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 ?

Correction: *Chaque bloc de données peut contenir 256 pointeurs (numéros de blocs). Il faut :*

$12 \times 1\text{k}$ *pour l'accès direct,*

$256 \times 1\text{k}$ *pour l'indexation simple,*

$256 \times 256 \times 1\text{k}$ *pour l'indexation double,*

$256 \times 256 \times 256 \times 1\text{k}$ *pour l'indexation triple.*

Au total $16,843,020\text{ko}$ *soit un peu plus de* 16Go.

2. On considère un fichier contenant $100,000$ octets. Combien de blocs de données faut-il (au total) pour représenter ce fichier sur disque ?

Correction: *Si on considère un fichier contenant* $100,000$ *octets, on voit que :* $100,000 = 97 \times 1024 + 672$. *Il faudra donc 98 blocs pour conserver les données de ce fichier.*

Le FCB ne dispose que de 12 pointeurs directs, il va donc falloir utiliser des blocs supplémentaires $98 - 12 = 86$ *pour conserver le reste des données. Comme* $86 < 256$, *ces blocs de données seront accessibles à partir du pointeur indirect simple du FCB et il faut un bloc supplémentaire pour stocker ces pointeurs.*

Un bloc est aussi nécessaire afin de stocker le FCB ce qui fait un total de $98 + 1 + 1 = 100$ *blocs.*

3. Combien de blocs doivent être chargés depuis le disque afin d'accéder à l'octet $1,000,000$ d'un fichier ?

Correction: *On refait un calcul similaire à celui de la question précédente :* $1,000,000 = 976 \times +576$. *Les 12 premiers blocs utilisent des pointeurs directs, le bloc qui nous intéresse est donc le* $977 - 12 = 965$*ème des blocs indirects.*

Le bloc d'indexation simple permet d'accéder au 256 blocs suivant, mais le notre n'est pas dans ceux la. On tente donc un accès au $965 - 256 = 709$*ème bloc via l'indexation double. Celle-ci permet d'acceder à* 256^2 *blocs, le bloc qui nous intéresse est donc bien accessible via l'indexation double.*

Il nous faut donc au total accéder au bloc du FCB, aux deux blocs qui contiennent l'indexation double et enfin au bloc de données, pour un total de 4 blocs accedé.

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 ?
   Correction: *Nombre de blocs multiplié par la taille des blocs :* $512 \times 2^{32} = 2^{41}$ *octets. Ce qui fait 2To.*
2. Quel avantage y a-t-il à utiliser un chainage plutôt que des niveaux d'indirection supplémentaire.
   Correction: *Il n'y a pas de limite à la taille maximale des fichiers.*
3. Combien de blocs sur le volume physique sont occupés par un fichier de 30 octets ?
   Correction: *L'espace utilisé par les 8 pointeurs fait 32 octets, il est donc possible de stocker ce fichier dans le FCB donc un seul bloc est nécessaire.*
4. Combien de blocs sur le volume physique sont occupés par un fichier de 1Mo ?
   Correction: *Il faut 2048 blocs pour les données. Les 7 premiers pointeurs sont stockés dans le FCB, et ensuite on peut en placer 127 par bloc d'index, il faut donc 17 blocs d'index. Au total :* $2048 + 1 + 17 = 2066$ *blocs.*
5. Combien de blocs doivent être chargés depuis le disque afin d'accéder à l'octet $1,000,000$ d'un fichier ?
   Correction: *Soit* $1,000,000 = 1953 \times 512 + 64$*, le bloc qui nous intéresse est donc le 1954ème. Les 7 premiers pointeurs sont stockés directement dans le FCB, nous cherchons donc le 1947ème dans la chaine de bloc d'index.*

   On peut placer 127 pointeurs par blocs d'index et $1947 = 15 \times 127 + 42$*, on cherche donc le 42ème pointeur du 16ème bloc d'index.*

   Donc, il faut chargé le bloc du FCB puis parcourir une chaine de 16 blocs d'index avant de chargé le bloc de donnée. Ce qui donne un total de 18 blocs chargés depuis le disque.

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 ?
   Correction: *Les numéros de secteurs du volume sont sur 6o = 48 bits, il y a donc au plus* 2^{48} *blocs de 4ko soit* $2^{48+12}o = 2^{60}o = 2^{20}$*Go ou* 2^{10}*To ou 1Eo.*

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

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

Correction: *Les numéros de bloc du fichier sont sur 4o = 32bits, il y a donc au plus 2^{32} blocs de 4ko soit $2^{44}o = 16To$.*

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 ?

Correction: *Chaque extent indique le nombre de blocs de l'extent sur 2 octets (16bits). Il peut donc indiquer au plus 2^{16} blocs. Pour 4 extents, nous avons donc au plus $4 \times 2^{16} = 2^{18}$ blocs indexés de 4ko chacun, soit en tout 1Go de données.*

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 ?

Correction: *Dans un bloc, on a $4096 = 341 \times 12$ octets (reste 4). On peut donc placer 340 extents de 12o en plus du header (de 12o aussi). Chaque extent addresse 2^{16} blocs de 4Ko. La taille maximale du fichier est donc de 340×2^{16} blocs, soit $22,282,240$ blocs ou 85 To.*

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 ?

Correction: *Chaque extent de l'inode peut référencer 2^{16} blocs. Chaque bloc d'extents peut référencer 340×2^{16} blocs de données. En tout, nous pouvons donc référencer $4 \times 340 \times 2^{16} \times 2^{16} = 85 \times 2^{36}$ blocs (ou $21,760To$ de données).*

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) ?

Correction: *Les plus gros fichier ont 2^{32} blocs (cf. question 2). Dans le meilleur cas, les données seront toutes à la suite les unes des autres et cela tient dans une indexation de niveau 1 comme vu ci-dessus. Dans le pire cas, ils sont tous séparés et il faut 2^{32} extents de 1 bloc, ce qui ne tient pas sur 1 seul niveau.*

Avec deux niveaux en utilisant uniquement des extents qui adressent un seul bloc dans les feuilles, nous avons $4 \times 340 \times 2^{16} \times 340 \times 2^{16}$ blocs adressés, ce qui est suffisant pour nos 2^{32} blocs.