Partie 8 : Systèmes de fichiers

Exercice 1 :

On considère un système de fichiers tel que l'information concernant les blocs de données de chaque fichier est donc accessible à partir du i-noeud de celui-ci (comme dans UNIX). On supposera que :

- Le système de fichiers utilise des blocs de données de taille fixe 1K (1024 octets) ;
- L'i-noeud de chaque fichier (ou répertoire) contient 12 pointeurs directs sur des blocs de données, 1 pointeur indirect simple, 1 pointeur indirect double et 1 pointeur indirect triple.
- Chaque pointeur (numéro de bloc) est représenté sur 4 octets.

a) Quelle est la plus grande taille de fichier que ce système de fichiers peut supporter ?
b) On considère un fichier contenant 100,000 octets. Combien de blocs de données sont-ils nécessaires (au total) pour représenter ce fichier sur disque ?

Exercice 2 :

On considère un système disposant d'un système de fichiers similaire à celui d'UNIX avec une taille de blocs de données de 4K (4096 octets) et des pointeurs (numéros de blocs) définies sur 4 octets. On supposera que le i-noeud de chaque fichier compte 12 pointeurs directs, 1 pointeur indirect simple, 1 pointeur indirect double et 1 pointeur indirect triple. On désire créer un fichier contenant un total de 20.000.000 (vingt millions) de caractères (caractères de fin de ligne et de fin de fichier compris).
Quelle est la fragmentation interne **totale** sur le disque résultant de la création de ce fichier.

Exercice 3 :

Considérez le système de fichiers d'UNIX et les tables de gestion suivantes : la table des i-nœuds, la table de tous les fichiers ouverts et les tables des descripteurs de fichier (une par processus).

1) Indiquez les informations récupérées, ajoutées dans chaque table ainsi que l'information retournée au processus demandeur, lors d'une demande d'ouverture, par un processus, d'un fichier ordinaire existant.
2) Sachant que chaque fichier ouvert a son propre pointeur de fichier, utilisé pour parcourir les données du fichier, indiquez, pour chacun des cas suivants, si les processus père et fils partagent le même pointeur de fichier ? Justifiez votre réponse.
   a) Un processus père ouvre un fichier ordinaire existant avant de créer un processus fils (avant d'effectuer un appel à fork).
   b) Un processus père crée un processus fils (effectue un appel à fork) puis chacun des deux processus ouvre un même fichier ordinaire existant.

3) Supposez que seule la table des i-nœuds est en mémoire et que chaque répertoire tient sur un bloc.
   a) Donnez le nombre d'accès au disque nécessaires pour récupérer l'i-nœud du fichier «/usr/cours/INF3600/tps/tp.pdf ».
   b) Donnez deux avantages qu'ont les liens physiques par rapport aux liens symboliques.

Exercice 4 :

1) Considérez un système de fichier ayant les caractéristiques suivantes et illustré par la figure 1 :
- Taille d'un bloc = 8 KO.
- Numérotation linéaire des blocs de disque de 0 à n-1.
- Utilisation de la technique de l'allocation indexée avec chaînage pour la gestion des blocs physiques : toutes les descriptions de fichiers créés sont regroupées dans la table des descripteurs. Chaque entrée dans la table contient les attributs d'un fichier et un pointeur vers le premier bloc d'index. Tous les pointeurs d'un bloc d'index, à l'exception du dernier, pointent vers des blocs de données du fichier. Le dernier pointeur d'un bloc d'index plein pointe vers le bloc d'index suivant.
- Taille d'un pointeur = 32 bits.

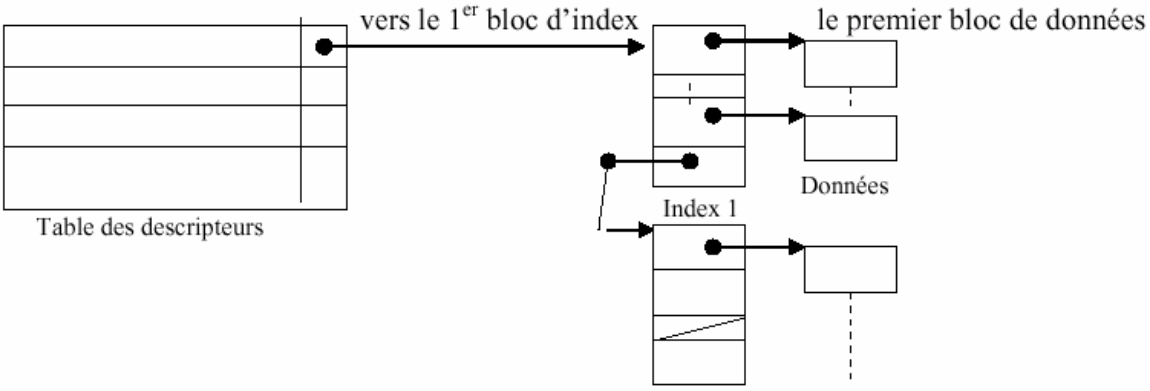


Figure 1 – Illustration d'un système de fichier

(a) Calculez la taille maximale en nombre de blocs que pourrait avoir un fichier en précisant les nombres de blocs d'index et de données.
(b) Déterminez le nombre de blocs qu'il faut lire à partir du disque pour pouvoir accéder à une donnée qui se trouve dans le 9000ième bloc de données du fichier.
Vous pouvez supposer qu'il y a K pointeurs par bloc d'index, la table des descripteurs est déjà chargée en mémoire, et les blocs d'index et de données ne sont pas encore chargés en mémoire.

2) Considérez les systèmes d'exploitation de la famille Unix.
(a) Expliquez la raison pour laquelle ces systèmes utilisent, en réalité, une structure de répertoire en forme de graphe acyclique plutôt qu'une structure de répertoire arborescente.

(b) Dans ces systèmes, tous les attributs de fichier sont décrits dans l'i-nœud (inode) du fichier sauf le nom.
(i) Expliquez pourquoi le nom n'est pas dans l'i-noeud.
(ii) Où sont donc sauvegardés les noms de fichiers?
(iii) Expliquez pourquoi la création de liens physiques sur des répertoires est interdite.

3) Expliquez la différence fondamentale qui existe entre une opération d'E/S par la technique du polling (i.e. scrutation) et une opération d'E/S par la technique des interruptions.

Exercice 5 :

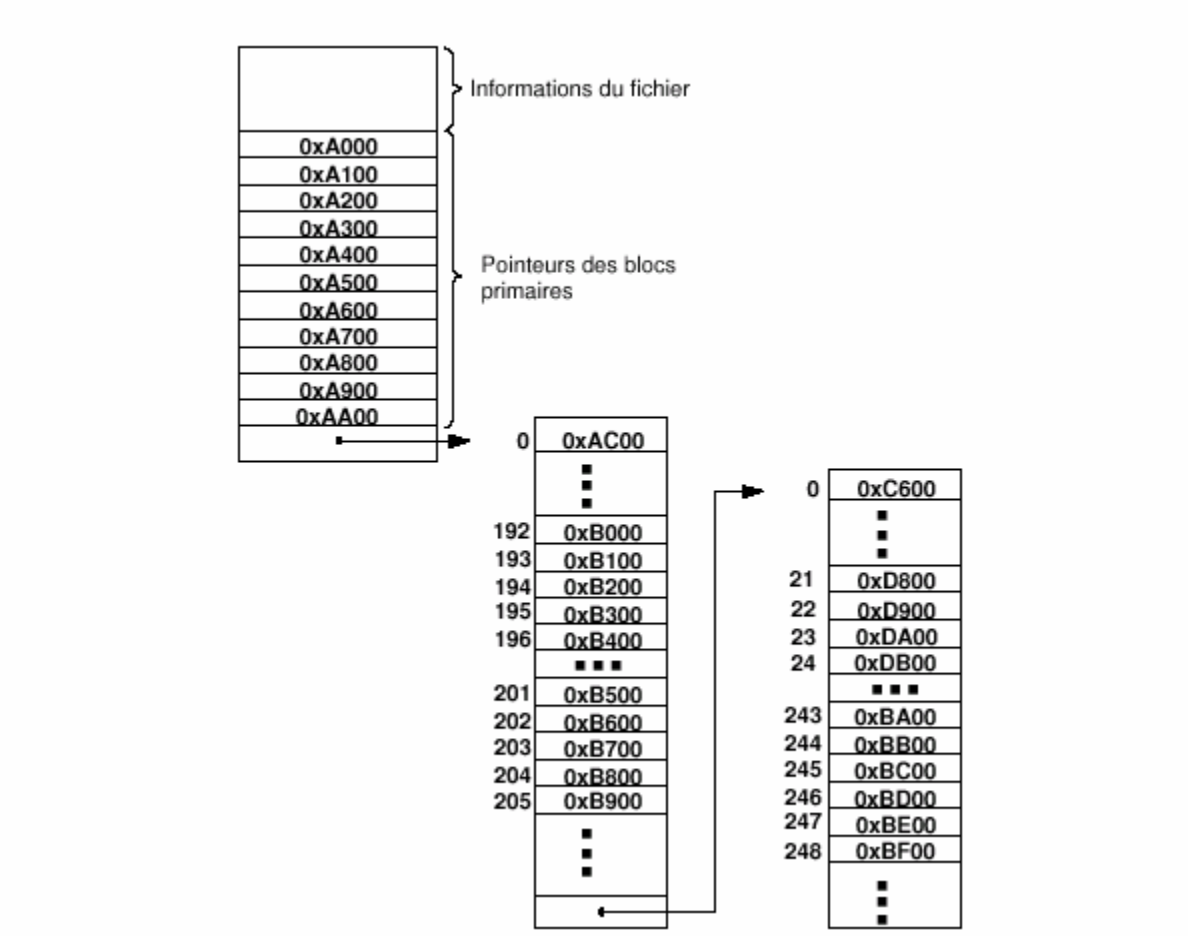
Dans le nouveau système de fichiers du dernier agenda électronique super puissant qui utilise PlamOS©, les concepteurs ont décidé d'utiliser une structure de fichiers semblable aux i-noeuds de BSD/SystemV.

La structure choisie, le plam-noeud©, ressemble à un i-noeud avec quelques différences. Ce plam-noeud©, d'une taille de 16 octets, est représenté comme suit :

- Trois champs de 8 bits chacun pour l'information sur le fichier,
- Onze pointeurs de blocs primaires,
- Un pointeur pour une indirection simple sur un bloc dont le dernier pointeur fait une seconde indirection simple.

Sachant que les blocs sur un PlamOS© sont de 1Koctets et que les adresses (pointeurs) sont codées sur 16 bits,

a) Quelle sera la taille maximale d'un fichier en octets ?
b) Dans quel bloc du disque dur retrouve-t-on le 209921e octet du fichier représenté par le plam-noeud© de la figure 2 ?



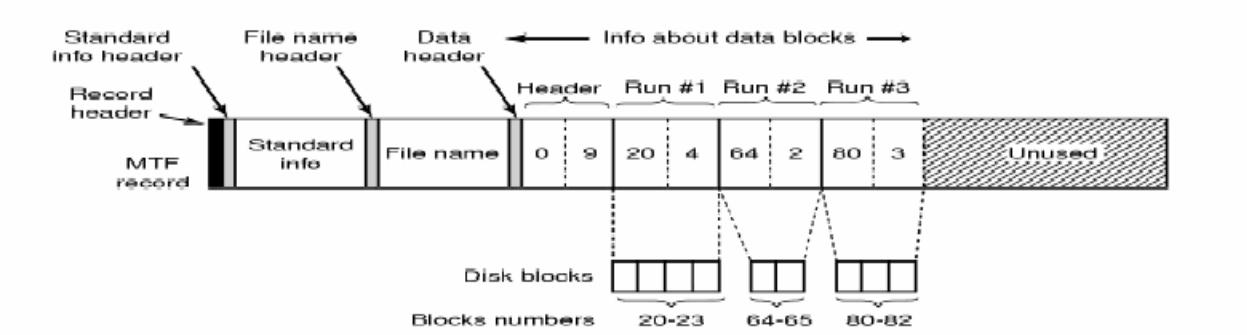
Exercice 6 :

Considérez le système de fichiers d'UNIX standard. La commande mv suivante permet de déplacer le fichier path1 vers le répertoire path2 :
mv path1 path2
où path1 et path2 sont respectivement les chemins d'accès d'un fichier et d'un répertoire.
1- Supposez que path1 et path2 appartiennent à un même système de fichiers et que vous avez les permissions nécessaires pour effectuer un tel déplacement.
Expliquez comment implanter la commande précédente au moyen des appels systèmes link, unlink et éventuellement d'autres (donnez sous forme de commentaires les différentes étapes).
2- Supposez maintenant que path1 et path2 appartiennent à deux systèmes de fichiers différents. Peut-on procéder de la même manière qu'en 1-. Expliquez pourquoi?

Exercice 7 :

Considérez le système de fichiers NTFS (version simplifiée). Ce système gère des partitions dont la taille peut atteindre 2^{64} octets. L'allocation se fait par blocs de taille fixe, encore appelés

*cluster*s, les numéros de clusters étant représentés sur 64 bits. L'espace libre est représenté par un fichier *bitmap* qui, à raison de 1 bit par cluster, définit son état d'allocation.
Une table, appelée MFT (Master File Table), contient un ou plusieurs enregistrement(s) par fichier existant sur le disque. Cet enregistrement, de la taille d'un cluster, a la structure suivante pour un fichier nécessitant un seul enregistrement :



On gère une partition de 2 Go par ce système, en utilisant une taille de cluster de 4 Ko.
a) **Déterminez** le nombre de clusters ainsi que la taille du fichier bitmap.
b) **Que contient le champ** « Header » qui précède les « Run »? À quoi sert-il ?