CHAPITRE 3 : Gestion des systèmes de fichiers

---

1. Les types de fichiers

Un fichier est un ensemble de données archivées en mémoire auxiliaire. Dans certains OS, ces données sont structurées en enregistrements suivant une organisation déterminée (séquentielle, calculée, séquentielle indexée, ...).
Sous UNIX ou Linux la notion d'enregistrement n'existe pas et par suite la notion d'organisation n'a pas de sens. Un fichier est simplement constitué d'une suite finie d'octets.
L'organisation logique du système de gestion de fichiers (**SGF**) d'UNIX ou Linux est simple, elle repose sur deux concepts : le **fichier** et le **répertoire** (ou catalogue ou dossier).

1.1. Fichier ordinaire

Un fichier ordinaire est constitué d'une suite finie d'octets. Il est caractérisé par un identifiant, un ou plusieurs noms, sa taille (nombre d'octets) et d'autres attributs qui seront décrits ultérieurement. Comme la notion d'enregistrement n'existe pas, c'est le programme d'application qui doit gérer la structuration des données et l'organisation des fichiers qu'il utilise.

On peut néanmoins distinguer deux catégories de fichiers ordinaires :
- les fichiers texte qui contiennent des données ou un script shell codés en ASCII sur 8 bits.
- les fichiers binaires qui contiennent des données ou des programmes codés en binaire dans le mode de représentation interne des informations du processeur.

1.2. Fichier spécial

Un organe périphérique réel ou virtuel est considéré du point de vue du système comme un fichier spécial dans lequel on peut lire et/ou écrire des données avec les mêmes opérations que celles utilisées pour les fichiers ordinaires (notion de fichier généralisé).
Un fichier spécial assure l'interface entre l'application et le niveau matériel (organe périphérique).
Toute opération d'E/S appliquée à un fichier spécial (`/dev/…`) active le pilote de périphérique (**device driver**) qui contrôle l'organe périphérique associé à ce fichier spécial.

Il existe deux types de fichiers spéciaux :

- les fichiers **blocs** pour lesquels les opérations d'E/S sont réalisées par blocs d'octets de taille fixe.

- les fichiers **caractères** pour lesquels les opérations d'E/S sont réalisées octet par octet.

Un fichier spécial est caractérisé de la même manière qu'un fichier ordinaire mais il ne contient pas de données. Il contient deux nombres :

- le **major number** qui identifie le pilote de périphérique

- le **minor number** qui peut être un numéro d'unité ou un paramètre du pilote.

1.3. Autres types de fichiers

Il existe d'autres types de fichiers qui peuvent être catalogués dans le SGF : tube nommé (FIFO), socket du domaine UNIX, lien symbolique, …..

1.4. Répertoire

Un répertoire (ou catalogue) est un pseudo-fichier qui contient une liste de noms de fichiers ou de répertoires avec leur identifiant. Cette définition **récursive** d'un répertoire se traduit par une structure en arbre du système de gestion de fichiers. Chaque répertoire correspond à un nœud de cet arbre.

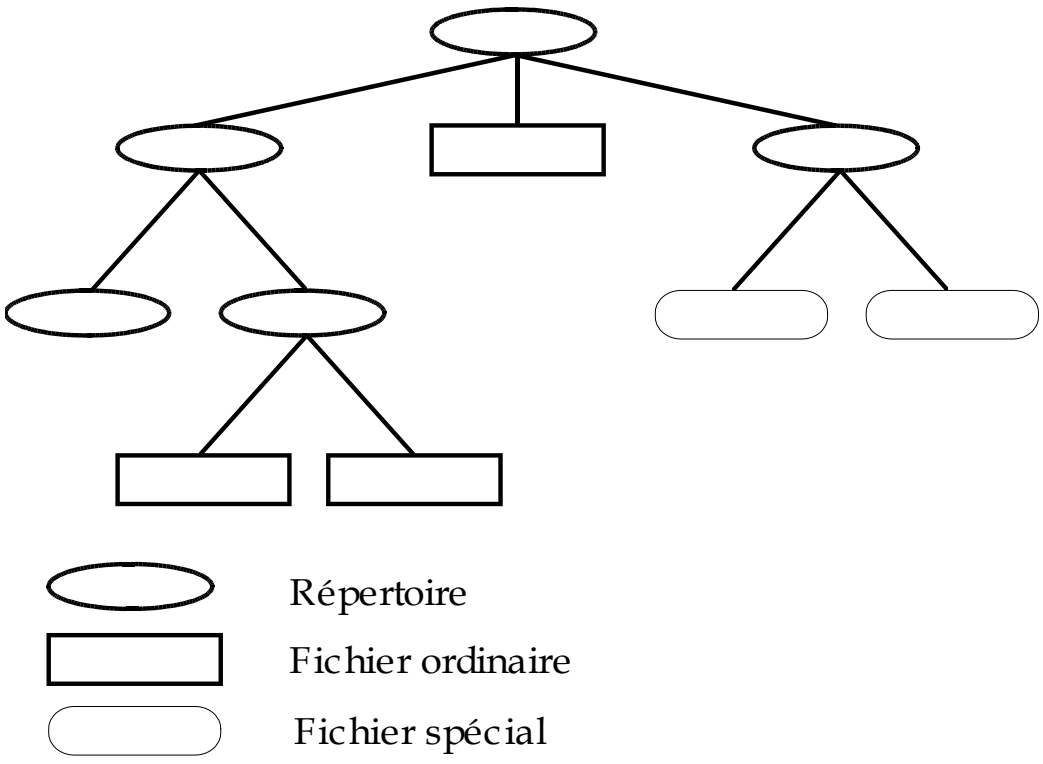
Un répertoire possède donc **un identifiant et un seul nom** dans le répertoire parent qui le contient (dans une structure en arbre un nœud ne peut pas figurer plusieurs fois dans la structure).

2. Organisation du système de gestion de fichiers (SGF)

2.1. Structure externe

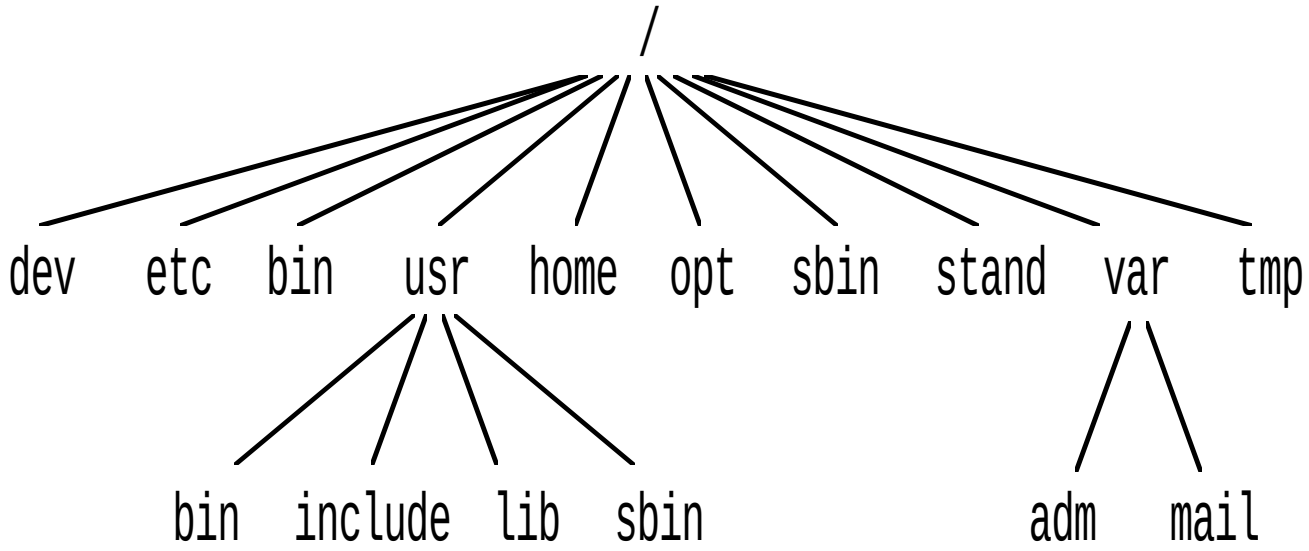
La définition récursive d'un répertoire conduit à une structure arborescente du **SGF**. Les répertoires correspondent aux **nœuds** de l'arbre et les fichiers correspondent aux **feuilles** : le SGF est dit **hiérarchisé**.

Cette structure est **dynamique**, les utilisateurs peuvent créer et détruire des fichiers ou des répertoires au gré de leurs besoins.



2.2. <u>Répertoires standards</u>

La hiérarchie standard du SGF d'UNIX ou Linux comporte un certain nombre de répertoires prédéfinis.
La figure suivante représente les principaux répertoires de cette hiérarchie pour la distribution Linux Debian :



`/dev` : contient les fichiers spéciaux qui sont associés aux périphériques réels ou virtuels
`/etc` : contient les fichiers d'administration du système (description des utilisateurs, des groupes, ...) et de configuration de la machine (table des processus à exécuter au démarrage de la machine, ...)
`/bin` : contient des commandes standard accessibles aux utilisateurs
`/home` : contient les répertoires des utilisateurs
`/opt` : contient les utilitaires additionnels (compilateurs, bases de données, logiciels d'application, ...)
`/sbin` : contient les programmes exécutés au démarrage du système
`/boot` : contient les fichiers du gestionnaire de chargement du noyau. Le noyau de Linux peut être stocké dans ce répertoire et dans la racine.
`/tmp` : contient les fichiers temporaires
`/usr` : contient divers répertoires :
- `/usr/bin` : contient des commandes standard accessibles aux utilisateurs
- `/usr/include` : contient les fichiers d'en-tête des programmes C
- `/usr/lib` : contient des bibliothèques
- `/usr/sbin` : contient les commandes d'administration du système
- `/usr/local` : contient des commandes standard et des fichiers spécifiques à la machine
- `/usr/X11R6` : contient les fichiers de configuration de l'environnement X-windows
- etc ….

`/var` : contient divers répertoires :
- `/var/log` : contient les fichiers de comptabilité et d'audit du système
- `/var/mail` : contient les boîtes à lettres locales des utilisateurs
- etc ….

`/lib` : contient des bibliothèques
`/mnt` : répertoire utilisé comme point de montage de **systèmes de fichiers** temporaires

2.3. Structure interne

Le SGF est enregistré en **mémoire auxiliaire** sur un ou plusieurs média de type:

- unités de disques magnétiques (disquettes, disques durs),
- unités de disques optiques (CD-ROM, DVD-ROM),
- cartes mémoires (clé USB, carte Memory Styck,carte Secure Digital, ...)
- ou autres.

Pour simplifier cette présentation, on se limitera aux média de type unités de disques magnétiques ou cartes mémoires qui sont gérées comme des disques virtuels.

La gestion des disques optiques étant un peu plus complexe ne sera pas abordée dans ce chapitre.

Après **formatage**, l'espace de stockage d'un disque magnétique ou d'une carte mémoire est constitué d'un ensemble de N blocs consécutifs de taille fixe (512, 1024, 2048, …octets).

Les disques durs ont une capacité de stockage beaucoup plus grande que les disquettes, c'est pourquoi il est possible de découper leur espace de stockage en plusieurs **partitions** alors qu'une disquette ne peut comporter qu'une seule partition.

Le nombre et la taille des partitions d'un disque dur peut varier en fonction de sa capacité et de l'organisation recherchée par l'administrateur.

Les cartes mémoires ont une capacité de stockage qui permettrait de découper leur espace de stockage en plusieurs **partitions**, mais elles ne comportent qu'une seule partition pour être portables sur différents OS.

Une partition est constituée par un intervalle de blocs consécutifs et est gérée comme un **disque logique** comprenant une suite de M blocs numérotés de 0 à M-1.

Le partitionnement des unités de mémoires auxiliaires sera abordé plus en détail au **chapitre 3** concernant l'installation d'un système Linux.

Une partition doit être structurée en **système de fichiers** (**système de fichiers** en anglais) pour pouvoir servir de support à tout ou partie de l'arborescence du SGF.

Un **système de fichiers** est une structure de données contenant toute l'arborescence de répertoires et de fichiers à partir d'un répertoire donné et qui est enregistrée dans une partition d'un organe périphérique du type mémoire auxiliaire.

Le système UNIX ou Linux gère une arborescence unique qui peut être enregistrée sur une ou plusieurs partitions structurées en **système de fichiers**.

L'organisation interne du SGF de **Windows** est différente de celle de UNIX ou Linux. Chaque partition d'un organe périphérique est vue par le système comme une **unité logique** supportant un seul **système de fichiers** dans lequel est enregistrée une arborescence de répertoires et de fichiers **indépendante** de celles qui sont enregistrées dans les autres unités.

Il est possible de faire fonctionner un système minimal avec un seul **système de fichiers** dans lequel est enregistrée l'arborescence du SGF en entier.

En pratique, il est nécessaire de distribuer le SGF sur plusieurs **systèmes de fichiers** pour répondre à différents objectifs ou contraintes ou les deux à la fois:

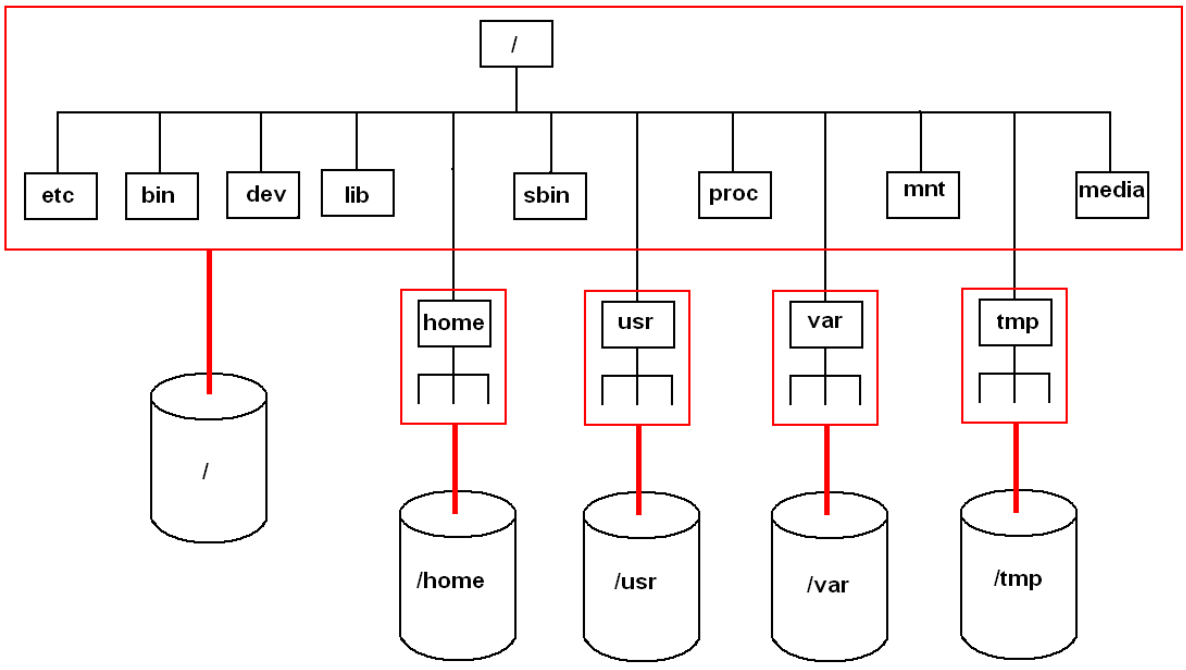
- une contrainte par exemple, est l'utilisation de supports amovibles comme les disquettes ou les CD-ROM.

  Il n'est évidement pas possible de faire des partitions à cheval sur des média différents, chaque média supporte une partition dans laquelle est enregistrée un **système de fichiers**. De plus les types de **systèmes de fichiers** utilisés pour les supports amovibles sont différents entre eux et différents de ceux utilisés pour les partitions de disques durs.

- dans le cas d'un serveur, un objectif par exemple, peut être de limiter l'espace disque de certains répertoires comme **`/usr`**, **`/home`**, **`/tmp`** ou **`/var`** qui peuvent grossir considérablement et bloquer le fonctionnement du système s'ils sont enregistrées dans la partition supportant la racine du SGF.

- dans le cas d'un serveur, un autre objectif par exemple, peut être de répartir plusieurs partitions sur deux disques durs pour paralléliser les accès aux fichiers et diminuer ainsi les temps d'accès.

- dans le cas d'un serveur, un autre objectif par exemple, peut être de faciliter l'administration de l'espace disque par sous-ensembles indépendants. En cas de problèmes sur un **système de fichiers**, il suffira de le réinstaller simplement à partir d'une sauvegarde plutôt que de réinstaller tout le SGF.



Exemple de systèmes de fichiers fixes usuels

2.4. Les différents types de systèmes de fichiers

Un **système de fichiers** est une structure de donnée qui contient d'une part les données des fichiers et répertoires de l'arborescence qu'elle supporte et d'autre part un ensemble de tables qui contrôlent l'accès à ces données.

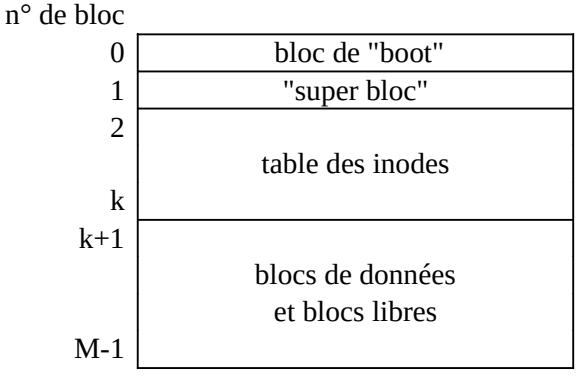
Il existe différents types de **systèmes de fichiers** en fonction des modes d'organisation des données, des types de tables d'allocation et des types de contrôle d'accès à ces données mis en oeuvre.

La commande **man fs** permet de connaître le types de **systèmes de fichiers** utilisables sous **Linux.**

Exemples :

- **minix** système de fichiers local des débuts de Linux, noms de fichiers de 14 ou 30 caractères.
- **ext2** système de fichiers local standard de Linux, noms de fichiers longs.
- **ext3** extension de ext2 avec journalisation.
- **sysv** système de fichiers local d'UNIX SystemV.
- **ffs** système de fichiers local d'UNIX BSD.
- **ufs** extension du ffs, système de fichiers local d'UNIX BSD.
- **msdos** système de fichiers local FAT16 pour les partitions MS DOS.
- **vfat** extension de msdos, système de fichiers local FAT32 pour les partitions WINDOWS.
- **ntfs** système de fichiers local pour les partitions WINDOWS NT/2000/XP.
- **hpfs** système de fichiers local pour les partitions HPFS.
- **iso9660** système de fichiers local pour les CD-ROM.
- **nfs** système de fichiers distant accessibles par réseau, développé par SUN Microsystem.

2.4.1. Structure du système de fichiers `sysv` d'UNIX systemV de AT&T



- le **bloc 0** contient l'identification du disque logique et le code de démarrage.

- le **bloc 1**, appelé "**super bloc"** contient les attributs du **système de fichiers**:

  - taille du **système de fichiers**,
  - taille de la table des **inodes**, qui est le nombre maximum de fichiers, tous types confondus, que peut contenir le **système de fichiers**,
  - un pointeur sur la liste chaînée des inodes libres,
  - un pointeur sur la liste chaînée des blocs libres (free list).

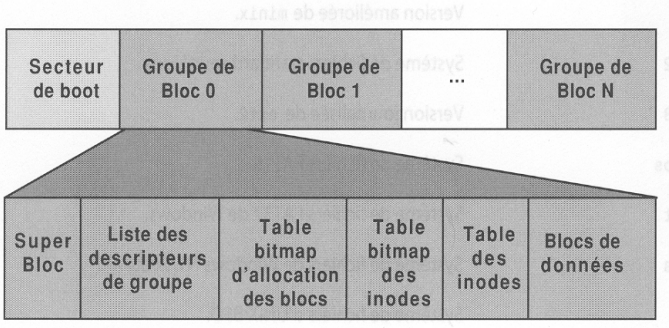
- les **blocs 2 à k** contiennent la table d'allocation des fichiers.
Tout fichier archivé dans le **système de fichiers** est représenté par un ensemble de blocs de données où sont stockés les octets constituant le fichier et par une structure, appelée **inode** (index node) contenant tous les attributs de ce fichier. Cet **inode** est stocké dans la table d'allocation.
L'indice de cet **inode** dans la table d'allocation sert d'**identifiant** de ce fichier dans le **système de fichiers**.
Un répertoire est simplement un fichier qui contient des couples (numéro de **inode**, nom de fichier).
A sa création, il contient deux couples, le premier avec le nom "**.**" et son propre numéro de inode,
le second avec le nom "**..**" et le numéro de inode du répertoire parent.

- les **blocs k+1 à M-1** contiennent d'une part les données des fichiers et répertoires de l'arborescence supportée par le système de fichiers et d'autre part ou sont libres.

2.4.2. Structure du système de fichiers `ext2` de Linux

Le **système de fichiers `ext2`** (Second extended File System) a été développé pour pallier les inconvénients des systèmes de fichiers originels **`minix`** et ensuite **`ext`**, en particulier le besoin de supporter des partitions de grande taille étant donnée la capacité sans cesse croissante des disques durs.

Il s'inspire du système de fichiers **`ffs`** de l'UNIX BSD. Quoique encore très utilisé, on lui préfère l'**`ext3`** qui est en fait un système de fichiers **`ext2`** à base de journaux, ce qui permet d'améliorer les performances et une correction plus rapide des erreurs.



La partition commence par un secteur de "boot" classique de 512 octets et est ensuite divisée en plusieurs groupes de blocs de même taille. Chaque groupe de blocs comprend 6 parties:

- le "**super bloc**" qui contient les informations de structure du systèmes de fichiers
- la **liste des descripteurs de groupes** qui localise dans la partition les informations essentielles de chaque groupe (localisation des tables)
- la **table bitmap d'allocation des blocs** du groupe qui associe à chaque bloc un bit d'état indiquant si le bloc est libre ou occupé
- la **table bitmap d'allocation des inodes** du groupe qui associe à chaque i-node un bit d'état indiquant si l'inode est libre ou occupé
- la **table des inodes** du groupe
- les blocs de données

Le "**super bloc"** contient les informations suivantes:

- la taille des blocs
- le nombre de blocs libres
- la taille en nombre de blocs du système de fichiers
- le nombre de blocs réservés à **root**
- le nombre total d'inodes
- le nombre d'inodes libres
- l'état du système de fichiers (monté ou démonté)
- le nombre de blocs par groupe de blocs
- le nombre d'inodes par groupe de blocs
- des informations sur les dates et les nombres de montages et de contrôle du système de fichiers

La répétition du "**super bloc"** et de la **liste des descripteurs de groupes** en début de chaque groupe améliore la fiabilité de ce type de système de fichiers. En effet, si un système de fichiers possède un seul "**super bloc"**, la moindre erreur dans le "**super bloc"** suffit à rendre le système de fichiers inutilisable.

3. Désignation des fichiers

3.1. Noms de fichier

Au niveau de l'utilisateur, un fichier (ou répertoire) est désigné symboliquement par un nom dans le répertoire où il est catalogué. Ce nom est composé d'une chaîne de caractères de longueur variable.
Ces caractères peuvent être quelconques, mais il est recommandé de ne pas utiliser les caractères spéciaux utilisés par le Shell.

La longueur maximale d'un nom de fichier dépend du type de **système de fichiers** qui le supporte.
Par exemple, elle est de 14 caractères pour un **système de fichiers `sysv`** et de 255 caractères pour un **système de fichiers `ext2`**.

Un nom de fichier n'a pas de structure prédéfinie au niveau du système, mais certains utilitaires nécessitent l'adjonction d'un suffixe, par exemple "**`.c`**" pour un fichier source en C.

3.2. Répertoire de connexion, répertoire de travail

Chaque utilisateur catalogué dans le système, possède un répertoire personnel qui lui à été attribué par l'administrateur du système. Lors de la connexion, l'interpréteur de commandes prend ce répertoire comme répertoire de travail (ou répertoire courant) initial.
Ce répertoire est appelé répertoire de connexion (**home directory**).

L'utilisateur peut ensuite à tout moment se positionner dans un autre répertoire qui devient le nouveau répertoire de travail (**working directory**) encore appelé répertoire courant.

3.3. Chemin d'accès à un fichier

Lorsqu'un fichier est désigné uniquement par son nom, l'interpréteur de commandes le recherche dans le répertoire de travail (ou répertoire courant).
Pour désigner un fichier qui n'est pas dans le répertoire de travail, il faut préfixer son nom par la désignation d'un **chemin d'accès** qui est la liste, séparée par des barres de division ("`/`"), des répertoires à traverser pour l'atteindre.
Ce chemin peut partir du répertoire racine (**root**), on le qualifie alors de **chemin absolu** et il doit commencer par une barre de division, ou partir du répertoire de travail, on le qualifie alors de **chemin relatif** et il ne doit pas commencer par une barre de division.

Exemples :

`/usr/sbin/useradd`	(chemin absolu)
`TP1/creerComptes.sh`	(chemin relatif)

Dans l'écriture d'un chemin d'accès, les notations particulières suivantes peuvent être utilisées :

- `/` désigne le répertoire racine (root)
- `.` désigne le dernier répertoire cité dans le chemin (répertoire courant)
- `..` désigne le répertoire **parent** du dernier répertoire cité dans le chemin

3.4. Liens physiques et liens symboliques

Au niveau interne du système, un fichier (ou répertoire) est identifié par son numéro du **inode** dans le **système de fichiers** qui supporte ce fichier. Le nom du fichier est simplement un identificateur symbolique employé par l'utilisateur pour désigner le fichier dans une commande.

Un répertoire est en fait un fichier particulier qui contient une liste de couples (**liens**) formés d'un nom et d'un numéro de **inode**.

Tout fichier **excepté les répertoires**, peut donc être catalogué dans des répertoires différents avec des noms identiques ou différents ou bien dans un même répertoire avec des noms différents. Supprimer un fichier d'un répertoire revient simplement à supprimer le lien associé, la suppression physique du fichier (libération du inode et des blocs de données) n'intervient que lorsqu'on supprime le dernier lien associé à ce fichier.

Ces **liens** dits **physique** (**hard links**) ne peuvent être créés que dans des répertoires supportés par un même **système de fichiers** (celui qui contient l'inode et les blocs de données du fichier).

Pour pallier à cette restriction, les versions récentes d'Unix ou Linux permettent de créer un lien spécial, appelé **lien symbolique**, sur un fichier contenu dans un **système de fichiers** quelconque. Un lien symbolique est un fichier qui contient une chaîne de caractères représentant un chemin d’accès vers un autre lien (symbolique ou physique) du fichier.

4. Attributs d'un fichier

Chaque fichier (ou répertoire) est caractérisé par des attributs dont les principaux sont :

- type de fichier (ordinaire, répertoire, …),
- droits d'accès,
- compteur de liens physiques,
- UID du propriétaire,
- GID du groupe,
- taille en nombre d'octets,
- dates et heures de création, de dernière modification et de dernier accès,
- références des blocs de données occupés par le fichier sur le média.

Ces attributs sont mémorisés dans l'inode qui est associé au fichier. Le système Unix ou Linux est un système multi-utilisateur qui gère la protection des informations de chaque utilisateur du système à partir de certains attributs des fichiers.

4.1. Les classes d'utilsateurs

Chaque utilisateur est affecté à un groupe initial lors de la création de son compte par l'administrateur. Il peut en plus être inscrit dans d'autres groupes
Le système distingue donc 3 classes d'utilisateurs relativement à l'accès à un fichier donné :

- la classe du **propriétaire** (**user**), qui correspond à l'utilisateur qui a créé le fichier ou qui en à hérité
- la classe des utilisateurs appartenant à un **groupe** (**group**) existant
- la classe des **autres utilisateurs** (**others**) qui ne font pas partie de 2 classes précédentes

4.2. Droits et modes d'accès

Pour chaque classe, trois types de **droits** d'accès au fichier sont définis avec un **mode** d'accès qui diffère suivant le type du fichier :

- droit **`r`** (**read**) qui autorise :
  - la lecture du contenu d'un fichier ordinaire ou spécial
  - la lecture du contenu d'un répertoire

- droit **`w`** (**write**) qui autorise :
  - l'écriture dans un fichier ordinaire ou spécial
  - l'ajout, la suppression ou la modification d'un lien dans un répertoire (supprimer ce droit pour un répertoire n'interdit pas d'écrire dans les fichiers qu'il contient et qui possèdent le droit w)

- droit **`x`** (**execute**) qui autorise :
  - l'exécution d'un fichier ordinaire (programme objet écrit dans le langage interne du processeur ou fichier de commandes shell)
  - l'entrée dans un répertoire pour accéder aux fichiers ou répertoires qu'il contient.
    Interdire ce mode d'accès pour un répertoire est le plus sûr moyen de le protéger.

Les droits d'accès à un fichier pour les trois classes d'utilisateurs sont matérialisés par neuf indicateurs binaires codés dans un mot de 16 bits de la façon suivante :

user			**group**			**others**		
bit 8	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
r	**w**	**x**	**r**	**w**	**x**	**r**	**w**	**x**

Si le droit est accordé, le bit correspondant vaut 1, il vaut 0 sinon.
Seuls le propriétaire du fichier et **root** peuvent modifier les droits d'accès accordés à chacune des trois classes d'utilisateurs (commande **chmod**).
Seul **root** peut changer le propriétaire ou le groupe propriétaire d'un fichier (commandes **chown** et **chgrp**).
Le SGF d'Unix (ou Linux) peut être très permissif, c'est à chaque utilisateur de protéger ses fichiers suivant ses désirs.

Il existe trois autres indicateurs codés sur les bits 9, 10 et 11, qui sont spécifiques aux fichiers exécutables.
Leur utilisation sera étudiée plus tard dans le cadre de la gestion des processus.

4.3. Masque de suppression des droits d'accès

Chaque fois qu'un fichier ou un répertoire est crée, il est affecté de droits d'accès par défaut suivants:

- **rw- rw- rw-** pour les fichiers ordinaires.
- **rwx rwx rwx** pour les répertoires.

L'utilisateur propriétaire de ce fichier possède un masque binaire de 9 bits (**user mask**) qui permet de supprimer un droit lorsque le bit correspondant vaut 1 et de le maintenir lorsque le bit correspondant vaut **0**.

Exemple :

Droits d'accès par défaut => **r w – r w – r w –**
Masque octal **077** = > **0 0 0 1 1 1 1 1 1**
Droits d'accès résultants => **r w – – – – – – –**

La valeur courante de ce masque peut être visualisée ou modifiée avec la commande **umask**.
La valeur de ce masque est en général initialisée par l'utilisateur à **077** ou **022** (en octal) dans le fichier de démarrage du shell (par exemple dans le fichier **.bash_profile** sous Linux).

5. Commandes d'administration des systèmes de fichiers

5.1. Fichiers spéciaux

Sous **Linux,** les fichiers spéciaux qui sont les interfaces avec les pilotes de périphériques (device driver) qui gèrent les opérations d'E/S dans les partitions des unités de disques magnétiques ou optiques sont:

- **/dev/hda1** pour la partition 1 du premier disque dur de type IDE
- **/dev/hda2** pour la partition 2 du premier disque dur de type IDE
- **…….**
- **/dev/hdb1** pour la partition 1 du deuxième disque dur de type IDE
- **…….**
- **/dev/sda1** pour la partition 1 du premier disque dur de type SCSI
- **…….**
- **/dev/fd0** pour le premier lecteur de disquette de type IDE (unité **A:** sous Windows)
- **/dev/cdrom** pour le lecteur de CD-ROM

5.2. Création d'un système de fichiers

Sous **Linux**, la commande **man fs** permet de connaître les types de **systèmes de fichiers** utilisables**.**

La création d'un **système de fichiers** sous Unix ou Linux s'effectue par le biais de la commande **`/sbin/mkfs`** dont la forme la plus standard est :

`mkfs` *options filesys*

où *filesys* est soit le nom d'un fichier spécial soit le nom d'un point de montage.

On consultera la page de manuel relative à cette commande pour de plus amples détails sur les options.

Au moment de la création d'un **système de fichiers**, l'administrateur peut choisir différents paramètres de configuration tels la taille des blocs logiques et le nombre de fichiers supportés par le **système de fichiers**.

Sous **Linux**, plusieurs alias de cette commande sont fournis, par exemple:

`mkfs.ext2` ou **`mke2fs`** qui équivaut à **`mkfs -t ext2`**

Les fichiers spéciaux associés au partitions des disques durs appartiennent à **root** et au **groupe root** avec les droits **rw** et n'ont aucun droits pour **others**. En effet par sécurité, seul **root** peut créer des **système de fichiers** sur ce type de partitions car on peut écraser un **système de fichiers** d'une partition en créant un nouveau **système de fichiers** dans cette partition.

La commande df permet d'afficher l'état d'un **système de fichiers** (cf. le **man**).

5.3. Montage et démontage d'un système de fichiers

Pour pouvoir manipuler les fichiers et répertoires supportés par un **système de fichiers,** ce **système de fichiers** doit être attaché (monté) à un **répertoire existant** de l'arborescence du SGF.
C'est la commande **`/sbin/mount`** qui permet d'attacher un **système de fichiers** à l'arborescence du SGF.
A l'inverse **`/sbin/umount`** le détachera à nouveau.

Plusieurs formes d'appel sont envisageables. La forme la plus standard de la commande **`mount`** est :

> **`mount`** *options fichier-spécial répertoire*
>
> Cette commande attache le **système de fichiers** se trouvant sur le *fichier-spécial* (**/dev/…**) au répertoire *répertoire*.
> Le contenu précédent du *répertoire*, ainsi que ses propriétaire et ses droits d'accès initiaux sont remplacés par ceux de la racine du **système de fichiers** se trouvant sur le *fichier-spécial*.

La commande **`mount`** sans paramètres affiche la liste des **système de fichiers** montés, liste qui est contenue dans le fichier **`/etc/mtab`**.

La commande **`umount`** permet de détacher le(s) **système de fichiers** mentionné(s) de la hiérarchie du SGF.
Un **système de fichiers** est spécifié soit en donnant le nom du répertoire où il est monté, soit en donnant le nom de **fichier spécial** sur lequel il se trouve.
Notez qu'un **système de fichiers** ne peut être démonté quand il est occupé, par exemple quand il y a des fichiers ouverts dessus ou quand certains processus y ont leur répertoire de travail.

Le fichier **`/etc/fstab`** contient des informations sur les différents **système de fichiers**.
Chaque **système de fichiers** est décrit sur une ligne indépendante. Les champs contenus sur chaque ligne sont séparés par des espaces ou des tabulations :

1. Le premier champ (**fs_spec**), décrit le **fichier spécial** ou le **système de fichiers** à monter.

2. Le second champ (**fs_file**), indique le répertoire de montage du **système de fichiers**.
   Pour les partitions de **swap** ce champ doit être spécifié comme ``**none**" (aucun).

3. Le troisième champ (**fs_vfstype**), décrit le type du **système de fichiers**.
   Remarque : si **vfs_fstype** est mentionné comme ``**ignore**", l'enregistrement n'est pas traité.
   Ceci permet de visualiser aisément les partitions non utilisées.

4. Le quatrième champ (**fs_mntops**), indique des options de montage associées au **système de fichiers**.
   Il s'agit d'une liste d'options séparées par des virgules. Ce champ contient au moins le type de montage, suivi éventuellement d'options appropriées au type de **système de fichiers**.

5. Le cinquième champ contient un indicateur de **sauvegarde** utilisé par la commande **`dump`**.
   Ce champ est mis à 1 pour les **systèmes de fichiers `ext2`** et **`ext3`**.

6. Le sixième champ contient un indicateur de **test** qui indique à la commande **`fsck`** de tester le **système de fichiers** avant qu'il soit monté. Ce champ est mis à 1 pour le **systèmes de fichiers racine** qui doit être vérifié en premier, à 2 pour les **systèmes de fichiers** qui doivent être vérifiés ensuite et

Ce fichier peut être consulté par tout utilisateur, mais seul **root** est autorisé à le modifier.

La commande **`mount`** vue précédemment, utilise le fichier **`/etc/fstab`** dans trois buts :

- La commande **`mount`** **-a** [**-t** *type*] exécutée dans un script de démarrage du système, monte tous les **système de fichiers** indiqués dans **`/etc/fstab`** (ou uniquement ceux du type indiqué), à l'exception de ceux ayant l'option **noauto**.

- Normalement, seul **root** peut monter des **système de fichiers**. Néanmoins, si la ligne de **`/etc/fstab`** contient l'option **user**, n'importe quel utilisateur peut monter le **système de fichiers** correspondant.
  Cette option est principalement utilisée pour les **systèmes de fichiers** des média amovibles
  du type CD-ROM, disquettes, etc ...

- Lorsque l'on monte un **système de fichiers** mentionné dans **`/etc/fstab`**, il suffit d'indiquer en paramètre de la commande **`mount`** le répertoire de montage, ou le **fichier spécial**.

5.4. Utilitaires de gestion de disquettes MSDOS

Les distributions **Linux** fournissent quelques commandes de base pour gérer des disquettes au format MSDOS :

- **mdir**
- **mcd**
- **mdel**
- **mdeltree**
- **mcat**
- **mtype**
- **mcopy**
- **mformat**

Consulter la documentation en ligne pour connaître les fonctionnalités de ces commandes.

Ces commandes ne sont utilisables que si l'administrateur du système (root) à attribué les droits **rw** pour **others** aux fichiers spéciaux qui gèrent les unités de disquettes (/dev/fd0, /dev/fd1, ...).