

Module : Administration Système Linux

**Cycle Ingénieur : Génie Informatique
Niveau : GI2**

Pr. Ahmad EL ALLAOUI

cyclegi2@gmail.com

2022/2023

Administration Systèmes Linux

Les processus

Les processus

- Un processus est une instance d'un programme en cours d'exécution, une tâche. Un processus a besoin de ressources matérielles : l'unité centrale, la mémoire centrale et l'accès à des périphériques d'entrées/sorties.
- Il possède un numéro unique sur le système pid
- Chaque processus appartient à un utilisateur et un groupe et à les droits qui leur sont associés
- Sous shell, un processus est créé pour exécuter chacune des commandes
- Le shell est le processus père de toutes les commandes.
- Caractéristiques statique:
 - identification (pid)
 - identification du proc. parent (ppid)
 - Un propriétaire déterminant les droits d'accès du processus aux ressources : ouverture de fichiers...
 - Un terminal d'attache pour les entrées/sorties
- Caractéristiques dynamiques :
 - Priorité, environnement d'exécution...
 - Quantité de ressources consommées (temps unité centrale utilisé...)

Les processus

- Un processus est toujours créé par un autre processus appelé processus parent.
 - Ainsi tous processus a un processus parent sauf le tout premier. Ce tout premier processus est appelé **init** et son identifiant est égal à 1 (*PID* = 1).
- Deux types de processus existent :
 - Les processus utilisateurs, tous issus du *shell* de connexion ;
 - Les processus démons :
 - Un démon est une traduction abusive de *daemon*.
 - Ces processus *daemon* assurent un service et sont souvent lancés au démarrage de la machine.
 - Les principaux services assurés par des processus *daemon* sont l'impression, les tâches périodiques, les communications, la comptabilité, le suivi de tâche.

Les processus

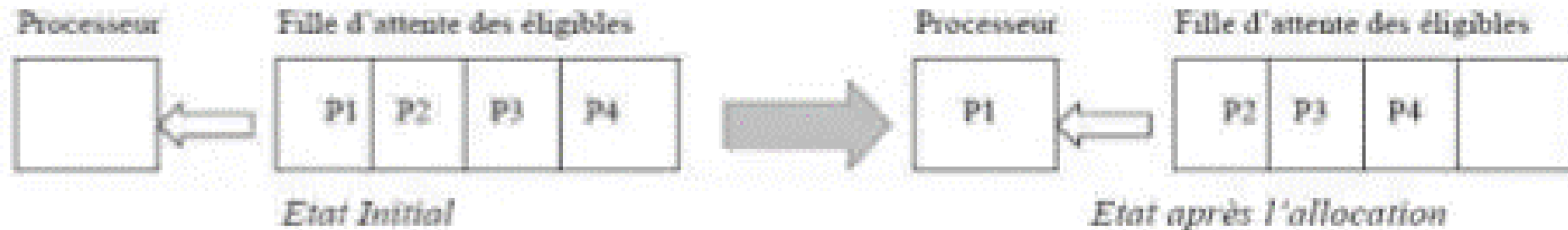
Statut d'un processus

- **Running en exécution**: le processus s'exécute
- **Waiting en attente**: Attend quelque chose pour s'exécuter
- **Ready prêt**: le processus a tout pour s'exécuter sauf le processeur
- **Suspendu**: Arrêté
- **Zombie**: état particulier

Les processus

L'ordonnancement

- **L'ordonnanceur** (scheduler) est le module du SE qui s'occupe de sélectionner le processus suivant à exécuter parmi ceux qui sont **prêts**
- Ordonnancement préemptif : l'Ordonnanceur peut interrompre un processus en cours d'exécution si un nouveau processus de priorité plus élevée est inséré dans la file des Prêts.
- Ordonnancement coopératif : Ordonnancement jusqu'à achèvement : le processus élu garde le contrôle jusqu'à épuisement du temps qui lui a été alloué même si des processus plus prioritaires ont atteint la liste des Prêts



Les processus

L'ordonnancement

L'algorithme FIFO (First In First Out)

- L'ordonnancement est fait dans l'ordre d'arrivée
- Le processus élu est celui qui est en tête de liste des Prêts: le premier arrivé.

L'algorithme SJF (Shortest Job First)

- SJF choisit de façon prioritaire les processus ayant le plus court temps d'exécution sans réellement tenir compte de leur date d'arrivée.

L'algorithme du tourniquet

- les processus sont rangés dans une file d'attente des éligibles, le processeur est alloué successivement aux différents processus pour une tranche de temps fixe Q appelé Quantum.

L'algorithme HPF(Highest Priority First)

- L'attribution de priorité à chaque processus. L'Ordonnanceur lance le processus prêt de priorité la plus élevée.
- En cas de priorités égales on utilise l'algorithme FIFO

- Pour voir les processus en cours: **ps**

- **Mini projet sur l'ordonnancement**

Les processus

- Infos retournées par **ps**:

Diagram illustrating the output of the **ps** command and the meaning of its fields:

	PID	TT	STAT	TIME	COMMAND
	3899	p1	S	0:00.08	-zsh
	4743	p1	S+	0:00.14	emacs
	4180	std	S	0:00.04	-zsh

Annotations:

- numéro de processus** (points to PID)
- terminal associé** (points to TT)
- état du processus:** (points to STAT)
- temps CPU utilisé** (points to TIME)
- commande exécutée** (points to COMMAND)

Legend for process states:

État	Description
R	actif
T	bloqué
P	en attente de page
D	en attente de disque
S	endormi
IW	swappé
Z	tué

```
ahmad@DESKTOP1:~$ ps -l
F S  UID  PID  PPID  C PRI  NI ADDR  SZ  WCHAN  TTY
0 S  1000   10    9   0  80   0  -   4285  -      tty1
0 R  1000   71   10   0  80   0  -   4625  -      tty1
```


Les processus: options

- a (*all*) : donne la liste de tous les processus, y compris ceux dont vous n'êtes pas propriétaire.
- g (*global, général...*) : donne la liste de tous les processus dont vous êtes propriétaire.
- u (*user, utilisateur*) : donne davantage d'informations (nom du propriétaire, heure de lancement, pourcentage de mémoire occupée par le processus, etc.).
- x : affiche aussi les processus qui ne sont pas associés à un terminal.
- w : le tronque pas à 80 caractères (peut être utilisée plusieurs fois pour tronquer plus loin)
- agux : est en fait souvent utilisé pour avoir des informations sur tout les processus.

Les processus

- Affichage des processus en cours:

- **ps -l** : affiche les processus utilisateurs en détails
- **ps -a** : liste tous les processus actifs
- **ps -u** : format d'affichage long
- **ps -x** : inclut les processus sans terminal
- **ps -aux** : affiche tous les processus d'un système

ps -ef :

- **-e** : affiche les processus en cours d'exécution de tous les utilisateurs
- **-f** : affiche la liste complète du format (affiche des informations supplémentaires sur les processus d'exécution)
- **PID** – l'ID de processus.
- **Tty** – le nom du terminal de contrôle du processus
- **TIME** – L'heure cumulative du processeur du processus, illustrée en minutes et en secondes.
- **CMD** – Le nom de la commande utilisée pour démarrer le processus.

```
root plus de détails, consultez ps(1).
ubuntu@ubuntu:~$ man ps
ubuntu@ubuntu:~$ ps -ef
UID          PID    PPID  C   STIME TTY          TIME CMD
root           1         0  0  09:58 ?        00:00:29 /sbin/init splash --- m
root           2         0  0  09:58 ?        00:00:00 [kthreadd]
root           3         2  0  09:58 ?        00:00:00 [rcu_gp]
root           4         2  0  09:58 ?        00:00:00 [rcu_par_gp]
root           6         2  0  09:58 ?        00:00:00 [kworker/0:0H-kblockd]
root           8         2  0  09:58 ?        00:00:00 [mm_percpu_wq]
root           9         2  0  09:58 ?        00:00:11 [ksoftirqd/0]
root          10         2  0  09:58 ?        00:00:22 [rcu_sched]
root          11         2  0  09:58 ?        00:00:01 [migration/0]
root          12         2  0  09:58 ?        00:00:00 [idle_inject/0]
```

Les processus

ps -aux :

- **USER** – l'utilisateur qui exécute le processus.
- **%CPU** – L'utilisation du processeur du processus.
- **%MEM** – le pourcentage de la taille du réglage de résident du processus dans la mémoire physique de la machine.
- **VSZ** – Taille de la mémoire virtuelle du processus en KIB.
- **RSS** – la taille de la mémoire physique utilisée par le processus.
- **STAT** – Le code d'état du processus, tel que Z (Zombie), S (Dormant) et R (en cours d'exécution). Démarrer – le moment où la commande a commencé.

```
ubuntu@ubuntu:~$ ps -aux
USER          PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.1  0.5 168956 11888 ?        Ss   09:58   0:30 /sbin/init
root           2  0.0  0.0     0     0 ?        S    09:58   0:00 [kthreadd]
root           3  0.0  0.0     0     0 ?        I<   09:58   0:00 [rcu_gp]
root           4  0.0  0.0     0     0 ?        I<   09:58   0:00 [rcu_par_gp]
```

Même résultat avec la commande: top

L'option -o vous permet de filtrer la colonne pour n'afficher que celle désirée.

- `ps -efo pid,comm`
- `ps auxo pid,comm`
- Enfin bien entendu, vous pouvez aussi utiliser **grep** sort pour filtrer ou trier les colonnes.
- `ps -ef | grep root`
- Processus en arrière-plan: **&**
(le terminal n'est pas bloqué)
exemple: `emacs monfichier.c &`

Les processus : pstree et top

- La commande **ps**tree permet de visualiser l'arborescence des processus.
- La commande **top** permet de visualiser dynamiquement les caractéristiques des processus (l'affichage est actualisé périodiquement).
- En plus des informations sur les processus, **top** donne des indicateurs sur l'état du système : occupation de la mémoire, de l'unité centrale...
- **top**: montre l'évolution de ces indicateurs en temps réel .

Processus en avant et en arrière plan

- Par défaut, une commande s'exécute **en avant-plan** (en anglais **foreground**).
 - Par exemple, l'utilisateur saisit la commande **\$ date**.
 - Le shell crée un processus enfant et **attend** qu'il se termine.
 - Le processus enfant exécute la commande **date**.
- ➔ Les processus parent et enfant s'exécutent séquentiellement (l'un après l'autre).
- ➔ Une seule commande est donc exécutée à la fois.

- Une commande peut aussi s'exécuter **en arrière-plan** (en anglais **background**).
 - Par exemple, l'utilisateur saisit la commande: **\$date &**
 - Le shell crée un processus enfant et **n'attend pas** qu'il se termine.
 - Le processus enfant exécute la commande **date**.
- ➔ Les deux processus, parent et enfant, s'exécutent alors **simultanément**.

Suspension et reprise d'un processus

- Sous Unix, il est possible de suspendre le processus en avant-plan en tapant CTRL-Z.
 - Le processus suspendu pourra reprendre ultérieurement.
- Il existe deux façons de reprendre un processus suspendu :
 - En avant-plan par la commande **fg (foreground)**,
 - En arrière-plan par la commande **bg (background)**.
- Par exemple :
 - \$ test → lancement de test en avant-plan
 - CTRL-Z → édition suspendue
 - \$ bg → reprise de l'édition en arrière-plan
- Un **job** est défini comme un processus en arrière-plan ou suspendu.
- La commande « **jobs** » permet de lister ces processus.

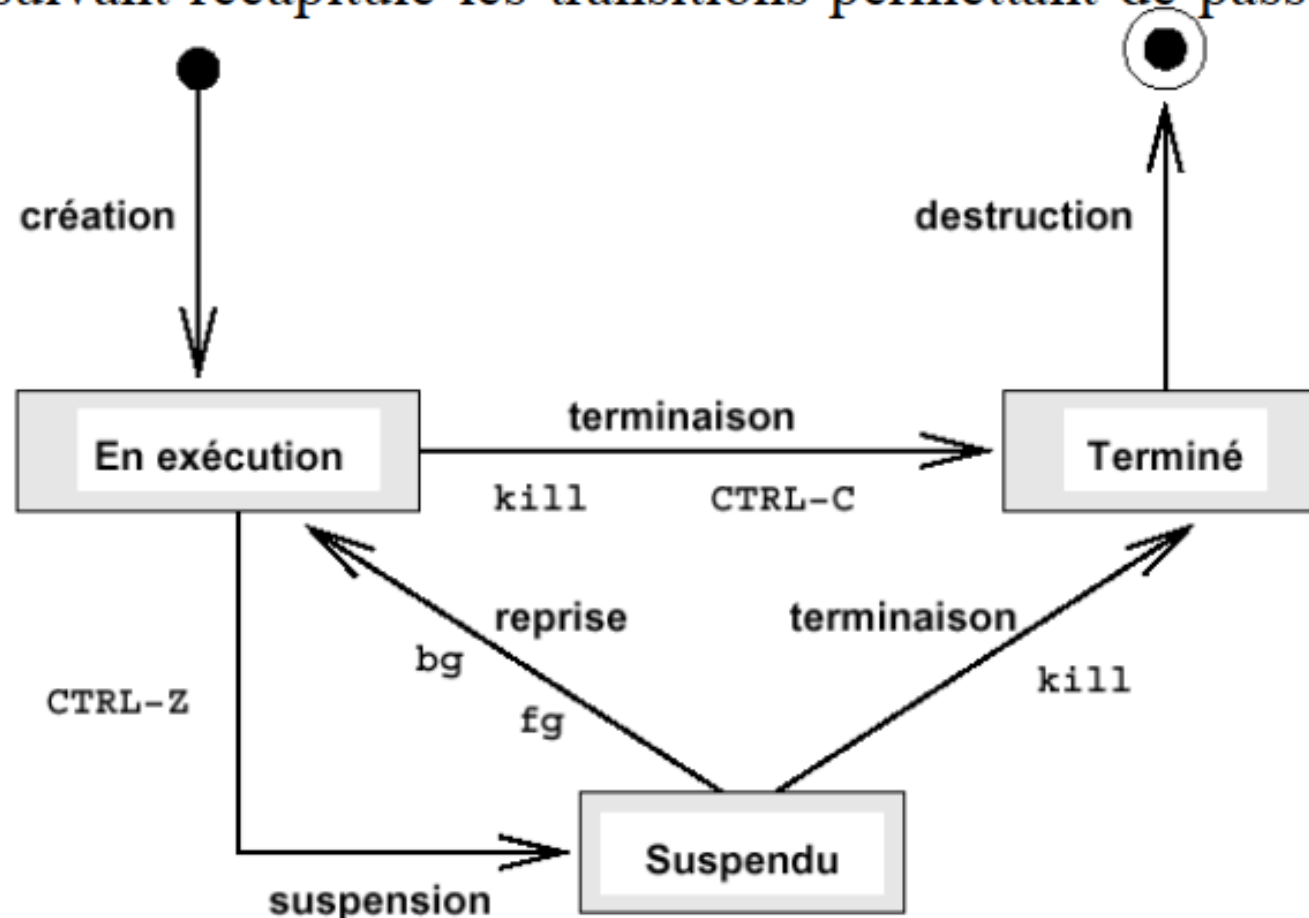
Tuer un processus: **kill -9 <PID>**

- Généralement un processus **se termine** à la fin de l'exécution de la dernière instruction ; il est alors **détruit** par le système d'exploitation.
- Un utilisateur peut terminer un processus en avant-plan en tapant **CTRL-C**.
- Un utilisateur peut aussi terminer un processus avec la commande **kill** envoyant un signal à un processus.
 - Par défaut, la commande **kill** envoie le signal 15 de terminaison (SIGTERM).
\$ kill PID
 - La commande **kill** peut aussi forcer la terminaison d'un processus en envoyant le signal 9 de destruction (SIGKILL).
\$ kill -9 PID_processus
- Notez que **le droit de détruire** un processus est **réservé à son propriétaire**.

Etat d'un processus

- Pour un utilisateur, un processus peut se trouver dans **trois états**:
 - **en exécution** (exécution de la commande), **suspendu** (CTRL-Z) ou **terminé**.

Le schéma suivant récapitule les transitions permettant de passer d'un état à un autre.



Les processus

Autres commandes pour la gestion des processus

- **ps** : liste des processus et de leurs caractéristiques
- **htop** : liste dynamique des processus et de ce qu'ils consomment
- **pgrep** : récupération d'une liste de processus par expression régulière
- **pidof** : récupération du pid d'un processus recherché
- **fuser** : informations sur les file descriptor d'un processus
- **lsof** : idem
- **pmap** : afficher le mapping mémoire d'un processus
- **strace** : liste les appels système du processus
- **ltrace** : liste les appels de fonction de bibliothèques dynamiques du processus
- **pstack** : affiche la pile d'appel du processus
- **gdb** : pour tout savoir et même modifier l'action d'un processus.
- **kill** : envoyer un signal à un processus connaissant son pid
- **killall** : envoie un signal à tous les processus portant un certain nom
- **pkill** : envoie un signal aux processus matchant une expression régulière
- **ctrl-z** : envoie le signal STOP au processus en avant plan du shell en cours
- **fg, bg** : envoie le signal CONT à un processus stoppé du shell en cours
- <https://fr.linux-console.net/?p=543#gsc.tab=0>

Les processus

Parfois, vous laissez trainer des processus sur la machine (par exemple si votre PC/Mac se plante. Dans ce cas il faut lister tous vos processus avec la commande suivante:

```
ps -ef | grep <votre login name>
```

Exemple : `ps -ef | grep ahmad`

Ensuite vous tuez chaque processus à votre nom avec la commande `'kill -9 <numero PID>'`, par exemple:

```
$ ps -ef | grep grob
grob 14356 14354 80 13:37:56 pts/2 0:02 -tcsh
grob 14535 14533 80 13:48:42 pts/11 0:03 -tcsh
grob 14475 14474 80 13:46:52 pts/2 0:04 /unige/gnu/bin/emacs -nw
root 15465 272 9 14:54:54 pts/0 0:00 grep grob
grob 14474 14356 10 13:46:52 pts/2 0:00 /bin/csh -f /unige/tecfa/util/bin/em....
```

```
$ kill -9 14474 (cette commande aurait tuée le emacs en cours ci dessus)
```

Les inodes

Les inodes (contraction de « index » et « node », en français : *nœud d'index*) sont des **structures de données** contenant des informations concernant les fichiers stockés dans certains systèmes de fichiers (notamment de type **Linux/Unix**).

À chaque fichier correspond un numéro d'inode (***i-number***) dans le système de fichiers dans lequel il réside, unique au périphérique sur lequel il est situé.

Les inodes peuvent, selon le système de fichiers, contenir aussi des informations concernant le fichier, tel que son créateur (ou propriétaire), son type d'accès (par exemple sous Unix : **lecture, écriture et exécution**), etc.

Les inodes



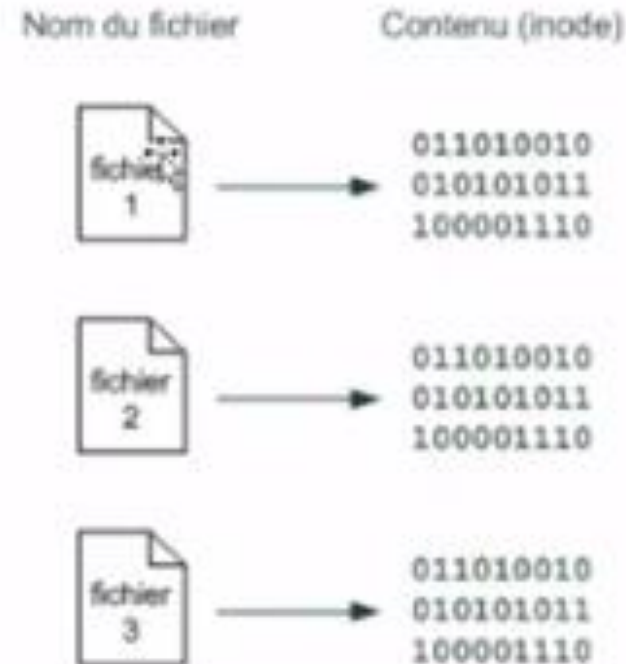
Les inodes contiennent notamment les **métadonnées** des systèmes de fichiers, et en particulier celles concernant les droits d'accès.

Les inodes sont créés lors de la création du système de fichiers.

La quantité d'inodes (**généralement déterminée lors du formatage et dépendant de la taille de la partition**) indique le nombre maximum de fichiers que le système de fichiers peut contenir.

Les inodes

Chaque contenu de fichier se voit attribuer un numéro d'identification appelé **inode**. Chaque nom de fichier est donc associé à un inode (son contenu).



Les inodes

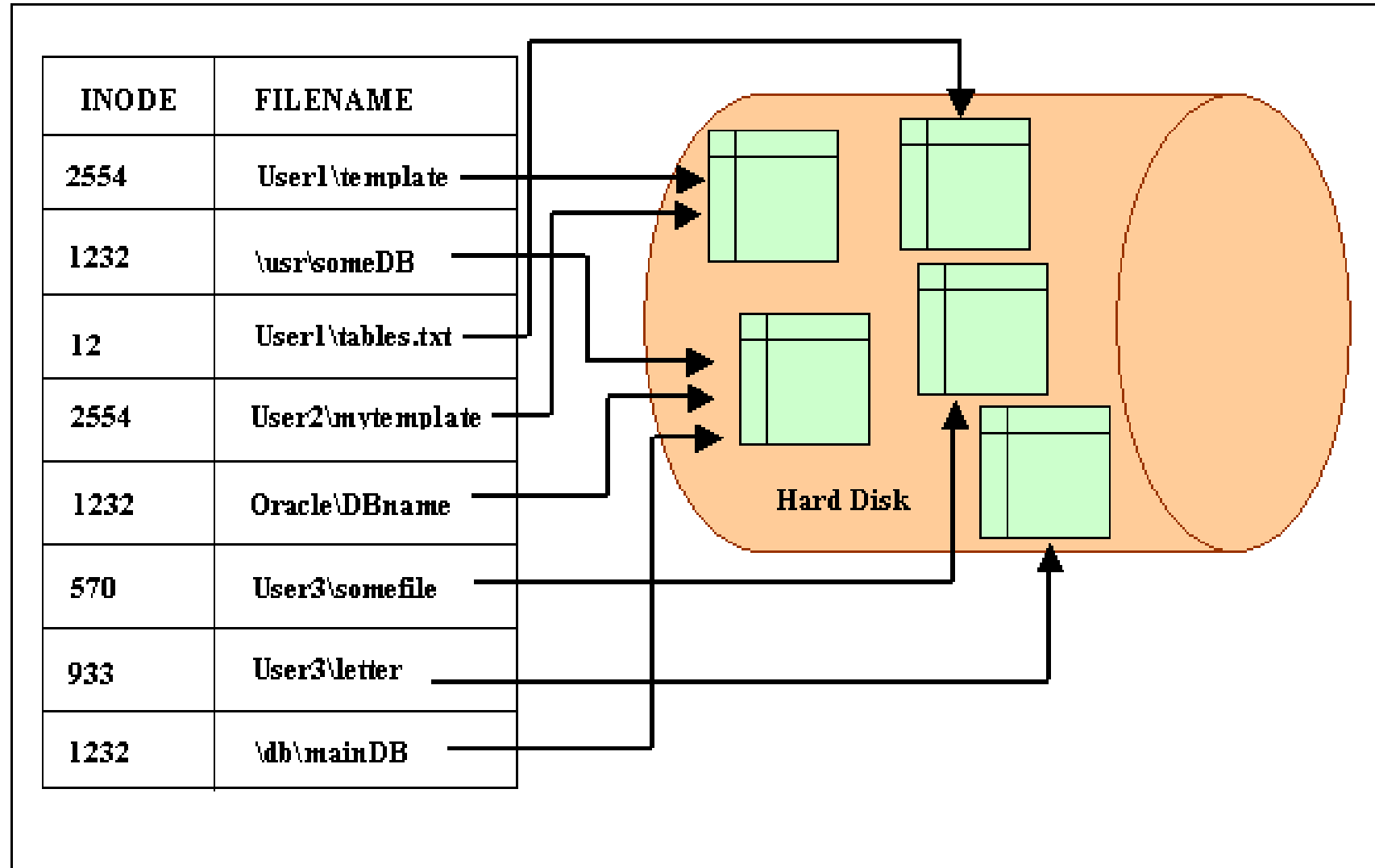


Le numéro d'inode est un entier unique pour le périphérique dans lequel il est stocké.

Le numéro d'inode d'un fichier **toto** peut être affiché avec la commande

ls -li toto

Les Inodes



Norme POSIX

La taille du fichier
en octets

L'identifiant du
groupe auquel
appartient le fichier

Le standard POSIX s'est basé sur les systèmes de fichiers traditionnels d'Unix. Cette norme impose donc que les fichiers réguliers aient les attributs suivants :

Identifiant du
périphérique
contenant le fichier

Le mode du fichier qui
détermine quel
utilisateur peut lire,
écrire et exécuter ce
fichier

Un compteur
indiquant le
nombre de liens
physiques sur cet
inode.

L'identifiant du
propriétaire du
fichier

Le numéro
d'inode qui identifie
le fichier dans le
système de fichier

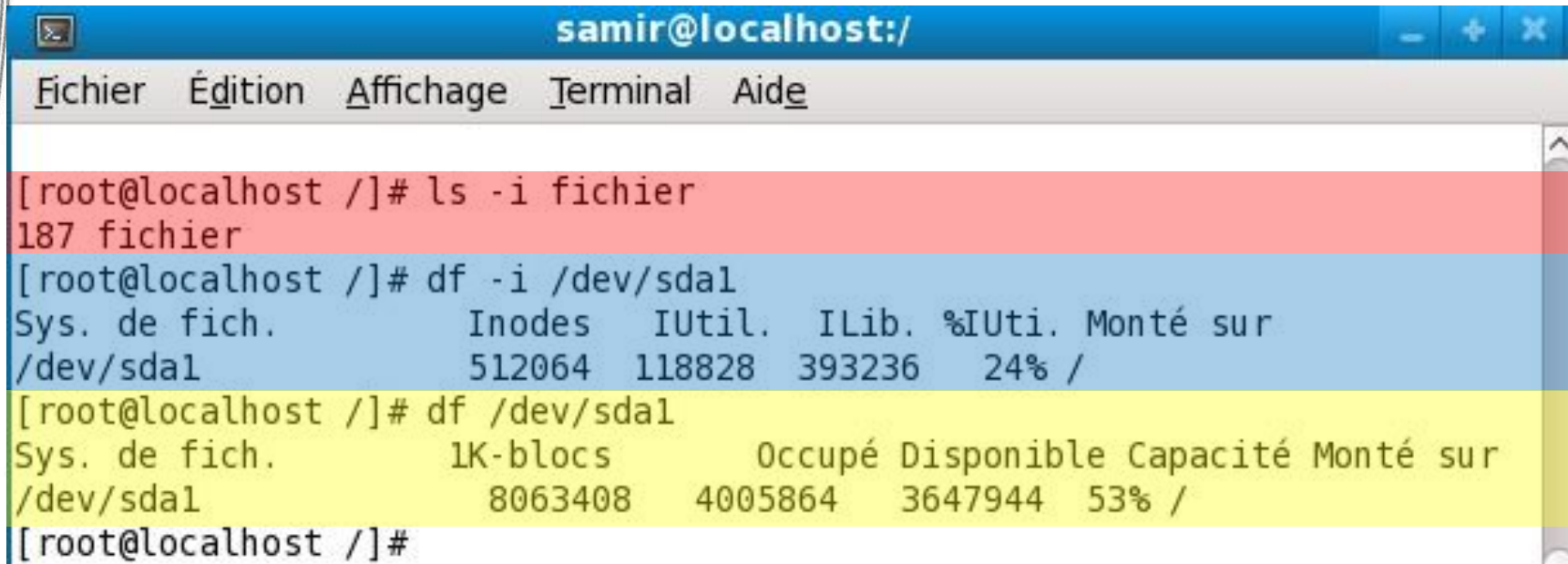
Horodatage

Les Inodes

Les Commandes

ls -i

df -i

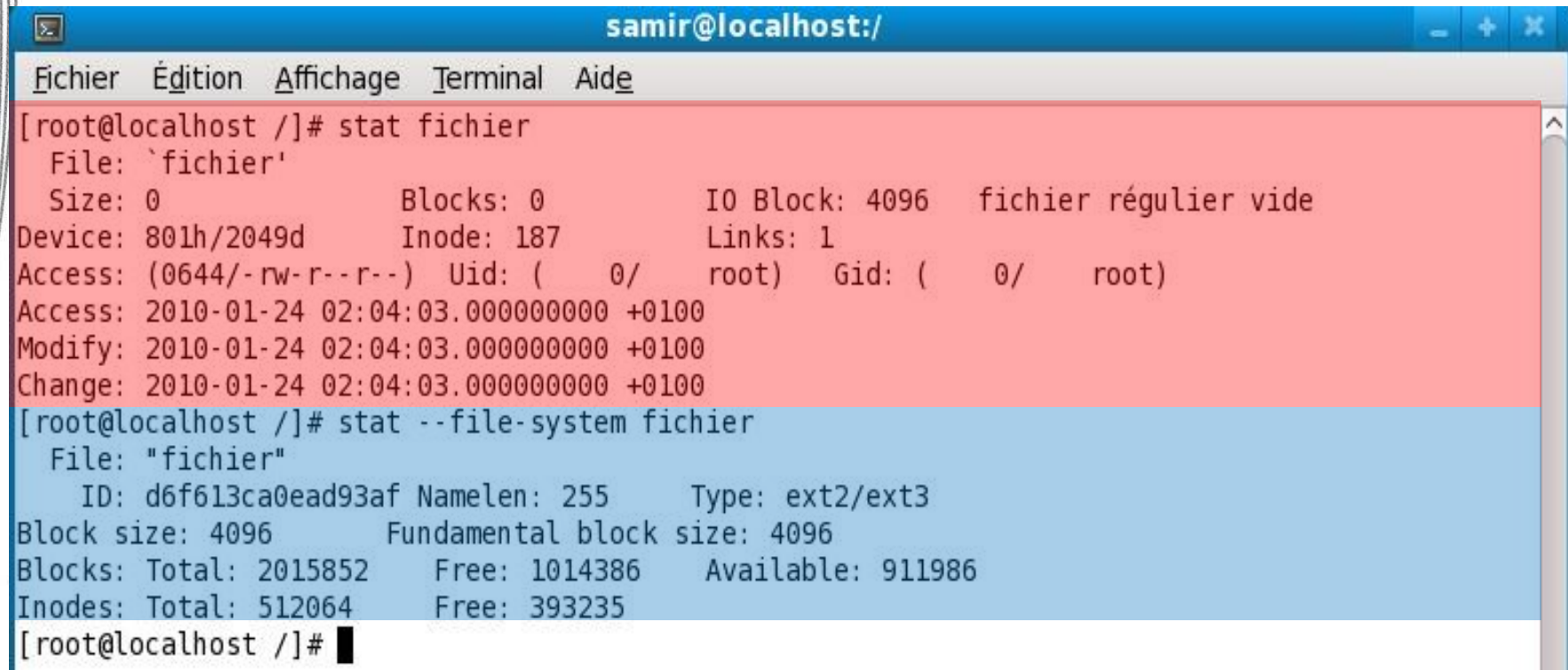


The screenshot shows a terminal window titled 'samir@localhost:/' with a menu bar containing 'Fichier', 'Édition', 'Affichage', 'Terminal', and 'Aide'. The terminal displays the following commands and their outputs:

```
[root@localhost /]# ls -i fichier
187 fichier
[root@localhost /]# df -i /dev/sda1
Sys. de fich.      Inodes    IUtil.  ILib.  %IUtil. Monté sur
/dev/sda1          512064   118828   393236   24% /
[root@localhost /]# df /dev/sda1
Sys. de fich.      1K-blocs    Occupé Disponible Capacité Monté sur
/dev/sda1          8063408    4005864   3647944   53% /
[root@localhost /]#
```

Les Inodes

La Commande stat

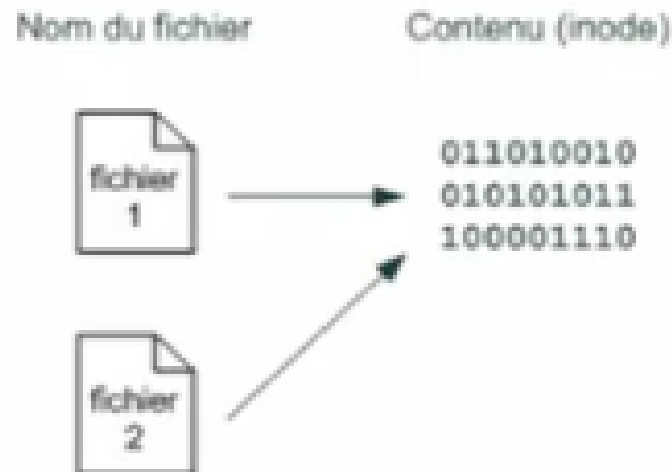


A terminal window titled 'samir@localhost:/' with a menu bar containing 'Fichier', 'Édition', 'Affichage', 'Terminal', and 'Aide'. The window shows the output of two 'stat' commands. The first command, 'stat fichier', displays file metadata for 'fichier' on a red background. The second command, 'stat --file-system fichier', displays filesystem statistics for 'fichier' on a blue background.

```
samir@localhost:/  
Fichier  Édition  Affichage  Terminal  Aide  
[root@localhost /]# stat fichier  
File: `fichier'  
Size: 0          Blocks: 0          IO Block: 4096   fichier régulier vide  
Device: 801h/2049d Inode: 187         Links: 1  
Access: (0644/-rw-r--r--)  Uid: (  0/   root)   Gid: (  0/   root)  
Access: 2010-01-24 02:04:03.000000000 +0100  
Modify: 2010-01-24 02:04:03.000000000 +0100  
Change: 2010-01-24 02:04:03.000000000 +0100  
[root@localhost /]# stat --file-system fichier  
File: "fichier"  
ID: d6f613ca0ead93af Namelen: 255   Type: ext2/ext3  
Block size: 4096      Fundamental block size: 4096  
Blocks: Total: 2015852 Free: 1014386   Available: 911986  
Inodes: Total: 512064 Free: 393235  
[root@localhost /]#
```

Créer un lien physique

Un lien physique permet d'avoir deux ou plusieurs noms de fichiers qui partagent exactement le même contenu, c'est-à-dire le même inode.



Créer un lien physique

In fichier 1 fichier 2

Cette commande permet de créer un lien physique de nom fichier 2 pour le fichier fichier1

touch fich1

ls -l

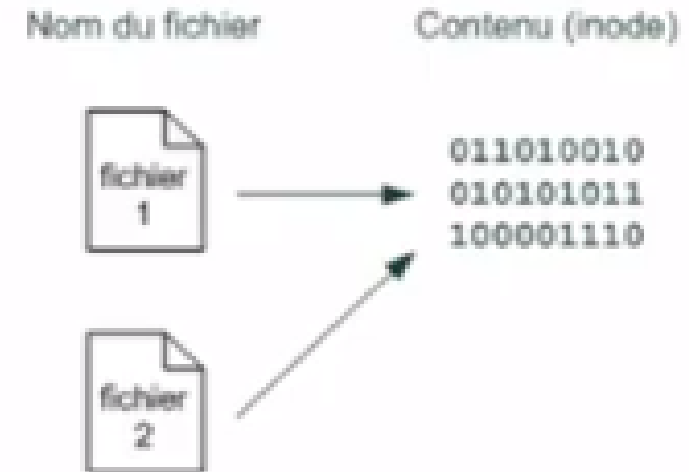
-rw-r--r-- 1 ahmad ahmad 0 Oct 31 23:25 fich1

In fich2 fich1

ls -l

-rw-r--r-- 2 ahmad ahmad 0 Oct 31 23:25 fich1

cat fich1



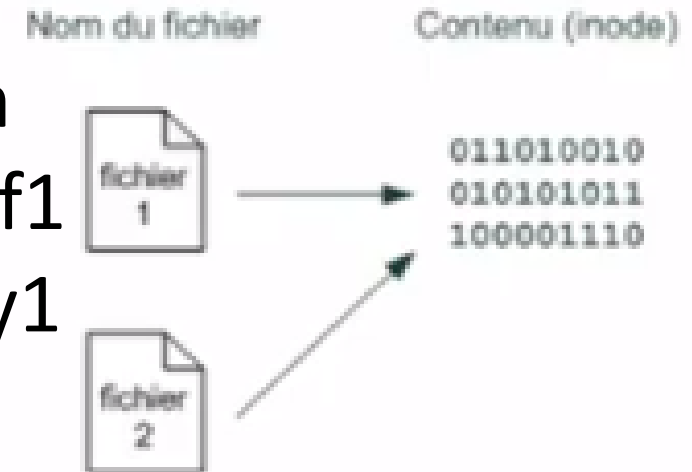
Savoir le inode

La commande `ls -i` permet de savoir les liens physiques (inodes) des fichiers

ls -i

```
ahmad@DESKTOP1:~$ ls -i
```

```
12103423998681339 fich 1125899907331478 m  
8725724278133611 doss1 15762598695900535 f1  
4222124650799754 fich1 2533274791305200 tty1  
4222124650799754 fich2
```



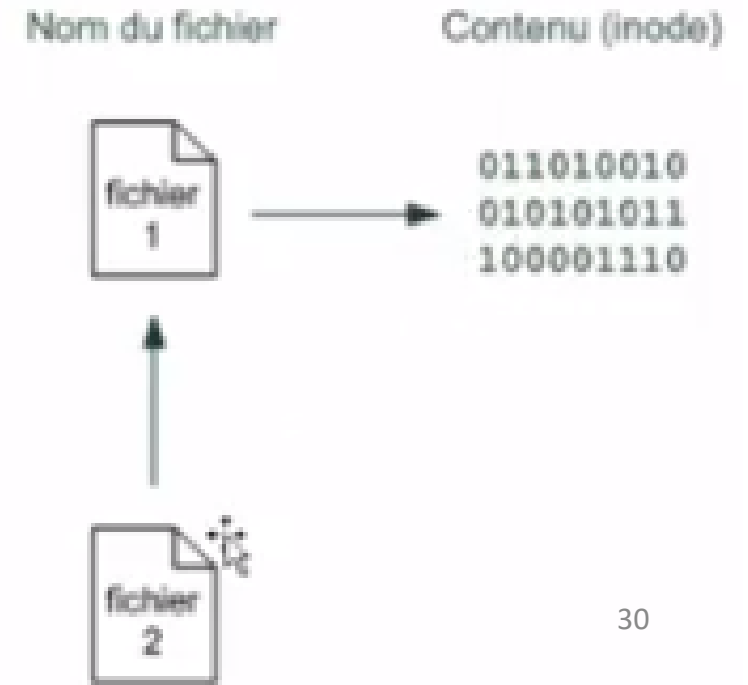
Créer des liens symboliques

- Les liens symboliques ressemblent plus aux « raccourcis »
- Son principe permet de créer un lien vers un autre nom de fichier. Cette fois, on pointe vers le nom de fichier et non pas vers l'inode directement
- `ln -s f1 f3`
- f3 est un lien symbolique de f1
- `ls -l`

```
-rw-r--r-- 1 ahmad ahmad 41 Oct 31 23:43 f1
```

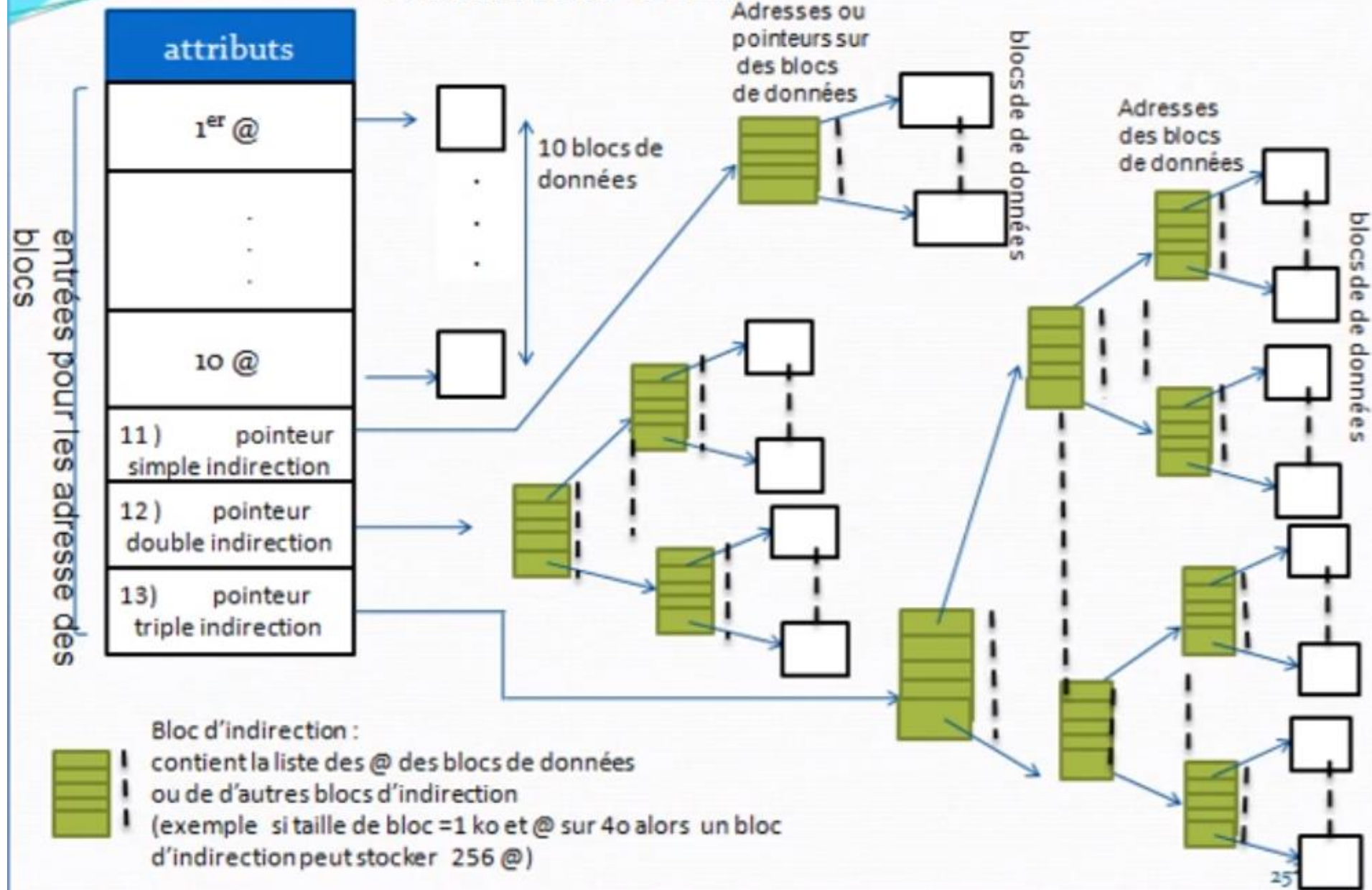
```
-rw-r--r-- 1 ahmad ahmad 18 Oct 26 10:09 f2
```

```
lrwxrwxrwx 1 ahmad ahmad 2 Oct 31 23:43 f3 -> f1
```



Allocation indexée : index node (i-node)

Solution Unix



Editeur de fichier

- Nano
 - Vim
 - Emacs
 - Gedit
-
- touch f1
 - vi f1
 - i pour insérer (pour écrire)
 - Echap puis deux points « : »
 - wq ou x enregistrer et quitter

Editeur de fichier : vi

- **vi** est un éditeur de fichiers qui contiennent des lignes de texte. Il fonctionne en mode écran; le nom **vi** provient du mot *visual*. l
- **Quelques commandes essentielles**
 - Démarrer l'éditeur : **vi nomdufichier_à_éditer** (mode **commande**)
 - Sauvegarder un fichier : **:w nom_du_fichier**
 - Quitter l'éditeur en sauvegardant le fichier: **:x**
 - Quitter sans sauvegarder : **:q**

Mode insertion

Ce mode est invoqué par une des commandes :

- **i** : insère des caractères après le curseur
- **A** : ajoute des caractères à la fin d'une ligne où que soit positionné le curseur
- **o** : insère une ligne après le curseur
- **O** : insère une ligne avant le curseur
- **a** : insère après le curseur
- **retour** insère une fin de ligne
- **Lorsque vous êtes en mode insertion taper ECHAP (ou ESC) pour revenir au mode commande**
- <https://linux.goffinet.org/administration/traitement-du-texte/editeur-de-texte-vi/>