IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Introduction to UNIX-like OSes and shell

DCL – 2025-2026

Creative Commons BY-NC-SA license

You are **free**:

- to **use**, **copy**, **distribute** and **transmit** this creation to the public;
- to **adapt** this work.

Under the following terms:

- **Attribution** (**BY**): you must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).
- **NonCommercial** (**NC**): You may not use the material for commercial purposes.
- **ShareAlike** (**SA**): if you modify, transform, alter, adapt or build upon this work, you may distribute the resulting work only under the same, similar or a compatible license.

Important notes

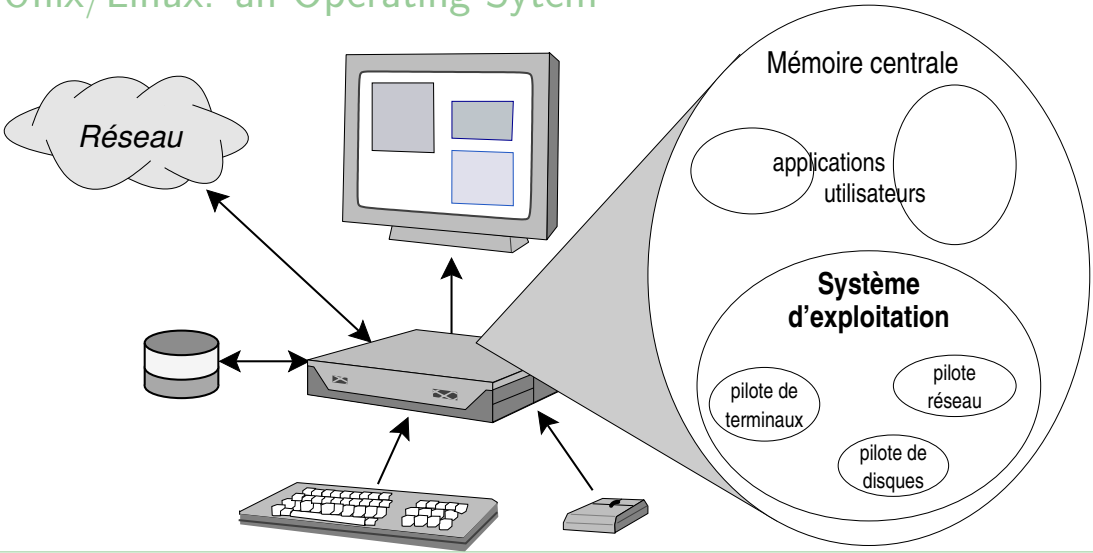
- If you do not understand something, please ask your questions. *We cannot answer the questions you do not ask. . .*
- If you disagree with us, please say it (politely)
- People don't learn computer science by only reading few academic slides: practicing is fundamental

Organization

- Morning
  - course (all): generalities to learn to understand your computer + shell (hopefully)
  - lab session (groups): environment installation (terminal, few useful tools, . . .)
- Afternoon
  - lab session (groups)
  - or shell course (all)i if I did not finished the course
- Note: if you feel comfortable with you computer and with command line, feel free to help your fellow students or to do something else (as long as you don't interfere with their work)

Progress

Unix/Linux: an Operating Sytem



Linux: a Unix system

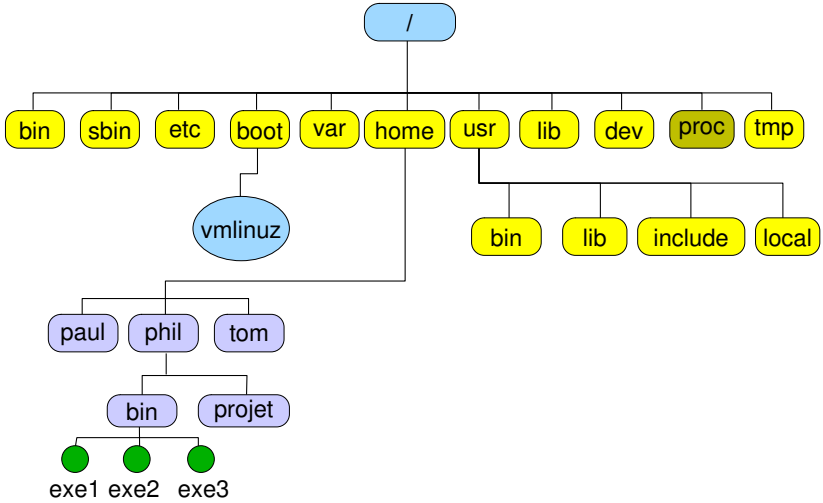
- Unix philosophy:
  - (almost) everything is a file
  - "Do one thing, do it well" (Doug McIlroy, l'inventeur des *pipes Unix*):
    - write programs that do one thing and do it well.
    - write programs to work together.
    - write programs that handle text streams, because that is a universal interface.
- Characteristics of a Unix operating system:
  - multi tasks (multi processus)
  - multi users
- Specific features (of Linux and of all Unix):
  - its file management system
  - its process management

→ Linux is (an implementation of) Unix

Unix Files

- Ordinary files
  - simple sequence of bytes, sometimes empty
- Directory
  - “file” containing references on files
  - allows one to create a tree structure of files and directories
- Link
  - reference on a file
- Special files
  - references on peripherals

The file system tree



Files – naming schemes

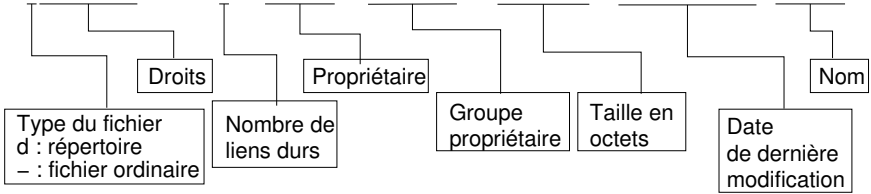
- Absolute name
  - relative to the root, the name starts with /
  - /home/phil/bin/exe1
- Relative name
  - relative to the current directory:

home/phil/bin/exe1	if we are in /
phil/bin/exe1	if we are in /home
bin/exe1	if we are in /home/phil
exe1	if we are in /home/phil/bin

Visualizing the content of a directory

- ls command
  - example:

```
[bash]$ ls -l
total
-rwxr-xr-x     1   clohr   ens-rec      7790   avr 11 2020 essai
-rw-rw-r--     1   clohr   ens-rec      1122   avr 24 2020 essai.c
-rw-rw-r--     1   clohr   ens-rec   9869052   avr 25 2020 test
```



Hidden files

- Files whose names begin with a period
  - example: .bashrc
- Not displayed by default by the tools used to show the content of directories
- Usually configuration files
- Equivalent to the registry database on alternative systems
- Hidden directories: their name also starts with a period
  - example: ~/.config/

Structure and content of a directory

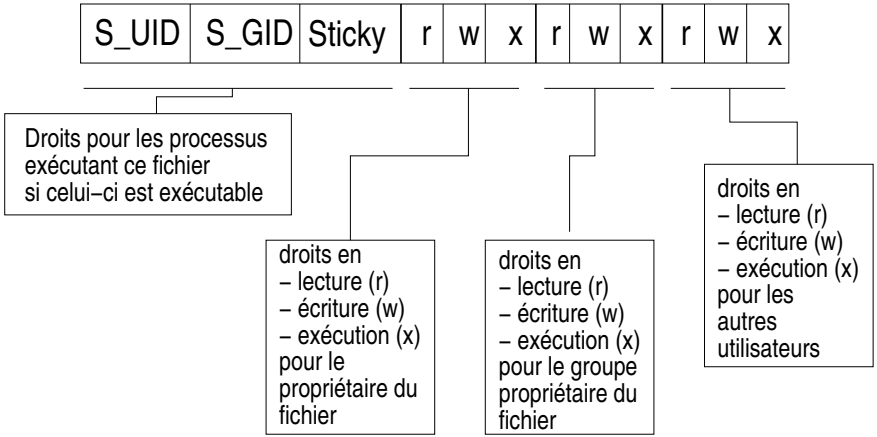
- A directory is primarily a file
- This file, of a specific type, contains references to files or subdirectories.
- A reference consists of:
  - the length of the reference
  - the length of the file name (< 256)
  - the file name
  - an index number called an **inode number** that identifies a structure in a table called the ***inode table***
  - This structure contains all the information about the file

“.” and “..” directories

- A directory is never empty, even when it is created, it already contains two references to directories named “.” and “..”.
  - the directory “.” (pieriod) is a reference on the directory itself.
  - the directory “..” is a reference on the directory immediately above it in the tree structure (the *parent* directory, so to speak)
- Usage
  - unambiguous naming of a local file `./test` for example
  - quick naming of a file above: `../fichier` for example

Files – permissions

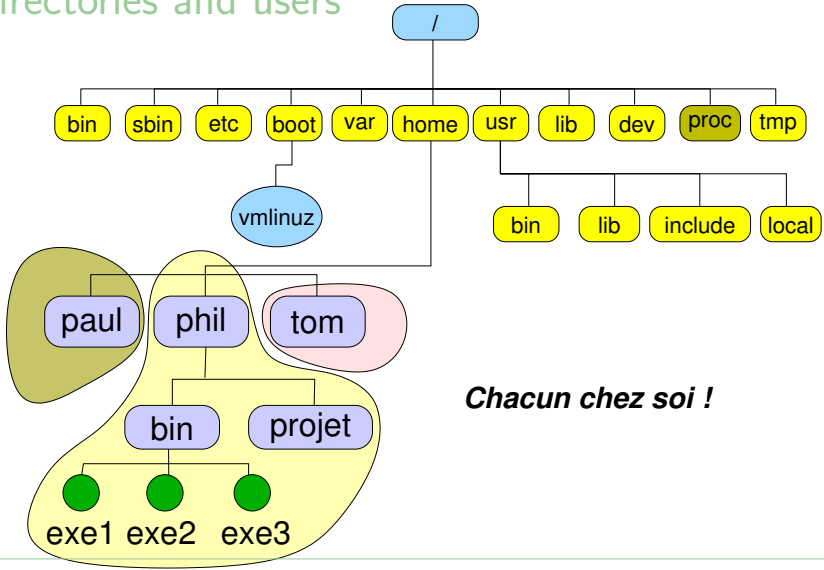
- In the mode field of the inode
  - 12 bits



Permissions on the directories

- These are the same types of rights as for files: `r`, `w` et `x` et même `t`
- However, the associated semantics is different
- Rights:
  - `r`: the directory is readable; its contents can be listed.
  - `w`: the directory is "writable," meaning you can create files or directories in it
  - `x`: The directory is not executable, but it is accessible: you can go into it or traverse it to access its contents (files, subdirectories)
  - `t`: The *sticky bit*, valid for a directory opened in `w` for everyone, indicates that only a file owner can delete that file.

Files, directories and users



Chacun chez soi !

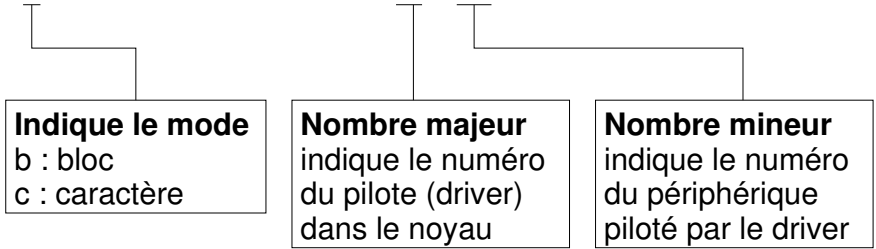
The special files

- They reference devices
  - Enable exchanges (reading/writing) with device drivers
  - Enable control of these devices
- Two types
  - Block mode devices
    - exchanges are made in blocks of bytes (by “pages”)
  - Character-mode devices, also known as transparent mode (more commonly referred to as *raw* mode)
    - data is exchanged byte by byte
  - Disks tend to be in block mode, while terminals are in *raw* mode

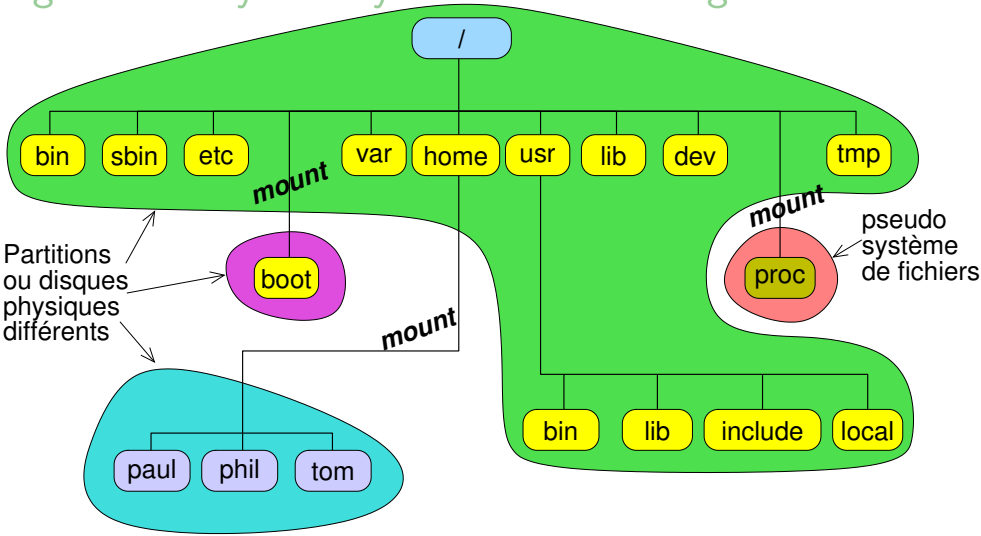
The special files – the /dev directory

- Input example in /dev

```
[bash]$ cd /dev; ls -l hda1
brw-rw---- 1 root  disk   3,  1  mar 24 2001  /dev/hda1
```



The global file system système de fichiers global



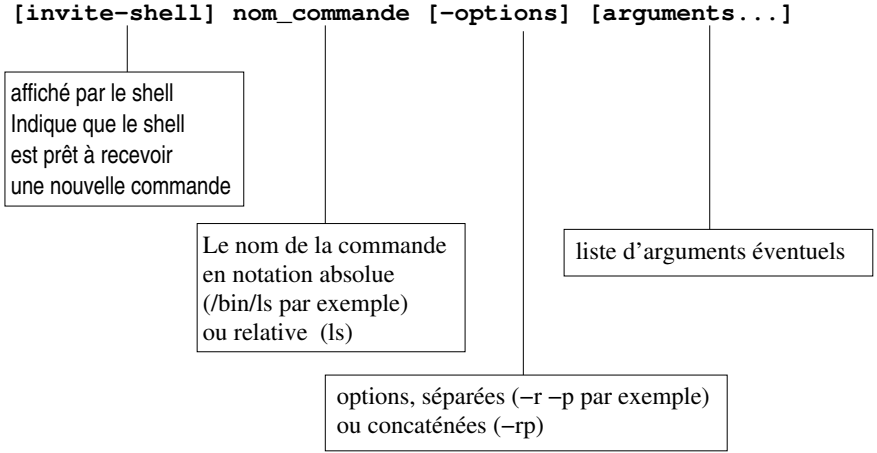
Progress

How to run a command

- Modern way
  - with the GUI
- Less modern way... but so effective!
  - via a terminal emulator (xterm, gnome-terminal, ...)
  - in a virtual console terminal (accessible via `Ctrl-Alt-F1` to `F6` from te graphical interface)
  - these methods launch a command interpreter process associated with the terminal, called *Shell*
  - the *Shell* is an interface between the keyboard and the system. It provides facilities for entering commands, but also a real programming language

The commandes

- General Syntax



The commands – the PATH variable

- The `PATH` is an environment variable
- It contains a list of directories where commands can be called by their shortest relative name: `ls` instead of `/bin/ls`, for example
- If the system tells you: "*command not found*," your `PATH` variable may be configured incorrectly...
- Configuration directly in the shell or in `$HOME/.bashrc`:
  - `export PATH=$PATH:/usr/local/bin` (for example)

The commands – a few refinements

- Background commands
  - the command line ends with the character `&`
  - the shell launches the command and immediately returns control without waiting for it to finish. The command continues to run in the background
- Redirections
  - redirecting the standard output file
    - *command* `>` *file*
  - redirecting the standard error file
    - *command* `2>` *file*
  - redirecting the standard output and error file
    - *command* `>` *file* `2>&1`

The commands – communication pipes

- To create filters
  - *command1* | *command2* | *command3* ...
  - the text normally displayed by *command1* is redirected to *command2* (it is read by *command2* and does not appear on the screen). The result is sent to *command3* and so on.
  - Exercise:
    - display the contents of the `/usr` directory using the `ls -l` command
    - using a pipe and the `head` command, display only the first 5 lines of the previous result
    - using another pipe and the `tail` command, display only the last line of the previous result
    - see the online manual for information on how to use these commands

The commands – wildcards

- A special character that replaces several others...
  - For file names
  - A kind of rational expression (but not POSIX)
- Example: `rm *` (remove the files within the directory)
- Recognized by the shell:
  - `?`: any character
  - `*`: zero or several characters
  - `[A1x]`: the character `A` or `1` or `x`
  - `[a-z]`: the characters from `a` to `z` (ASCII code)
  - `{ab,ac}`: the character string `ab` or `ac`

The classical Unix commands

(non-exhaustive list...)

- Create, browse files and directories
  - `ls cd pwd cp mv rm mkdir rmdir touch`
- Display — edit files
  - `more less` — `nano`
- Text filtering
  - `echo cat grep sort uniq sed tail tee head cut tr split paste printf`
- File comparison
  - `comm cmp diff patch`

The classical Unix commands

...

- Basic administration basique (user level)
  - `chmod chown ps su w who`
- Communication
  - `mail telnet ftp finger ssh`
- Shells
  - `sh csh ksh zsh bash tcsh fish`
  - Bash is the default shell on most Linux systems
  - Zsh is the default shell on recent MacOS systems

The commands – documentation

- The `man` command, know how to read man syntax
  - example: `man rm`

```
RM(1)          Manuel de l'utilisateur Linux     RM(1)
NOM
       rm - Effacer des fichiers.
SYNOPSIS
       rm   [-dfirvR] nom...
```

Nom de la commande

Liste des options
Entre [] signifie que ces opttions ne sont pas obligatoires.
(Si on les indique cependant, on ne mettra pas les crochets)

Le ou les arguments.
Ici les caractères ... indiquent que nom peut être répété plusieurs fois

File search tools

- The `locate` command
  - searches for occurrences of the character string passed to it as arguments in a database updated via the `updatedb` command
- The `find` command
  - searches for anything anywhere IN REAL TIME (slow)
- The `whereis` command
  - searches for the command name passed as an argument in a number of standard directories
- The `which` command
  - searches in the `PATH` where the command specified as an argument is located

Processes

- Concept of process
  - a process is a program that is running in a given environment in the main memory.
- Environment
  - a set of information that complements the running program and configures its execution
  - mainly three types of environment
    - environment variables
    - the identity of the user on whose behalf the process is running
    - open files (particularly standard input/output files)

Creation of a process

- A process is always created by the kernel, at the request of another process
- The process requesting creation is called the parent, and the created process is called the child
- The child process is created in main memory in a memory area separate from the parent process
- The child process is an exact copy of the parent process. However, a technical detail (PID: Process IDentifier) allows the developer to differentiate between the two processes.

Environment variables

- Character string, using uppercase by convention
  - syntax: `NAME=value`
  - grouped together in a memory space called the “environment variables table”
  - this table is inherited by child processes and persists when a command is executed (see below)
- Some standard variables
  - `PATH`, `HOME`, `USER`, `LOGNAME`, `DISPLAY`, ...

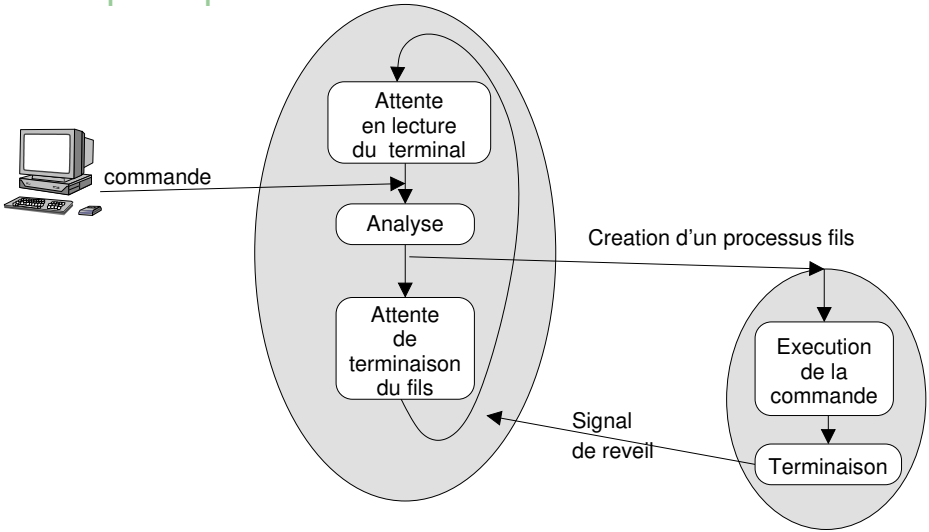
The standard input-output files

- Three standard files
  - the standard input file
    (descriptor 0, FILE Pointer stdin)
  - the standard output file
    (descriptor 1, FILE Pointer stdout)
  - the standard error output file
    (descriptor 2, FILE Pointer stderr)
- Open by default when launching an executable
- Virtually associated with the keyboard for standard input and with the screen for the other two
- They can be redirected to actual files or communication pipes

Process control

- List the processes
  - the `ps` command
    - many options: `ax`, `axl`, `axf`, `-ef`, etc.
  - the `top` command
    - like `ps axu`, with regular redisplay, plus information on memory load and usage
- Terminate a process
  - if you have control (foreground processes in a terminal)
    - without killing it (temporary stop): `<Ctrl-Z>` (`fg` to resume)
    - by deleting it: `<Ctrl-C>`
  - the `kill` command (out of the scope of this introduction)

The Shell principle



Conclusion

- An introduction to understanding your machine and discovering the command line, to demystify the terminal, not enough to become a *power user*
- Overview of uses and commands
- Mastering your machine and the terminal requires practice, time, and a little perseverance

Gentle reminder

⚠ Important notes

- if you do not understand something, please ask your questions. *We cannot answer the questions you do not ask...*
- if you disagree with us, please say it (politely)
- people don't learn computer science by only reading few academic slides: practicing is fundamental