Ecole Nationale des Sciences de l'Informatique

Correction TD SGF

Exercice 1 :

● On rappelle que sous Unix, un fichier est représenté de façon interne par la structure de données i-node. Cette structure comprend:
– un ensemble de 10 pointeurs directs qui pointent vers des blocs contenant les premières données du fichier
– un pointeur indirect simple,
– un pointeur indirect double,
– et enfin un pointeur indirect triple (un degré de plus d'indirection).
● En supposant que la taille d'un bloc est de 512 octets et que la taille d'un index est de 4 octets, indiquer (en nombre de blocs et en octets) :
– 1) la taille minimale d'un fichier ?
– 2) la taille maximale d'un fichier ?

Correction :
1) Un fichier une fois créé va occuper au minimum 01 bloc ; sa taille est au moins égale à 512 octets.
2) Avec 128 entrées pour chaque table de pointeurs et
512 octets par bloc, cela donne des fichiers d'au maximum 10 + 128 + 128*128 + 128*128*128= 2.113.674 blocs soit une taille très confortable de 512*2.097.152 = 1.082.201.087 octets.

Exercice 2 :

Considérons un disque dur ayant une capacité de 256 Go et un système de gestion de fichier utilisant FAT-32
1) Donner la taille minimale de bloc physique en Kilo octets pour indexer tout l'espace disque.
2) Quelle est alors la taille minimale d'un fichier dans un tel système ?
3) Calculer le nombre de blocs nécessaires pour stocker la table FAT sur le disque.
4) Pour la mémorisation de blocs libres de cette unité disque, la méthode table de bits (bitmap) est utilisée. Quelle est la taille de la table de bits en blocs ?
5) Vous décidez de formater ce disque dur en FAT-16 en choisissant des blocs physiques de 32 Koctets. Expliquer pourquoi il est déconseillé fortement de faire cette opération ?
Correction :

1) • Taille du disque = 256 Go = $2^8 * 2^{30} = 2^{38}$octets.
• Une entrée de la FAT-32 a 28 bits. Nombre d'unité d'allocations (nombre d'entrée possibles) = 2^{28}
• Taille minimale de bloc physique = 1 bloc physique = $2^{28}/2^{28} = 2^{10}$ octets = 1 Koctets
2) ● Taille minimale d'un fichier = 1 bloc physique = 1 Koctets
3) ● Taille de la table FAT = 2^{28} * 28 bits.
● Nombre de blocs nécessaires = 2^{28}* 28 /8*2^{10}= 2^{28}* 28 /2^{13}= 28 *2^{15} blocs
4) ● Taille de la table de bit = nombre de blocs du disque = 2^{28} bits.
● Taille de la table en bloc =2^{28}/ 2^{13}= 22^{15} blocs
5) taille max = $2^{16} * 2^{15}$= 2^{31}= 2 GB <<< 256 Go

Exercice 3 : (Exam 2018)

Nous allons considérer une version un peu simplifiée d'ext4 (utilisé sur Android), un système de fichier très répandu sous GNU/Linux. Dans ce système, le volume (par exemple une partition d'un disque dur) est découpé en blocs de 4 Ko. Comme vu dans le cours, et par analogie avec les disques durs, nous parlerons de secteurs pour désigner les emplacements de 4Ko sur le volume dans lequel 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 disque **(voir figure ci_dessous)**.

Extent

Numéro du 1er bloc Logique du fichier (4 octets)	Taille (2 octets)	Numéro du 1er bloc Physique (6 octets)

Les extents font 12 octets organisés de la manière suivante :

- sur les 4 premier octets : le numéro du premier bloc logique du fichier
- sur les 2 octets suivants : la taille, en nombre de blocs contigus, de cet extent
- sur les 6 derniers octets : le numéro du bloc physique du disque ou commence cet extent

Exemple

Prenons un fichier de 15 Ko, et qui a donc 4 blocs de fichier et supposons que ses trois premiers blocs sont : 1000, 1001 et 1002 du volume et que son quatrième bloc est le bloc 42 du volume. Il suffit de deux extents pour indiquer où se trouve la totalité du fichier sur le disque :

- (0, 3, 1000) : les trois premiers blocs logique (0, 1, 2) du fichier sont stockés à partir du bloc physique n°1000.
- (3, 1, 42) : le dernier bloc (bloc n°3) du fichier est stocké sur le bloc physique n°42.

I. Questions Diverses

1) Quelle est la taille maximale que peut avoir ce disque, en blocs et en octets ?
2) Quelle est la taille maximale d'un fichier, en blocs et en octets ?
3) 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.

L'inoeud d'un fichier en extfs 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 dans l'inode ?

II. Indexation indirecte

En pratique, on procède à une indexation à plusieurs niveaux : la liste d'extents de l'inode 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 l'inode, 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 (ou simple indirection).
- 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 (ou double indirection) ;
- 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.

Questions

1) Combien d'extents peut-on mettre dans un bloc complet ? Une liste commence nécessairement par un header. Les quelques octets restant sur le bloc ne sont pas perdus ; ils servent à faire des checksums.

2) 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 ?

3) 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 :

Questions Diverses

1) Les numéros de bloc du volume sont sur 6 o = 48 bits, il y a donc au plus 2^{48} blocs de 4 Ko soit 2^{48+12}o = 2^{60}o =2^{20}Go = 2^{10}To =1Eo.

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

3) Chaque extent indique le nombre de blocs de l'extent sur 2 octets (16 bits). 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 4 Ko chacun, soit en tout 1 Go de données.

Indexation indirecte

1) Dans un bloc, on a 4096 = 341 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=22282240 blocs ou 85 To.

2) 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 21760 To de données).

3) Les plus gros fichiers 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.