

GNU/Linux

Command line

Nicolas Delanoue

Université d'Angers - Polytech Angers



POLYTECH[°]
ANGERS



- 1 Shell
- 2 Using the file system
- 3 User account
- 4 Permissions et users

Definition

A *shell* is a software that interprets commands.

Remarks

- There are several shells. `Bash` is one, but there are others like for example `sh`.
- A shell is executed in a *console* which also called a *terminal*.

Definition

A *command line* is an instruction given to a shell.

What is this for ?

Everything !

Definition

The *prompt* is a message prompting you to type a command.

```
nico@pc:~$ ls
adminstratif2      gurobi.lic      perso
Animation_scientifique gurobi.log      perso2
articles           images          recherche
Bureau            liste_pour_tim.txt~ Steam
congres_seminar   livre_de_vadim.txt Téléchargements
dosbox            livres          tmp
dosbox.conf       monlivre        tout.pdf
eclipse           monter_partage  video
nico@pc:~$
```

Keyboard shortcuts

Tab completes the file / directory name if there is only one that matches

Tab Tab in the case where a single tab press did not display anything, displays the list of all possibilities,

Arrow _Up recalls the previous command,

Shift Page _Up goes up one page in the console,

Ctrl C stop the process running in the console (sending the SIGINT signal to the process),

Ctrl Z suspends the running process in the console (sending the SIGSTOP signal to the process),

Ctrl L clears the terminal screen,

Selection selects the text (= copy)

Middle Click inserts it at the current position of the text cursor (= paste)

Organization

- Hierarchical file system, ie a tree structure.
- Files and directories are entries.

Constraints with ext4 Linux

- Names are case sensitive
- File names do not necessarily have an extension

Note for the teacher : build with the students a system of dummy files.

Organisation du système de fichiers de Linux

- `/bin` basic system executable commands
- `/boot` Linux kernel boot files
- `/dev` machine peripherals as files
- `/etc` system configuration files
- `/home` home directories for users to save their documents,
 - `/lib` shared libraries and kernel modules
- `/media` mount point for removable devices
- `/mnt` standard mount point for external systems
- `/opt` additional applications
- `/proc` system resource information
- `/root` home directory of the super-user (root)
- `/sbin` executable commands reserved for root
- `/tmp` temporary directory
- `/usr` contains applications, shared libraries, documentation
- `/var` the “ variable ” part of the system : log files, mail queues, printer ...

Definition

A *path* identifies a file or a directory.

Remarks

- A path
 - lists all crossed directories,
 - terminates in the destination.
- The crossed directories are separated by /

Definition

A path that starts from the root / is called *absolute*.

Examples of absolute paths :

- /home/etudiant/
- /bin/find
- /root/doc/presentation.ppt

Special paths

The **current directory**, noted `.`, is the one in which we work.

The **parent directory**, noted `..`, is the one above the current directory,

The **home directory**, accessible via `~`, is the directory of documents of the logged user.

Remarks

Paths `.` et `..` are relative paths with respect to the current folder.

Exercice

Let us suppose that the current folder is `/home/etudiant`, give the absolute paths of the following files :

- 1 `polytech/rapportTP.doc`

Exercice

Let us suppose that the current folder is `/home/etudiant`, give the absolute paths of the following files :

① `polytech/rapportTP.doc`

solution : `/home/etudiant/polytech/rapportTP.doc`

Exercice

Let us suppose that the current folder is `/home/etudiant`, give the absolute paths of the following files :

- 1 `polytech/rapportTP.doc`
solution : `/home/etudiant/polytech/rapportTP.doc`
- 2 `../prof/texteTP.pdf`

Exercice

Let us suppose that the current folder is `/home/etudiant`, give the absolute paths of the following files :

- 1 `polytech/rapportTP.doc`
solution : `/home/etudiant/polytech/rapportTP.doc`
- 2 `../prof/texteTP.pdf`
solution : `/home/prof/texteTP.pdf`

Exercice

Let us suppose that the current folder is `/home/etudiant`, give the absolute paths of the following files :

- 1 `polytech/rapportTP.doc`
solution : `/home/etudiant/polytech/rapportTP.doc`
- 2 `../prof/texteTP.pdf`
solution : `/home/prof/texteTP.pdf`
- 3 `/usr/bin/firefox`

Exercice

Let us suppose that the current folder is `/home/etudiant`, give the absolute paths of the following files :

- 1 `polytech/rapportTP.doc`
solution : `/home/etudiant/polytech/rapportTP.doc`
- 2 `../prof/texteTP.pdf`
solution : `/home/prof/texteTP.pdf`
- 3 `/usr/bin/firefox`
solution : `/usr/bin/firefox`

Exercice

Let us suppose that the current folder is `/home/etudiant`, give the absolute paths of the following files :

- 1 `polytech/rapportTP.doc`
solution : `/home/etudiant/polytech/rapportTP.doc`
- 2 `../prof/texteTP.pdf`
solution : `/home/prof/texteTP.pdf`
- 3 `/usr/bin/firefox`
solution : `/usr/bin/firefox`
- 4 `./jeux/stupides/pileouface`

Exercice

Let us suppose that the current folder is `/home/etudiant`, give the absolute paths of the following files :

- 1 `polytech/rapportTP.doc`
solution : `/home/etudiant/polytech/rapportTP.doc`
- 2 `../prof/texteTP.pdf`
solution : `/home/prof/texteTP.pdf`
- 3 `/usr/bin/firefox`
solution : `/usr/bin/firefox`
- 4 `./jeux/stupides/pileouface`
solution : `/home/etudiant/jeux/stupides/pileouface`

Exercice

Let us suppose that the current folder is `/home/etudiant`, give the absolute paths of the following files :

- 1 `polytech/rapportTP.doc`
solution : `/home/etudiant/polytech/rapportTP.doc`
- 2 `../prof/texteTP.pdf`
solution : `/home/prof/texteTP.pdf`
- 3 `/usr/bin/firefox`
solution : `/usr/bin/firefox`
- 4 `./jeux/stupides/pileouface`
solution : `/home/etudiant/jeux/stupides/pileouface`
- 5 `..`

Exercice

Let us suppose that the current folder is `/home/etudiant`, give the absolute paths of the following files :

- 1 `polytech/rapportTP.doc`
solution : `/home/etudiant/polytech/rapportTP.doc`
- 2 `../prof/texteTP.pdf`
solution : `/home/prof/texteTP.pdf`
- 3 `/usr/bin/firefox`
solution : `/usr/bin/firefox`
- 4 `./jeux/stupides/pileouface`
solution : `/home/etudiant/jeux/stupides/pileouface`
- 5 `..`
solution : `/home/`

Exercice

Let us suppose that the current folder is `/home/etudiant`, give the absolute paths of the following files :

- 1 `polytech/rapportTP.doc`
solution : `/home/etudiant/polytech/rapportTP.doc`
- 2 `../prof/texteTP.pdf`
solution : `/home/prof/texteTP.pdf`
- 3 `/usr/bin/firefox`
solution : `/usr/bin/firefox`
- 4 `./jeux/stupides/pileouface`
solution : `/home/etudiant/jeux/stupides/pileouface`
- 5 `..`
solution : `/home/`
- 6 `.`

Exercice

Let us suppose that the current folder is `/home/etudiant`, give the absolute paths of the following files :

- 1 `polytech/rapportTP.doc`
solution : `/home/etudiant/polytech/rapportTP.doc`
- 2 `../prof/texteTP.pdf`
solution : `/home/prof/texteTP.pdf`
- 3 `/usr/bin/firefox`
solution : `/usr/bin/firefox`
- 4 `./jeux/stupides/pileouface`
solution : `/home/etudiant/jeux/stupides/pileouface`
- 5 `..`
solution : `/home/`
- 6 `.`
solution : `/home/etudiant`

Some commands to know

Let us suppose that the current folder is `/home/etudiant`
`/home/etudiant` :

Where am I?	<code>pwd</code>	print working dir	<code>\$ pwd</code> <code>/home/etudiant/</code>
What is there here?	<code>ls</code>	list	<code>\$ ls</code> <code>Bureau rapportTP.odp</code> <code>cv.pdf</code>
Change folder	<code>cd</code>	change directory	<code>\$ cd /home</code> <code>/home/\$</code>
View the contents of a file	<code>cat</code>	catalog	<code>\$ cat /etc/passwd</code>

Copy	cp	copy	<pre>\$ cp rapportTP.doc ../prof \$ cp Bureau/linux.pdf . \$ cp -r /tmp/dossier .</pre>
Rename	mv	move	<pre>\$ mv comptes.txt compta.txt \$ mv Bureau/linux.pdf . \$ mv dossier/ Bureau/</pre>
Delete	rm	remove	<pre>\$ rm comptes.txt \$ rm Bureau/linux.pdf . \$ rm -rf /tmp/dossier/</pre>

Create an empty file	touch	<pre>\$ touch test0.txt \$ ls Bureau test0.txt</pre>
Create a folder	mkdir	<pre>\$ mkdir photos</pre>
Find a file	find	<pre>\$ find / -name "compt*"</pre>

puis :

diff, locate, whereis, ...

Definition

An *user account* is the set of resources assigned to a user or device (computer, peripheral ...).

Remarks

- A user can be a person or a role,
- one person can use multiple accounts, for example I use the `nico` account and the `root` account,
- an account can be used by more than one people.

Why multiple accounts ?

- Everyone wants to protect their data,
- not everyone can do everything on the machine.

Security mechanisms

- We protect the files.
- File permissions are granted to users.

Location of the list of user accounts

- The accounts are listed in the `/etc/passwd` file.
- The passwords are in the `/etc/shadow` file.

```
user@pc:~$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
...
gdm:x:108:118:gdm:/var/lib/gdm:/bin/false
nico:x:1000:1000:nicolas,,,:/home/nico:/bin/bash
```

Explanation of the last line of this file

- `nico` is the name of the user account,
- `1000` is the unique identifier of the user, also called `uid`,
- `nicolas , , ,` corresponds to the user's description,
- `/home/nico` is his home directory,
- `/bin/bash` is the program that runs at logon.

Accounts : create / delete

- with the GUI : easy
- from the command line :
 - useradd to add,
 - usermod to modify,
 - userdel to delete,
 - passwd to change the password.

Note

All these modifications impact the passwd files, shadow in the /etc directory.

Accounts : change user

The same person may, to fulfill different roles, have to change accounts

- graphically : disconnect or reconnect,
- from the command line : with the command `su` and `exit`.

Example :

```
nico@pc1:~$ su xavier
```

```
Password: *****
```

Result, we are connected as xavier :

```
xavier@pc1:/home/nico$
```

To make sure, use `whoami` :

```
xavier@pc1:/home/nico$ whoami
```

```
xavier
```

Accounts : change user

```
xavier $ su reivax
```

```
reivax $ ...  
reivax $ ...  
reivax $ su root -
```

```
xavier $ su reivax
```

```
reivax $ su root -  
root # ...  
root # ...  
root # exit
```

```
xavier $ su reivax  
xavier $
```

```
xavier $ su reivax
```

```
reivax $ su root -  
reivax $ ...  
reivax $ ...  
reivax $ exit
```

Two ways to become the super user, i.e. root

- 1 Change user with the command `su root` -
 - requires the password of the root account.
- 2 Use of the `sudo` mechanism
 - allows to execute a command under root, (just the time of the command)
 - requires authorization ...



FIGURE – And there I am the king!

sudo mechanism demonstration

```
nico@pc:~$ ls /dossier_sensible
```

```
ls: cannot open the '/ root' directory:  
Permission not granted
```

```
nico@pc:~$ sudo ls /dossier_sensible
```

```
[sudo] Password of nico :  
fichier_1_important.pdf truc_sensible.txt
```

Here, you must type the password of nico and not that of root.

Definition

A *user* is « somebody » of `/etc/passwd`.

Definition

A *group* is « something » of `/etc/group`. A group gathers users.

Propriétés du système de fichier ext4

- A user necessarily belongs to at list one group (which carries his name, in general),
- a file (or directory) has one user and one group (owners).

Definition

A file or directory has *permissions*

ext4 permissions

There are 3 sets of permissions :

- Regarding the owner noted *u*
- Concerning the group noted *g*
- Regarding the others noted *o*

Each permission set allows (or not) :

- The reading (or listing) noted *r*
- The writing (or creation) noted *w*
- The execution (or entry) noted *x*

```
user@pc:~/ $ ls -l
drwxr-x--- 27 nico g1      4096 nov.   5 09:25 articles
-rw-rw-r--  1 nico nico   1619 oct.   2  2017 monip.txt
```

Explanation

The file articles

- is a folder (because of the first **d**),
- is owned by user **nico** who can
 - list the files it contains (**r**)
 - create files inside this folder (**w**)
 - enter this folder (**x**)
- is attached to the **g1** group, users of the group can
 - list the files it contains (**r**)
 - not create files inside this folder (**w**)
 - enter this folder (**x**)
- and the rest of the world can't do anything.

```
user@pc:~/ $ ls -l
drwxr-x--- 27 nico g1      4096 nov.   5 09:25 articles
-rw-rw-r--  1 nico nico   1619 oct.   2 2017 monip.txt
```

Explanation

The file `monip.txt`

- is a regular file (because of the first `-`),
- is owned by user `nico` who can
 - open file (`r`)
 - modify this file (`w`)
 - not run this file (`x`)
- is attached to the `nico` group, users of the group can
 - open file (`r`)
 - not modify this file (`w`)
 - not run this file (`x`)
- and the rest of the world can only read the file.

Notation

Permissions can also be written in octal. Indeed, each permission corresponds to a value :

- The reading, noted r , corresponds to the value 4,
- The entry, noted w , corresponds to the value 2,
- The execution, noted x , corresponds to the value 1,

Example

The permission rwX $r-x$ $r--$ therefore corresponds to 7 5 4, effects

- rwX matches $4 + 2 + 1$, i.e. 7.
- $r-x$ matches $4 + 0 + 1$, i.e. 5.
- $r--$ matches $4 + 0 + 0$, i.e. 4.

Change permissions

The command to change permissions is `chmod`.

Remark

You can change the permissions if :

- We are the owner,
- on is root.

Examples

```
nico@pc:~/ $ chmod 750 rapport.doc
```

```
nico@pc:~/ $ chmod ug+w rapport.doc
```

We can use the `-R` option to call it recursively.

Change the owner

The command to change the owner is `chown`.

Remark

You can change the owner if :

- We are the owner,
- `on` is `root`.

Example

```
nico@pc1:~/ $ chmod nico:group document.txt
```

We can use the `-R` option to call it recursively.

Important commands

