SYSTEMES DE FICHIERS

INTRODUCTION

Le concept de fichier, sous UNIX, présente des caractéristiques qui expliquent pour partie la puissance du système. Il est caractérisé par sa simplicité et son universalité.
Par ce principe, UNIX propose une vision uniforme des objets mis à la disposition des utilisateurs. Un fichier peut être un document ou le programme d'un utilisateur, le noyau d'UNIX lui-même, les utilitaires ou encore des fichiers représentant les périphériques. Quel que soit son type, un fichier est géré avec les mêmes primitives et selon les mêmes principes par le système.

Le système de fichiers d'UNIX reste un système de référence qui a inspiré les concepteurs des autres systèmes d'exploitation.

PRESENTATION DU SUPPORT

Ce support de cours se veut générique dans la mesure où il ne s'appuie pas sur une version d'UNIX particulière. Quelques exemples reposant sur les versions existantes à l'E.S.A.T. seront néanmoins donnés.

Dans ce support, les sujets suivants seront traités :

- Organisation d'un disque
- Structure d'un système de fichier
- Organisation du système d'exploitation
- Les différents types de fichiers
- Attributs des fichiers

SOMMAIRE

DEFINITION

Un système de fichiers est une entité regroupant un ensemble de fichiers sur un ***disque logique***. Il contient non seulement les données des fichiers mais également un certain nombre d'informations qui permettent au système d'assurer la gestion du système de fichiers.

I- ORGANISATION D'UN DISQUE

I-1 Disques Physiques / Disques Logiques / Volumes

Un espace physique d'un *disque* peut être divisé en ***disques logiques*** pour plusieurs raisons :

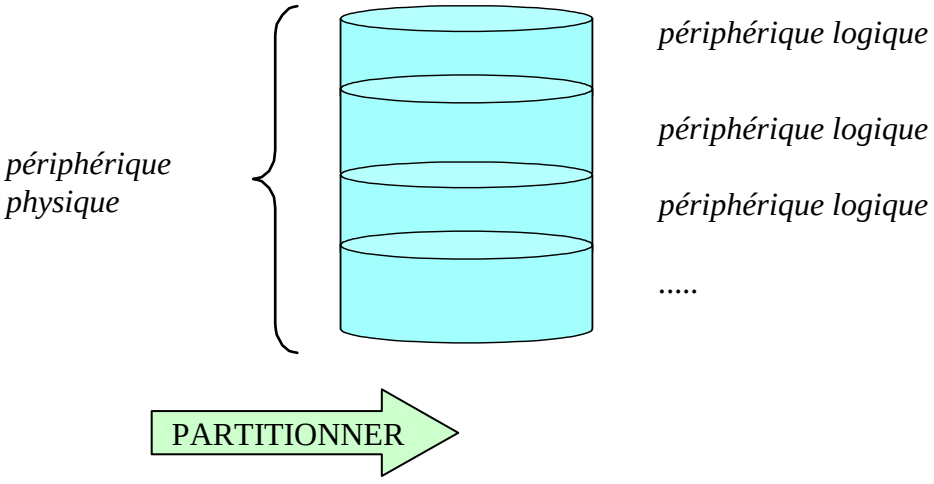
- Installer plusieurs systèmes d'exploitation sur le même disque dur
- Diviser l'espace physique d'un même système d'exploitation en plusieurs zones (système, données) pour des raisons importantes de sécurité ou plus fines d'optimisation d'accès

L'opération qui consiste à découper ce disque dur s'intitule le **partitionnement**.

Ces disques logiques sont aussi appelés ***périphériques logiques*** si l'on désigne le disque comme étant un *périphérique physique*.
En pratique, il est coutume d'appeler le résultat de cette opération une ***partition***.

Une partition = partie du disque sans structure

Cette opération est réalisée avec des outils d'administration : commande ***fdisk*** commune aux systèmes MS-DOS et UNIX, mais aussi avec des utilitaires spécifiques comme Partition Magic.



Il est possible de mettre en place une structure de système de fichiers (ou SDF = acronyme couramment utilisé en cours) sur chaque disque logique.
Cette opération est possible avec la commande ***mkfs*** (commande format sous MS-DOS).

Système de Fichiers = structure apte à supporter des fichiers

L'un des disques logiques a cependant un rôle privilégié dans la mesure où il contient le noyau et les fichiers système (le système d'exploitation).
Ce disque logique "système", créé lors de l'installation du système d'exploitation, doit être toujours actif pour être toujours accessible : on dit qu'il est *"monté"*.

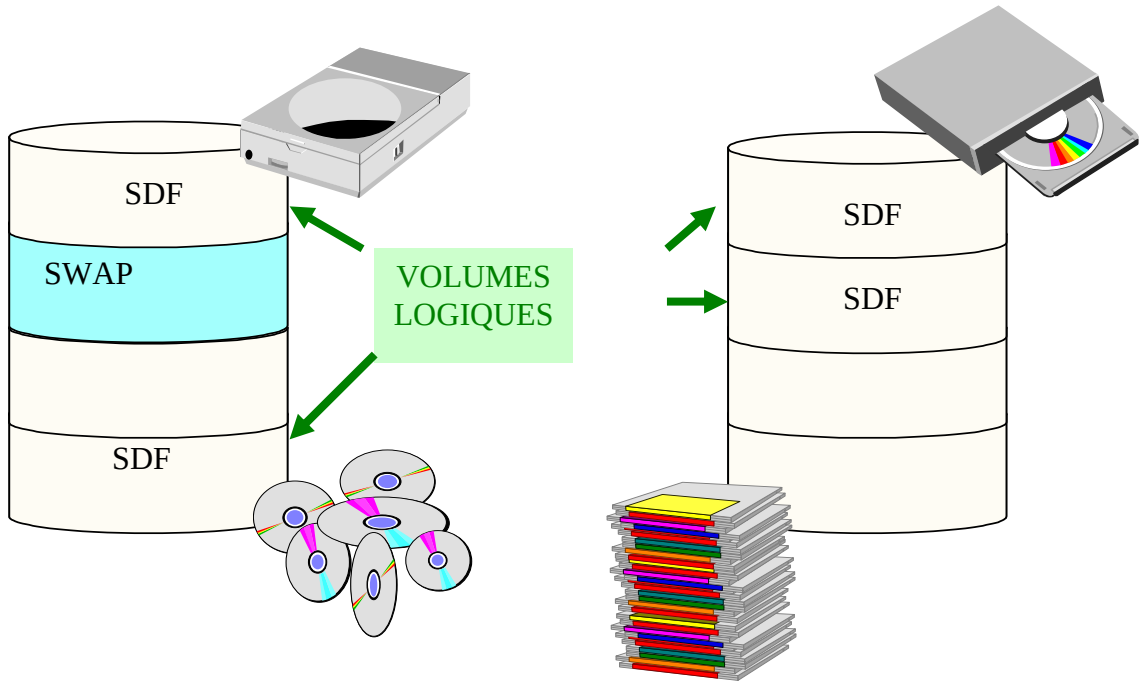
Remarques :
- Tout système d'exploitation doit se trouver sur une partition primaire (ou principale)
- Les partitions non système peuvent se trouver installées sur des partitions étendues

Par ailleurs, tout support magnétique adressable (disquette, Cd-Rom...) peut servir de support à un système de fichiers *supplémentaire*.

Remarque : la taille maximum théorique d'une partition primaire dépend du système d'exploitation : MS-DOS = 2Go, Windows 95 (4Go à 16.000Go), UNIX (16.000Go).

La notion de système de fichiers est élargie en introduisant celle de *volume* (notion que l'on retrouve sous d'autres systèmes d'exploitation comme GCOS de Bull) :
- un *volume* est un disque logique sur lequel un système de fichiers a été créé.

Volume Logique = partition + S.D.F.



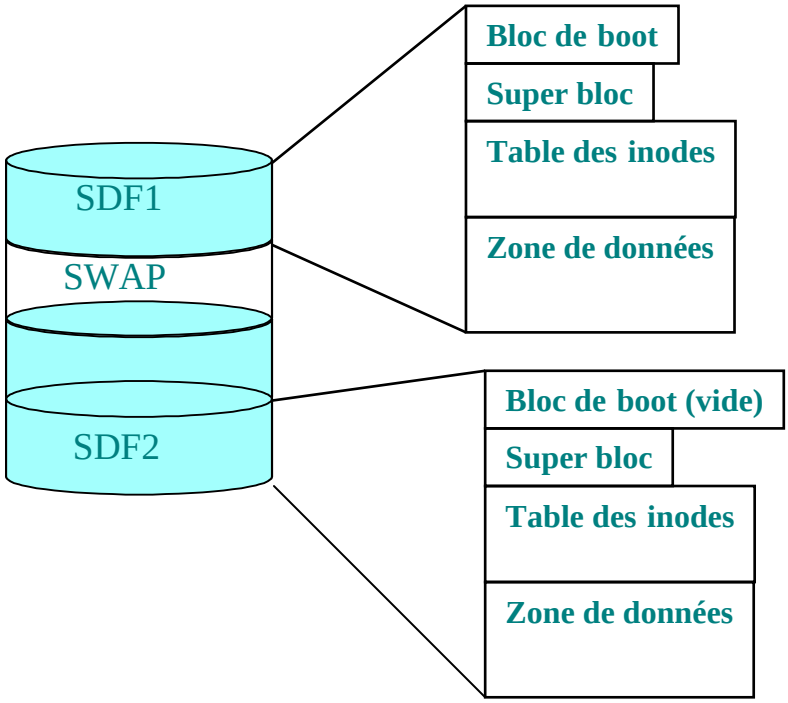
I-2 La Zone de SWAP

La zone de SWAP est une zone de mémoire virtuelle sur disque dur ayant pour rôle celui d'étendre le rôle de la RAM.
Cette zone, sous UNIX, occupe généralement l'espace d'une partition spéciale, créée lors de l'installation du système d'exploitation.
Elle ne contient pas de système de fichiers et est donc inaccessible à l'utilisateur.

Remarques :
✓ La zone de SWAP peut être un fichier (rôle similaire à celui du fichier d'échanges du monde Windows).
✓ Si celui ci n'est pas créé par l'administrateur, le système le créera automatiquement.
✓ Il existe une commande réservée à l'administrateur permettant de visualiser l'occupation de cette zone : par exemple, « swapon –s » sous LINUX.

I-3 Implantation d'Ensemble sur le disque

Un disque peut être découpé en partitions selon le schéma suivant :



II- STRUCTURE D'UN SYSTEME DE FICHIERS

Un système de fichiers est composé de blocs contenant les données des fichiers et de blocs contenant les données techniques.

Un bloc logique peut correspondre à un ou plusieurs secteurs (en général 2, 4 ou 8 blocs physiques). Sa taille est un multiple de 512 octets.

Les blocs d'un même fichier ne sont pas forcement contigus : c'est le système qui mémorise leurs emplacements. De cette manière, les blocs sont alloués dynamiquement au fur et à mesure de l'évolution de la taille du fichier.

Remarques :

✓ La taille d'un système de fichiers est figée à sa création.

✓ La taille des blocs logiques est paramétrable. Sur la version System V Release 4, ces blocs ont une taille de 4 Ko.

Bloc 0	BOOT BLOC	Contient le programme de démarrage de la machine
Bloc 1	SUPER BLOC	Contient des tables de contrôle
Bloc 2 à n	TABLE DES INODES	ou I-liste (un inode => un fichier)
Blocs n+1 à …	ZONE DE DONNEES	Ensemble de blocs de données et de liens entre les blocs ; espace disque où sont stockés les différents fichiers.

II-1 Le Bloc de Démarrage (Boot Bloc)

Ce bloc, "bloc 0" sur le disque, contient le(s) ***programme***(s) *(boot strap)* ***assurant le démarrage*** (ou l'amorçage) et l'initialisation du système d'exploitation.
Tous les volumes logiques possèdent ce bloc de démarrage.
Seul celui du volume "système" contient les informations de démarrage (*boot).* Sur les autres volumes, le bloc est vide.

Sa taille est un multiple de 512 octets : 512 ou 1024 (Linux) ou...

II-2 Le Super-Bloc

Ce bloc ***décrit le système de fichiers***. Il contient des tables, générées lors de la création du système de fichiers, qui permettent de contrôler celui-ci.

Voici quelques éléments de description :

- Nom du Volume Logique
- Nom du Système de Fichiers
- Type du Système de Fichiers
- Etat du Système de Fichiers
- Taille du système de fichiers
- Nombre de blocs libres
- Pointeur sur le début de la liste des blocs libres
- Taille de la liste des inodes
- Nombre et la liste des inodes libres.

Sa taille est un multiple de 512 octets : 512 ou 1024 (Linux) ou...

II-3 La Table des Inodes

Lors de sa création, un fichier est ***référencé de manière unique*** dans la ***table des inodes*** (rôle similaire de la FAT sous MS-DOS).

Les entrées de la table sont appelées ***inodes*** (index-node ou noeud d'index) et sont accessibles par l'index en question qu'on appelle ***numéro d'inode***.

Cette inode est en réalité une structure (langage C) contenant les informations nécessaires à la gestion du fichier référencé. En particulier, cette structure contient, directement ou non, les adresses des blocs de données formant le fichier.

Le numéro d'inode est unique au sein d'un système de fichiers.
C'est par lui que le fichier sera géré par le système (et non par son nom).

L'inode décrivant le fichier comporte :
- Numéro d'inode,
- Type de fichier et permissions d'accès,
- Numéro d'identification du propriétaire,
- Numéro d'identification du groupe propriétaire
- Nombre de liens sur ce fichier,
- Taille du fichier en octets,
- Date du dernier accès au fichier,
- Date de la dernière modification du fichier,
- Date de la dernière modification de l'inode (= création),
- Tableau de plusieurs pointeurs (table de blocs) sur les blocs logiques contenant les morceaux du fichier

Remarques :

✓ La taille de la table des inodes détermine le nombre maximum de fichiers que peut contenir un système de fichiers.

✓ Il n'y a pas dans un inode de renseignements concernant le nom du fichier.

✓ Le numéro d'inode est codé sur 4 octets. Le nombre maximal de fichiers au sein du système de fichiers est donc de 2^{32}-2**,** soit 4.294.967.294 fichiers.

✓ La taille d'une inode est un multiple de 64 octets (128 sous LINUX). En fait, elle dépend de la taille des adresses utilisées par le système.

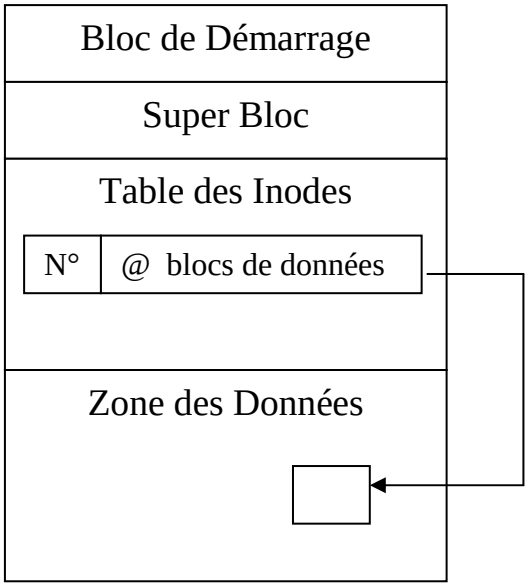
Cohérence du système de fichiers

UNIX, à l'image des autres systèmes d'exploitation, utilise une technique de zones tampons en mémoire centrale afin de minimiser le nombre d'accès disque.

Une image du super-bloc et de la table des inodes sont chargées en mémoire (RAM) lors du chargement du système d'exploitation afin que toutes les opérations d'E/S se fassent à travers ces tables du système.
Lorsque l'utilisateur crée ou modifie des fichiers, c'est en premier lieu cette image mémoire qui est actualisée. Le système d'exploitation doit donc à un moment donné actualiser le super-bloc du disque logique.
La mise à jour des inodes des fichiers sur disque (dans la table des inodes restée sur disque) ne s'effectue que lorsque le fichier est fermé et que l'entrée dans la table des inodes actifs a été libérée.
Cette mise à jour se réalise donc périodiquement et en dernier lieu au moment où l'utilisateur arrête le système d'exploitation.
Si cette dernière mise à jour ne peut être réalisée, le système de fichiers perd sa cohérence et cela peut provoquer des pertes de données. C'est pour cette raison qu'il ne faut **jamais arrêter brutalement un système d'exploitation Unix**.
Il est donc important d'arrêter une station UNIX selon des procédures précises.

II-4 La Zone de Données

Il s'agit de l'espace disque résiduel dans lequel sont stockés les différents fichiers (blocs de données et liens de chaînage entre les blocs).



III- ORGANISATION DU SYSTEME D'EXPLOITATION

III-1 Organisation du Système de Fichiers

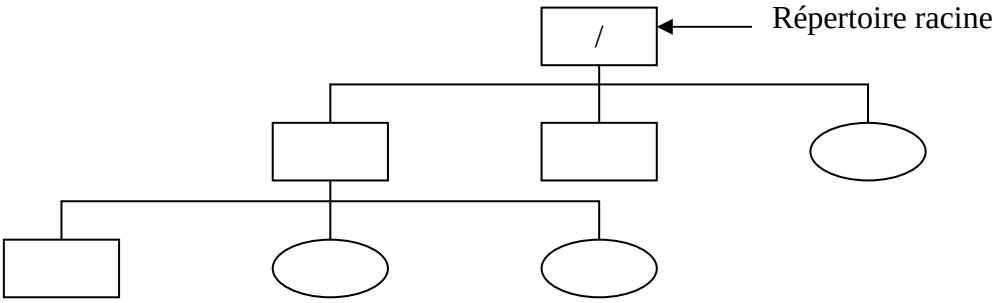
Un système de fichiers est vu par 1'utilisateur comme une structure hiérarchique et arborescente de répertoires (arbre inversé) qui permet d'accéder à n'importe quel fichier du disque.

Les nœuds de cette structure arborescente sont les fichiers répertoires (cf. VI-2) et les feuilles, des fichiers. Les fichiers ainsi désignés pouvant être eux même des répertoires (ou fichiers répertoires).

L'arbre est construit à partir d'un répertoire racine unique (ou root).

Dans le cas particulier du système de fichiers contenant le système d'exploitation, ce répertoire racine appelé *" root "* est représenté par le caractère " / ".

Tous les autres répertoires sont reliés plus ou moins directement à ce répertoire racine. Ainsi, tous les fichiers appartiennent à la même arborescence.



III-2 Les Systèmes de Fichiers Supplémentaires

Une station UNIX peut posséder plusieurs disques durs, un lecteur de disquette et un lecteur de cédérom. Chacun de ces supports peut contenir un ou plusieurs systèmes de fichiers.

Un système de fichiers, créé sur un périphérique logique représentant une unité de stockage, est caractérisé, entre autres, par le nom du périphérique en question, ainsi que par sa taille en nombre de blocs et en nombre d'inodes.

III-3 Montage d'un Système de Fichiers

Un disque dur de la station contient le système de fichiers du système d'exploitation proprement dit ou système de fichiers racine (*root file system).* Les références de ce système de fichiers sont toujours chargées en mémoire centrale au démarrage, et sont donc toujours accessibles.

Pour accéder à un autre système de fichiers, il faut effectuer ce qu'on appelle le *"montage"* (ou **attachement**) du système souhaité. Il s'agit de rattacher les références (Super bloc, Table des Inodes) de ce nouveau système de fichiers.

Le montage d'un système de fichiers consiste à connecter le *répertoire racine* de ce système à un répertoire du système de fichiers accessible . Cette opération est réalisée grâce à la commande ***mount***. L'action qui vise à déconnecter un système de fichiers pour le rendre inaccessible (ou démontage ou détachement) est réalisée par la commande ***umount***.

Remarques :

✓ L'ensemble des fichiers et répertoires du système de fichiers monté est à considérer, par l'utilisateur, comme faisant partie du système de fichiers racine : il n'y a plus, en fait, qu'une seule structure hiérarchique de fichiers.

Le système d'exploitation DOS (Windows) est différent car chaque volume logique (correspondant à une unité de stockage) est affectée d'une lettre (A: , C:, ...) et garde sa propre structure hiérarchique.

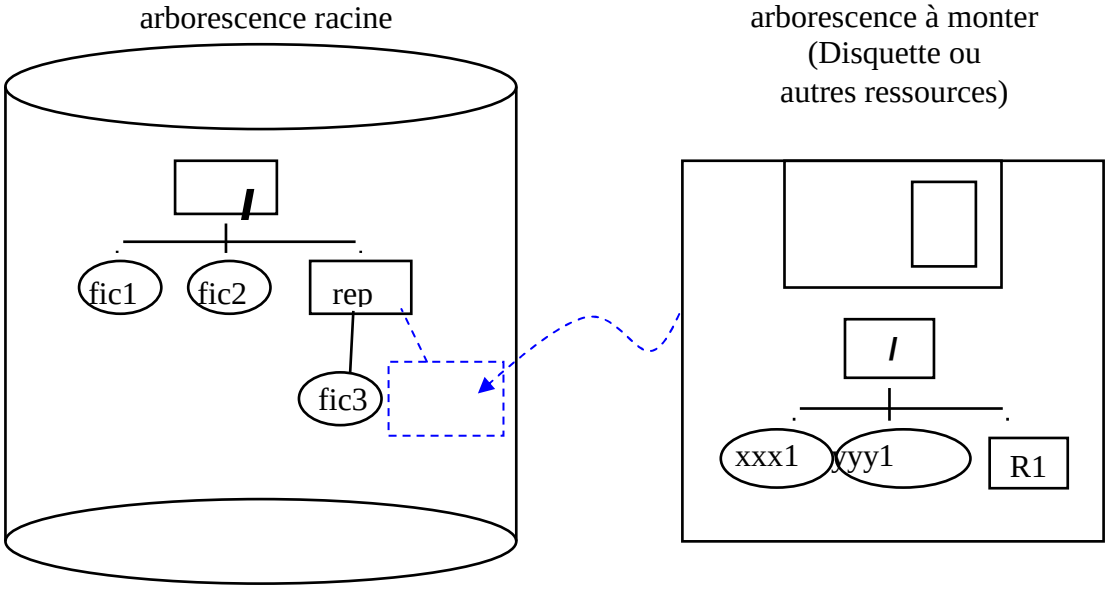
✔ Les opérations de montage prédéfinies (au minimum le système de fichiers "racine" contenant le système d'exploitation) s'effectuent en général au démarrage de la machine. Ces opérations sont décrites dans des fichiers de configuration que seul l'administrateur peut paramétrer.

La syntaxe de *mount* est la suivante :

mount [-options] [source] [destination]

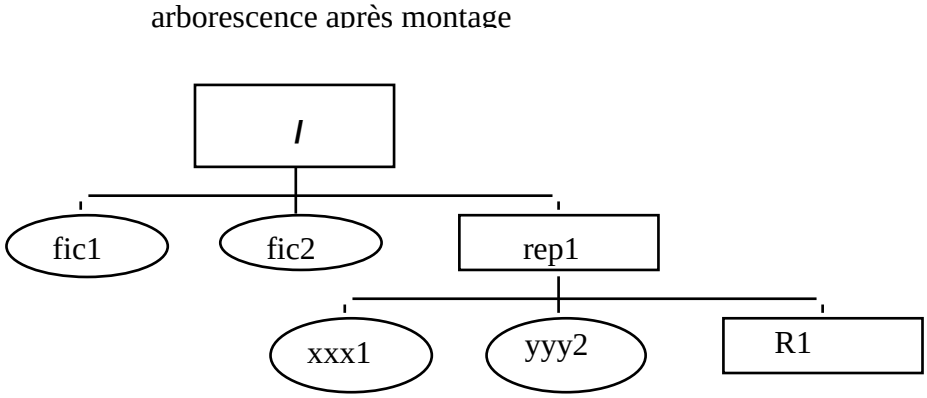
où

source est le nom du périphérique logique représentant le SDF à monter
destination désigne le répertoire où est monté le SDF (en général le répertoire */mnt*)



Remarque :

✔ Lorsqu'un système de fichiers est monté sur un répertoire qui contient déjà des fichiers, ces derniers deviennent inaccessibles au système tant que le montage est actif. Ces fichiers ne sont pourtant pas "écrasés". Dès que le système de fichiers "hôte" est démonté, ces fichiers sont à nouveau accessibles.



le fichier fic3 est occulté après montage

Remarque :
✓ En général le système de fichiers racine comporte un répertoire dédié nommé /mnt (répertoire de montage temporaire) sur lequel des systèmes de fichiers supplémentaires (disquettes, Cd Rom) et distribués (NFS) pourront être montés.

Exemples :

mount -r /dev/fd0 /mnt/floppy
montage d'une disquette (volume logique /dev/fd0) en lecture (-r) sur...

mount /dev/cdrom /mnt/cdrom
montage d'un cédérom (volume logique /dev/cdrom) sur le répertoire ...

Remarques :
✓ Tout programme en cours d'exécution sur un système de fichiers empêche tout montage sur celui-ci.
✓ En conséquence des remarques précédentes, le système interdit le montage d'un système de fichiers supplémentaire au niveau de la racine du système d'exploitation « / ».

✓ Deux SDF différents ne peuvent être attachés sous le même répertoire
→ il faut démonter le volume logique précédent avant de monter le volume suivant.

✓ **On ne fait que charger en mémoire l'arborescence du volume logique, et non le contenu de ses répertoires.**

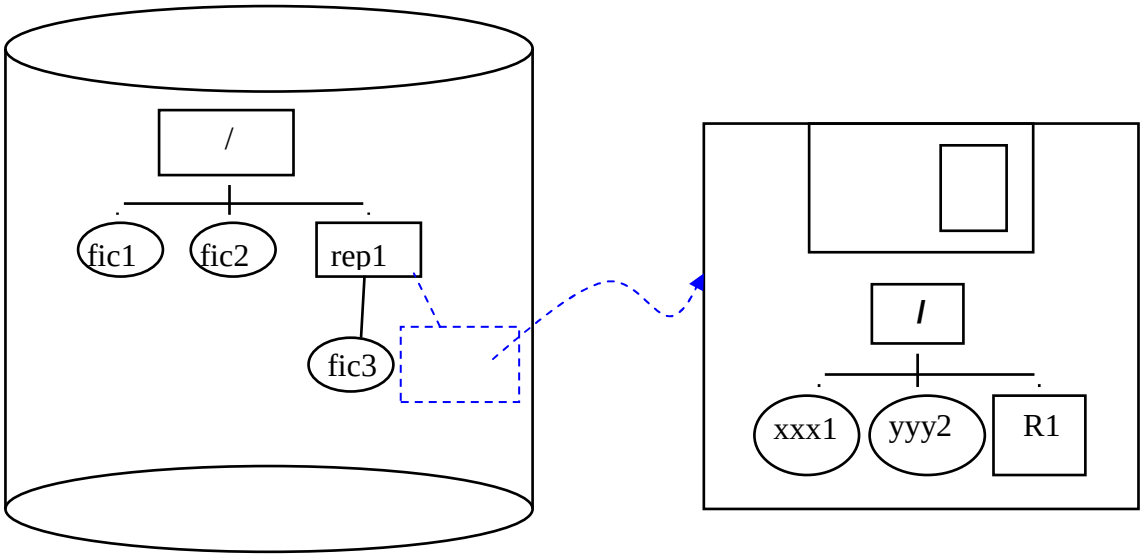
Les syntaxes de u*mount* sont les suivantes :
umount [source] *ou* **umount [destination]**

✓ source est le nom du fichier spécial représentant le SDF à monter
✓ destination est le répertoire où est monté le SDF (en général le répertoire /*mnt*)

Exemple :
umount /dev/fd0
ou
umount /mnt/floppy

Remarque :
✓ Pour le démontage, ne pas rester en dessous du point de montage, sinon affichage d'un message d'erreur rendant compte de l'impossibilité de démonter :
Sous LINUX : « device is busy ».

arborescence après démontage



💣 Pour assurer la cohérence du système de fichiers d'une disquette**, il ne faut pas retirer cette disquette du lecteur avant d'avoir effectué la commande umount** (afin d'assurer la mise à jour du super-bloc et de la table des inodes de la disquette). En cas d'oubli, il faudra remettre la même disquette et non une autre, afin de réaliser cette mise à jour.

🗎 Accessibilité des utilitaires mount et umount par un utilisateur

Les commandes mount et umount ne sont accessibles, en exécution, que par l'**administrateur** du système, ceci pour des raisons de sécurité. Il n'est donc pas possible, en théorie, à un utilisateur de stocker ses programmes et ses données sur une disquette puisqu'il ne pourra pas "monter" celle-ci.

Pour palier cette limitation, l'administrateur dispose d'options et de fichiers de configuration qui rendent possible un montage pour un utilisateur :

- Sous AIX, le fichier de configuration s'appelle /etc/filesystems.
- Sous LINUX, il s'appelle /etc/fstab.
- Sous SCO, il s'appelle /etc/default/filesys.

Remarques :

✓ La commande **mount** sans argument provoque l'affichage des volumes logiques actuellement montés :

Node	Mounted	Mounted over	Vfo	date	options
Machine	Partition montée	Point de montage	Type de SdF	…	…

✓ Il est possible de monter des systèmes de fichiers de type différents. Il faut utiliser les commandes ci-dessus avec une option variable selon le type d'UNIX : -n (AIX), -t (LINUX), -F (SCO).

✓ La commande **df** (disk free) permet aussi de visualiser les volumes logiques montés et de mesurer la quantité d'espace libre par SDF :

File	1K-blocks	Used	Available	Use%	Mounted On
/dev/hda1	4096	3904	24	2%	/
/dev/hda6	20480	11388	1021	16%	/home

III-4 Les Systèmes de Fichiers Distribués

Les réseaux prennent de plus en plus de place dans l'organisation des systèmes d'information.

Chaque station a besoin d'accéder à certaines sources d'informations situées sur d'autres stations ou serveurs, et de rendre ces sources ou ressources disponibles sur cette station.

Afin d'optimiser l'ensemble, il est souhaitable de mettre au point des systèmes qui évitent de posséder plusieurs exemplaires d'une même ressource.

Des logiciels supportant le concept de *fichiers partagés* ont été mis au point. Grâce à ce système, un fichier de données ou un programme ne doit plus exister qu'à un seul exemplaire au sein du réseau. De même, les unités de stockage ou de sauvegarde sont utilisables par tous les éléments du réseau.

Le partage des informations parmi les éléments d'un réseau se fait suivant le principe *clients/serveur :*

- Un client est une station qui sollicite une ressource (requête) via le réseau auprès d'une autre station (serveur).
- Un serveur est la station susceptible d'offrir cette ressource.

Dans le monde UNIX, plusieurs solutions ont été mises au point et certains de ces logiciels sont devenus des standards de fait. RFS, NFS, VFS (UNIX System 5.4) peuvent être cités.

Le système NFS

NFS *(Network File System),* développé par *Sun Micro Systems* (1980), est et reste le plus employé.

Il permet de partager les ressources au sein d'un réseau reposant sur l'emploi du protocole Internet.

Il peut inclure des systèmes de fichiers provenant de systèmes d'exploitation hétérogènes (Windows NT, GCOS,...). Grâce à lui, des machines de constructeurs différents, disposant de systèmes d'exploitation incompatibles, peuvent échanger des informations si NFS est disponible sur chacune d'entre elles.

Pour pouvoir reconnaître des systèmes de fichiers différents, NFS a conçu une organisation appelée " Système de fichiers virtuels " *(Virtual File Svstem)* qui sert d'interface entre une requête d'information et le système de fichiers local.

Le mécanisme de fonctionnement est le suivant :

✓ Le serveur du réseau détermine ce qu'il offre en ressources (répertoires de systèmes de fichiers) aux clients du réseau.

✓ Ceux-ci peuvent alors monter cette ressource grâce à une option de la commande *mount*.

✓ Pour exécuter cette opération, la connaissance de l'adresse IP du serveur ou de son nom de domaine équivalent [ex : server99] est nécessaire.

Exemple sous LINUX :

mount -t nfs 160.192.41.52:/Linux/Mandrake7.1 /mnt/nfs

caractère ":" entre l'adresse IP et le nom du répertoire, pas d'espace de part et d'autre.

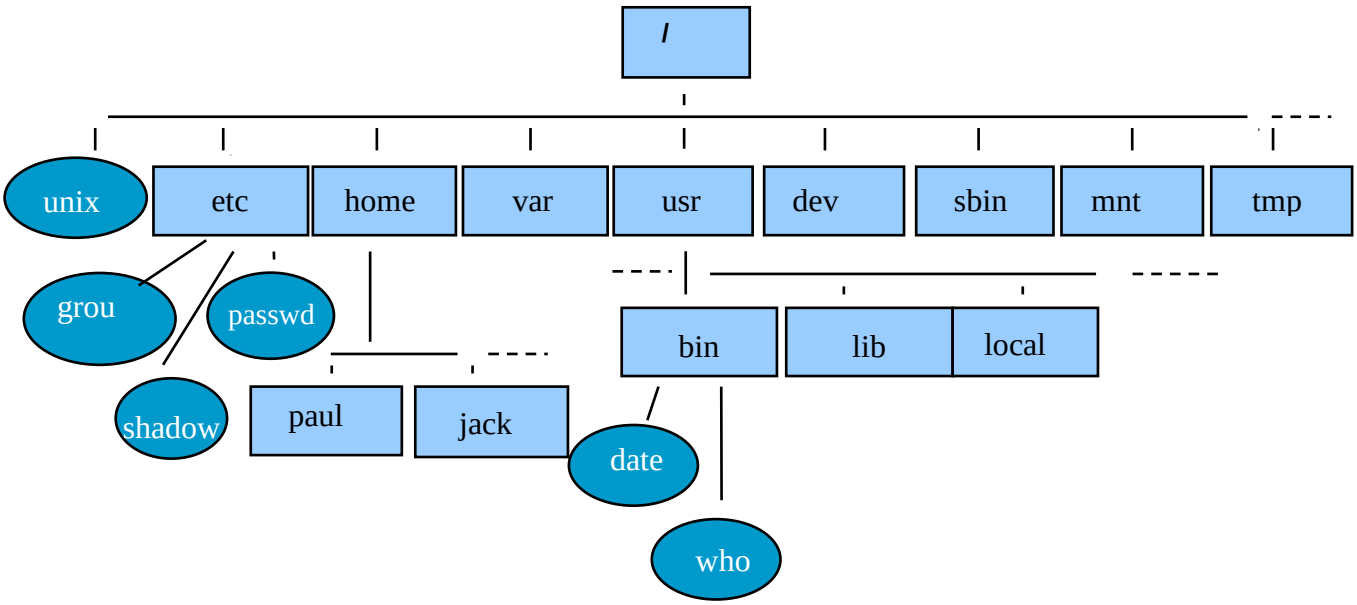
Cette commande permet à une station cliente de monter le répertoire distant /Linux/Mandrake7.1 se trouvant sur la station faisant office de serveur "nfs" d'adresse IP 160.192.41.52 sur sa propre arborescence (/mnt/nfs).
Il effectue pour cela un montage de type "nfs", représentant le type du système de fichiers monté.

Pour visualiser les ressources (ou structures) exportables d'un serveur "nfs" :

showmount -e @IP station

III-5 Arborescence Type d'un Système UNIX

Le standard



Chaque répertoire standard est affecté à un domaine bien précis. Cette standardisation permet de retrouver les informations organisées de la même manière d'un système à l'autre.

Dans le détail, cette arborescence varie légèrement selon le système utilisé.
Les répertoires sont utilisés comme suit :

/bin contient la plupart des utilitaires et commandes UNIX (les plus utiles)

/dev contient les fichiers spéciaux (pilotes de périphériques ou ***devices***)

/etc contient les fichiers de configuration (tables du système)

/home contient tous les comptes utilisateurs

/lib contient les bibliothèques (libraries) des langages de programmation

/mnt contient les montages temporaires (mount) de systèmes de fichiers additionnels

/tmp regroupe les fichiers temporaires.

/usr contient certains sous-répertoires utilisés par le système ou des utilitaires orientés " utilisateurs "

/usr/bin commandes de l'utilisateur

/usr/local applications locales et utilitaires liés au système d'exploitation

/sbin commandes dédiées au système et à l'administrateur

/var contient l'ensemble de fichiers de gestion non figés (files d'attente des imprimantes, boites aux lettres de messagerie, files d'attente des taches, bases de données, fichiers historiques du système,…)

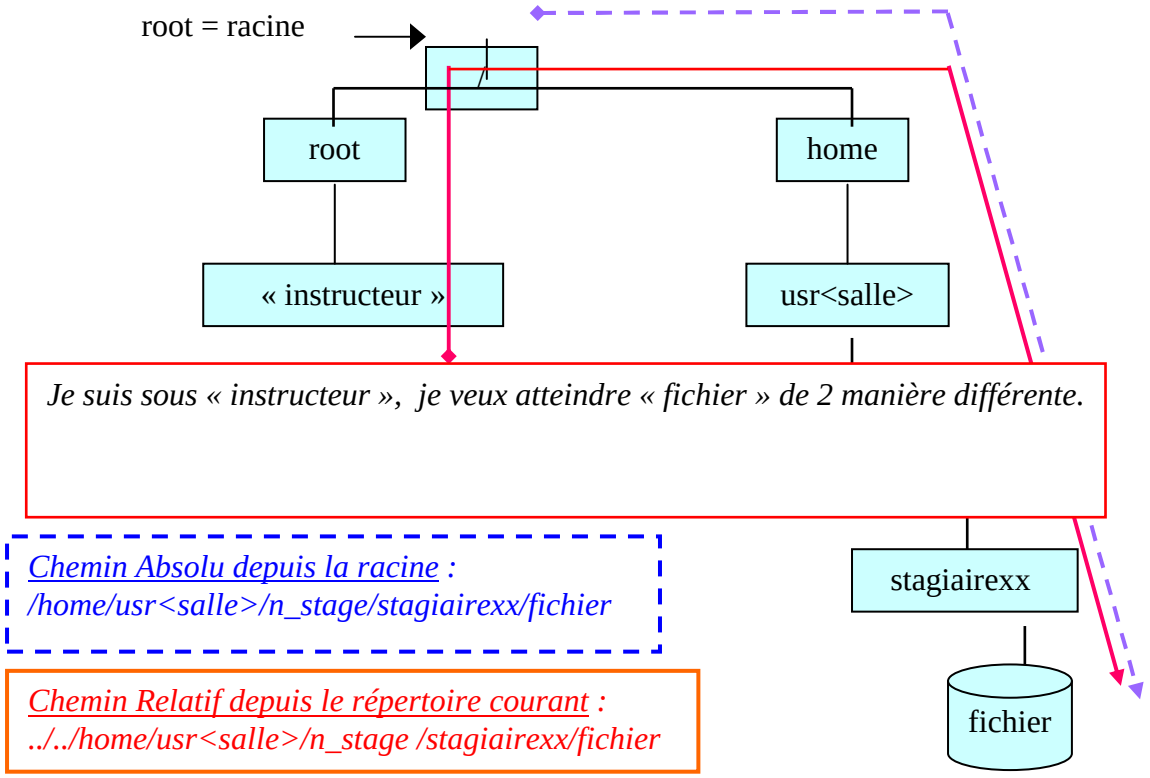
III-6 Chemin d'Accès à un Fichier *(pathname)*

Pour accéder à un fichier particulier, il faut utiliser, en plus de son nom, un chemin d'accès (*pathname en anglais).* Ce chemin d'accès contient les différents répertoires par lesquels le système doit passer pour accéder au fichier.

✓**Répertoire courant** : c'est le répertoire dans lequel se trouve un utilisateur à un moment donné (variable d'environnement PWD). Il est symbolisé par un point « . ».

✓**Chemin d'accès absolu** : c'est le chemin d'accès complet (*full pathname*). Il commence toujours par le répertoire racine, indiqué par un " / ". Ce caractère est suivi par la liste des noms des répertoires à parcourir, séparés par un " / " et se termine par le nom du fichier.

✓**Chemin d'accès relatif** : UNIX maintient la trace du répertoire courant de 1'utilisateur (cf. variable d'environnement **PWD**). Toutes les références aux fichiers situés "en aval" de ce répertoire courant peuvent donc se faire par rapport à ce répertoire dans la structure hiérarchique.



✓ A l'issue de la procédure de connexion *(login)*, 1'utilisateur se trouve dans un répertoire par défaut défini par le système (variable d'environnement **HOME**) : son répertoire d'accueil *(home directory).*

✓ La variable d'environnement **PATH** contient une chaîne de caractères composée des différents chemins d'accès aux commandes accessibles à un utilisateur donné.

IV- LES DIFFERENTS TYPES DE FICHIERS

Pour le système UNIX, tout fichier est une suite d'octets sans aucune hypothèse sur la structuration ou la nature des informations stockées. Toute organisation interne des données contenues dans un fichier est laissée au choix de l'utilisateur.

IV-1 Désignation des Fichiers

- Codé sur 255 caractères depuis la version UNIX "System V Release 4" (1989)
- Tout caractère de la table ASCII est utilisable
  - Pour un groupe de mots, il faut entourer l'expression choisie de guillemets.
- Majuscules et Minuscules sont différenciées par le système de gestion de fichiers

- Il n'y a pas de notion d'extensions, même si certaines extensions tendent à devenir des standards. Par conséquent, si une extension est donnée à un fichier, elle fait partie intégrante de son nom et doit être spécifiée lorsqu'on l'appelle.

Exemples de conventions :

.c	fichier source en langage C
.h	fichier d'entête C et Fortran
.o	fichier objet en langage C
.Z et .z	fichier de données compressées
.tar	fichier de données archivées par l'utilitaire "tar"
.gz et .tgz	fichier de données archivées par l'utilitaire "gzip" et "tar+gzip"
.html	page Web

Le caractère « . » n' a pas de signification particulière (comme sous DOS), toutefois les fichiers commençant par un point se trouvent **cachés** à certaines commandes du shell lancées avec leurs options par défaut :

- .profile fichiers de configuration d'environnement

  ➔ utiliser la commande « ls » avec l'option "-a"

Cependant la nature des taches à effectuer sur eux conduit le système UNIX à distinguer sept types de fichiers : ordinaires, répertoires, spéciaux bloc, spéciaux caractère, liens symbolique, tubes, sockets.

IV-2 Les Fichiers Ordinaires

Nous distinguerons plusieurs types de fichiers ordinaires suivant leur utilisation :

✓ **Les fichiers textes** (fichiers ASCII) ou **fichiers de données** et **fichiers multimédias**

✓ **Les fichiers de programmes (ou sources)** :

- fichiers source de langage à compiler en binaires exécutables (C, Cobol, Fortran,…).

✓ **Les fichiers exécutables** de deux types :

- soit les fichiers binaires résultant de la traduction par un compilateur approprié des fichiers sources décrits ci-dessus ;
- soit des programmes écrits en *langages interprétés* (Basic, Shell); les instructions pouvant être directement exécutables par la station.

✓ **Les fichiers multimédias, etc…**

Exemple : ls –l fic

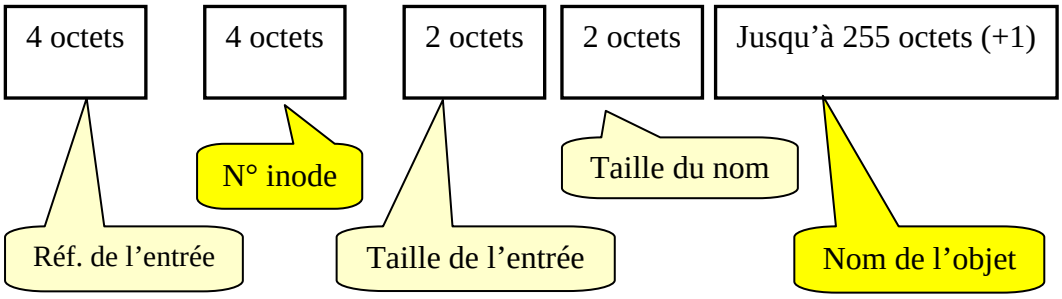
```
-rwxr-xr-x   1  prof   instruc    26    nov 31   15 :21   fic
```

le tiret caractérise un fichier ordinaire

IV-3 Les Fichiers Répertoires ou Répertoires

Un répertoire ou catalogue (directory en anglais) est une table assurant la correspondance **nom de fichier** logique avec le **numéro d'inode**.

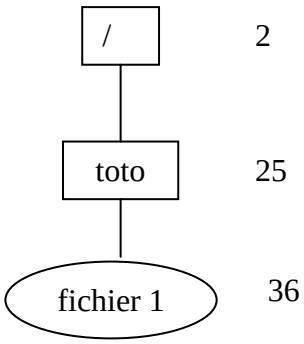
La taille d'une entrée est de longueur variable sur les types d'UNIX récents (System V Release 4, BSD 4.3, AIX) :



Tout répertoire possède à sa création deux entrées dont les noms sont « • » et « •• ». Ils désignent respectivement le répertoire lui-même et son ascendant ou répertoire père.

Exemple : Contenu du répertoire /toto

25	•	
2	••	
36	fichier 1	
42	essai. c	
…		



Remarque : le numéro d'inode du répertoire racine est toujours 2.

Exemple : ls –l rep

drwxr-xr-x 2 prof instruc 1024 nov 31 15 :25 rep

le "d" caractérise un fichier répertoire.

IV-4 Les Fichiers Spéciaux

Ce sont pour la plupart des fichiers associés aux pilotes de périphériques intégrés dans le noyau UNIX. En conséquence, la gestion d'un périphérique quelconque peut être réalisée à travers l'utilisation de ce fichier.

Ces fichiers ne contiennent pas de données.
Exceptée l'affichage du fichier (commande more), toutes les opérations effectuées sur un fichier normal peuvent être envisagées sur un fichier spécial.

Les fichiers spéciaux sont regroupés dans le répertoire */dev.*

Il existe deux types de fichiers spéciaux caractérisés par le mode d'accès au périphérique :

✓ Fichier spécial mode bloc :
les entrées sorties se font par bloc (utilisation de zone tampon)
exemple : périphériques de stockage (disque dur, bandes,…)

```
ls –l /dev/fd0
brw-rw-rw-   1   root     system      15,  0     nov 3          15 :26     fd0
```

le "b" caractérise un fichier spécial mode bloc

✓ Fichier spécial mode caractère :
les entrées sorties se font caractère par caractère
exemple : clavier, écran, …

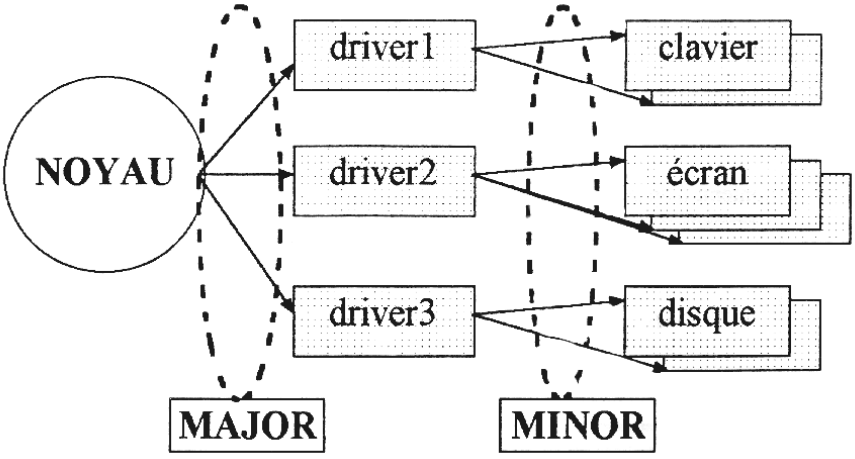
```
ls –l /dev/tty0
crw-------   1   root     system      16,  0     nov 3          15 :27     tty0
```

le "c" caractérise un fichier spécial caractère

Remarque : il existe des pseudo-périphériques
➔ */dev/mem* : la mémoire physique;
➔ */dev/null* : le "broyeur" pour rediriger les données vers le néant.
➔ */dev/zero* : un fichier qui ne contient que des "zero";

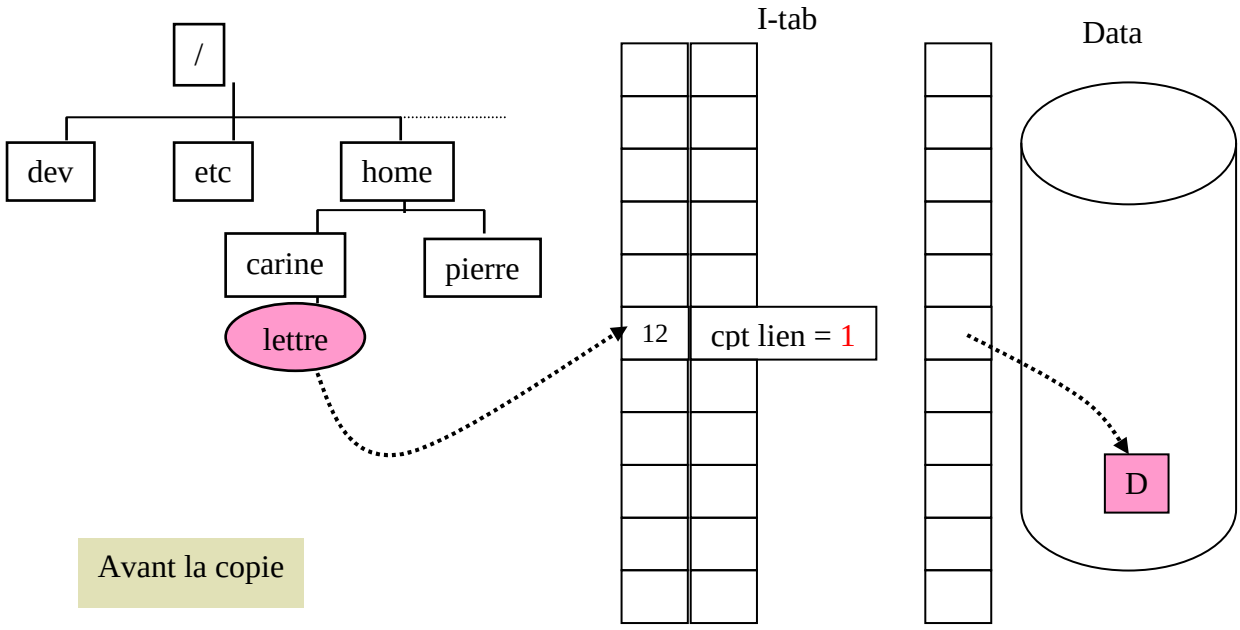
L'inode d'un fichier spécial ne contient pas de référence à des blocs de données. Ce sont les majeur et mineur qui sont mémorisés à cet emplacement.

✓ le majeur (major number) : ce nombre indique le classe de périphérique à utiliser pour communiquer avec un pilote de périphérique donné (puis son contrôleur)
✓ le mineur (minor number) : ce nombre est utilisé pour différencier les différents périphériques d'une même classe utilisant le même pilote (même "major number").



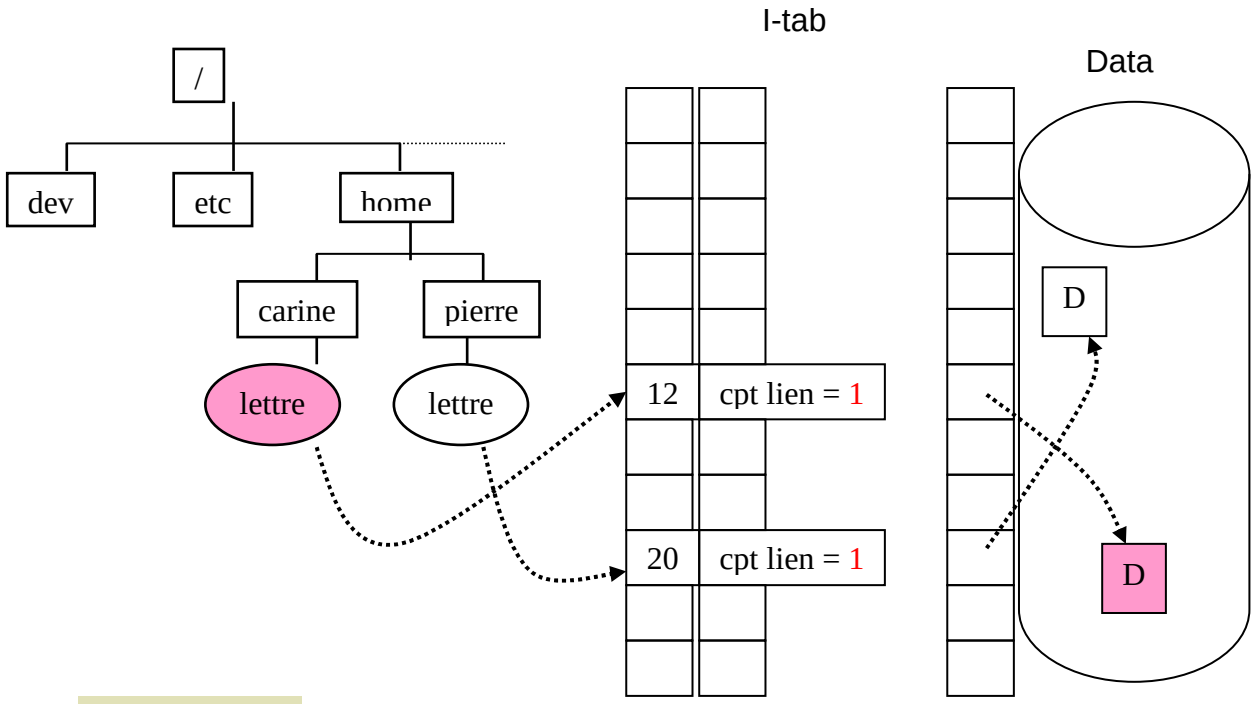
IV-5 Les Liens sur les Fichiers

Le système UNIX offre la possibilité de donner plusieurs noms à un même fichier physique, sans que celui-ci ait plusieurs copies. Il est dit qu'un nouveau **point d'accès** au fichier est créé. Contrairement à la copie de fichiers, il n'y pas de duplication du fichier physique.



Avant la copie

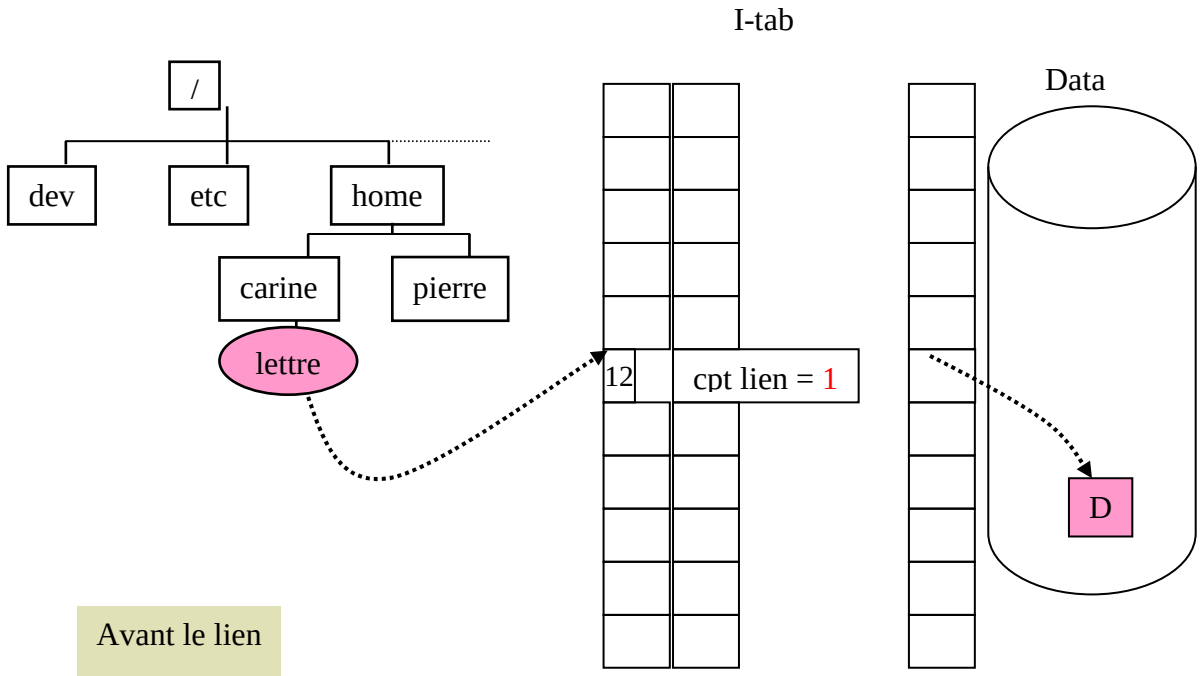
Le **compteur de liens** représente le nombre de chemins d'accès à un fichier donné dans un **même système de fichiers**. Par défaut le compteur de lien est égal à 1.



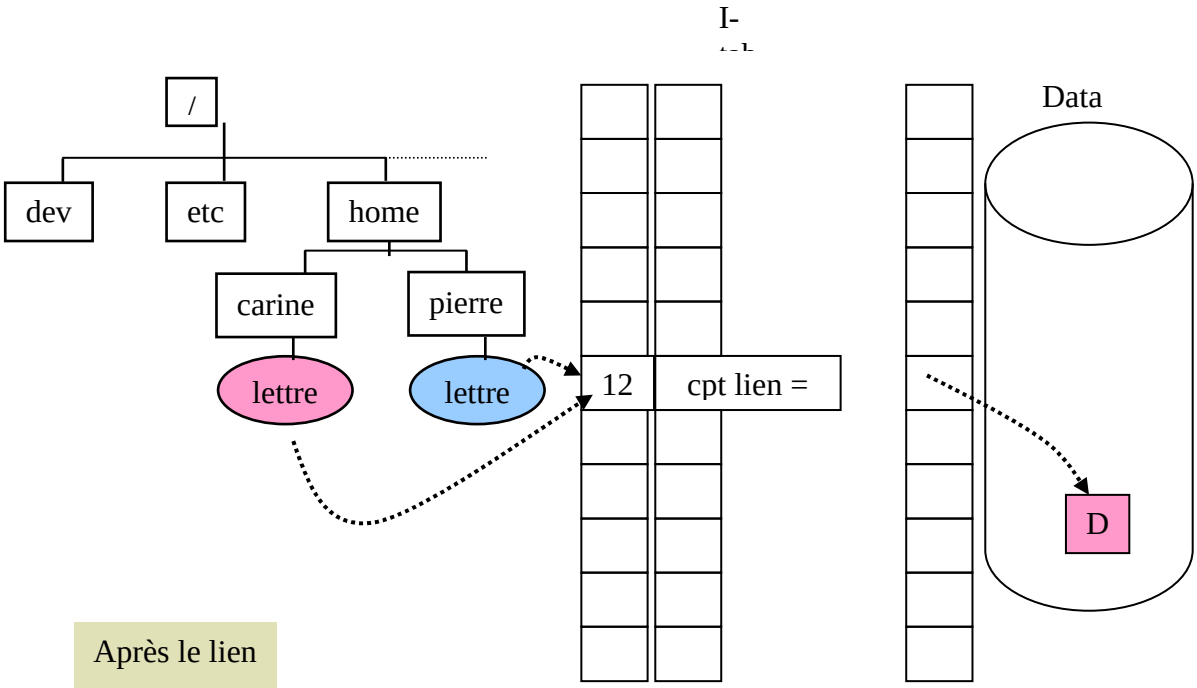
Après la copie

IV-51 Les liens physiques

Pour chaque lien réalisé dans un répertoire souhaité, une référence contenant le nouveau nom du fichier et le numéro d'inode du fichier originel est créé. Chacune des entrées, dans les répertoires respectifs, contient donc un nom particulier auquel est adjoint le même numéro d'inode. Le fichier reste "physiquement" unique.



Avant le lien



Après le lien

Exemple : ls –l /home/carine/lettre

-rwxr-xr-x **2** carine eleve 26 nov 31 15 :21 lettre

compteur de liens

Remarques :

✓ Les caractéristiques d'un fichier étant stockées dans l'inode, elles sont les mêmes pour tous les fichiers liés.

✓ Il peut être intéressant de lier un fichier de données afin qu'il soit accessible à plusieurs utilisateurs sans qu'on ait à le copier. Néanmoins, sans mise en place de dispositifs spécifiques, le problème des accès simultanés à un fichier peut surgir.... avec les risques que cela peut produire sur la cohérence des données.

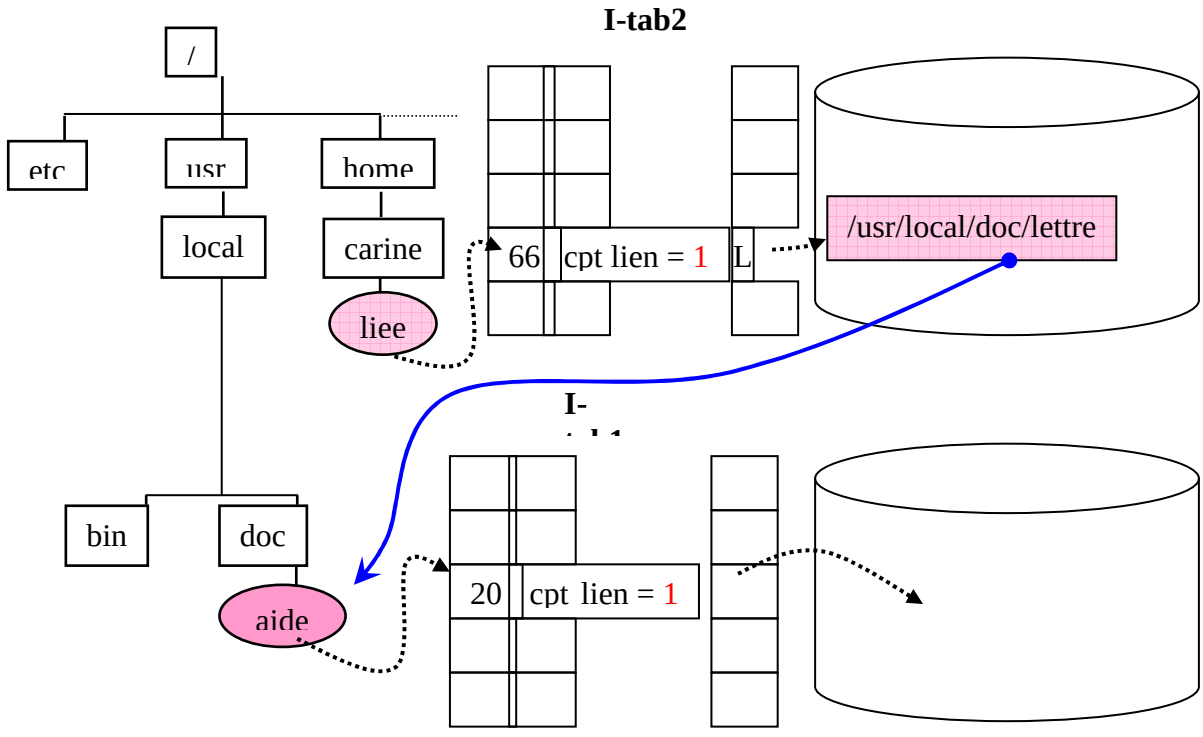
✓ La notion de ***lien physique*** a quelques limitations : comme elle dépend étroitement de l'unicité des inodes, il est impossible de lier des fichiers appartenant à des systèmes de fichiers différents. De même, il est impossible de lier des répertoires .

✓ Ce procédé date du temps (1975 : Version 6 d'AT&T) où les capacités des disques durs étaient moindres. Il est de moins en moins utilisé de nos jours.

IV-52 Les liens symboliques

Contrairement au lien physique, un lien symbolique implique la création d'un nouvel inode. Un lien symbolique définit un **chemin d'accès** synonyme du chemin d'accès du fichier (ordinaire ou répertoire).

Le fichier créé ne contient qu'une indication sur le chemin permettant d'atteindre le fichier "pointé". Cette notion n'a plus les limitations des liens physiques : il est désormais possible de lier des répertoires et des fichiers appartenant à des systèmes de fichiers différents.



Après le lien

Dans le schéma ci-dessus, l'on suppose que le répertoire /home représente le sommet de l'arborescence d'un deuxième système de fichiers.

Exemple :

ls –l /home/carine/liee

lrwxr-xr-x 1 carine eleve 23 nov 31 15 :21 liée→ /usr/local/doc/aide

Le "l" caractérise un fichier lien symbolique.

Remarque :
Il est possible de déplacer le fichier lien comme le fichier origine, mais alors le lien est coupé (clignotement sous LINUX) car le chemin à décrire n'est plus le même.

IV-53 La commande ln

Cette commande crée un lien entre une entrée de répertoire (un fichier) et un fichier existant physiquement. Le lien peut se faire avec un fichier ayant un nom différent ou un fichier ayant le même nom mais situé dans un autre répertoire.

syntaxes: ***ln fich_1 fich_lien2***
Création d'un lien physique, avec le nom de fichier *fich_lien2* sur le *fich_l* dans le même répertoire

ln fichier répertoire
Création d'un lien physique avec le même nom de fichier dans ***répertoire***

Rappel : impossibilité de créer des liens physiques sur un répertoire ou sur un fichier appartenant à un autre volume.

✓ Un répertoire a toujours au moins deux liens ("**.**" et "**..**") créés par le système.
Le compteur de liens dans ce cas, représente alors le nombre de sous-répertoires (y compris "." et "..").

✓ L'option *"-s"* de la commande *ln* permet de créer un ***lien symbolique*** :
ln -s nom_fich lien_symb
Création d'un lien symbolique *lien_symb* sur le fichier *nom_fich*

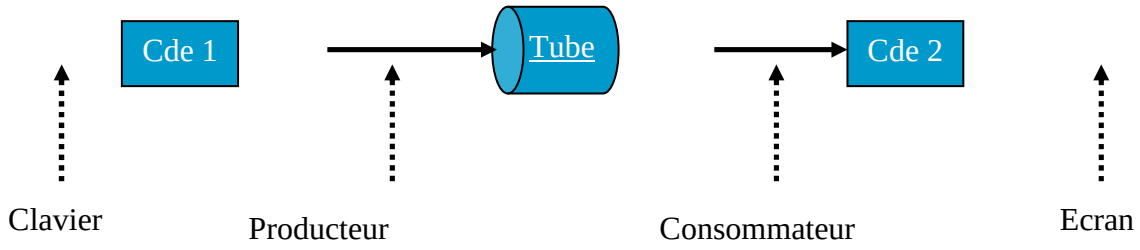
Conseil : lors de la création d'un lien symbolique depuis un répertoire quelconque, pour décrire le chemin, il faut se placer au préalable du côté du fichier lien (fichier final) et voir comment rejoindre le fichier lié (fichier origine).

IV-6 Les Fichiers de Communications

Il s'agit des fichiers *tubes (ou pipe),* et des fichiers *sockets*.

IV-61 Les fichiers tubes

Ils permettent de passer des informations entre des processus selon une technique fifo (first in first out) : un processus (le producteur) écrit des informations transitoires dans un fichier *pipe*, et un autre (le consommateur) les lit. Après lecture, les informations ne sont plus accessibles.



Les fichiers tubes sont appelés des pipes nommés (ou pipes fichiers). Ils peuvent être créés par l'utilisateur grâce à la commande *mknod*.

Exemple :

```
mknod  fic_tube  p
ls  –l  fic_tube
prwxr--r--     1  prof   instruc        0        nov 31   15 :21    fic_tube
```

le "p" caractérise un fichier pipe nommé

Remarque :

- ✓ Il existe un deuxième type de tube de communication appelé pipe du shell (ou pipe mémoire) : le système d'exploitation crée une zone partagée en mémoire où les deux processus communiquent. (cf. cours processus).
- ✓ Pour créer cette zone et relier la sortie standard du premier processus sur l'entrée standard du second, l'utilisateur emploie le raccourci clavier « altgr 6 » ou « | ».

Exemple de deux commandes « pipées » : *ls -l | more*

IV-62 Les fichiers sockets

Les fichiers *sockets* sont des tubes de communication bi-directionnels inter-processus qui utilisent une inode du système de fichiers. De ce fait, ils ne peuvent faire communiquer que des processus situés sur une même station.

D'un autre côté, ces fichiers peuvent être en rapport avec l'outil *socket* de B.S.D. qui permet de communiquer avec d'autres stations via la carte réseau (protocoles TCP/IP ou UDP/IP).

Exemple :

```
ls  –l  /dev/log
srw-rw-rw-   1   root   system   0    dec 15   06 :20   log
```

le "s" caractérise un fichier socket

(Les fichiers sockets sont étudiés plus avant dans un cours C-système).

Remarque :
Les fichiers de communication ne contiennent que des données transitoires, le temps de la transaction. La commande « ls –l » affichera toujours une taille du fichier nulle.

V- ATTRIBUTS DES FICHIERS

V-1 Droits d'Accès

V-11 Cas générique des fichiers

L'accès des fichiers est contrôlé par un principe de permissions d'accès.

Il existe 4 types de protections qui déterminent les modes d'emploi autorisés d'un fichier :

r	droit en lecture
w	droit en écriture
x	droit d'exécution
-	aucun droit.

Exemples :
- ✓ Pour éditer un fichier sous Vi et le modifier, il faut que ce dernier ait non seulement les droits en écriture mais aussi les droits en *lecture* (ouverture de fichier).
- ✓ Pour exécuter un script shell, il faut que ce dernier ait non seulement les droits en exécution mais aussi les droits en *lecture* (ouverture de fichier).

V-12 Cas particulier des répertoires

Les significations des droits d'accès sont différentes lorsqu'il s'agit d'un fichier répertoire :

r	autorise la lecture et le listage du répertoire (commande *ls*)
w	autorise la création ou la destruction d'un fichier (un fichier protégé en écriture n'est pas protégé contre la destruction)
x	autorise le passage (accès aux fichiers et aux sous-répertoires; commande *cd*)
-	aucun droit.

Remarques :
- ✓ La commande " ls -l" lancée sur un répertoire distant ne peut aboutir si les droits en lecture et en *exécution* ne sont pas posés sur lui, alors que la commande "ls" est possible sur un répertoire sans le droit x (r suffit).
- ✓ Poser les droits en écriture et en *exécution* sur un répertoire permet non seulement de supprimer le répertoire lui-même, mais aussi de créer, modifier et supprimer des fichiers (ou sous-répertoires) sous ce dernier.

Attention :
- ✓ Un fichier protégé en écriture peut se faire supprimer par un autre utilisateur dans la mesure où les droits du répertoire qui le contient autorise cet utilisateur à effectuer cette opération.

V-13 Classement des utilisateurs

Le système d'exploitation UNIX gère des groupes d'utilisateurs.
Tout utilisateur déclaré appartient à un groupe déclaré.

La commande "ls -l" affiche :

- le nom du propriétaire du fichier, le groupe d'utilisateurs auquel ce propriétaire appartient.

Exemple : -rwxrwxrwx 1 paul instruc 60 Nov 2 09:01 fichier

En effet, **l'utilisateur qui crée** un répertoire ou un fichier en devient le **propriétair**e.

Le système d'exploitation autorise tout **propriétaire** et tout **administrateur** à modifier et supprimer un fichier.

En conséquence, le système d'exploitation UNIX classe les utilisateurs potentiels en trois catégories qui déterminent les modes d'emploi possibles des fichiers :

user	pour le propriétaire du fichier
group	pour le groupe d'utilisateurs auquel appartient le propriétaire
other	pour le reste des utilisateurs du système.

En fonction de cela, chaque catégorie dispose de droits d'accès en lecture et /ou écriture et / ou exécution spécifiques sur le fichier.

```
 ls -l fichier
-rwxrwxrwx  1  paul  instruc  60  Nov 2 09:01  fichier
  U  G  O
```

Synthèse :
Chaque inode comporte un champ de 12 bits mémorisant les différents droits d'accès : chaque bit est le drapeau d'un droit d'accès.

suid	sgid	sticky	r	w	x	r	w	x	r	w	x
1	1	1	1	1	1	1	1	1	1	1	1
Attributs spéciaux			*Droits du propriétaire*			*Droits du groupe*			*Droits des autres*		

Les bits spéciaux à 1 positionnent les attributs spéciaux.

V-2 Attributs Spéciaux

V-21 Sticky-bit

Cet attribut spécial a deux fonctions différentes selon qu'il est posé sur un fichier ordinaire ou sur un répertoire.

Lorsque le bit de permanence *(sticky-bit* littéralement "bit collant" ou bit de résidence) est positionné **sur un fichier**, le code du programme reste **résidant** en mémoire après qu'il ait

été exécuté. Cette possibilité permet d'éviter des transferts disques à des programmes qui sont appelés fréquemment.

Exemple : ls –l monfic

-rw-r--r-**T** 1 prof instruc 12 dec 25 23:59 monfic

↖ le "T" représente un sticky bit positionné

Remarques :
✓ Le *sticky-bit* positionné sur un fichier qui n'a pas le droit en exécution pour les autres, est représenté par un "T". Si ce droit "x" est positionné, le sticky est représenté par un "t".

Dans le cas d'un fichier de type répertoire, le fait de positionner le *sticky-bit* protège l'ensemble des fichiers de ce répertoire qui ont le même propriétaire que le répertoire en question, des actions de modification des autres utilisateurs autorisés à accéder à ce répertoire. Les fichiers du propriétaire sont ainsi protégés quels que soient les droits d'accès posés sur eux par ce dernier.

Rappel : un fichier protégé en écriture n'est pas protégé contre la destruction si les droits posés sur le répertoire qui le contient autorise les modifications de celui-ci !

Exemple : ls –ld /tmp

drwxrwxrw**t** 5 root system 1024 fev 29 12:18 /tmp

V-22 Set-User-ID (SUID) et Set-Group-ID (SGID)

Il s'agit d'une possibilité supplémentaire de permission d'accès attribuée à un utilisateur lorsqu'il est nécessaire qu'il dispose des mêmes droits que ceux du propriétaire d'un fichier, ou ceux du groupe concerné.
En effet, un utilisateur peut avoir à exécuter un programme (en général un utilitaire système) mais ne pas avoir les droits d'accès nécessaires.
En positionnant les bits adéquats [bits "s" au niveau du propriétaire et (ou) au niveau du groupe], l'utilisateur du programme possède, pour le temps d'exécution de celui-ci, l'identité du propriétaire (ou celle du groupe) du programme.

Par exemple, le fichier */etc/passwd* qui contient des informations sur les différents utilisateurs ne peut être modifié par ceux-ci.
Le programme /usr/*bin/passwd* est un utilitaire, appartenant au super-utilisateur, qui permet à chaque utilisateur d'effectuer la modification de son mot de passe.
Cela est possible par le fait que cet utilitaire a le SUID positionné [droit d'exécution du propriétaire à "s"].
L'utilisateur de /usr/*bin/passwd* bénéficie donc le temps de la commande *passwd* des permissions d'accès de "root": il peut donc modifier le fichier */etc/passwd*.

Exemple : ls –l /usr/bin/passwd

-r-**s**--x--x 1 root system 12 dec25 23 :59 usr/bin/passwd

le "s" représente un SUID positionné

Remarques :

✓ Le *SUID* positionné sur un fichier qui n'a pas le droit en exécution pour le propriétaire, est représenté par un "S"; si ce droit "x" est positionné, le SUID est représenté par un "s".

✓ Le *SGID* positionné sur un fichier qui n'a pas le droit en exécution pour le groupe, est représenté par un "S"; si ce droit "x" est positionné, le SUID est représenté par un "s".

V-3 La Commande chmod

Cette commande permet de changer les permissions d'accès à un ou plusieurs fichiers. La commande peut utiliser des paramètres *absolus ou symboliques.* Elle ne peut être utilisée que par le propriétaire du fichier ou par l'administrateur.

syntaxes: ***chmod** [-R] **mode fichier***
chmod [ugoa] [+ | - | =] [droits] fichier

L'option **–R** appliquée à un répertoire rend possible la **R**écursivité de la commande. Dans le cas présent, la commande change les droits du répertoire et de tous les fichiers se trouvant dessous.

V-31 Mode (ou méthode) symbolique

Principe :
Dans ce mode, trois éléments représentant le destinataire de la commande *chmod*, l'opération à effectuer, le type de permissions sont utilisés.

Les valeurs prises par ces trois indicateurs sont :

- indicateur de destination	:	u (user) pour le propriétaire, g (group) pour le groupe, o (others) pour les autres, a (all) pour tous (u+g+o)
- indicateur d'opération	:	+ ajoute la ou les permissions, - retire la ou les permissions, = assigne la ou les permissions
- indicateur de permission	:	r en mode lecture, w en mode écriture, x en mode exécution, s permet l'activation du S(U)(G)ID concerné, t met en place un *sticky bit.*

Exemples :

chmod a+r nom-fichier : autorise la lecture pour tous

chmod ug-x nom-fichier : retire le droit d'exécution au propriétaire et au groupe

chmod o=rwx nom-fichier : donne tous les droits aux autres utilisateurs

V-32 Mode (ou méthode) absolu(e) ou octal(e)

Principe : r = 4 w = 2 x = 1

Exemple : chmod 734 monfic

=>	r w x	- w x	r - -
	4+2+1	0+2+1	4+0+0
	7	3	4

Le paramètre en mode absolu est représenté par 4 chiffres en octal :

4000	SUID activé
2000	SGID activé
1000	sticky-bit activé
0400	accès lecture pour le propriétaire
0200	accès écriture pour le propriétaire
0100	permet l'exécution pour le propriétaire
0040	accès lecture pour le groupe propriétaire
0020	accès écriture pour le groupe propriétaire
0010	permet l'exécution pour le groupe propriétaire
0004	accès lecture pour les autres
0002	accès écriture pour l'es autres
0001	permet l'exécution pour les autres
.....	

L'addition des valeurs donne le mode global.
Exemple : 0761 signifie
mode lecture / écriture / exécution pour le propriétaire,
mode lecture / écriture pour le groupe,
mode exécution pour les autres

V-4 La Commande umask

Cette commande permet de modifier les permissions d'accès données par défaut à un fichier lors de sa création. Sans paramètre, la commande affiche la valeur d'umask en cours.

syntaxe : ***umask [valeur_octale]***

principe : *complément à 7*

777	masque	777	masque
-022	valeur umask	-755	droits
755	droits	022	valeur umask

Le paramètre de umask est constitué par trois chiffres en octal.
Les permissions à masquer sont spécifiées par les mêmes valeurs que celles utilisées par la commande chmod (en mode absolu). Le 1er chiffre représente le propriétaire, le 2ème le groupe et le 3ème les autres.

- Notons que umask fonctionne à l'inverse de chmod en mode absolu : les droits spécifiés sont ôtés dans umask alors qu'ils sont positionnés dans chmod.

- **Contrairement à un répertoire, un fichier ne possédera jamais les droits d'exécution dès sa création !**

Exemple : umask 022

➔ Droits d'un répertoire à la création : 755 (rwx r-x r-x)
➔ Droits des fichiers à la création : 644 (rw- r-- r--)

V-5 Synthèse des Attributs

ls -lisa /bin/passwd

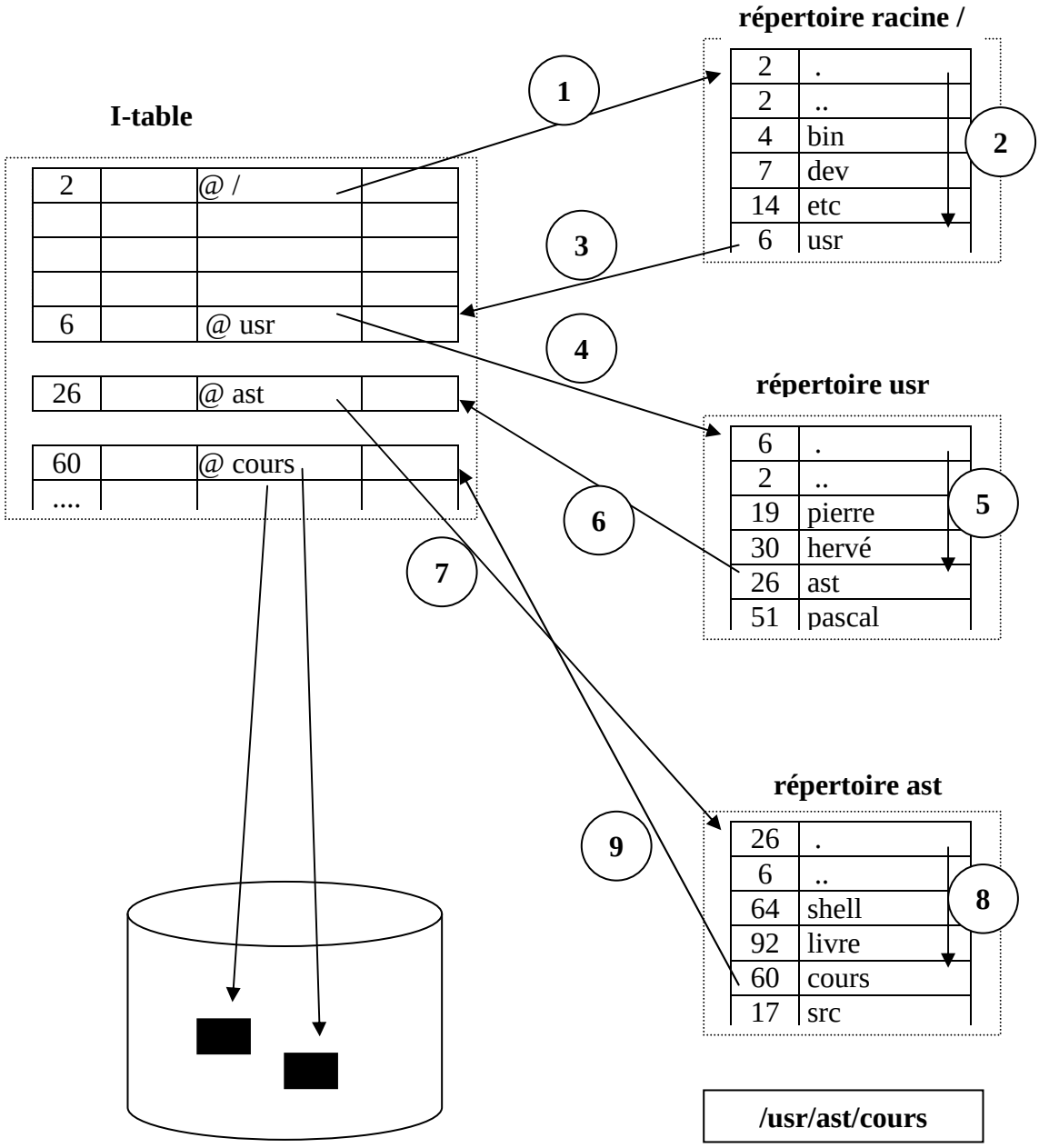
```
18  48  -rwsr-sr-t  1  root  sys  23572  Sep 19 2000 /bin/passwd
```

- Date de dernière MAJ
- Taille en octets (Fic ordinaire) / Major ,Minor (Fic spécial)
- Nom du groupe (GID)
- Nom du propriétaire (UID)
- Nombre de liens (Fic normal) / Nombre de sous-répertoires (Fic répertoire)
- Permissions
- Type du fichier
  - \- => normal
  - d => répertoire
  - b => spécial bloc
  - c => spécial caractère
  - p => tube
  - s => socket
  - l => lien symbolique
- Taille du fichier en blocs physiques
- N° d'inode

VI COMPLEMENT : PROCEDURE D'ACCES A UN FICHIER

Compte tenu de la manière dont est constitué le système de fichiers, la procédure suivie par le système pour retrouver l'adresse d'un fichier et de ses différents blocs de données est relativement complexe.

Exemple : accès au fichier */user/ast/cours.*



1. " / " est le répertoire racine. Il est référencé, en tant que tel dans le 1er inode de la table des inodes dont le numéro (n°2) est connu par le système. Comme il s'agit d'un répertoire, l'inode contient l'adresse d'un bloc de données correspondant à la première entrée du catalogue du répertoire (table de correspondance *nom de fichiers / inodes*).

2./3. Au champ *user de* cette table correspond le numéro d'inode 6.

4. "usr" est aussi un répertoire dont l'inode n°6, situé dans la table des inodes, contient une adresse vers le catalogue de ce répertoire.

5./6. Le champ *ast* de cette table est lié au numéro d'inode 26.

7./8./9. *"ast"* est encore un répertoire. Selon le même raisonnement, l'entrée du catalogue "ast" permet de lier le champ *cours* à l'inode portant le numéro 60. Dans l'inode n°60 sont stockés les pointeurs vers les blocs de données constitutifs du fichier "cours".

Le procédé parait extrêmement complexe mais se révèle extrêmement efficace.

CONCLUSION

L'étude des systèmes de fichiers sous UNIX est relativement complexe. Ce support de cours permet d'effectuer un solide tour d'horizon du sujet, et ne prétend pas être exhaustif.
Pour bénéficier de compléments sur le sujet, la consultation d'ouvrages spécifiques est conseillée.

Bibliographie sélective :

- "UNIX sous tous les angles" d' A.Janssens (Editions Eyrolles).
- "Les bases de l'administration système" d'A. Frisc (Editions O'Reilly)