

## Le système de gestion de fichiers

Toutes les applications informatiques doivent enregistrer et retrouver des informations. La solution consiste à stocker les informations dans des fichiers sur des disques. UNIX, et donc Linux, reconnaît deux types de dispositifs : les périphériques à accès direct par **blocs (comme les disques)**, et les périphériques **caractères (comme les bandes et les liaisons séries)**, dont certains peuvent être à accès direct et d'autres à accès séquentiel. Chaque périphérique (\*) supporté est représenté dans le *système de fichiers* (filesystem) par un fichier spécial sous /dev. Lorsqu'on lit ou écrit dans un tel fichier, les données proviennent du, ou vont dans, le pilote logiciel du périphérique qu'il représente.

```
ex : less /dev
      /dev/hda, /dev/cdrom, /dev/ramdisk, /dev/console, /dev/mouse0, /dev/lp0
```

(\*) Note : parmi les périphériques on distingue d'une part le disque dur (HD, *Hard Disk*) et groupe de disques (RAID, *Redundant Array of Independent (or Inexpensive) Disks*) ou des supports tels que les bandes, les disquettes (FDD, *Floppy Disk Drive*), le CD-ROM (*Compact Disk – Read Only Memory*), et d'autres part les terminaux, les imprimantes et les réseaux qui sont de type caractères .

Un disque dur peut être divisé en plusieurs partitions. Chaque système de fichiers qui réside sur une partition fonctionne comme si elle était sur un disque dur séparé. L'idée est que, si l'on a un seul disque dur et que l'on veut avoir, par exemple, deux systèmes d'exploitation dessus (GNU/Linux et M\$-Windoze) ou deux systèmes de fichiers (/ et /boot), on peut diviser le disque en deux partitions. Chaque système d'exploitation ou système de fichiers utilise sa partition comme il le désire et ne modifie pas automatiquement celle de l'autre. Les distributions GNU/Linux réalisent automatiquement des partitions et des systèmes de fichiers. Ils peuvent être aussi créés manuellement.

```
ex : fdisk -l (ATTENTION : uniquement avec l'argument -l en mode root)
      /dev/hda1 Linux (/boot), dev/hda2 Linux (/), dev/hda3 Linux Swap
```

Un système de fichiers regroupe les méthodes et les structures de données qu'un système d'exploitation utilise pour gérer les fichiers sur un disque ou une partition. Les systèmes de fichiers sous GNU/Linux tels que ext2, ext3 (ext2 journalisé) sont peu sensibles à la fragmentation contrairement au système comme FAT ou NTFS. C'est à dire que les données ne sont pas éparpillées sur tout le disque dur l'accès aux données est donc plus rapide puisqu'elles sont toutes regroupées dans la même zone. L'espace disque est administré par le système d'exploitation en unités de blocs et fragments de blocs. En ext2, fragments et blocs doivent être de la même taille. Dans les versions récentes de Linux une interface dans le système de fichiers /proc, permet d'afficher le fichier /proc/filesystems grâce à la commande cat :

```
ex : cat /proc/filesystems
      nodev    sysfs
      nodev    rootfs
      nodev    proc
              ext2
```

```
nodev   ramfs
        ext3
nodev   usbdevfs
```

Particularité du système de fichiers périphériques :

Dans le noyau 2.4 chaque bus gère ses périphériques indépendamment, il n'y aucune relation d'ordre ou de dépendance. C'est à dire qu'il n'est pas possible d'avoir une gestion commune des périphériques. Pour y remédier un *modèle unifié de périphériques* (model device) à été implémenté dans le noyau 2.6, permettant de connaître la structure du bus, l'ensemble des périphériques, des pilotes, et exporte son état sous la forme d'un fichier virtuel `sysfs` monté sous `/sys`.

## Représentation générale

Un système de fichiers est représenté par une arborescence : la plupart des systèmes d'exploitation font appel au concept de *répertoire* (directory) pour regrouper des fichiers. Un répertoire peut contenir soit des fichiers, soit d'autres répertoires. Ce modèle engendre un système de fichier avec une hiérarchie de répertoires et de fichiers organisés en arbres. Différents systèmes de fichiers, existent :  
ex : Ext2, Ext3, ReiserFS, Linux Swap (GNU/Linux). UFS, *UNIX File System* (BSD, Sun Solaris ...)

La commande permettant de monter des systèmes de fichiers s'effectue grâce à `mount` :

ex : `mount /mnt/cdrom` montage d'un `cdrom` sur le répertoire spécifique des périphériques `/mnt`.

## Fichier

Les fichiers sous GNU/Linux sont stockés sur un système de fichiers basé sur la gestion d'*unité d'allocation* (clusters) ou blocs : la plus petite unité du disque que le système d'exploitation sait gérer. Rappel : un bloc == 1 à 16 secteurs, un *secteur* (sector) = 512 *octets* (Bytes ou octets).

Différents types de fichiers existent : fichiers ordinaires (informations : ASCII ou binaires), fichiers catalogues (répertoires), fichiers spéciaux bloc (modélisent les disques) et fichiers spéciaux caractères (modélisent les périphériques E/S). Contenu d'un fichier : suite d'octets. Ces octets peuvent représenter des caractères, des nombres entiers, flottants, des codes d'opérations machines, des adresses mémoires, etc...

## Noeud d'information (inode)

Les fichiers et les répertoires sont identifiés par des inodes. Un inode contient toutes les informations concernant un fichier, sauf son nom. Le nom est stocké dans le répertoire avec le numéro d'inode. Une entrée de répertoire est composée d'un nom de fichier et du numéro d'inode qui représente le fichier. L'inode contient les numéros de plusieurs blocs de données, ceux qui sont utilisés pour stocker les données du fichier. Toutes les inodes sont identiques au sein du même partition. On peut déduire le nombre de fichiers présents sur le disque à partir du nombre d'inodes alloués :

ex : `ls -li` et `df -li`

## Le nom

Un nom élémentaire composé de caractères peut être employé dans le nom de fichiers différents. La longueur des noms de fichiers peut atteindre 255 caractères. La casse est significative (nom est différent de Nom, NOM, NoM); la tradition d'UNIX est de n'utiliser en que des minuscules. Le nom d'un fichier possède en général un *suffixe* (extension) séparé par un point qui est fonction du contenu du fichier : `.txt` pour du texte, `.c` pour du C, `.cc` ou `.C` `.cpp` pour du C++ (non limité à 3 caractères). Le nom complet d'un fichier est formé de liste des répertoires qu'il faut traverser à partir du haut de la hiérarchie, le *répertoire racine* (root directory) ou *chemin d'accès absolu* (path), plus le `nom_du_fichier`. Les répertoires sont séparés par un caractère qui dépend du système d'exploitation : `>` pour Multics, `/` pour UNIX, `\` pour Dos et Winxx, `:` pour MacOS :

ex : `/home/toto/titi/nom_du_fichier.txt` chemin d'accès absolu plus `nom_du_fichier.txt`

Le *système de fichiers racine* (root file system), soit le système de fichiers primaire est associé au répertoire le plus haut `/` :

`/bin` commandes binaires utilisateur essentielles (pour tous les utilisateurs)

`/boot` fichiers statiques du chargeur de lancement

`/dev` fichiers de périphériques

`/etc` configuration système spécifique à la machine

`/home` répertoires personnels des utilisateurs (optionnel)

`/lib` bibliothèques partagées essentielles et modules du noyau

`/mnt` point de montage pour les systèmes de fichiers montés temporairement

`/proc` système de fichiers virtuel d'information du noyau et des processus

`/root` répertoire personnel de root (optionnel)

`/sbin` binaires système (binaires auparavant mis dans `/etc`)

`/sys` *état des périphériques* (model device) et *sous-systèmes* (subsystems)

`/sys/block`, `/sys/bus`, `/sys/class`, `/sys/devices`, `/sys/firmware`, `/sys/module`,

`/sys/power`,

`/tmp` fichiers temporaires

La hiérarchie `/usr` :

`/usr/X11R6` système X Window, Version 11 Release 6

`/usr/bin` commandes utilisateurs principales

`/usr/etc` configuration système à l'échelle d'un site

`/usr/include` répertoire pour les fichiers include standards

`/usr/lib` bibliothèques pour la programmation et les packages

`/usr/local` hiérarchie locale

`/usr/man` pages de manuel

`/usr/sbin` binaires système standard non essentiels

`/usr/share` données indépendantes de l'architecture

`/usr/src` code source

La hiérarchie /var :

- /var/lib information sur l'état des applications
  - /var/lib/dhcp, /var/lib/samba, /var/lib/xdm
- /var/local données variables des logiciels de /usr/local
- /var/lock fichiers lock
- /var/log fichiers et répertoires de rapports
- /var/named fichiers de DNS
- /var/nis fichiers de la base de données du Service d'Information Réseau (NIS)
- /var/preserve fichiers sauvés après un crash ou un blocage de `ex` ou `vi`
- /var/run fichiers variables d'exécution
- /var/spool répertoires de spool fichiers d'impressions, courriers, news ...
  - /var/spool/lpd, /var/spool/cups, /var/spool/mail, /var/spool/news
- /var/tmp fichiers temporaires, utilisés pour garder /tmp petit

## Le répertoire de login (utilisateur)

A l'exception de l'administrateur système qui accède aux répertoires tels que /bin, /boot, /dev, /etc, /sbin, /usr, /var, chaque personne doit au moins pouvoir accéder à son répertoire utilisateur situé dans /home. Elle peut-être super utilisateur (#) dans le cas de root ou simple utilisateur (\$). Les mécanismes de sécurité UNIX interdisent aux utilisateurs non autorisés de commettre des bêtises par des manipulations dangereuses pouvant détruire des fichiers essentiels au système, de manière irréversible. Nous utiliserons donc le login simple utilisateur.

La configuration sur mesure est un point fort d'UNIX et de GNU/Linux. Toutes les applications essentielles (éditeurs, compilateurs, courrier, client X Window) sont configurables par des fichiers externes, afin que chaque utilisateur puisse choisir ses préférences au lieu de se contenter des valeurs par défaut. Des fichiers d'initialisation (\*) au nom précédé d'un point et suffixé rc, *ressource configuration*, se trouvent en général dans votre répertoire personnel (le . interdit la commande `ls` de les afficher en temps normal) :

`.bashrc` : script destiné au shell `bash`, susceptible de contenir des commandes ou des éléments de programmation. Exécuté à chaque lancement d'un nouveau shell.

ex : `PS1='personnal directory user /home/toto/ ] € '` insère un nouveau nom d'invite

`.dmrc` : script destiné à sélectionner le dm, *desktop manager* lors de l'ouverture de la session utilisateur lançant une interface graphique (KDE, GNOME...). Exécuté que lorsque vous vous connectez.

`.xinitrc` : script (s'il est créé par l'utilisateur) permet de déterminer quel *gestionnaire de fenêtre* (Window Manager : `gnome-session`, `startkde`, ...) sera lancé avec la commande `startx`.

*.bash\_profile* : script destiné au shell `bash`, permet de différencier les shells interactifs des shells lancés par des programmes fonctionnant en tâche de fond. Exécuté que lorsque vous vous connectez.

(\*) Notes : ces fichiers ne sont pas indispensables; tous les programmes sont suffisamment bien conçu pour appliquer les valeurs par défaut.