TYPES DE FICHIERS

L'utilisateur a une vision *uniforme* des entrées sorties que celles-ci s'effectuent sur des fichiers, des *périphériques* ou des IPC *(outils de communication interprocessus).*
La notion de fichiers est donc très générale puisque qu'un fichier est la représentation abstraite d'un périphérique (par exemple le disque */dev!dkO)* ou bien d'une unité logique de stockage (par exemple */home/prof/livre*).

La sémantique des appels systèmes d'accès aux fichiers et leurs protections est en principe identique quelque soit leur type.

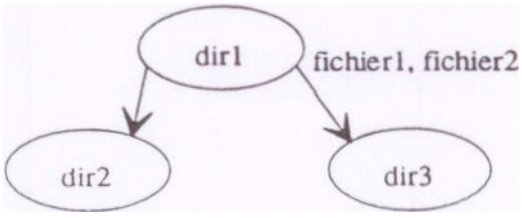
Un système de fichiers est *arborescent* et défini *récursivement.* On distingue quatre types de fichiers :

I. Fichiers ordinaires

Un *fichier ordinaire* (ou encore régulier, banalisé) est constitué d'une suite finie d'octets *sans organisation particulière.* Il ne faut pas confondre les fichiers *ordinaires* et les fichiers *standards.*

II. Fichiers répertoires

Un *fichier répertoire* assure la correspondance entre un nom logique de fichiers et sa situation géographique sur le disque. Un répertoire est constitué de fichiers de n'importe quel type.



L'utilisateur perçoit le système de fichiers virtuel d'UNIX sous la forme d'un *arbre* où les *noeuds* sont des *répertoires hiérarchisés* et les *feuilles* des fichiers *normaux* ou *ordinaires.*
Le noeud le plus élevé est appelé *racine* (**root**). C'est le répertoire à partir duquel l'utilisateur va désigner un fichier par son chemin d'accès *(pathname)* représenté par un *arc orienté.*

Un *répertoire* se distingue des fichiers normaux par son type et la structure de ses blocs structurés en entrées de 16 caractères chacun (14 caractères pour le nom. 2 caractères pour le numéro de l'inode dans les anciennes versions SYSTEM V). valable jusqu'à 255 caractères dans les versions 4.2 BSD et SYSTEM V R4), correspondant chacune à un fichier. ou à un sous répertoire.

Dans la version UNIX 4 3 BSD. un répertoire est alloué par unité appelée *chunk* dont la taille est déterminée de telle sorte que chaque allocation peut être transférée sur disque en une seule opération. Un répertoire est un ensemble de blocs contenant des pointeurs sur des structures de répertoire de longueur variable composées des champs : taille de l'entrée, *longueur de son identificateur, pointeur sur l'inode* correspondant.

III. Lien matériel et lien symbolique

III.1. Lien matériel

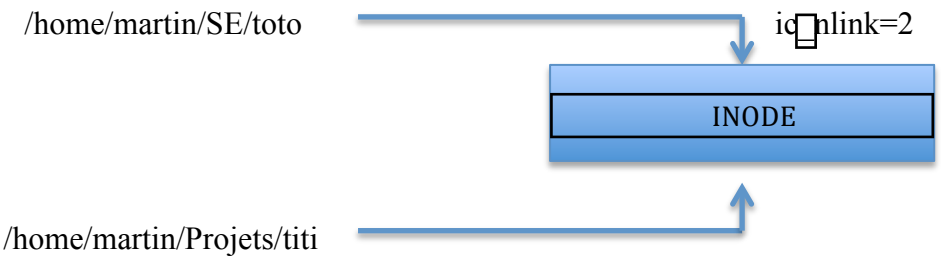
Un *lien matériel* représente le nombre de chemins d'accès à un fichier donné dans un même système de fichiers. Le fichier est supprimé si ce nombre devient nul.

Le lien matériel est réalisé entre deux fichiers associés à un même inode. C'est un élément d'un répertoire qui fait référence à un fichier.
Un même fichier (avec ses caractéristiques propres : taille. protections. etc) peut comporter plusieurs liens. On ne peut établir un lien matériel sur deux fichiers situées sur des partitions différentes. La seule façon de reconnaître un lien matériel entre deux fichiers est de visualiser et constater qu'ils sont identiques.
Quand deux fichiers portent sur la même inode, le fait d'en modifier on modifie automatiquement l'autre.

Une inode est accessible par une entrée d'un répertoire qui crée un *lien matériel* entre le fichier et l'inode le matérialisant. Il peut avoir plusieurs chemin d'accès à un inode *dans son système de fichier* donc plusieurs liens matériels sur une même inode.

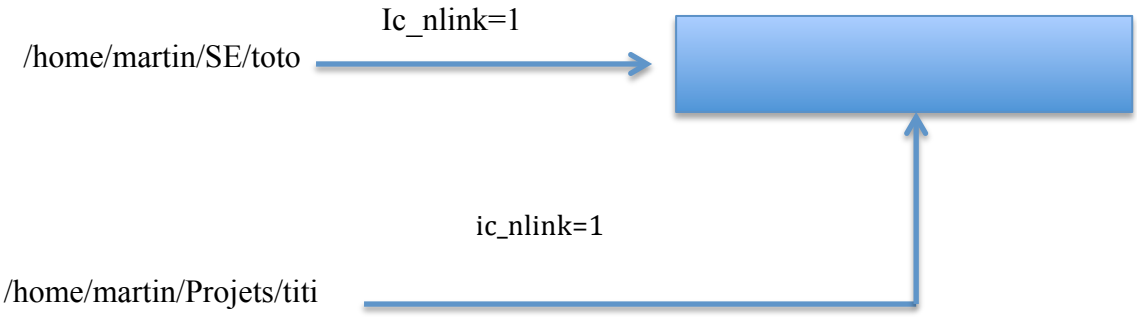
Le concept de lien matériel est fondamental : les inodes ne font pas partie d'un répertoire donné. ils existent séparément et sont accessibles par les liens matériels. Quand le nombre de liens d'une inode devient nul, l'inode est dés-allouée.

Un lien matériel permet de définir un deuxième chemin d'accès à un fichier sans en faire de copie effective. Il faut bien comprendre que le fichier est unique même si plusieurs chemins d'accès différents permettent d'y accéder.



III.2. Liens symboliques

Il existe un deuxième type de liens : le *lien symbolique*
Un *lien symbolique* définit un *chemin d'accès synonyme* du chemin d'accès du fichier (ordinaire ou répertoire). C'est une entrée d'un *répertoire quelconque* de l'arborescence définissant *une nouvelle inode* contenant un nouveau chemin d'accès (absolu ou relatif) d'une inode existante (banalisé ou répertoire) d'un *système de fichiers quelconque de l'arborescence.* On a la représentation suivante:



ic_nlink : est le nombre de liens

Contrairement à un lien matériel, le déplacement ou la suppression d'un fichier ne modifie pas un lien symbolique sur ce fichier qui ne pointera sur rien si le fichier n'est pas remplacé ou qui pointera sur le nouveau fichier dont le nom est identique au fichier antérieur.

Le lien symbolique sur un répertoire est très utile mais n'est pas symétrique. Un changement de répertoire relatif à un fichier accédé par un lien symbolique peut laisser l'utilisateur perplexe sur sa position dans l'arborescence.

Commandes de gestion des liens

. Liens matériels

ln fichier_Origine lien_destination

. Liens symboliques

ln -s fichier_Origine lien_destination

Le lien symbolique. entre deux fichiers. s'exécute sur des numéros d'inodes différents et peut associer des fichiers se trouvant sur des systèmes de fichiers différents.

IV. Les fichiers spéciaux

Un *fichier spécial* est référencé dans le répertoire **/dev** ou un de ses sous répertoires.
Il matérialise un *périphérique* et a un des modes de fonctionnement suivant:
-mode c : tous les pénphén4uc.s.
-mode bloc*:* ce sont les supports, de stockage (disque, bande, etc.)

La représentation par la commande : **ls –l** de ces types de fichiers est la suivante :

- pour les fichiers ordinaires
d pour les fichiers repertoires
b pour les fichiers spéciaux en *mode bloc.*
c pour les fichiers spéciaux en *mode caractères.*

V. Les fichiers standards :

Trois fichiers : "appelés fichiers standards", sont automatiquement ouverts au début de chaque session de travail, ce sont :

- le fichier standard d'entrée : le clavier, dont le pointeur est *stdin* et le *descripteur* 0,
- le *fichier standard de sortie* : l'écran, dont le pointeur est *stdout* et le descripteur 1.
- le *fichier standard d'affichage des diagnostics d'erreurs* : l'écran, dont le pointeur est *stderr* et le descripteur 2.

On travaille implicitement avec ces trois fichiers que l'on peut toujours rediriger quand on utilise l'interprète de commandes.

V.1 Redirection des entrées/sorties standards

La *redirection* d'un fichier consiste à modifier son descripteur de telle sorte que *l'inode cible* son modifiée.

L'interprète de commandes *shell* attend une commande tapée au clavier et affiche le résultat et les messages d'erreur sur la sortie standard.

Exemples :

La commande ls : affiche à l'écran le contenu du répertoire courant
wc lit le texte tapé au clavier jusqu'à Ctrl D et affiche ensuite à l'écran le nombre de lignes, de mots et de caractères tapés.

Il est possible de rediriger les trois fichiers standards.
Une redirection n'est valable que pour la commande courante.

Exemple :

```
ls > liste
ls > /dev/tty12
```

Le contenu du répertoire courant n'est plus affiché à l'écran mais stocké dans le fichier liste.
Le contenu du répertoire courant est affiché sur le terminal associé au périphérique tty12.

Remarque :
La redirection de la sortie permet de créer un fichier vide, il suffit, en shell de Bourne ou en KSH shell d'exécuter la commande :
> fichier_vide
Cette possibilité n'est pas acceptée par le *C-shell* Une autre solution est de lancer la commande (qui fonctionne aussi en shell Bourne et KSH):

```
cp /dev/null fichier_vide
```

Ordre d'exécution des opérateurs

Le fichier de redirection est détruit puis recrée avant l'exécution de la commande.
Exemple :

ls –l > liste
En début d'exécutons. le fichier cible a une taille nulle car il est détruit s'il existe.
Cette destruction préalable du fichier de redirection peut être lourde de conséquences.

Soit la commande :
cat f1 f2 > f1

Pour concaténer les fichiers f1 et f2 en appelant f1 le résultat. Le fichier f1 d'origine est détruit et le fichier f1 recrée est l'identique à f2. Il est cependant possible d'éviter cette destruction initiale en utilisant le symbole >> au lieu de > selon le principe suivant: si le fichier existe. L'information est ajouté en fin de fichier: il est crée sinon.

Exemple :
ls > liste
date >> liste