Intro to Linux

- Lectures / Labs -

Scanné avec CamScanner

1 - Intro to Linux

- Operating system (there is no one formal definition)

→ A system software that allows hardware and software to communicate

→ A layer between hardware and software, it manages tasks and resources (hardware is limited so we need OS to distribute equaly the computer resources). Kernel : primary module / heart of an OS.

→ A layer of system that sits between the complex hardware and the user-level application.

- The role / functionalities of an operating system:

1 - Hides hardware complexity and diversity.

2 - Manages resources/hardware 3 - Multitasking : several tasks at a time

- Provides isolation and protection for applications.

- OS's main features: 5As

1 - Help / assistance

- hide complexity (API of devices are very complex)
- simplify access to devices / all features (even those that are rarely used).
- manage all situations.
- factorize reccuring
- modularity and maintainability of code.

* Drivers: set of APIs (provided for different software / network)

2- Abstraction:

- Different devices provide similar services.
- Different brand / hard drive capacity / magnetic tap.
- Reduced functionality.
- Increased reusability.
- Code portability.

3- Augmentation:

- Improved performance.
- Extand ressources virtually.
- Multitasking.

4- Arbitration:

- Distribute access to resources.
- Reliability of system.

5- Authorization management:

- Multi-user context.
- Privacy and system integrity.
- Limit access to resources.
- Security.

• Linux:

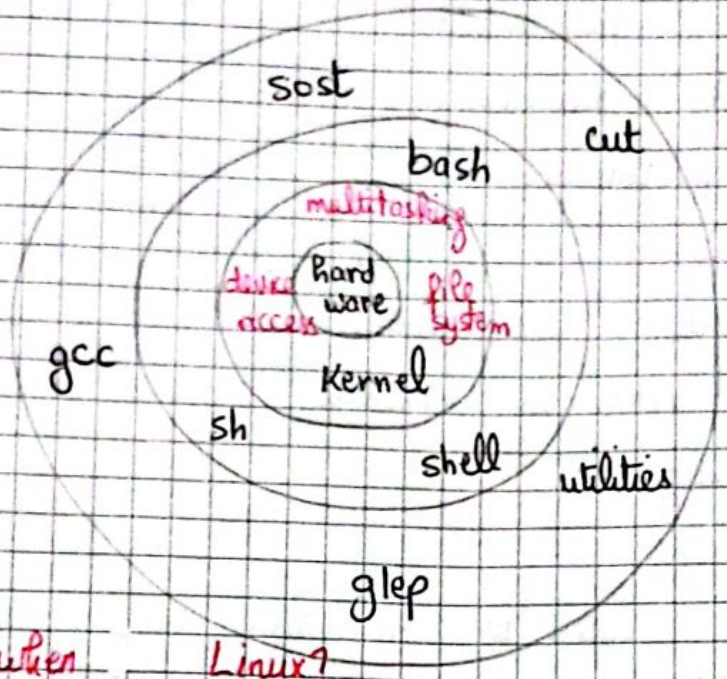
→ Linux is a Unix-like operating system assembled under the model of free and open source software development and distributions.

→ These operating systems share the Linux kernel having the same GNU utilities. (Kernel + GNU utilities = OS)

→ Several distributions = different purposes.

Scanné avec CamScanner

Kali → security ; Mint → education



Who and when Linux?

Linus Trovalds → Linux kernel
Richard Stallman → GNU utilities
} Linux OS

Where is Linux?

- World wide web: 67% of world's web servers use Linux
- Research / high-performance computing
- Linux runs on: IOT, enterprise businesses, websites and computers.

Why Linux?

- Free and open source
- Powerful for research datacenters
- Personal use (phones, desktop)
- Universal
- Community (business) driven

Scanné avec CamScanner

Linux : Many distributions

- Customized versions of Linux with the same Kernel.
- It's a collection of packages / applications grouped and pre-installed on standard Linux to meet specific requirements.

https://eylenburg.github.io/linux_comparison.htm

1) Beginners : easy to use and work out of the box.
   mint - ubuntu - solus.

2) Beginners to intermediate : some post-install tweaks needed (codecs) : Fedora - openSUZE - mageia.
   to install codecs video files

3) Advanced users : anyone can use it but needs reading the documentation and getting comfortable with the shell :
   debian - RHEL - Arch.

4) Experts : aditionnal distro-specific knowledge to learn :
   slackware - NixOS - Gentoo

Main creteria :

1) Target audience and systems (desktop system / old machine ..)
2) Process of installing and updating applications (package managment tools)
   APT for mint
3) Supports , updates : some distributions are maintained by a volenteers community , others by commercial vendor.

Most Popular distributions

Scanné avec CamScanner

Red Hat Entreprise Linux:

- Most popular commercialy
- Used by corporate organisations : server platforms due to its dedicated support services and updates.

CentOS Linux:

- Derivation of RHEL.
- No brand name and support. (for free)
- Relies on the developer community

Fedora Linux:

- Sponsored by RHEL.

Set of definitions:

1. GNU:

- GNU operating system is a complete free software system, upward compatible with Linux.
- GNU stands for "GNU's not Unix".
- The Free Software Foundation "FSF" is founded to raise funds to help develop GNU.
- It's goal is making high - quality freely available software. emacs - other programs.
- Bash: GNU shell
- GCC: GNU compiler
- GDB: GNU Debugger - The Gimp : GNU Image Manipulation Program.
- Gnome : GNU desktop enviroment.
- Emacs: powerful editor.

Scanné avec CamScanner

- Ghostscript and Ghostview: interpreter and graphical frontend for post script files. '.ps' intermediate format before PDF / define images
- GNU Photo: software for interaction with digital cameras.
- Octave: programming language, primarily intended to perform numerical computations and image processing.
- GNU SQL: relational database system.
- Radius: remote authentification and accounting server.

2) Open Source:

- Users have the freedom to run, copy, distribute, study, change and improve the software.
- The term "free software" is preferred by Stallman since the goal is to develop humanity (matter of liberty and principles).

3) Software license:

- A legal instrument (contract law) governing the use or redistribution of a software.
- GNU designed GNU GPL to release free software under a license specifically to protect freedom for all users of the program.
- Linux was placed under the GNU GPL (general public license).
- Openservce licenses: GNU GPLv3 - GNU - LGPLv3 - MIT,..
- GPL governs the distribution of a software, not its use.
- Copyleft: assure that the software remains redistributed under the same opensource license.

Scanné avec CamScanner

Linux : The Shell

Definition of the shell:

- A software / program that interprets commands and sends them to the OS.
- Acts as an interface between the user and the kernel.
- Provides : → Built-in commands : comes with the kernel can't be updated or removed.
  → Programming control structures : program of commands (script) to run once ; and structures (loops - conditions).
  → Environment variables : predefined variables - new defined
- Linux supports multiple shells. (Bash (BOURNE Again Shel
- csh - Ksh - zsh - sh) ; the default one is: Bash.

The prompt:

- In linux "~" is a shorthand for your home directory.

[username @ sccl ~] $ ...

↑ username ↑ system name ↑ current directory ↑ input

$: standard user, normal: (not root) (not super).

: root user, super.

* better to use "sudo" to temporarly become root user for performing some commands".

Commands:

[username @ sccl ~] $ command -- option argument.

Scanné avec CamScanner

- Command: a program that does one thing.
- Option: change the way a command does that one thing:
  - → Short form: single-dash and one letter: ls -a
  - → Long form: double-dash and a word: ls --all
- Argument: provides the input/output that the command interacts with.

Examples of commands.

- whoami: the username of the account using the terminal (mylogin)
- hostname: the name of the machine. (name of this computer)
- echo "Hello world": prints a string (set of characters)
- echo $HOME: prints a string: path to home directory (HOME is an environment variable) (print environment variable)
- echo my login is $(whoami) replace $(x) with output of x comman
- date: print the date and the time
- cal: print this month's calander
- youtube: bad command
- To know more about "date":
  - → type date (built in /)
  - → date --help (--h = it's an option to get more details about the command).
  - → help date (help: command)
  - → man date
  - → apropos date
  - → whatis date
  - → info date.

→ Built-in commands: commands embadded into the kernel.

Scanné avec CamScanner

→ The type command.

→ Shell built-in command displays the kind of the command : type cmd

→ --help

→ Display usage information : a discription of the command's supported syntax and options. command --help / -h

→ Help command:

→ Get help for Built-in commands

help shell-builtin -cmd

→ Man command:

→ Display command's manual page: man program

→ Most executable programs intended for command-line use provide peice of documentation called: manual - man page.. A special paging program is used to view them "man".

→ A title - synopsis of the command's syntax, a discription of the command's purpose, a listing and description of each of its options.

→ Man page doesn't include examples.

→ Used by everyone, amateurs, developers, professionals, experts.

→ Reading the command line manual page for a linux command, a configuration file.

→ On most linux systems, man uses less to display the manual page.

* less: display the content of a text file / reading text files.

Scanné avec CamScanner

- Ways to scroll through text:

Space , f	page forward
b	page backward
<	go to first line
>	go to last line
/	search forward (n to repeat n times)
?	search backward (n to repeat)
h	display help
q	quit help

- Manual pages are usually installed along with installation of a software in Linux.

- There is a defined syntax for man pages.

test.sh

```
#!/bin/bash
if ["$1" == "h"]; then
        echo "Hello"
fi
if ["$1" == "b"] ;then
        echo "Bye"
fi
```

the first argument

$./test h g

$1 $2

- To execute the script: ./test in a bash shell
- To transmit a parameter to a script: ./test h
  → the script prints "Hello" if "h" is specified as argument
  → the script prints "Bye" if "b" is specified as argument

How to create a manual page:

1. Create a text file: test.1
2. man page uses a very old roff markup language
3. It has commands (read markers) for various titles and sections:

Scanné avec CamScanner

→ TH : first command, used to specify heading of the man file.

→ SH : Section heading

→ B : display text next to it in bold.

→ TP : display information about an argument (flag) to the command

→ BR : display text in bold and in normal roman font.

eg:

```
.TH test.sh 1
.SH NAME
test.sh \- Print Hello or bye
.SH SYNOPSIS
.B test.sh
[h]
[b]
.SH DESCRIPTION
.B test.sh
this is our man page
"Bye" if argument is 'b'
.SH OPTIONS
.TP
.BR h
Print Hello
.TP
.BR b
Print Bye
```

call ./test.1

→ for more info:

man man_page

→ man ./test.1

to get its manual page

→ The sections of the manual:

1. general commands manual
2. system calls manual
3. library functions manual
4. kernel interfaces manual
5. file formats manual
6. games manual
7. miscellaneous information manual
8. system manager's manual
9. kernel developper's manual.

→ less command:

source code of the manual page of the "command"

→ man command.

interpreted man page.

Scanné avec CamScanner

- The system stores its man pages in /usr/share/man/
- The directory /usr/share/man/man1 stores man pages for user shell commands: cd /usr/share/man

ls -l

zcat ls 1.gz => { zcat is used to read compressed files without decompressing (zip / gz).

- It's recommended that your own man page be stored at: /usr/local/man

→ Apropos: Display appropriate commands.
Search the list of man pages for possible matches based on a search term. apropos keyword

→ Whatis: Display a breif description of a command (in 1 line)
: the name + one line from man command : whatis command.

→ Info: Display program's info entry : info command

→ The GNU project provides an alternative to man pages called info pages.

→ Displayed with a reader program named "info"

→ Info pages are hyperlinked much like webpages

Notes:

- man -k keyword <=> apropos keyword
- man -f <=> whatis

Scanné avec CamScanner

User management :

- User : individual / entity that interacts with the operating system by logging in and performing various tasks.
- User management : plays a crucial role in ensuring secure access control, resource allocation and system administration.
- A user is associated with a user account.
- Account : consists of several properties defining the identities and previleges within the system:

- username
- UID (user ID)
- GID (group ID) <u>not unique</u>
- Home directory
- Default shell
- Password : root user can't see normal users's passwords but can reset them

Types of Linux accounts :

1) Root
2) Standard (normal - guest)
3) System account.

Ty

1 Root

control c

unfettered

admi

2 Sys

system

Scanné avec CamScanner

Types of users

1. Root account / superuser, complete unfettered control of the system, it's assumed as an administrator (one root user).

2. System accounts: created by the system during installation used to run system services and applications. Any modification in them could affect the system.

3. User accounts: created by the administrator, can access the system and its resources based on their permissions, have limited access to critical system files and directories

User Groups:

- Collections of users, creating and managing groups is a simple way to deal with multiple users simultaneously (when dealing with permissions).
- Permissions are assigned to a collection of users who are organized to logical groups with identical permissions.

Scanné avec CamScanner

- Admins manage permissions for entire user group instead of managing individual user accounts.
- A single user can belong to multiple logical groups.

How does user management work?

- Linux stores data in files and directories:
  - user accounts info.
  - user passwords.
  - group configurations.
- System administrators can interact with these files, control and modify user and group setting.

Main files and directories for storing user data

1 - /etc/passwd : info of useraccounts, readable by most users, only root and sudo can modify.

2 - /etc/group: list of user groups, each line represents a group.

3 - /etc/sudoers: users with elevated permissions on the machine, admins can use this files to configure permissions for users and groups to use sudo command.

4 - /etc/shadow: stores encrypted user password information and other password-related data.

5 - /etc/gshadow: stores encrypted group password information and other password-related data.

6 - /etc/skel: contains default configuration scripts and templates ".bashrc" and "bash_profile". The templates are copied to the user's home directory when a new user is created.

Scanné avec CamScanner

I- /etc/ login.defs: contains system wide user account policy, administrators can refer to and modify this file to enforce security and user managment rules.

Editing user management files directly is not recommended, it's better to use commands.

Commands:

- id — show ID
- groups — view membership
- finger — show user info
- getent — fetch " " from database
- grep — search for patterns
- last — show most recent login attempts
- lslogins — display user info
- users — list current logged in users
- cat — list all users
- who — info of current logged in users
- w — current login
- Pastb — show failed login attempts

Create a user: useradd / adduser

→ requires sudo or root previleges:

sudo useradd test_account

→ No output

→ Checking by: cat /etc/ passwd

sudo adduser test_account

→ Output: creation done correctly

Modify default user settings: usermod

→ Modify various attributes of an existing user account.

-d: changes the home directory

-s: changes the default shell

-e: sets expiry date

Scanné avec CamScanner

-c : add comments for user's entry.

-u : changes the ID.

-aG : adds the user to supplementary groups without removing existing group membership.

Delete user: userdel

- removes a user from the /etc/passwd file.

  sudo userdel test_account

- To remove all related files from system

  sudo userdel -r test_account

Add members to a group: sudo adduser test_acc test_group

Create a group: sudo groupadd test_group

Remove members from a group: sudo deluser test_acc test_group

Scanné avec CamScanner

Linux file system

- A simple discription of the UNIX system, also applicable to linux : « On a UNIX system, everything is a file; if something is not a file, it's a process »

Linux file system.

→ Data in Linux is organized into files.

→ Files are organized into directories

→ Directories are organized into a tree-like system: file system.

→ Linux and Unix make no difference between files and directories (a directory is just a file containing names of other files).

→ Programs, services, texts, images are files.

→ Comunication with hardware via texts (keybords - screens)

Types of files:

1) Ordinary files: file on the system that contains data, text or program instructions.

2) Directories: store both special and ordinary files (directories (Unix) ⟺ folders (windows / macOS)

3) Special files: provide access to hardware (hard drive, modems, ...). Other special files (links) are similar to aliases or shortcuts that gives access to a single file using different names.

Scanné avec CamScanner

→ To list the content of a directory ⇒ command ls.

→ The option "-l" helps to get more information about the listed files.

Symbols & meanings

- ⇒ regular file.
d ⇒ directory
l ⇒ link
c ⇒ special file.
s ⇒ socket: (way to create communication link between processes) ← bidirectional connection
p ⇒ named pipe
b ⇒ block device.

- named pipe: same as sockets but communication within the same system: unidirectional communication
- block device: describes a device that provides data storage.

→ ls:

1) Type of file
2) Permissions of different types of users.
3) Number of hard links to this file.
4) ID of the owner
5) The group for which the file is associated: by default it's the owner's group.
6) The size of the file in bytes.
7) The date of creation or last modification.
8) The file's name.

→ ls -F + "/=*|@" +name: to get the file type.

nothing: regular
/ : directory
* : executable
@ : link
= : socket
| : named pipe

Scanné avec CamScanner

Metacharacters with special meanings.

- * to match 0 or more characters.
- ? to replace on single character (unknown one)

Hidden files:

- An invisible file is one, the first character of which the dot or the period character.
- Used to store configuration information.
- Use: " ls -a" to list all the files (including invisible ones).

Permissions r (read) - w (write) - x (execute)

→ Owner permissions determining what actions the owner can perform on the file

→ Group permissions determining what actions a user, member of the group that a file belongs to, can perform on the file.

→ Other "world" permissions determining what actions other users, can perform on the file.

nine characters for the permissions:

3 → owner (r-w-x)

3 → group (r-w-x)

3 → other (r-w-x)

"-" if not permetted instead of the (r/w/x)

→ To change permissions:

command ⇒ chmod

1) Symbolic mode

$ chmod u-x, g=rx, o+wx testfile

$ ls -l

testfile -rw -r -x rwx 1 amrood users 1024 Nov 2 00:10 testfile

2) Absolute mode:

$ chmod 743 testfile

$ ls -l

Scanné avec CamScanner

Exemple -rwxr--r--x-wx
111 100 100 001 011
7 4 3

The owner and the group:

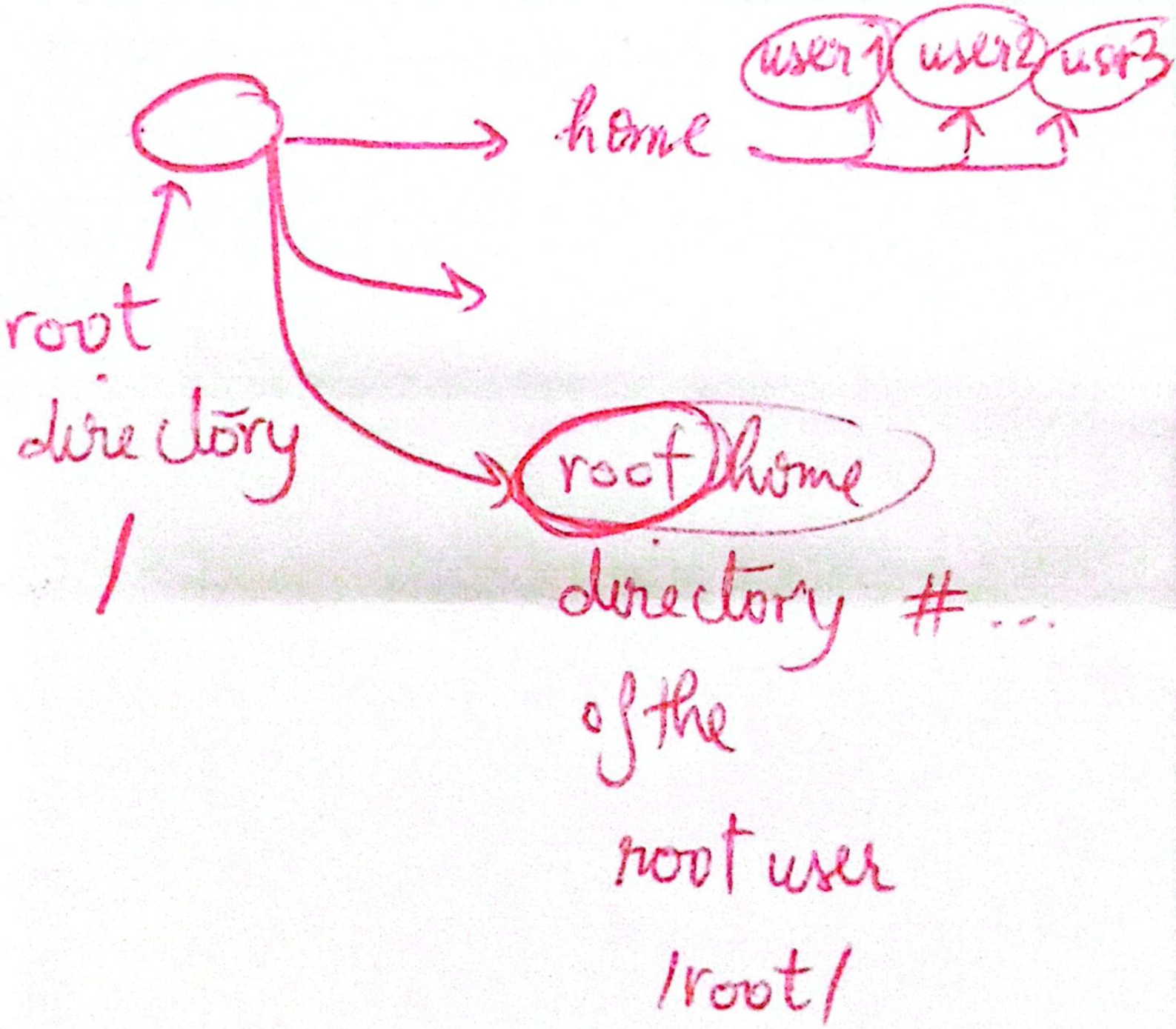
- chown ⇒ change the owner of the file.
- chgrp ⇒ change the group of the file.

Linux file system layout:

- The tree starts at the trunk (slash /).
- This directory, containing all underlying files and directories. "the root directory".
- Make difference between the root directory "/" and the root user home directory "/root/".

Directories:

1) /bin : common programs (manual files) shared by the system administrator.
2) /boot : startup files and kernel.
3) /dev: references to all CPU peripherals hardware which are represented as files with special properties.
4) /etc: most important system configuration files, contains data similar to those in Control Panel in windows.
5) /home: home directories of the common users.
   /usr: programs, libraries, documentations.
6) /lib: library files for all kinds of programs needs by the system and users.
7) /lost+found: files saved during failures are stored here.
8) /mnt: standard mount point for external file systems for short



Scanné avec CamScanner

periods of time.

8) /media : standard mount point for removable media : disks.

Remarks

- /dev is essential for OSs, can't be removed.
- /dev contains special files (devices files) corresponding to physical or system components.
- /mnt or /media can be erased and the hardware devices still working, but the applications using these devices may stop implementing these tasks.

Partitions of storage:

- Linux uses more than 1 partition per disk.
- Achieve higher data security in case of disaster.
- Partitioning allows the disk to be seen as a group of independant storage areas.
- Partitions make backups easier and help to restrict and confine potential problem areas.
- Hard disk space is not infinite, so the administrator need to manage it wisely.

File systems in Linux : $X_1, X_2, ...$

Kinds of major partitions:

- Data partition: normal linux system data, including the root partition containing all the data to start up and run the system.
- Swap partition: expansion of the computer's physical memory / extra memory on hard disk.

* Linux systems use fdisk at installation time to set the partition type

- The division of hard disks into partitions is determined by the system administrator.
- In Reality
  - A file system : a file is represented by an inode
  inode: kind of serial number containing information about the actual data that makes up the file to whom this file belongs and where it's located.
  - Each inode describes a data structure on the hard disk, storing the properties of a file including the physical location of the file data.
  - When a hard disk is initialized to accept data storage, usually during the initial system installation process / when adding extra disks to an existing system, a fixed number of inodes per partition is created.
  - 1 inode / 2 ~ 8 KB of storage
  - The number of inodes is the maximum number of files of all types that can exist at the same time in a partition.
  - When a new file is created, it gets a free inode. It contains:
    - owner - group of owner - type of the file - permissions - size - date + time of creation - number of links - adress defining the actual location of the file data.
    (name of file and directory not included)

    ⇓

    stored in the special directory files

Scanné avec CamScanner

- To display inodes numbers: ls -i (ID of inode in a partition)
- Inodes have their own spaces separately in the memory.
- The system can make up a tree-structure that the user understands.

→ Some directories share the same inode (same identifier) as they're in different partitions of the disk.
- df name-file: to know the partition where file is stored.

The path:

- The way to follow in the tree structure to reach a given file.

1- Absolute path: /usr/... / file : makes no assumptions about, begins from the hard drive describing each step until the target location.

2- Relative path: /dir / file depends on current directory the location of a file in relation to the current working directory.

Linking files:

a way of matching 2 or more file names to the same data. (shortcut)

- Hard link: like another file pointing to the same data, if one removed, the other works
- Soft link/symbolic: another file contains a path to the original file, if removed, the new file will be useless

Scanné avec CamScanner

Processes :

- Whenever there is an issue, a process is started.
- A process is an instance of a running program.
- It's a program in execution
- When a process is either created by a done of an existing process or loaded from a binary executable file.
- The OS tracks processes through a 5 digit ID number (PID)
- At any point of time, each process in the system has a unique PID.
- Linux is based on UNIX: multiple users running multiple commands, at the same time and on the same system

- By running commands
- By executing binary files
- Each process has a parent process (heirarchy)

⇒ Measures have to be taken to have CPU manage all these processes

⇒ Functionality has to be provided so users can switch between processes

⇒ Resource arbitration and competition between processes to be handelled

⇒ Processes have to continue to run even if the user who started them logged out.

⇒ Users need a means to reactive interrupted processes

Scanné avec CamScanner

Types of processes.

① Interactive processes.

- Initialized and controlled through a terminal session (started by a logged in user).

② Automatic / Batch process.

- Not connected to a terminal.
- Tasks that can be queued into a spooler area where they wait to be exicuted on a FIFO basis. (first in first out)

→ at a certain date and time "at".

→ At a time when the total system load is low enough to accept extra jobs "batch".

* response time:
delay between asking the process to execute and its executable time. (not fixed)

③ Deomons: (end with d)

- chrone D: executed every minutes (specific tasks)
- A program that runs continuously in the background of the OS.
- Most are intialized at a system startup and wait until their services are required. (crond - httpd)

Main process states:

- (R) Running: the execution in the CPU (executing the program line by line).
- (S) Waiting / Sleeping: waiting for an event to complete (user input / system resource becoming available).

Scanné avec CamScanner

- (T) Stopped : the process is stopped by recieving a signal (while debugging).
- (z) zombie : terminated process, which has finished executing but still has an entry in the process table to report to its parent process. (If a parent is terminated, the kid process must die)
- (X) dead : It's a transient state set just before the process descriptor is deallocated (killed / terminated)

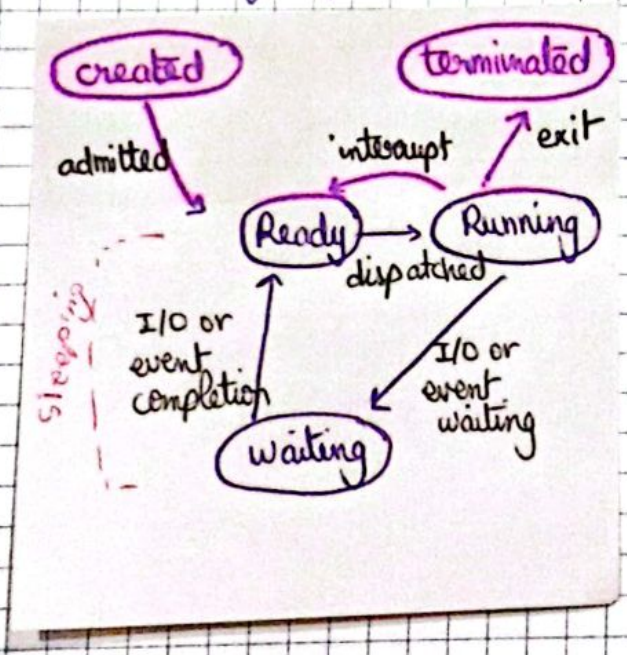
restart the machine to hardly erase a zombie process

=> If we use "ps" with "-l" the "Stat" column displays the status of all running processes

Top parent process in Linux system : init

-> By printing the processes tree.

-> By running "ps" command.



Ending processes:

- When a process is ended normally (not killed or interrupted) the program returns the exit status to its parent.
- The exit status is a number returned by the program providing the result of the execution.

Scanné avec CamScanner

- Processes can be ended because of a signal : kill command for example.
- The system of returning information upon executing a job has its origins in the C programming language (UNIX written in C).

Process attributes :

- PID
- the parent process ID : PPID.
- TTY : terminal to which the process is connected.
- the owner's username
- the group of the owner
- the nice number : $"nice - renice" : a parameter introduced by the user to increase priority of executing a process (degree of friendliness of this process towards other processes)

→ process priority is calculated basing on nice number and recent usage of the CPU

Scanné avec CamScanner

Note: the terminal is not a process!

Ending processes:

- A process can exit using _exit system call, this will free up the resources that the process was using.
- When a process is terminated it lets the kernel know why it's terminating: termination status.
- The parent process has to aknowledge the termination of the child process using wait system call and that checks the termination status of the process.
- When a parent process is terminated before the child pro the kernel attaches the orphin processes to the -init- process (always working)

Terminals:

- "Terminal" was reffering to a physical hardware device used to send and recieve I/O from computing devices.
- Now, terminal is a software program (terminal emulator)
- TTY in linux: TeleTYpewriter refers to an interface that enables access to terminal

* Only relationship between parent and child processes is that they share the same ID (of the parent)

LAB NOTE

Scanné avec CamScanner

Types of terminals:

1) Regular terminal device TTY: native terminal that you can type into and that can send output to the system.

2) Pseudoterminal PTS: emulates terminals with the shell terminal window. (terminal closed <=> process terminated)

- Processes are usually bound to a controlling terminal (but not all eg: init is not).

(To communicate with terminals)

- ls /dev
- ls -l
- ls /dev/pts/
- ls -l
- echo "bonjour" > /dev/pts/1

name of terminal as numbers

→ To write something to a terminal (Mac OS)

echo "... " > /dev/name of terminal

get from the command ps

Processes in C

- include the header "system"
- fork():
- exec():

→ fork(): child process (node) executing same program as parent.

→ exec(): just execute the command/program using the parameter command (...).

Scanné avec CamScanner

Process scheduling:

- Some processes are started automatically (service processes the computer needs to do its work).
- Others started manually (by typing a command)
- If we need a certain task to start automatically at pre-defined intervals, using cron:

* The cron daemon crond: process automatically starts on all computers and will check configuration every minute to see whether it has an issue with certain tasks.

- The configuration file /etc/crontab is read by crond
- This file contains:

Minute - hour - day of month - month - day of week.

- Indication of when a command should run.
- Name of user whose permissions to execute this file
- The command to run.

« # m h dom mon dow user command »

"*" means every

eg:

10 5 3 12 * nobody /usr/bin/false

=> the command starts 10 minutes after 5a.m on every December 3

Linux boot sequence:

- When the computer boots, it will start a kernel.
- The kernel itself is responsible for starting the first process, which normally is the process <u>init</u> (PID = 1)
- This process is responsible for all other processes.
- In Linux process managment, there's a <u>parent - child</u> relationship
- Waiting for the machine to boot is: une vache qui regarde le train passer
- The boot process can be broken into 4 stages:
  - → bios post
  - → boot loader (GRUB2)
  - → kernel initialization
  - → INIT process (systemd)

• The bios: "basic input/output system". The program the microprocessor to start the computer after it's powered on.

- Can't be reinstalled cuz it's on the ROM (read only memory).

• It also manages data flow between the computer's OS and attached devices (hard disk, printer, mouse).

- If you don't have a Bios, you can't start the machine.
- BIOS comes included with computers (stored in the read only memory).

(Firmware: a software that provides low-level control of computing device hardware)

Scanné avec CamScanner

1_ Bios post : it has very little to do with Linux itself.

- POST : power on self test.
- BIOS runs POST test to ensure the hardware components that are attached to the device are functionning correctly.
- If POST fails, the computer may not be operable and the system will not proceed to the next stage of the boot process.

* Some parameters can be changed in the BIOS : speed, order, password.
* BIOS parameters are saved out of the ROM to be modified.

2_ Boot loader :

- When the post test is passed successfully
- Boot loader is stored in the first sector of the hard disk.
- The boot loader will provide the user with the boot screen, often with multiple options to boot into.
- Once the selection is made at the boot screen, boot loader loads kernel into the memory.
- A popular eg of boot loader : GRUB2
- BIOS loads and executes the boot code from the boot device.

Scanné avec CamScanner

→ Processes with PID = 0 are part of the kernel.

3- Kernel initialization:

- Once the OS selected, the kernel is loaded to the memory. The kerel carries out Tasks (initializing hardware and memory tasks)
- Once it's completly operational, the kernel looks for INIT to launch

4. The init process:

- In most linux distributions, init process calls systemd deamon.
- systemd is responsible for mounting the file systems starting and managing system services
- To check the init system used run ls -l /bin/init if it's a system then you'll see a pointer to /lib/system/system

Managing processes:

→ Starting processes: background and foreground.
→ Killing processes with kill - pkill - killall.
→ Adjusting process priority with nice - renice.
→ Process management using top.
→ Multitasking and scheduling.

Scanné avec CamScanner

Bash shell scripting

- A bash script is a file containing a sequence of commands that are executed by the bash program.
- It allows to perform actions as navigating to a directory, creating a folder....
- By saving the commands a script, you can repeat the same sequence of steps multiple times and execute them by running the script
- The bash interprets the script line by line.

Advantages.

1) A powerful tool for automatic system administration tasks, managing resources and performing other routine tasks
2) Automation: allows to automate repetitive tasks and processes saving time and reducing the risk of errors
3) Portability: can be run on various platforms and OSs Unix, Linux, MacOS and even Windows using emulators or VMs.
4) Flexibility: easily modified and highly customizable.
5) Accessibility: easy to write and doesn't require specific tools /softwares.
6) Integration: can be integrated with other tools such as: data bases, web servers, allowing more complex automation.

Scanné avec CamScanner

7) Debugging: easy to debug and most shells have built in debugging and error reporting tools

How to create and execute bash scripts?

- Script naming conversions: name.sh
also can run without the extension ".sh".
- Adding the Shebang: #! /bin/bash
- Bash scripts starts with ↑.
- Shebang tells the shell to execute it via bash shell
- It is an absolute path to the bash interpreter.

- Executing bash script: ./name.sh or sh name.sh or bash nsh
- To execute a script: "./name.sh" after modifying the permissions (add execute permission) to user

chmod u+x name.sh

- Comments in bash scripting: # obliged!
- Any line starting with # will be ignored by the interpreter.
- Comments are very helpful in documenting the code and it's a good practice to add them to help others understand the code.
- Comments in scripts are more important than in C (auto documented programming language)

Scanné avec CamScanner

- Variables and data type:
- Allow to store data, used to read, access and manipulate data.
- To access $variable_name.
- There are no data types.
- Variables can store numerical values, individual chars or strings.
- Variables can be set in 2 ways:

1) Assign the value directly: country = algeria
2) Assign the values based on another variable: same_country = $country

- We can also access the value of a variable by using {} ← allow for a more controlled parsing. It's used:

1) when concatenating variable with other strings without introducing whitespaces

eg: A="World" : echo "hello ${A}!" delimits the variable A from the "!"

2) For a parameter expansions (substring extraction or length calculation).

eg: ${A:0:3} extracts the first three chars of A.

Variable naming:

- Variables' names should start with a letter or an underscore.
- Variables' names can contain letters, numbers, underscores.
- Variables' names are case sensitive.

Scanné avec CamScanner

- Variables names shouldn't contain spaces or special characters.
- Use of descriptive names that reflect the purpose of the variable.
- Avoid using reserved keywords (if - then - else - fi) like C++.

Notes !

① echo " Hi " ; echo 'Hi'

⇒ same output : Hi

② N=10

echo " User earns $N" ; echo 'User earns $N'

⇒ User earns 10. ⇒ User earns $N

- PATH variable contains the paths to specific programs so you can execute them without specifying the full path.

Arithmetic operations using ((

- The double parenthesis construct in bash allows for arithmitic expansion

- a = $((2+3)) / let a = 2+3 ⇒ in both cases a = 2+3
  - let will treat them as integers -
- ((a++)) / ((a = a+1)) ⇒ to increment.

! Spaces between the parentheses are important

Input and output in bash scripts :

- Command line arguments :

In a bash script or function, command line arguments are accessed using positional parameters :

Scanné avec CamScanner

- $0 : represents the name of the script or program being executed.
- $1, $2... : represents the positional parameters st $1 is the first argument, $2 is the second one...

Other functions.

1) $* : represents <u>all</u> command line arguments as a single string.
2) $@ : represents all command line arguments as separate strings, preserving their original whitespace and quoting.
3) $# : represents the number of command line arguments passed to the script or function.
4) $$: represents the PID of the current script or argument.
5) $! represents the PID of the last background command executed.

Read command: read

- Has many useful options:

1) -p prompt: displays the specified prompt before reading input.
2) -s: silent mode, input is not echoed in the terminal
3) -n count: reads only <u>count</u> characters before stopping.

Test command: test

Scanné avec CamScanner

- Also known as [ ]
- Always return either True(0) or false(1) (similar to the exit() context).

Ternary-Like Condition:

1) && : executes the command on its right only if the command on its left succeds.
2) || : executes the command on its right only if the command on its left fails.

test $A = "hello" && echo true || echo false

Test -testing integers:

1) -eq : tests if 2 integers are equal.
2) - ne : tests if 2 integers are not equal.
3) - lt : tests if the first integer is less than the second.
4) -le : tests if the first integer is less than or equal to the second integer.
5) -gt : tests if the first integer is greater than the second integer.
6) -ge : tests if the first integer is greater than or equal to the second integer.

"Test" - testing file types:

1) -f : Tests if a file exists and is a regular file.
2) -d : Tests if a file exists and is a directory.
3) -e : Tests if a file exists.
4) -s : Tests if a file exists and is not empty.

Scanné avec CamScanner

5) -r: Tests if a file exists and is readable.
6) -w: Tests if a file exists and is writable.
7) -x: Tests if a file exists and is executable.
8) -p: Tests if a file exists and is a named pipe.
9) -b: " " " " " " is a block device.
10) -c: " " " " " " is a character device.

"test" - advanced test using

- [[ ]] offers more robust and feature rich solution for writing conditional expressions in bash
- It allows enhanced string comparaison.
  - Pattern matching "==" and "!=".
  - We can use "||" and "&&" for OR / AND operations.
  - Regex matching using =~ operator [[$a =~ ^[0-9] $]] (start with a digit from 0 to 9)
  - Complex comparaisons [[... && ... || ...]]

① [[$a =~ ^ [0-9]* $]] $a is composed of digits only
② [[$a =~ ^ [0-9] $]]: $a is a single digit
③ [[$a =~ ^ [0-9]+$]]: $a is composed of one digit or more

* : 0 or more
\+ : 1 or more

Scanné avec CamScanner

Conditional statments :

① If condition ; then
statemen 1
statement 2
fi

② If condition ; then
statement
else
statement
fi

③ If condition ; then
statement
elif condition ; then
statement
else statement

fi

Case statements : $variable name

```
case expression in
  case1)  statement 1
          stament 2 ;;
  case2)  statements .
          statement 2 ;;
  *)      statement 1 ;;
esac
```

if there is no match, the case else statement will be matched.

The for loop. for is shell keyword

(a)
```
for item in [List]
do
    [commands]
done
```

* for loop can be executed in the terminal or inside a script.

→ $* : all arguments concatenated in one string.

→ $(basename $0) : extract name of the script only (without path).

→ $A : name_name1 (when we iterate, considered 2 strings) A= "name name1"

→ "$A" : name name1 (" " " , " 1 string).

```
for (( intialization; test ; step ))
do
    [ commands]
done
```

The while loop

```
while [condition]
do
    [ commands]
done
```

stop when the condition becomes false

≠

The until loop

```
until [condition]
do
    [commands]
done
```

stop when the condition becomes true

Scanné avec CamScanner

The while loop + reading from a file:

```
while read line
do
    echo $line
done < input.txt
```

prints the content of the file line by line.

Break and continue statements:

- Loops allow us to run one or more commands multiple times until a certain condition is met.
- When we need to alter the flow of the loop and:

→ terminate the loop break

→ terminate only the current iteration continue

Functions: set of commands called many times

- The purpose: → More readable script.
  → Avoid writing repeatedly the same code.
- Always try to keep your function names discriptive.
- A comment is required for each new function.

```
function_name () {
    commands
}
```

OR

```
function function_name {
    commands
}
```

- Variable scope:

→ Global variables can be accessed from anywhere in the script regardless of the scope.

→ All variables by default are global, even if declared inside a function.

Scanné avec CamScanner

- Add keyword "local" when creating a variable to not make it global (used only in the script where it's created).

- Return value.
- Functions don't allow to return value when called.
- The <u>return status</u> can be specified by using the <u>return</u> keyword and it's assigned to the variable $?
- The return statement terminates the function.

- Arguments:
  - The passed parameters are $1, $2, ..., $n corresponding to the position of the parameter after the function's name.
  - $# holds the number of positional parameters passed to the function.

<u>awk:</u>

- The awk command is a mainstay in the command suite in both Unix and Linux
- Developed by Bell Labs in the 1970's by:
  Aho - Weinberger - Kernighan.
- The awk command allows access to awk programming language which is designed to process data within text stream
- awk - a programming language:

```
awk 'BEGIN { print "Hello world!" }'
```

- In general, the awk command is used to filter content from files.

Scanné avec CamScanner

awk ' BEGIN { FS = ":" } { print $0 } END {print NR}' /etc/passwd

prints all lines + num of lines

awk program → can contain up to 5 parts

file

→ begin part (begin block) : executed once at the beginning
→ main part (code block) : executed line by line on the file
→ end part (end block) : executed once at the end (after processing the file)

1) Begin block: optional
executed before starting processing of the file, used to set up environment variables, initialize counters, set field separators.

2) Code block:
series of commands to execute for each line.

3) End block: optional
used for any final calculations or cleanup (display summary / results of processing).

- Working on selected lines only.

1) awk 'NR<9 {print NR, $1 }' /etc/passwd ⇒ process only first eight lines

2) awk 'NR==5, NR==11' /etc/passwd ⇒ process lines 5 through to 11.

3) awk '/bash$/' /etc/passwd ⇒ process lines that end in the word bash

Scanné avec CamScanner

* The pattern (condition) is added between the begin block and the code block.

- Output formatting:

o awk' BEGIN {FS = ":"} { print $1, $3, $7 }' /etc/passwd

⇒ print the username ($1), the user ID ($3) and the default shell ($7)

- An improved print ⇒ printf!

o awk' BEGIN { FS = ":" } { printf "%-10s %-4d %-17s\n", $1, $3, $7 }' /etc/passwd

(format: %-10s → 10 digits, %-4d → 4 int, %-17s → 17 digits)

⇒ the printing is alligned

o awk -F":" '$3 > 999' /etc/passwd

(-F: option; $3 > 999: UID > 999)

↑ field separator as an option.

Built-in functions:

1) String functions:

→ length(s) : the length of s

→ substr (s, start, length) extracts a substring from s

→ index (s, t) : returns position of substring t in s

→ toupper (s) converts all chars of s to uppercase

→ tolower (s) opposite

2) Numeric functions:

→ sqrt(x) : returns the square root of x

→ sin(x) cos(x) tan(x): trigonometric functions

→ exp(x) log(x): exponential and logarithm functions

Scanné avec CamScanner

→ int (x) converts x to an integer by truncating the fractional part.

User defined functions.

→ awk

// here we write the whole code (like C++)

we write it in the terminal as a whole command.

→ OR: we write the program in a file: name.awk
to execute: awk -f name.awk. (✓ reusability)

Regular expressions.

wts models > Patterns used to match character combinations in strings.

- Reg ex can be used in various contexts, such as with tools like grep - sed - awk and within Bash itself using [[ ]] conditional expressions.

1) Literal: matches the exact character. "Hello" matches "Hello": [[$V == "hello"]] doesn't match Hello
case sensitive

2) Metacharacters: characters with special meanings.

.	maches	any single char except a newline
^	"	the beginning of a line
$	"	the end of a line
*	"	0 or more occurences of the preceding element
+	"	1 " " " " " "
optional ?	"	0 or 1 occurence of the preceding element
|	"	alternation : logical OR

Scanné avec CamScanner

3- Character classes: matches anyone of the chars in the set

[abc]: matches 'a', 'b', or 'c'

[^abc]: " any character except 'a', 'b' or 'c'

[a-z]: " any lower case character

[0-9]: " any digit.

4. Anchors: specifies positions within strings

^ : beginning of a line

$: end " " "

5- Groups and ranges:

(abc) : matches the exact sequence "abc"

a{2,4}: " 'a' repeated 2 to 4 times

```
grep -E '^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]
        + \.[a-zA-Z]{2,} $' file.tsct
```

repeated at least 2 times

it's a real point not metacharacter

⇒ valid email adress

```
[[ $V =~ ^([0-9]{1,3}\.){3}[0-9]{1,3}$]]
```

⇒ valid IP adress

nnn.nnn.nnn.kkk

sin

not sin

not sin

Scanné avec CamScanner

* Sed command:

1) Print 3 lines.

sed -n '1,3p' /etc/passwd

sed '3q' /etc/passwd

2) Print lines doesn't contain word "root".

sed '/root/d' /etc/passwd

3) Print last line

sed -n '$p' /etc/passwd

4) Print lines 5 → 10.

sed -n '5,10p' /etc/passwd

sed -n '5,10p;10q' /etc/passwd

5) sed 's/2/to/g' f1.txt > f2.txt

changes "2" in f1.txt by "to"

and puts the new version in f2.txt

g : changes all occurences: wherever there's a "2"

in: f1, it'll change it to "to". (In the same file)

Scanné avec CamScanner

6) To correct both:

sed 's/iz/is/g ; s/2/to/g' f1.txt

sed -e 's/iz/is/g' -e 's/2/to/g' f1.txt

• -i to save the changes in f1.txt.

7) sed '3,4s/iz/is/g' f1.txt => apply on only lines 3 and 4.

8) sed '3,4s/z/s/g ; 3,4s/:/,/g' f1.txt

to replace z by s and : by , in lines 3,4

<u>or:</u> sed '3,4y/z:/s,/' f1.txt

y: translation ; s: substitution

Scanné avec CamScanner