VI- GESTION DES FICHIERS

INTRODUCTION

- Un fichier désigne un ensemble d'informations stockées sur le disque.
- Le système de fichiers est la partie du système d'exploitation qui se charge de gérer les fichiers.
- La gestion consiste en la création (identification, allocation d'espace sur disque), la suppression, les accès en lecture et en écriture, le partage de fichiers et leur protection en contrôlant les accès.

QU'EST CE QU'UN FICHIER ?

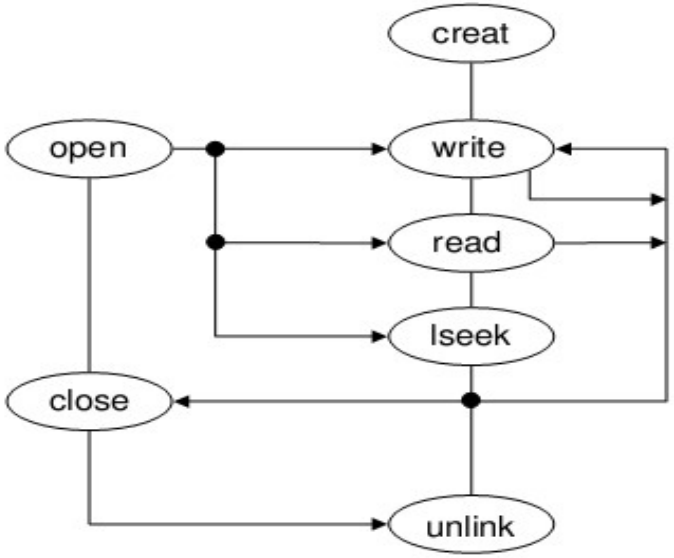
- Pour le système d'exploitation, un fichier est une suite d'octets.
- Par contre, les utilisateurs peuvent donner des significations différentes au contenu d'un fichier (suites d'octets, suite d'enregistrements, arbre, etc.).
- Chaque fichier est identifié par un nom auquel on associe un emplacement sur le disque (une référence) et possède un ensemble de propriétés : ses attributs.
- Le nom est en général, composé de deux parties séparées par un point.

QU'EST CE QU'UN FICHIER ?

- La partie qui suit le point est appelée extension (prog.c, stdio.h, fich.doc, archivo.pdf, etc.).
- L'extension peut être de taille fixe, comme dans MS-DOS ou variable comme c'est le cas d'Unix/Linux ; obligatoire ou non.
- L'extension est nécessaire dans certains cas. Par exemple, le compilateur C rejettera le fichier prog.txt même si son contenu est un programme C.
- Le nom d'un fichier peut être sensible à la typographie : ainsi en Unix/Linux Tonton et tonton.

CYCLE DE VIE DES FICHIERS

- Les fichiers, comme bien d'autres composants, ont un cycle de vie.
- Ils sont crées (ou ouverts), on les fait modifier (écrire), on lit à partir d'eux, et finalement, peut être, ils meurent (sont effacés).



TYPES DE FICHIERS

- Dans un système, il existe plusieurs types de fichiers. Unix/Linux et MS-DOS ont des fichiers ordinaires, des répertoires, des fichiers spéciaux caractère et des fichiers spéciaux bloc.
- Les fichiers ordinaires contiennent les informations des utilisateurs. Ils sont en général des fichiers ASCII ou binaires.
- Les répertoires sont des fichiers système qui maintiennent la structure du système de fichiers.
- Les fichiers spéciaux caractère sont liés aux Entrées/Sorties et permettent de modéliser les périphériques d'E/S série tels que les terminaux, les imprimantes et les réseaux. Les fichiers spéciaux bloc modélisent les disques.

TYPES DE FICHIERS

- Dans le système Unix/Linux, "-" désigne les fichiers ordinaires, "d" les répertoires, "c" les périphériques à caractères, "b" les périphériques à blocs, et "p" les tubes avec nom (named pipe).
- Les périphériques sont des fichiers désignés par des références.
- Les fichiers spéciaux bloc et caractère vont identifier les dispositifs physiques du matériel : les disques, les rubans magnétiques, les terminaux, les réseaux, etc.
- Chaque type de dispositif a un contrôleur responsable de sa communication.

TYPES DE FICHIERS

- Dans le système d'exploitation il y a une table qui pointe vers les différents contrôleurs de dispositifs. Tous les dispositifs seront alors traités comme de fichiers.
- Dans Unix/Linux les périphériques comme les terminaux sont des fichiers spéciaux qui se trouvent sous le répertoire /dev.
- **Exemple** : on copie le texte : "abcd" tapé au clavier, **dans un fichier fictoto** par la commande :

toto> cat > fictoto

abcd

^D

et **sur un terminal,** par la commande :

toto> cat > /dev/tty

abcd

^D

ACCÈS AUX FICHIERS

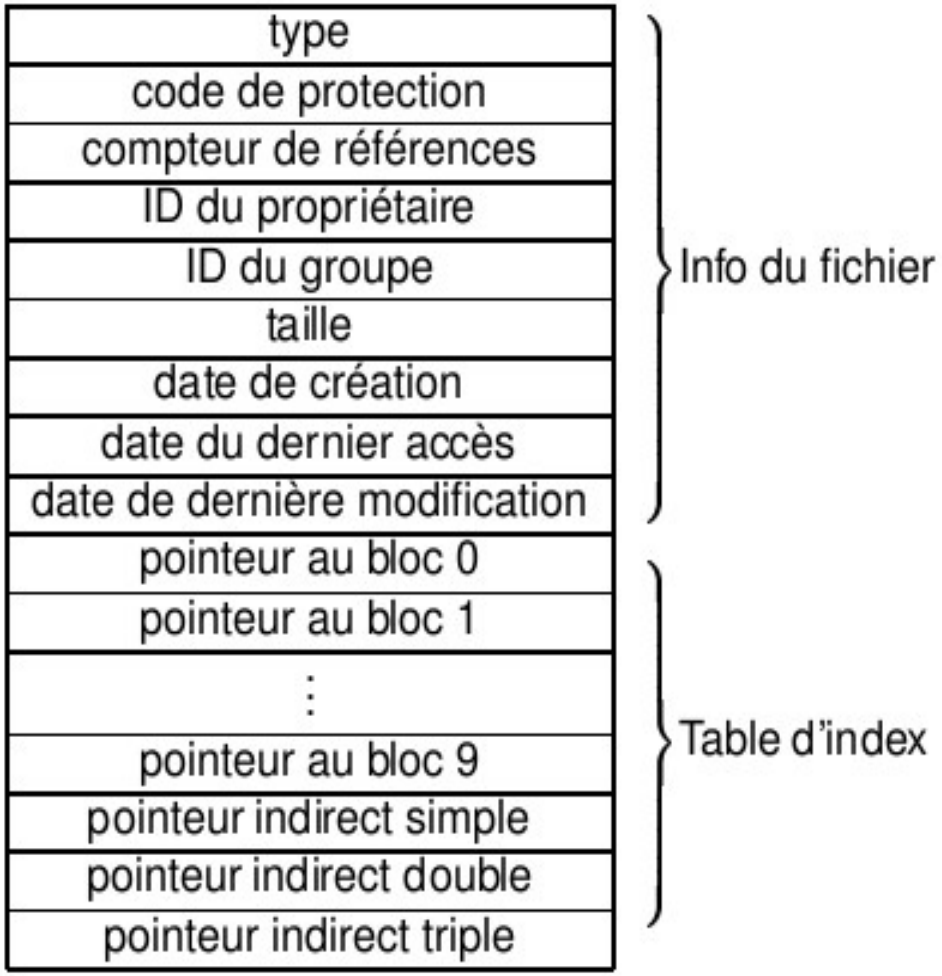
- Pour accéder à un fichier il faut fournir au système de fichiers les informations nécessaires pour le localiser sur le disque, c'est-à-dire lui fournir un chemin d'accès.
- Les systèmes modernes permettent aux utilisateurs d'accéder directement à une donnée d'un fichier, sans le parcourir depuis le début du chemin.
- On parle de chemin relatif par opposition au chemin absolu qui part de la racine de l'arborescence au fichier.

ATTRIBUTS DES FICHIERS

- Les attributs des fichiers diffèrent d'un système à un autre.
- Cependant, ils peuvent être regroupés en deux catégories :
  1. Les attributs qui servent à **contrôler les accès** comme le code de protection, le mot de passe, le propriétaire, etc.
  2. Les attributs qui définissent le **type et l'état courant** du fichier : indicateur du type ASCII/binaire, taille courante, date de création, date de la dernière modification, etc.

I-NŒUD DE FICHIER

- Dans les systèmes comme Unix/Linux toutes les informations des fichiers sont rassemblées dans une structure associée au fichier, appelée noeid d'information, i-noeud ou i-node.
- L'i-noeud contient les informations suivantes :
  - le type du fichier (régulier, répertoire, caractère, bloc ou tube) ;
  - le code de protection sur 9 bits ;
  - l'identificateur et groupe du propriétaire ;
  - les dates de création, du dernier accès et de la dernière modification ;
  - le compteur de références ;
  - la taille et finalement la table d'index composée de 13 numéros de blocs et de pointeurs.


type
code de protection
compteur de références
ID du propriétaire
ID du groupe
taille
date de création
date du dernier accès
date de dernière modification
pointeur au bloc 0
pointeur au bloc 1
pointeur au bloc 9
pointeur indirect simple
pointeur indirect double
pointeur indirect triple
Info du fichier
Table d'index

SERVICES POSIX SUR LES FICHIERS

- Les fichiers permettent de stocker des informations et de les rechercher ultérieurement.
- Les systèmes fournissent un ensemble d'appels système relatifs aux fichiers.
- Dans le cas d'Unix/Linux, les principaux appels système Posix relatifs aux fichiers sont :
  - creat() et open() pour la création et l'ouverture d'un fichier.
  - close() pour la fermeture d'un fichier.
  - read() pour la lecture d'un fichier.
  - write() pour l'écriture dans un fichier.
  - lseek() pour déplacer le pointeur de fichier.
  - stat() pour récupérer des informations d'un fichier.
  - link() pour créer un lien entre deux fichiers.
  - unlink() pour supprimer un lien ou un fichier.

RÉPERTOIRES

- Les systèmes d'exploitation modernes adoptent une structure arborescente pour représenter le système de fichiers. Les noeuds de l'arbre sont des répertoires et les feuilles sont des fichiers.
- Un répertoire est composé de fichiers et de sous répertoires.
- Pour accéder à un fichier, il suffit de spécifier les répertoires du chemin qui mène de la racine de l'arborescence au fichier (chemin d'accès).
- Dans le système Unix/Linux, chaque répertoire contient aussi sa propre référence "." et celle du répertoire immédiatement supérieur "..".
- Un répertoire est aussi considéré comme un fichier particulier composé des fichiers.

SERVICES POSIX SUR LES RÉPERTOIRES

- Les répertoires d'Unix/Linux possèdent une entrée par fichier.
- Chaque entrée possède le nom du fichier et le numéro de l'i-noeud.
- Le système Unix/Linux offre des appels système Posix pour manipuler les répertoires :
  - mkdir() créer un répertoire.
  - rmdir() supprimer un répertoire vide.
  - opendir()ouvrir un répertoire.
  - closedir()fermer un répertoire.
  - readdir()lire les entrées d'un répertoire.
  - rewindir()placer le pointeur d'un répertoire.
  - link() créer une entrée dans un répertoire.
  - unlink() effacer une entrée d'un répertoire.
  - chdir() changer de répertoire de travail.
  - rename() renommer un répertoire.
  - getcwd() obtenir le nom du répertoire actuel.

i-nœud	Nom du fichier
20	.
3	..
100	chap1
2378	chap2
125	scheduler
⋮	⋮

RÉPERTOIRE MSDOS.

- Les répertoires de MS-DOS possèdent une entrée par fichier.
- Chaque entrée a les informations suivantes :
  - Nom du fichier
  - Extension
  - Attributs
  - Réservé
  - Heure de la dernière modification
  - Date de la dernière modification
  - Numéro du premier bloc
  - Taille

Nom du fichier
Extension
Attributs
Réservé
Heure Der. Mod.
Date Der. Mod.
N° du 1er bloc
Taille

STOCKAGE ET PARTAGE DE FICHIERS

- Les fichiers de données sur les disques se répartissent dans des blocs de taille fixe.
- La lecture ou l'écriture d'un élément d'un fichier impliquera donc le transfert du bloc entier qui contient cet élément.
- Pour un accès rapide, on aura donc intérêt à prendre des blocs de grandes tailles.
- Cependant, les fichiers, y compris les fichiers de 1 octet, ont une taille minimale de 1 bloc.
- Si un disque comprend beaucoup de fichiers de petites tailles et si les blocs sont de grandes dimensions, l'espace gaspillé sera alors considérable.
- Il faut donc implanter des techniques adéquates de stockage de fichiers. Il y a trois manières de faire l'allocation des fichiers : l'allocation contiguë, la liste chaînée de blocs et l'indexation par i-noeuds.

STOCKAGE DES FICHIERS

- Des études sur de nombreux systèmes ont montré que la taille moyenne d'un fichier est de 1 Ko.
- En général, les tailles de blocs fréquemment utilisées sont de 512, 1024, ou 2048 octets.
- Chaque disque conserve, dans un ou plusieurs blocs spécifiques, un certain nombre d'informations telles que le nombre de ses blocs, leur taille, leurs états, entre autres.
- À chaque fichier correspond une liste de blocs contenant ses données.
- L'allocation est en général, non contiguë et les blocs sont donc répartis quasi aléatoirement sur le disque.
- Les fichiers conservent l'ensemble de leurs blocs suivant 2 méthodes : la liste chaînée et la table d'index par i-noeuds.

ALLOCATION CONTIGUË

- La table d'allocation de fichiers contient seulement une entrée par fichier, avec le bloc de début et la taille du fichier.
- L'allocation contiguë est simple et performante, mais a le grave défaut de remplir rapidement l'espace avec des zones inutilisables (à cause de la fragmentation externe), et dans ce cas le compactage n'est pas envisageable avec des temps raisonnables.
- En plus il faudrait déclarer la taille du fichier au moment de sa création.

LISTE CHAÎNÉE

- Les blocs d'un même fichier sont chaînés sous une représentation de liste chaînée
- Chaque bloc contiendra des données ainsi que l'adresse du bloc suivant.
- Le fichier doit mémoriser le numéro du 1er bloc. Par exemple, si un bloc comporte 1024 octets et si le numéro d'un bloc se code sur 2 octets, 1022 octets seront réservés aux données et 2 octets au chaînage du bloc suivant.

- Cette méthode rend l'accès aléatoire aux éléments d'un fichier particulièrement inefficace lorsqu'elle est utilisée telle quelle. En effet, pour atteindre un élément sur le bloc n d'un fichier, le système devra parcourir les n-1 blocs précédents.

LISTE CHAÎNÉE

- Le système MS-DOS utilise une version améliorée de listes chaînées.
- Il conserve le premier bloc de chacun des fichiers dans son répertoire.
- Il optimise ensuite l'accès des blocs suivants en gardant leurs références dans une Table d'Allocation de Fichiers (FAT)
- Chaque disque dispose d'une table FAT et cette dernière possède autant d'entrées qu'il y a de blocs sur le disque.
- Chaque entrée de la FAT contient le numéro du bloc suivant.

LISTE CHAÎNÉE

- Par exemple, dans la table suivante :

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
x	x	EOF	13	2	9	8	L	4	12	3	E	EOF	EOF	L	...

- "x" indique la taille du disque, "L" désigne un bloc libre et "E" un bloc endommagé.
- Le fichier commençant au bloc 6, sera constitué des blocs : 6-8-4-2
- Le parcours de la FAT est nettement plus rapide que la chaîne de blocs. Cependant elle doit constamment être tout entière en mémoire, afin d'éviter les accès au disque pour localiser un bloc.

TABLE D'I-NOEUDS

- Le système Unix associe à chaque fichier un numéro unique d'identification.
- A chaque numéro est associé un ensemble d'informations appelé noeud d'information ou i-noeud.
- Parmi les champs de l'i-noeud, la table d'index indique l'emplacement physique du fichier.
- Elle est composée de 13 entrées. Les 10 premières contiennent les numéros des 10 premiers blocs composant le fichier.

TABLE D'I-NOEUDS

- Pour les fichiers de plus de 10 blocs, on a recours à des indirections. Le bloc n° 11 contient le numéro d'un bloc composé lui-même d'adresses de blocs de données. Si les blocs ont une taille de 1024 octets et s'ils sont numérotés sur 4 octets, le bloc n° 11 pourra désigner jusqu'à 256 blocs. Au total, le fichier utilisant la simple indirection aura alors une taille maximale de 266 blocs.
- De la même manière, le bloc n° 12 contient un pointeur à double indirection, et le bloc n° 13 un pointeur à triple indirection.
- Sur la figure suivante, on montre l'usage des indirections simples, doubles et triples d'un i-noeud.

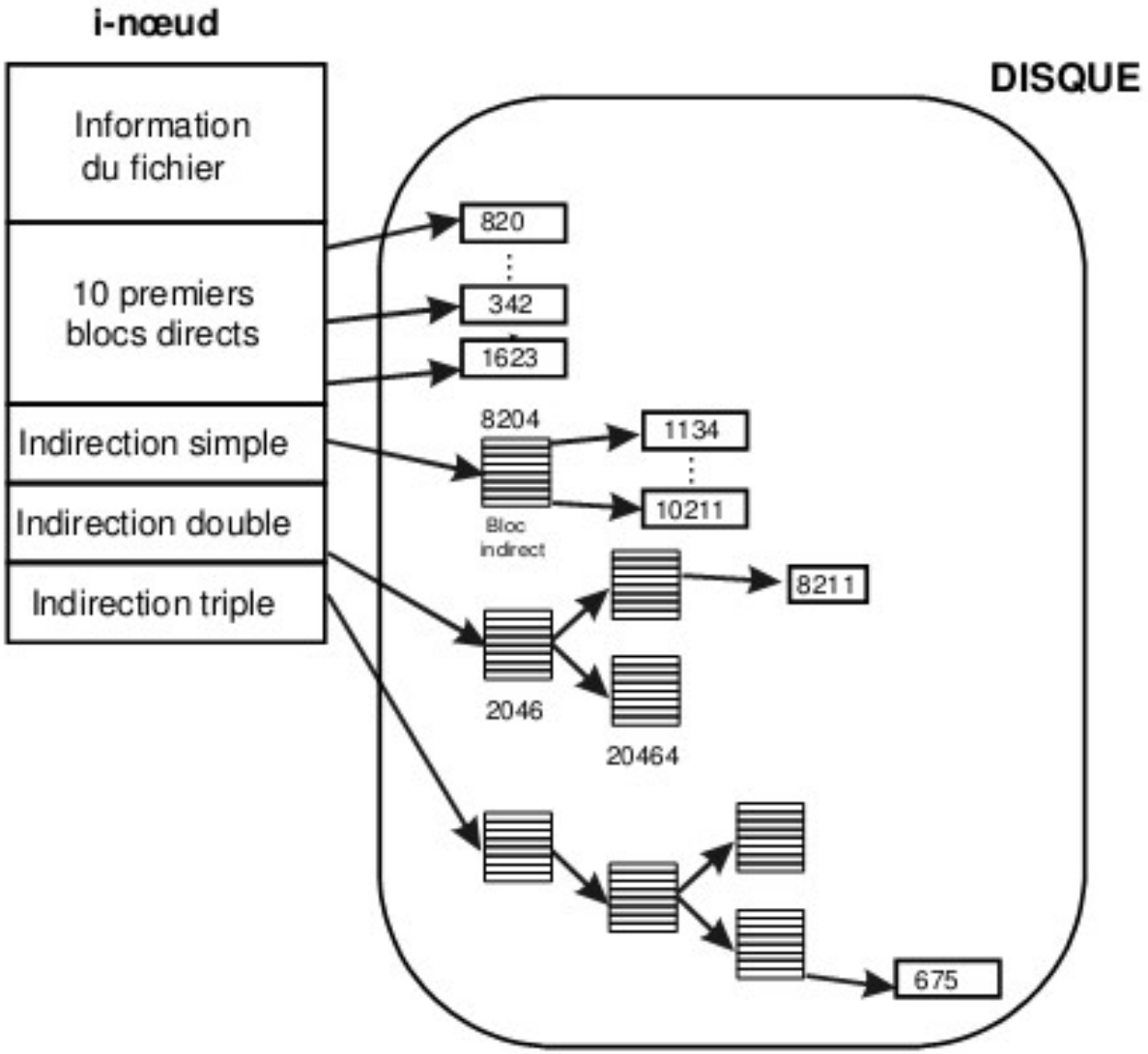

i-nœud
Information du fichier
10 premiers blocs directs
Indirection simple
Indirection double
Indirection triple
DISQUE
820
342
1623
8204
1134
10211
Bloc indirect
8211
2046
20464
675

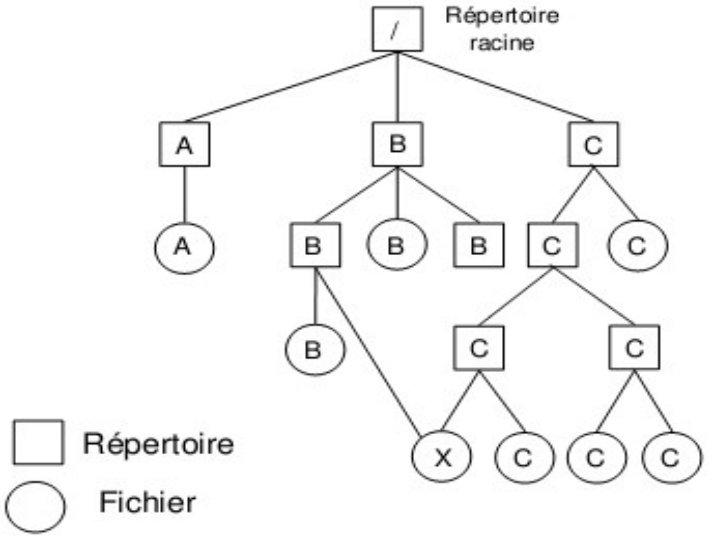
TABLE D'I-NOEUDS

- Un fichier peut avoir une taille maximale de 16 Go quand il utilise des pointeurs à triple indirection.
- La table d'index comporte jusqu'à 4 niveaux d'indexation, mais seul le premier niveau est présent en mémoire lorsque le fichier correspondant est ouvert.
- Dans Linux, la table d'i-noeuds diffère un peu de celle utilisée dans Unix System V.
- Sous Linux la table est composée de 12 entrées pour les blocs et de 3 entrées pour les indirections.

PARTAGE DE FICHIERS

- Lorsque plusieurs utilisateurs travaillent sur un même projet, ils doivent souvent partager des fichiers.
- Pour permettre le partage d'un fichier existant, le système Unix/Linux offre la possibilité d'associer plusieurs références à un fichier, en créant des liens physiques ou des liens symboliques.

Par exemple, sur la figure, le fichier X est partagé entre les répertoires /B/B/X et /C/C/C/X



LIENS PHYSIQUES

- Pour réaliser le partage d'un fichier se trouvant sur un disque, il suffit d'associer à son numéro d'i-noeud plusieurs chemins d'accès ou références.
- Ce type de lien est physique.
- Les lien physiques sont possibles seulement si les chemins d'accès font références uniquement aux répertoires gérés par un même système de fichiers. En d'autres mots, on ne peut pas créer de lien à partir d'un répertoire d'une machine vers un fichier se trouvant sur une autre machine.

LIENS PHYSIQUES

- L'appel système qui permet de créer un lien physique à un fichier existant est link() :int link(char* oldpath, char* newpath);
- La commande shell équivalente à link() est ln : ln oldpath newpath
- La commande shell qui permet de supprimer un lien est rm.

LIENS SYMBOLIQUES

- Les liens symboliques permettent de créer des liens vers des fichiers qui ne sont pas forcément gérés par le même système de fichiers.
- Ils présentent donc l'avantage de pouvoir constituer des liens vers des fichiers situés sur n'importe quel ordinateur à distance.
- Outre le chemin d'accès sur la machine même, il faut placer l'adresse réseau de la machine dans le chemin d'accès.
- Un lien symbolique est un pointeur indirect vers un fichier existant.
- La destruction d'un lien symbolique vers un fichier n'affecte pas le fichier.

LIENS SYMBOLIQUES

- La commande shell d'Unix qui crée des liens symboliques est ln avec l'option s pour symbolique.
- Par exemple, la commande : ln -s /usr/include/stdio.h stdio.h
- crée une nouvelle entrée dans le répertoire courant pour un nouveau fichier dont le nom est stdio.h. Le contenu de ce nouveau fichier est le chemin d'accès /usr/include/stdio.h.

PROBLÈMES DES LIENS

- Les liens permettent aux fichiers d'avoir plusieurs chemins d'accès.
- Les programmes qui parcourent tous les catalogues pour en traiter les fichiers peuvent rencontrer les fichiers partagés plusieurs fois.
- Un même fichier peut donc subir le même traitement plusieurs fois (par exemple, ils peuvent être recopiés plusieurs fois).
- Une fois les liens établis, plusieurs utilisateurs peuvent accéder à un même fichier. Les accès peuvent être concurrents.
- Le système Unix fournit des appels système qui permettent de contrôler les accès aux données d'un même fichier.

VERROUILLAGE DE FICHIERS

- Les processus qui désirent effectuer des accès exclusifs à un fichier peuvent verrouiller, en une seule opération atomique, aussi bien un octet qu'un fichier entier.
- Deux sortes de verrouillage de fichiers sont disponibles : partagé et exclusif.
- Si une portion d'un fichier est verrouillée par un verrou partagé, une seconde tentative d'y superposer un verrou partagé est autorisée, mais toute tentative d'y poser un verrou exclusif sera rejetée tant que le verrou n'est pas relâché.
- Si une partie de fichier contient un verrou exclusif, toute tentative d'en verrouiller une quelconque portion sera rejetée tant que le verrou n'est pas relâché.

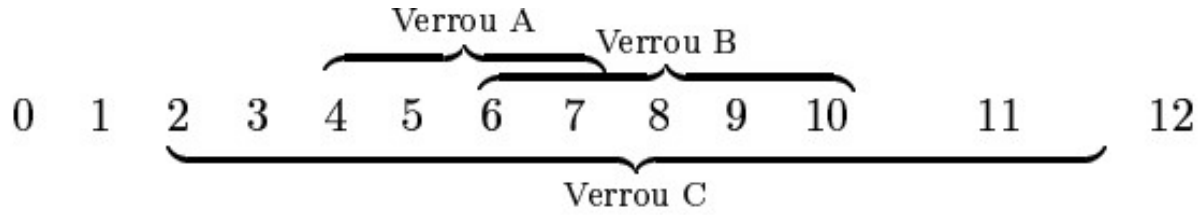
VERROUILLAGE D'UNE PARTIE DE FICHIER

- Lorsqu'un processus demande la pose d'un verrou, il doit spécifier s'il veut être bloqué si le verrou ne peut pas être posé.
- S'il choisit d'être bloqué, lorsque le verrou existant sera levé, le processus sera débloqué et son verrou pourra être posé. S'il ne veut pas être bloqué, l'appel système se termine immédiatement, avec un code qui indique si le verrouillage a pu être ou non effectué.

VERROUILLAGE D'UNE PARTIE DE FICHIER

- Par exemple, soient quatre processus concurrents A, B, C et D qui partagent un même fichier. Les processus font dans l'ordre les demandes de verrouillage suivantes :
- Le processus A demande la pose d'un verrou partagé sur les octets 4 à 7 du fichier.
- Le processus B demande de mettre un verrou partagé sur les octets 6 à 9.
- Le processus C demande la pose d'un verrou partagé sur les octets 2 à 11.
- Enfin, le processus D demande un verrouillage exclusif de l'octet 9 avec une requête bloquante en cas d'échec.

VERROUILLAGE D'UNE PARTIE DE FICHIER



Les processus A, B et C arrivent à poser leurs verrous partagés. Par contre, le processus D, est bloqué jusqu'à ce que l'octet 9 devienne libre, c'est-à-dire jusqu'à ce que B et C libèrent leurs verrous.

COHÉRENCE DU SYSTÈME DE FICHIERS

- La plupart des ordinateurs ont un programme utilitaire qui vérifie la cohérence du système de fichiers.
- Ce programme est peut être exécuté à chaque démarrage du système surtout à la suite d'un arrêt forcé.
- La vérification de la cohérence peut se faire à deux niveaux : blocs ou fichiers.

AU NIVEAU DES BLOCS

- Au niveau des blocs, le vérificateur construit deux tables.
  - La première indique pour chaque bloc occupé, le nombre de fois où le bloc est référencé dans les i-noeuds des fichiers.
  - La seconde indique pour chaque bloc libre, le nombre de fois où il figure dans la liste des blocs libres (ou la table de bits des blocs libres).
- Si le système de fichiers est cohérent, chaque bloc a un 1 soit dans la première table, soit dans la seconde.

AU NIVEAU DES BLOCS

- Si un bloc n'apparaît ni dans la première table, ni dans la deuxième, le bloc est dit manquant. Le vérificateur ajoute les blocs manquants à la liste des blocs libres.
- Si le bloc apparaît deux fois dans la liste des blocs libres, le vérificateur reconstruit la liste des blocs libres
- Le pire qui puisse arriver est qu'un bloc appartienne à deux fichiers (ou plus). Si on détruit l'un des deux fichiers, le bloc sera placé dans la liste des blocs libres (il sera alors à la fois libre et utilisé).

AU NIVEAU DES BLOCS

- Si on détruit les deux fichiers, le bloc figurera deux fois dans la liste des blocs libres. Dans ce cas, la meilleure solution consiste à allouer un bloc libre, à y copier le bloc commun et à remplacer le bloc commun dans l'un des deux fichiers par le nouveau.
- La dernière possibilité d'incohérence dans les deux tables est qu'un bloc soit, à la fois, utilisé et libre. La solution dans ce cas consiste à retirer le bloc de la liste des blocs libres.

AU NIVEAU DES FICHIERS

- Le vérificateur vérifie la cohérence du point de vue des liens. Chaque numéro de noeud d'index doit apparaître autant de fois dans la structure arborescente qu'il possède de liens.
- Il commence au catalogue racine et parcourt récursivement toute l'arborescence pour déterminer pour chaque numéro d'i-noeud le nombre de références. Il compare ensuite le nombre de références obtenu pour chaque numéro d'i-noeud avec celui contenu dans l'i-noeud correspondant.

AU NIVEAU DES FICHIERS

- Si les deux nombres sont différents, le système est dans un état incohérent. La solution consiste à corriger dans l'i-noeud, la valeur du nombre de références.
- Dans la plupart des systèmes Unix, le programme ***fsck*** effectue cette tâche à chaque démarrage, si nécessaire.

PROTECTION

- Les systèmes de fichiers contiennent parfois des informations très importantes.
- Ils doivent donc protéger ces informations contre les accès non autorisés et les pertes.
- Les causes des pertes sont les catastrophes naturelles, les erreurs matérielles ou logicielles, les erreurs humaines.
- La plupart de ces problèmes peuvent être résolus si l'on effectue des sauvegardes assez régulièrement.

PROTECTION

- Le problème des intrus est beaucoup plus complexe.
- Pour assurer la protection contre les intrus, plusieurs mécanismes ont été mis en œuvre, parmi lesquels l'identification de l'utilisateur (mots de passe ou identification physique) et les codes de protection des objets.

FIN