

A (very) short introduction to command line interface

Linux/Unix commands

Guillaume Duvillé
Henallux - UIB

2020 - 2021

1. First touch
2. Generalities
3. Exploring
4. Modifying
5. Executing a command

The interface

```
mfreeze@sonic:/tmp/workshop$  
mfreeze@sonic:/tmp/workshop$ whoami  
mfreeze  
mfreeze@sonic:/tmp/workshop$ cat /etc/hostname  
sonic  
mfreeze@sonic:/tmp/workshop$ █
```

- ① Bash **prompt** (username@hostname:location\$)
- ② After prompt, **commands** can be executed (whoami)
- ③ Result of commands whoami
- ④ Command (cat) with parameter (/etc/hostname)
- ⑤ Result of command (cat /etc/hostname)
- ⑥ New prompt

```
command [-opts] [--long-opts] param1 [param2 ...]
```

① **Case sensitive**

② **Options** usually aims at modifying command behaviour.

③ Example:

- `ls`: list content of the current directory
- `ls /tmp`: list content of the `/tmp` directory
- `ls -a`: list content of the current directory including hidden files
- `ls --all`: same as `ls -a`

1. First touch
2. **Generalities**
3. Exploring
4. Modifying
5. Executing a command

File tree structure (1)

```
/
|- bin/
|  |- ls
|  |- cd
|  `-- pwd
|- home/
|  |- mfreeze (*)/
|  |  |- poema/
|  |  |  |- slides.pdf
|  |  |  `-- bash.sh
|  |  `-- emptydir/
|  `-- olaf/
|      `-- file
`-- tmp/
```

- Unix/Linux filesystem resembles a **tree structure** (rooted at /)
- Every shell is **located** in the tree (usually user directory).
- . refers to the **current location**
- .. refers to the **parent directory**
- ~ refers to the personal directory
- files whose name beginning with . are **hidden files**.

File tree structure (2)

```
/
|- bin/
|  |- ls
|  |- cd
|  `-- pwd
|- home/
|  |- mfreeze (*)/
|  |  |- poema/
|  |  |  |- slides.pdf
|  |  |  `-- bash.sh
|  |  `-- emptydir/
|  `-- olaf/
|      `-- file
`-- tmp/
```

- Each file can be identified either by:
 - **absolute path**: location in the tree starting from /
 - **relative path**: location in the tree starting from current directory
- slides.pdf absolute path: /home/mfreeze/poema/slides.pdf
- slides.pdf relative path: poema/slides.pdf
- ls absolute path: /bin/ls
- ls relative path: ../../bin/ls

1. First touch
2. Generalities
3. Exploring
4. Modifying
5. Executing a command

Exploration commands

- `cd [dest_dir]`
 - change directory
 - **set current location** to `dest_dir`
 - if no `dest_dir` is given, set it to personal directory
- `ls [-a] [file1] [file2 ...]`
 - list information of files
 - if `file1` is a directory list its content (**without hidden files**)
 - if no `file1` is given, list **current directory**
 - `-a` prints hidden files
- `pwd`
 - print working directory

Discovery script

- Download and unzip the archive
- Open a terminal (if not already done)
- Start a fresh bash instance: `bash`
- Source the discovery workshop: `source path/to/discovery_mode.sh`
- Follow instructions
- Quit the bash instance with `exit`.

1. First touch
2. Generalities
3. Exploring
- 4. Modifying**
5. Executing a command

File creation commands

- `touch path/to/file`
 - creates a **new file**
- `mkdir [-p] path/to/dir`
 - creates a **new directory**
 - if called without `-p`, the parent `path/to/` must exist
 - if called with `-p`, every missing directory is created. Similar to:
 - `mkdir path`
 - `mkdir path/to`
 - `mkdir path/to/dir`

Copy commands

- `cp [-R] src dest`
 - **creates** `dest` which is a **copy** of `src`
 - `dest_dir` **musts not exist**,
 - if called with `-R`, copy is recursive.
- `cp [-R] src1 [src2 ... srcn] dest_dir`
 - **copies** `src1, src2, ...` into `dest` directory. **Creates:**
 - `dest/src1`
 - ...
 - `dest/srcn`
 - `dest_dir` **musts exist**,
 - if called with `-R`, copy is recursive.

File deletion commands

- `rmdir [-p] path/to/dir`
 - removes an **empty directory**
 - if called with `-p`, it also removes the parents. Similar to:
 - `rmdir path/to/dir`
 - `rmdir path/to`
 - `rmdir path`
- `rm [-r] path/to/file`
 - removes a **file**
 - if called with `-r`, remove a directory **recursively**
 - please be careful no confirmation is needed by default.

Modification commands

- `mv src dest`
 - **moves** `src` into `dest`,
 - if `dest` does not exist, `src` is **moved and renamed** `dest`,
 - if `dest` exists and is a directory, `dest/src` is created and `src` is deleted,
 - if `dest` exists and is not a directory, **overwrites**
 - `mv` is recursive.

- `mv src1 [src2 ... srcn] dest_dir`
 - **moves** `src1, src2, ...` into `dest` directory.
 - `dest_dir` must exist,

Build script

- Download and unzip the archive
- Open a terminal (if not already done)
- Start a fresh bash instance: `bash`
- Source the discovery workshop: `source path/to/buildmode.sh`
- Follow instructions
- Quit the bash instance with `exit`.

Other useful commands

- `cat file1 [file2 file3 ...]`
 - prints content of files given in parameters,
 - if multiple files are given, it concatenates the contents
- `man command`
 - display the manual page of the command,
 - RTFM!

1. First touch
2. Generalities
3. Exploring
4. Modifying
5. Executing a command

How it works

- To **execute** a file just call its name (relative or absolute):
`'path/to/executable/file'`
- Commands are generally executable files,
- The command `which` returns the file executed by a command:
 - `which ls` return `/usr/bin/ls`
 - `which which` return `/usr/bin/which`
- How do we go from `ls` to `/usr/bin/ls` ?

- Regular bash **variables**
- Monitor, defines and/or modifies bash environment and its behaviour.
- List of variables is given by `env` command
- Quite a large number:
 - `$HOME`: personal repository of the current user
 - `$SHELL`: default shell for current user
 - `$PWD`: current work directory
 - ...
 - `$PATH`: list of directories that will be **searched to find executables**.

The PATH variable

- Contains a **list of directories** separated by `:`,
- **Order matters!**
- Content can be display with command: `echo $PATH`
- `path_part.sh` introduces command `print_path` to nicely display the `PATH` variable:
 - source the file `source path/to/path_part.sh`
 - call the command `print_path`
- To add new directory modify the variable:
 - `export PATH=$PATH:/path/to/dir`

Permanent modification

- Modifications on `PATH` variable last **as long as bash instance**.
- To make “permanent” modifications, we generally use a `bashrc` file:
 - a **bash script** that acts like a **configuration file**,
 - it is read **any time a new bash instance is created**,
 - can be found at multiple locations including:
 - `/etc/bash.bashrc`: system-wide configuration
 - `$HOME/.bashrc`: per-user configuration
- Just add `export PATH=$PATH:/path/to/dir`.

Julia installation

- Usually in the packages repositories,
- Latest stable version can be downloaded:
`https://julialang.org/downloads/`
- If needed, installation instruction can be found:
`https://julialang.org/downloads/platform/`
- Don't hesitate to ask for help,
- You should have a working instance of Julia after this slide.