Systèmes de fichiers

Exercice 1. Questions de cours

1. Quels sont les éléments constitutifs de la structure d'un fichier ? Expliquez leur rôle.
2. Quelles sont les opérations de bas niveau offertes par l'OS (sous forme d'appels systèmes) pour manipuler les fichiers ?
3. Qu'est-ce que le Master File Table ?
4. Quel sont les principaux inconvénient de l'allocation chainée ?

Exercice 2. Tailles de fichiers On considère un système de fichiers tel que l'information concernant les blocs de données de chaque fichier est accessible à partir du FCB de celui-ci (comme sous UNIX). On supposera que :

- le système de fichiers utilise des blocs de données de taille fixe 1k (1024 octets) ;
- le FCB 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.

1. Quelle est la plus grande taille de fichier que ce système de fichiers peut supporter ?
2. On considère un fichier contenant $100,000$ octets. Combien de blocs de données faut-il (au total) pour représenter ce fichier sur disque ?
3. Combien de blocs doivent être chargés depuis le disque afin d'accéder à l'octet $1,000,000$ d'un fichier ?

Exercice 3. Système de fichier On considère un système de fichier utilisant différentes méthodes afin d'indexer les blocs physiques composant les fichiers. Le volume physique est organisé en blocs de 512 octets et les numéros de blocs sont représentés sur 32bits.

Le FCB d'un fichier contient 7 pointeurs directs et 1 pointeur vers un bloc d'index. Les blocs d'index sont chainés, dans chacun d'entre eux le premier pointeur indique quel est le prochain bloc d'index ou contient la valeur 0 indiquant la fin de la chaine.

Afin d'optimiser le stockage des très petits fichiers, ceux-ci peuvent-être stockés directement dans le bloc du FCB. En effet, pour un ficher n'occupant que quelques octets, il est possible de le stocker dans l'espace normalement utilisé pour les pointeurs vers des blocs physiques.

1. Quelle est la taille maximale d'un volume physique supportée par ce système de fichiers ?
2. Quel avantage y a-t-il à utiliser un chainage plutôt que des niveaux d'indirection supplémentaire.
3. Combien de blocs sur le volume physique sont occupés par un fichier de 30 octets ?
4. Combien de blocs sur le volume physique sont occupés par un fichier de 1Mo ?
5. Combien de blocs doivent être chargés depuis le disque afin d'accéder à l'octet $1,000,000$ d'un fichier ?

Exercice 4. Le système ext4fs

Nous allons considérer une version un peu simplifiée d'ext4, un système de fichier très répandu sous Linux. Dans ce système, le volume (par exemple une partition d'un disque dur) est découpé en blocs de 4ko. Comme vu dans le cours, et par analogie avec les disques durs, nous parlerons de *secteurs* pour désigner les emplacement de 4ko sur le volume dans lequels sont placés les *blocs* de fichier.

L'allocation des blocs dans un système de fichier extfs4 se fait de manière indexée en utilisant ce qu'on appelle un *extent*. Un *extent* indique une suite de blocs de fichier qui sont stockés les un à la suite les un des autres sur le volume. Les *extent* font 12 octets organisés de la manière suivante :

- sur les 4 premier octets : le numéro du premier bloc de fichier couvert par cet *extent* ;
- sur les 2 octets suivants : la taille, en nombre de blocs, de cet *extent*[1] ;
- sur les 6 derniers octets : le numéro du secteur sur le volume ou commence cet *extent*.

Exemple : Prenons un fichier de 15ko, et qui a donc 4 blocs de fichier (le dernier bloc ayant 1ko non utilisé) et supposons que ses trois premiers blocs sont placés dans les secteurs 1000, 1001 et 1002 du volume et que son quatrième bloc est sur le secteur 42 du volume. Il suffit de deux *extents* pour indiquer où se trouve la totalité du fichiers sur le volume :

(0, 3, 1000) : trois blocs de fichier à partir du bloc 0 sont stockés à partir du bloc de volume 1000.
(3, 1, 42) : l'unique bloc de fichier 3 est stocké sur le bloc de volume 42.

Liste d'extents : Pour indexer plusieurs parties non-contiguës d'un même fichier, on utilise une *liste d' extents*. Celle-ci est composée d'un *extent_header* sur 12 octets, qui indique notamment le nombre d'*extents* dans la liste, puis des *extents* eux même.

1. Quelle est la taille maximale d'un volume, en blocs et en octets ?

2. Quelle est la taille maximale d'un fichier, en blocs et en octets ?

3. Le *File Control Block* d'un fichier en ext4 contient une *liste d'extents* sur 60 octets (la liste contient donc au plus 4 *extents* en plus du *header*).

Quelle est la taille maximale (en blocs ou en octets) d'un fichier dont tous les *extents* sont stockés de cette manière, uniquement du FCB ?

Indexation indirecte : En pratique, on procède à une indexation à plusieurs niveaux : la liste d'extents du FCB peut pointer soit directement sur des blocs de données (comme nous l'avons vu à la question précédente), soit sur des blocs contenant à leur tour une liste d'*extents* :

- Si la liste d'*extents* contenue dans le FCB pointe directement vers des blocs de fichier, on parle d'accès direct (c'est le cas que nous avons vu à la question précédente) ;
- si elle pointe vers des listes d'*extents* qui, eux, pointent vers des blocs de fichiers, on parle d'accès indirect de niveau un ;
- Si elle pointe vers des listes d'*extents* qui pointent à leur tour vers des listes d'*extents* qui pointent vers des blocs de fichiers, on parle d'accès indirect de niveau 2 ;
- Et ainsi de suite...

Le niveau d'accès est indiqué dans le header de la liste d'extents sur l'inode. **Notez bien que lorsqu'un bloc contient une liste d'*extents*, ceux-ci occupent tout le bloc**.

4. Combien d'*extents* peut-on mettre dans une liste qui occupe un bloc complet ?

5. En vous aidant des réponses aux questions précédentes, quelle est la taille maximale (en blocs ou en octets) d'un fichier indexé en accès indirect de niveau un ?

6. De combien de niveaux d'indirection a-t'on besoin dans le pire des cas pour stocker les plus gros fichiers possibles (dont la taille a été calculée à la deuxième question) ?

[1] Dans la vraie spécification de ext4 il y a une petite différence ici.