F.S

Questions principales

Décrivez la structure d'une partition formatée en F.A.T Le schéma ci-dessous représente une partition formatée en F.A.T

```
-----------------------------------------------
| BA | FAT | Copie FAT | Tableau de clusters |
-----------------------------------------------
```

- BA = Boot Area[1]
- FAT = Index

Le tableau de clusters est constitué de clusters de même taille. Un cluster ne peut contenir qu'un seul fichier.

1 cluster == n * secteur défini[2]

La FAT est l'index du tableau de clusters, il est chargé en RAM au démarrage pour des raisons de performance. La FAT contient autant d'entrée que de clusters présent dans le tableau de clusters. Cependant une entrée n'occupe pas nécessairement un cluster. La FAT contient l'adresse des clusters. Elle fait le lien entre les différents clusters d'un fichier.

Une entrée dans la fat peut avoir plusieurs valeurs :

- 0 si le cluster est libre [3]
- Une valeur positive pour désigner l'adresse du cluster suivant.
- -1 si c'est le dernier cluster.
- -2 si le cluster est défectueux. Comment savoir si un cluster est défectueux ? On écrit dedans, on lit ce qu'on a écrit, si les valeurs sont différentes, alors le cluster est défectueux.

Le répertoire racine se trouve trouve toujours au cluster 0 dans une partition formatée en FAT16. En FAT32, le cluster est donné par le MBR.

Les **répertoires en FAT sont des fichiers** particuliers contenant les métadonnées des fichiers présent dans le répertoire. Pour chaque fichier, le répertoire possède un descripteur de 32 bytes.

Les champs d'un descripteur :

[1] BA == BR == Boot Record
[2] cluster == bloc
[3] ne veut pas dire que le cluster est vide, seulement qu'il est libre

- Nom : permet de connaître le nom du fichier.
- Cluster : l'adresse du premier cluster du fichier.
- Dates : dates de modification, création, dernier accès.
- Extension : l'extension du fichier
- Taille : la taille du fichier.[4]

Un répertoire a donc toujours au moins deux descripteurs : le descripteur du répertoire courant et le parent (sauf la racine qui n'a pas de parent).

Avantages de la FAT :

- Très simple à intégrer.

Inconvénients de la FAT :

- Lourd en mémoire si la partition est trop grande.[5]
- Si le nom est trop long, il peut être étalé sur plusieurs entrées.
- Pas de droits d'accès.
- Fragmentation externe, les clusters ne sont pas toujours contigus lorsqu'on écrit et efface beaucoup.
- Fragmentation interne, un cluster ne peut contenir qu'un seul fichier, si celui-ci n'est pas remplit, on perd de la place.

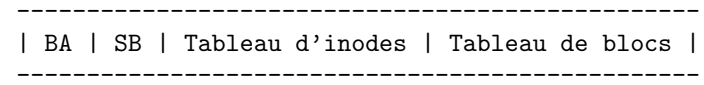
Voir les questions secondaires FAT.

Décrivez la structure d'une partition formatée en EXT/EXT2 EXT :

En EXT, un mini-disque (une partition) est composée de quatres zones :

- Boot Area : Contient les informations sur le démarrage
- Super-bloc : Contient les informations sur le mini-disque
- Tableau d'inodes : contient les inodes (métadonnées des fichiers)
- Tableau de blocs : contient les blocs (les données)

[4] 32Bits !, raison de la taille max d'un fichier de 2^{32}
[5] #bloc * taille entrée FAT



Un fichier contient une inode. Pour accéder à un fichier, il faut connaître l'inode, cet inode comprend :

- Nom du propriétaire
- Droits d'accès
- Dates
- Taille
- Type (fichier / dossier)
- Liste de tous les blocs du fichiers

Le chaînage est donc fait dans l'inode lui même. La liste de tous les blocs contient 10 pointeurs de blocs direct, et 3 pointeurs avec indirection.

- Le 11ème pointeur contient un seul niveau d'indirection, il pointe vers un tableau de blocs qui pointe vers les blocs de données.
- Le 12ème pointeur contient deux niveaux d'indirection, il pointe vers un tableau de blocs qui pointent eux même sur un tableau de blocs qui pointent sur les blocs de données.
- Le 13ème pointeur contient trois niveaux d'indirection. il pointe vers un tableau de blocs qui pointent eux même sur un tableau de blocs qui pointent eux même sur un tableau de blocs qui pointent vres les blocs de données.

Détaillez comment l'OS permet de découper un disque en plusieurs partitions (primaire, logique, étendue). Quelles sont les limites de cette technique ? Comment évolue-t-elle ? Quels sont les outils utilisés pour gérer ces partition (mkfs, fdisk, mount, unmount, df, /dev, ...)

Questions secondaires

(FAT) Détaillez comment l'OS retrouve un fichier, ajoute des données à ce fichier, efface ce fichier Exemple pour retrouver un fichier : int h = open("/home/user/f1");

- En FAT16 : Le cluster 0 contient la racine. Dans la liste des descripteurs présent dans le répertoire racine, l'OS cherche un descripteur avec le nom "user". Si il le trouve, il cherche le descripteur avec le nom "f1" dans user. Dans le descripteur de f1, on a le premier cluster du fichier ainsi que sa taille. Les clusters suivant sont chainés et sont retrouvés grâce à la FAT.
- En FAT32 : Même principe qu'en FAT16, sauf que le répertoire racine est retrouvé grâce au MBR.

Supprimer un fichier : On met 0 dans la FAT aux clusters correspondant au fichier.

Ecrire dans un fichier : Il est très simple d'écrire dans un fichier, il suffit de changer le chainage en fat si on utilise de nouveaux clusters.

(FAT) Détaillez les situations d'incohérence et montrer comment l'OS récupère cette situation. Lors d'un arrêt brutal du système, la FAT peut comporter des erreurs par rapport au contenu des clusters. Le système analyse le disque :

- Si la suite d'un fichier est un cluster vide, on remplace l'adresse par -1.
- Si on trouve un cluster qui contient des données mais ne semble appartenir à aucun fichier, et qu'il n'est pas libre. L'OS dépose alors le fragment de fichier dans le dossier lost+found (il y crée un fichier avec l'adresse du premier cluster).

(EXT2) Détaillez comment l'OS retrouve un fichier, ajoute des données à ce fichier, efface ce fichier.

(EXT2) Détaillez les situations d'incohérence et montrer comment l'OS récupère cette situation

(EXT2) Détaillez le contenu d'un super-bloc et l'utilité des champs qui s'y trouvent

(EXT2) Détaillez le contenu d'un inode et l'utilité des champs qui s'y trouvent

(EXT2) Détaillez les appels système qui permettent d'utiliser le système de fichier (open, read, write, close, dir, dup, lseek, stat, ...) et comment l'OS implémente ces appels système (handle, TDFO, ...)

(EXT2) Détaillez comment l'OS mémorise les liens à l'aide d'exemple (soft, hard)

(EXT2) Détaillez la notion de fichier creux à l'aide d'un exemple (création, taille, occupation du disque)

(EXT-EXT2) Détaillez la structure d'une partition formatée en EXT et EXT2. Détaillez les avantages d'EXT2 et l'implémentation de ces avantages. Commen EXT2 a t-il évolué ensuite ?