A. Les fichiers

On entend par fichier sous Unix, la structure contenant des données utilisateur. Le terme "fichier" est remplacé par "document" sous d'autres systèmes d'exploitation.

Les fichiers standards sont constitués d'une suite de caractères ou flux d'octets dont le format n'est pas imposé par le système mais par les applications.

L'analogie entre octet et caractère tient au fait qu'un caractère est codé avec un octet ; la taille d'un fichier peut donc être donnée indifféremment en octets ou en caractères.

Les répertoires sont des fichiers particuliers pouvant contenir plusieurs autres fichiers, répertoires ou non. Cela permet d'organiser les fichiers de façon hiérarchique et arborescente.

Il existe en fait sept types de fichiers sous Linux, repérés par une lettre différente en début de ligne à l'affichage de la commande **`ls -l`** :

- **`-`** : fichier standard ;
- **`d`** : répertoire (*directory*) ;
- **`b`** : périphérique de type bloc ;
- **`c`** : périphérique de type caractère ;
- **`l`** : lien symbolique ou logique (*soft link*) ;
- **`p`** : tube nommé (*named pipe*) pour la communication entre processus ;
- **`s`** : socket, comme les tubes nommés mais généralement dans un contexte réseau.

Voici un exemple de fichier de chaque type :

```
[root]# ls -ld /etc/motd /lib /dev/hda /dev/null /etc/rc.local /dev/initctl /dev
/log
brw-r-----   1 root disk 3, 0 jun 16  2005 /dev/hda
prw-------   1 root root    0 jun 16 16:59 /dev/initctl
srw-rw-rw-   1 root root    0 jun 16 16:59 /dev/log
crw-rw-rw-   1 root root 1, 3 jun 16  2005 /dev/null
-rw-r--r--   1 root root    0 jan 13  2000 /etc/motd
lrwxrwxrwx   1 root root   13 jun 13 20:12 /etc/rc.local -> rc.d/rc.local
drwxr-xr-x  11 root root 4096 jun 16 16:24 /lib
```

Les fichiers sont stockés et référencés dans un système de fichiers. Ils peuvent être référencés par plusieurs noms.

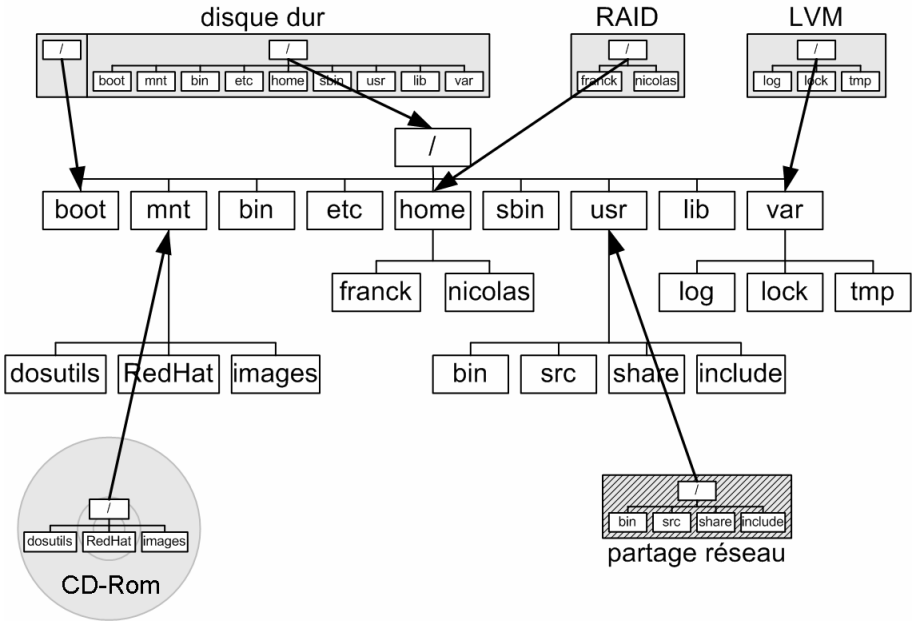
B. Arborescence Linux

L'arborescence de fichiers Linux respecte, à quelques répertoires près, le FHS (*Filesystem Hierarchy Standard*) mis en place dans le but d'homogénéiser la structure des systèmes de fichiers Unix. Ce document, qui détaille le nom et le contenu des répertoires, est disponible à l'adresse http://www.pathname.com/fhs.

C. Systèmes de fichiers

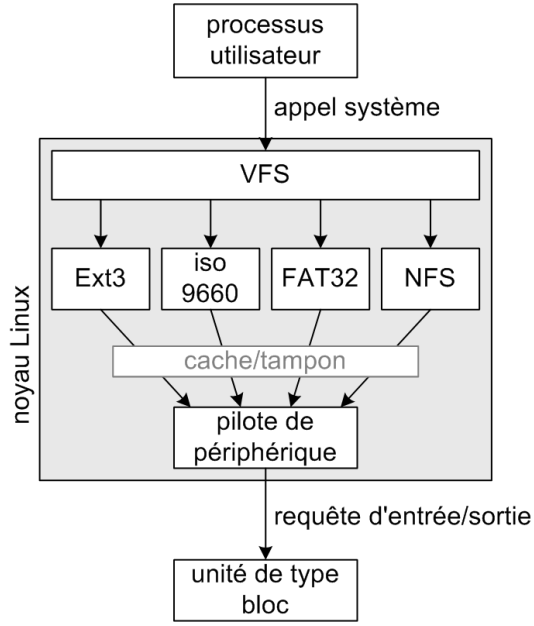
Un système de fichiers est une structure dotée d'une organisation hiérarchique permettant de stocker des fichiers sur toute unité de type bloc ; en particulier les disquettes, les disques durs, les volumes RAID et les matrices LVM que nous avons vus au chapitre précédent.

L'arborescence de fichiers Linux peut être composée de plusieurs systèmes de fichiers stockés sur des unités de bloc distinctes. Une fois le système de fichiers racine (contenant le répertoire / du système) chargé au démarrage, les autres sont chargés à leur tour et rattachés à l'arborescence sur des points de montage matérialisés par des répertoires généralement vides.



Chaque système de fichiers impose son propre format d'enregistrement des fichiers sur le périphérique de stockage.

Linux supporte plusieurs systèmes de fichiers grâce à l'utilisation d'un système de fichiers virtuel (VFS) acceptant tous les appels système de base inhérents à la manipulation des fichiers sous Unix ; ce système de fichiers virtuel permet de faire le lien entre les applications et les différentes implémentations des systèmes de fichiers.



Le VFS met donc en place un modèle de fichier commun à tous les systèmes de fichiers. Ceci dit, ce modèle est intimement lié aux systèmes de fichiers Unix et une translation avec perte de données sera effectuée pour certains systèmes de fichiers non Unix. Par exemple, il ne sera pas possible d'enregistrer les informations concernant les droits des fichiers sur un système FAT32 car celui-ci n'intègre pas de notion d'inode.

1. Concepts communs aux systèmes de fichiers Unix

Tous les systèmes de fichiers de type Unix sont constitués avec les mêmes éléments de base et fonctionnent sur le même principe. Nous allons donc présenter ici les fondements de ces systèmes de fichiers utilisés par Linux.

Les partitions accueillant un système de fichiers Unix sont segmentées en blocs de 1 ko par défaut. Chaque bloc peut prendre la forme d'une des quatre structures suivantes :

- superbloc (*superblock*) ;
- inode ;
- bloc d'indirection ;
- bloc de données.

a. Superbloc

Ce bloc contient les informations relatives au système de fichiers ; on y trouvera entre autres :

- la taille des blocs ;
- la taille du système de fichiers ;
- le nombre de montages effectués pour ce système de fichiers ;
- un pointeur vers la racine du système de fichiers ;
- les pointeurs vers la liste des inodes libres ;
- les pointeurs vers la liste des blocs de données libres, etc.

Ce bloc très important est le premier bloc du système de fichiers. Par mesure de sécurité, il est dupliqué tous les 8 ou 16 Mo sur le système de fichiers ; des copies de celui-ci seront donc disponibles aux emplacements 8193, 16385... Ainsi, il sera toujours possible de monter le système de fichiers même si le premier bloc est physiquement endommagé.

b. Inodes et blocs d'indirection

Chaque fichier est représenté par une structure appelée inode (*index node*). Ces inodes sont regroupés dans une table et sont identifiables par un numéro. La commande **`ls -i`** retourne le numéro d'inode associé à un fichier.

L'inode contient la totalité des informations sur le fichier, excepté son nom :

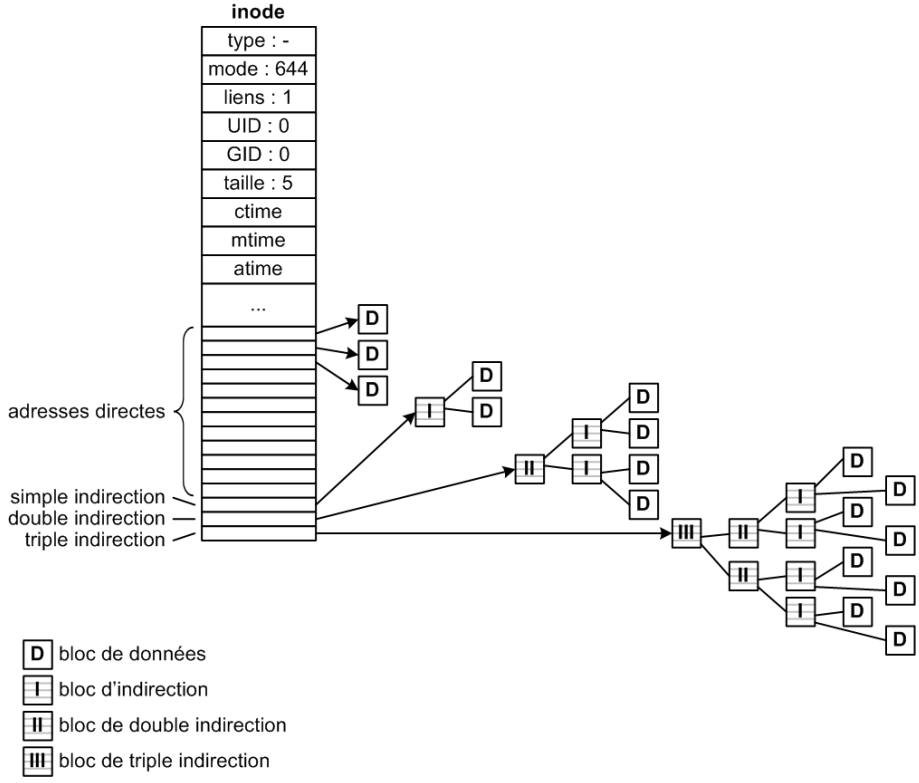
- type de fichier : **`-`**, **`d`**, **`b`**, **`c`**, **`l`**, **`p`**, **`s`** ;
- mode ou droits d'accès ; par exemple : **`rwxr-x---`** ;
- nombre de liens physiques (*hard links*) : correspondant au nombre de références, c'est-à-dire au nombre de noms ;
- UID du processus créateur ou affecté par **`chown`** ;
- GID du processus créateur, affecté par **`chgrp`** ou hérité du répertoire (bit SGID) ;
- taille du fichier en nombre d'octets ;
- date de dernier accès (atime), de dernière modification du contenu (mtime) et de dernière modification des informations contenues dans l'inode (ctime) ;
- adresses pointant vers les blocs de données qui constituent ce fichier.

Les inodes sont tous de même taille : 128 octets. Il y en a donc 8 par bloc d'inodes sur le système de fichiers.

La place pour les informations étant limitée, il n'y a que 15 adresses, codées sur 4 octets (32 bits), présentes dans l'inode :

- Douze d'entres elles pointent directement vers des blocs de données de 1 ko : cela permet donc d'enregistrer des fichiers d'une taille maximale de 12 ko.
- Une adresse vers un bloc d'indirection de 1 ko contenant lui-même 256 adresses vers des blocs de données. Cette adresse permet d'augmenter la taille maximale des fichiers de 256 ko.

- Une adresse vers un bloc de double indirection pointant à son tour vers 256 blocs de simple indirection. La taille maximale des fichiers adressables croît encore de 65 536 ko (64 Mo).
- Une adresse de triple indirection qui, cette fois-ci, pointe vers un bloc adressant 256 blocs de double indirection. La taille maximale des fichiers atteint alors 16 Go supplémentaires.



Ce système d'adressage permet donc en théorie d'adresser des fichiers d'une taille maximale de plus de 16 Go (16 840 020 ko) pour des blocs de 1 ko et de dépasser 4 To avec des blocs de 4 ko. Cependant, en raison de limitations ISO/ANSI C portant sur la taille des fichiers sur les architectures 32 bits, la taille maximale d'un fichier est usuellement de 2 Go.

c. Blocs de données

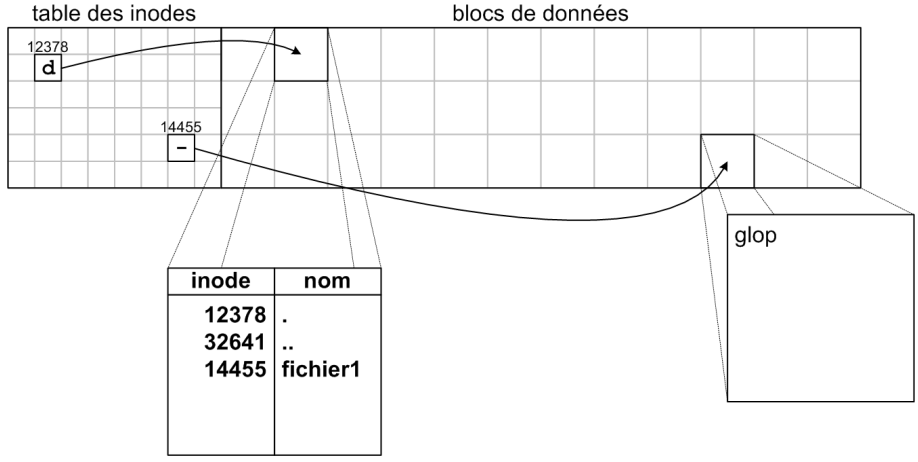
Les données réelles des fichiers sont stockées dans les blocs de données.

Dans le cas d'un fichier répertoire, les données enregistrées peuvent être représentées comme une table mettant en correspondance les noms des fichiers contenus dans ce répertoire avec leurs numéros d'inodes.

Par exemple :

```
[root]# ls -ali
total 3
12378 drwxr-xr-x    2 root   root     1024 jun 16 03:15 .
32641 drwxr-x---    4 root   root     1024 jun 16 03:13 ..
14455 -rw-r--r--    2 root   root        5 jun 16 03:14 fichier1
[root]# cat fichier1
glop
```

Peut être représenté par :



Le nom d'un fichier étant stocké dans les données du répertoire et non dans l'inode de celui-ci, il est facile de donner plusieurs noms (références ou liens durs) à ce même fichier.

L'ajout d'un nouveau nom à un fichier se fait à l'aide de :

`ln [-s] <nom du fichier> <nom supplémentaire>`

Sans l'option **`-s`**, cette commande créera un lien dur (*hard link*) tandis qu'avec l'option **`-s`**, elle créera un lien symbolique (*soft link*) :

```
[root]# ln fichier1 fichier2
[root]# cat fichier2
glop
[root]# ln -s fichier1 fichier3
[root]# ls -ali
total 4
12378 drwxr-xr-x   2 root  root   1024 jun 16 03:15 .
32641 drwxr-x---   4 root  root   1024 jun 16 03:13 ..
14455 -rw-r--r--   2 root  root      5 jun 16 03:14 fichier1
14455 -rw-r--r--   2 root  root      5 jun 16 03:14 fichier2
16214 lrwxrwxrwx   1 root  root      8 jun 16 03:15 fichier3 -> fichier1
```