```

Look at your prompt, you are probably seeing `~` (*tilde*) symbol instead of your home directory *path* I have just found with `pwd`. In Linux CLI `~` is a short-hand for the *home directory*.

Hands-on

Print your working directory into the terminal.

Moving between directories

Intro

`cd` command stands for **c**hange **d**irectory. With `cd` you can move from your *current directory* to any directory you want, if you provide a *path* to the target directory as an argument.

```
cd /data3      # change your working directory to data3
pwd            # print your working directory
```

Hands-on

- Move from your *working directory* to the `/data3/` directory;
- Move back to your home directory using `~` instead of the *full path* to your home directory;

Relative vs absolute paths

Intro

Use `.` to shorthand your *current* directory, and `..` to shorthand the *parent* directory of your current directory.

```
cd ./projects  # Move to the 'projects' directory inside your working directory
cd ../rawdata  # Move to the 'rawdata' which is located in the parent directory
```

Hands-on

Move one directory above (parent directory) from your current directory. Use `pwd` to veryfy your location, and move back to your home directory using `~` instead of the *full path*.

Listing the directories

Intro

`ls` stands for **lis**t. The command general format is `ls [OPTION] ... [FILE]...`
By default, `ls` will list the content of your current directory.

- `-a`, `--all` do not ignore *hidden* files;
- `-l` use a long listing format;
- `-h` human-readable (e.g. show file size in Kb or Mb);

```
ls -alh
```

Hands-on

List content of your home directory.

Making directories

Intro

To make a directory use `mkdir` command (stands for **make dir**ectory). It accepts the *path* to the directory you are going to create, and several optional arguments.

```
mkdir sandbox                           # create a new directory 'sandbox'
ls -lh                                  # list files and directories
cd sandbox                              # change directory to 'sandbox'
pwd                                     # print working directory
mkdir -p ./data/in                      # create subdirectory
mkdir -p ./src/r                        # create subdirectory
tree                                    # explore the dir structure
```

Hands-on

Create a subdirectory `sandbox` with `data` subdirectory in in your home directory with *one command*.

Removing files and directories

Intro

The command `rm` is used for removing *files* or *directories*. To remove directories containing other files and directories, use `-r` or `-R` switch.

```
rm ./sandbox                     # Returns an error since directory is not empty
rm -rv ./data/raw02              # Succesfully removes directory with its subdirectories
```

Above we use `-v` switch to make `rm` verbose and `-r` switch to recursively remove the content of the directory.

Hands-on

Remove whole `sandbox` directory and it's subdirectories.

Note about the project structure

```
sandbox
├── data
│   ├── in
│   │   ├── csv
│   │   ├── fasta
│   │   ├── gbff
│   │   └── txt
│   └── out
│       ├── csv
│       ├── fasta
│       └── txt
└── src
    ├── r
    └── sh
```

More information: Vince Buffalo, 2015 and Noble, 2009.

Using brackets expansion with mkdir

Intro

We can use brackets expansion to create a above described directory structure with one command.

```
mkdir -pv sandbox/{src/{sh,r},data/{in/{csv,txt,fasta,gbff},out/{csv,txt,fasta}}}
```

Hands-on

Repeat the example above. After that create a subdirectory `img` in `./sandbox/data/out/` directory. `img` must contain subdirectories `png`, `pdf`, `svg`. Use *brackets expansion*.

Creating files

Intro

You can create an empty file with `touch` command. The command accept several optional arguments, and mandatory argument(s) - a file name(s) you are going to create:

```
touch logfile
tree
```

Hands-on

It's usefull to have a ReadMe.md file in a root of your project directory, to document your work. Create a ReadMe.md file in a `sandbox` directory.

Copying files

Intro

`cp` stands for **co**py. It accepts the path to the file or directory you want to copy, the destination path, and several optional arguments (e.g. `-R`, `-r` for *recursive*, `-t` for *destination* path).

```
cp /home/dst20/cli00/sandbox/data/txt/ncbi_links.txt -t ./data/in/txt
```

Hands-on

Repeat the command above to copy the `ncbi_links.txt` file. But instead of copying it into your `./data/in/txt` directory, copy it into `./data/in/csv` directory. Verify that file has been copied succesfully.

Moving files

Intro

Use `mv` command (stands for **mo**v**e**) for moving files and directories. This command accepts path(s) to the files (directories) you are going to move, and destination path(s) as well as several optional arguments.

```
mv -v ./data/in/csv/ncbi_links.txt -t ./data/in/txt
```

We use `-v` switch to make `mv` verbose, and `-t` switch to specify *target* directory.

Hands-on

You realized that you copied a `.txt` file into the wrong (`.csv`) directory. Use the example above to move `ncbi_links.txt` into the directory it belongs to. Don't forget to verify that the operation was sucessful.

Renaming files and directories

Intro

The fact that we can specify the name of our *destination file* allows us not ony to *move* but also to *rename* files and directories with `mv`:

```
mv -v file_with_old_name.txt file_with_new_name.txt
```

Notice that we use `-v` switch to make `mv` verbose.

Hands-on

Rename your `./data/in/txt/ncbi_links.txt` file into `./data/in/txt/ncbi_pa_assemblies_gbff.txt` using example above.

Reading the files with cat

Intro

The simplest command you can use for reading the content of the files is `cat`. Supply the `cat` with the path to your file, and read the content of the file on the screen.

```
cat ./data/in/txt/ncbi_pa_assemblies_gbff.txt
```

Hands-on

Using command above try to find out if all of the first ten links point to the `.gbff.gz` files.

Heads and tails

Intro

We can use `head` and `tail` commands to print the head and tail of the text file respectively. Commands accept few useful arguments for instance `-n` argument followed by the number of lines you want to print instead ten first (last) lines (default).

Hands-on

- Print only first 5 lines of `./data/in/txt/ncbi_pa_assemblies_gbff.txt` file;
- Print only last 5 lines of `./data/in/txt/ncbi_pa_assemblies_gbff.txt` file;

Pipes

Intro

What if want *to chain* commands? Can we *pipe* the output of one command into another command? We can use `|` known as a *pipe operator* for this purpose. Imagine we are interested only in *last 50 lines* of our `./data/in/txt/ncbi_pa_assemblies_gbff.txt` file, however not all of them, but only *first 3 lines*?

```
tail ./data/in/txt/ncbi_pa_assemblies_gbff.txt -n 50 | head -n 3
```

Hands-on

Repeat the example avbove, but this time, try to find what are the first 2 lines of the last 123 lines of the `./data/in/txt/ncbi_pa_assemblies_gbff.txt` file.

more on reading the files

Intro

Instead of `cat` we can use `more` to read text files (and more). The basic workaround is the same as for `cat`, however `more` accepts more optional arguments, and allows you to navigate through the text document in more comfortable way.

```
more -d ./data/in/txt/ncbi_pa_assemblies_gbff.txt
```

Hands-on

Repeat the command above to see `more` in action.

less on reading the files

Intro

The `less` is `more`! With tones of options, it allows you to scroll up and down through the document, and even to *search* a specific string: just hit `/`, type your string and hit `<ENTER>`. `more` will highlight all the patterns it found in the document, and you can jump from one match to another pushing the `n` key!

```
less ./data/in/txt/ncbi_pa_assemblies_gbff.txt
```

Hands-on

Browse the `./data/in/txt/ncbi_pa_assemblies_gbff.txt` with `more`, and locate the link to the `Pseudomonas_aeruginosa_PAKAF` assembly.

To be continued...