EX04 – Systèmes de gestion des fichiers

Systèmes d'exploitation / Classes T-2adfg

– Solutions –

1. L'appel système «open» est-il absolument essentiel sous UNIX ? Quelles seraient les conséquences si'il n'existait pas ?

   Solution : Pour commencer, s'il n'y avait pas de «open», il serait nécessaire à chaque «read» de spécifier le nom du fichier à ouvrir. Le système devrait alors récupérer l'i-node correspondant, bien qu'il puisse se trouver dans le cache. Il se poserait rapidement le problème de savoir à quel moment retourner l'inode sur le disque. Il pourrait expirer. Ce serait peu commode, mais cela fonctionnerait.

2. Les systèmes qui supportent les fichiers séquentiels ont toujours une opération pour rembobiner les fichiers. Les systèmes qui supportent des fichiers à accès direct ou aléatoire en ont-ils aussi besoin ?

   Solution : Non. Pour relire un fichier, il d'effectuer un accès aléatoire à l'octet 0.

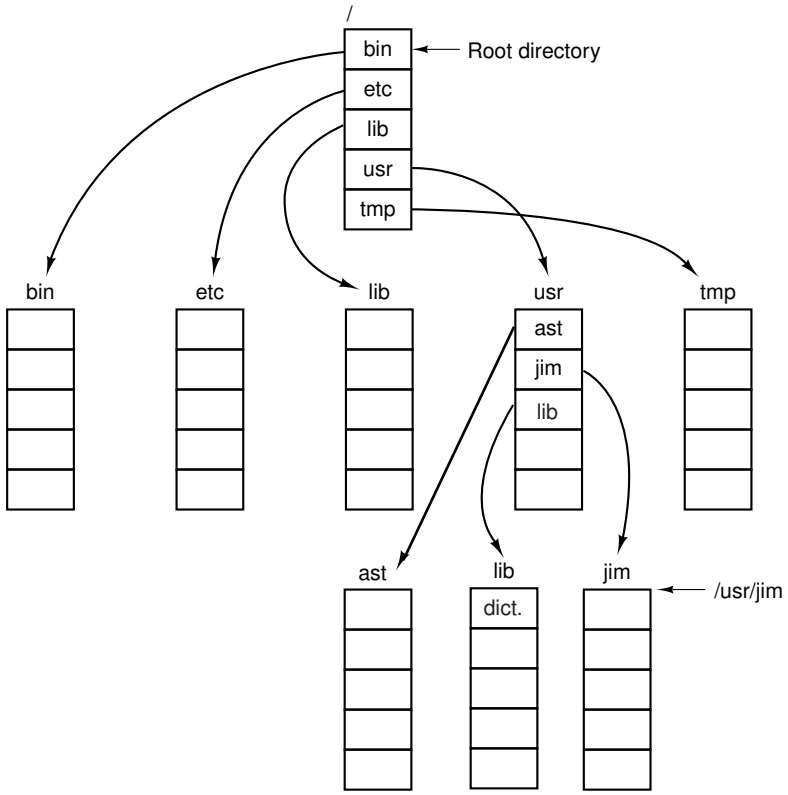
3. Certains systèmes d'exploitation proposent un appel système «rename» pour donner un nouveau nom au fichier. Existe-t-il une différence entre cette opération et celle qui consiste à faire une copie du fichier dans un nouveau fichier avec un nouveau nom et à ensuite effacer le premier fichier ?

   Solution : Oui. L'appel «rename» ne change pas l'heure de création ou l'heure de la dernière modification. En revanche, un nouveau fichier se voit attribuer l'heure actuelle comme heure de création et heure de la dernière modification. En outre, si le disque est plein, la copie peut échouer.

4. Un système d'exploitation élémentaire dispose uniquement d'un seul répertoire, mais lui permet d'avoir beaucoup de fichiers avec des noms longs. Peut-il simuler une sorte de système de fichiers hiérarchique ? Comment ?

   Solution : En utilisant des noms de fichiers comme «`/usr/ast/file`» (ou «`:usr:ast:file`»). Si cela ressemble à un chemin d'accès hiérarchique, il s'agit en réalité d'un nom unique contenant des barres obliques imbriquées (ou des deux-points).

5. Considérons l'arborescence de la figure ci-dessous. Si le répertoire de travail est `/usr/jim`, quel est le chemin d'accès absolu pour les fichiers dont le chemin d'accès relatif est `../ast/x` ?



Solution : Le composant point point (`..`) déplace la recherche vers `/usr` ainsi `../ast` le place dans `/usr/ast`. En conséquence, `../ast/x` est identique à `/usr/ast/x`.

6. Pour utiliser une allocation contiguë de fichiers sans pâtir des trous, il est possible de compacter le disque chaque fois qu'un fichier est effacé. Comme tous les fichiers sont contigus, la copie d'un fichier implique un déplacement du bras et un délai de rotation pour lire le fichier, puis un transfert à pleine vitesse. De même pour la réécriture du fichier. En supposant que le temps de déplacement est de 5 ms, que le délai de rotation est de 4 ms, que le taux de transfert est de 8 MB/s et que la taille moyenne d'un fichier est de 8 KB, combien de temps faut-ii pour lire un fichier depuis le disque vers la mémoire et le réécrire dans un autre emplacement du disque ? Avec ces mêmes paramètres, combien de temps prendrait la défragmentation (c'est à dire le compactage) de la moitié d'un disque de 16 GB ?

Solution : Il faut $9\,ms$ pour démarrer le transfert. Pour lire 2^{13} octets à un taux de transfert de 2^{23} octets/s, il faut $2^{-10}\,s$ ($0.977\,ms$), soit un total de $9.977\,ms$. Pour réécrire le fichier, on a besoin de $9.977\,ms$ de plus. Ainsi, pour copier un fichier, il faut $19.954\,ms$. Pour compacter la moitié d'un disque de 16 GB, on doit copier 8 GB de stockage, soit 2^{20} fichiers. À $19.954\,ms$ par fichier, on obtient un total de $20'923\,s$, Soit 5.8 heures. Le compactage du disque après chaque suppression de fichier n'est donc pas une excellente idée.

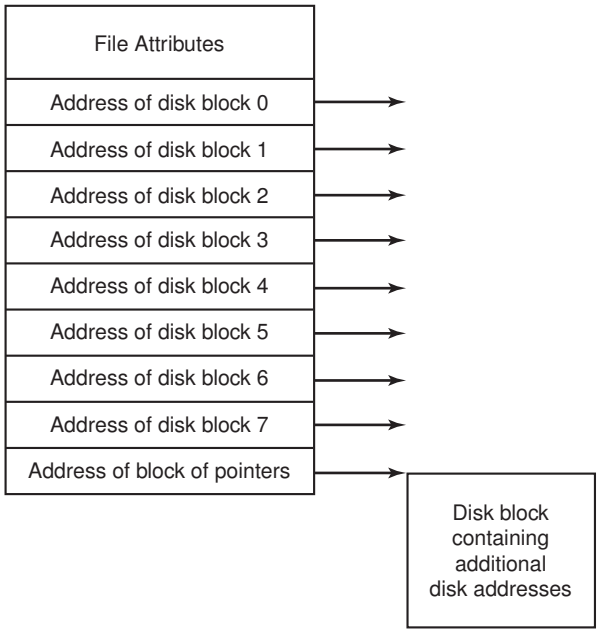
7. Certains dispositifs numériques grand public ont besoin de stocker des données, par exemple des fichiers. Indiquez un dispositif numérique récent qui nécessite l'enregistrement de fichiers et pour lequel une allocation contiguë serait une bonne idée.

   Solution : Un appareil photo numérique enregistre un certain nombre de photographies successives sur un média de stockage non volatil (par exemple, une mémoire flash). Lorsque l'on réinitialise l'appareil photo numérique, le moyen de stockage est vidé. Par la suite, on enregistre les images une à une successivement, jusqu'à ce que le média soit plein, puis on les copie sur un disque dur. Pour cette application, l'idéal est un système de fichiers contigu dans l'appareil photo numérique (par exemple, sur le média de stockage des images).

8. De quelle manière MS-DOS implante-t-il l'accès aléatoire aux fichiers ?

   Solution : Il trouve l'adresse du premier bloc dans l'entrée du répertoire, puis il suit la chaîne des pointeurs de blocs dans la FAT jusqu'à ce qu'il localise le bloc recherché. Il mémorise ensuite ce numéro de bloc pour le prochain appel système «read».

9. Considérons l'i-node de la figure ci-dessous. S'il contient 8 adresses directes de 4 octets chacune et si tous les blocs sont de 1024 octets, quelle est la taille maximale d'un fichier ?



   Solution : Le bloc indirect peut contenir 256 adresses disque (1024 / 4). Avec les 8 adresses disque directes, le fichier contient au maximum 264 blocs. Dans la mesure où chaque bloc a une taille de 1 KiB, la taille maximale d'un fichier est de 264 KiB.

10. Donnez un avantage des liens matériels par rapport aux liens symboliques et un avantage des liens symboliques vis-à-vis des liens matériels.

    Solution : Les liens matériels ne demandent pas d'espace disque supplémentaire, mais simplement un compteur dans leur i-node pour suivre leur nombre, alors que les liens symboliques demandent de l'espace pour stocker le nom du fichier pointé. Les liens symboliques peuvent pointer vers des fichiers situés sur d'autres ordinateurs, voire sur l'Internet, alors que les liens matériels sont limités aux fichiers de leur propre partition.

11. Le début d'une table de blocs libres juste après le formatage d'une partition disque ressemble à `1000000000000000` (le premier bloc est utilisé par le répertoire racine). Le système recherche toujours les blocs libres à partir du bloc qui a le plus petit nombre ; ainsi après l'écriture du fichier A, qui requiert 6 blocs, la table des blocs libres est de la forme : `1111111000000000`. Donnez la table après chacune des opérations suivantes :

    a) Le fichier B est écrit en utilisant 5 blocs.

    b) Le fichier A est effacé.

    c) Le fichier C est écrit en utilisant 8 blocs.

    d) Le fichier B est effacé.

    Solution : Le début de la table de blocs ressemble à :

    a) Après avoir écrit le fichier B : `1111 1111 1111 0000`

    b) Après avoir supprimé le fichier A : `1000 0001 1111 0000`

    c) Après avoir écrit le fichier C : `1111 1111 1111 1100`

    d) Après avoir supprimé le fichier B : `1111 1110 0000 1100`

12. Que se passe-t-il si la liste des blocs libres ou la table sont perdues à la suite d'un plantage ? Existe-t-il un moyen de les reconstruire ou le contenu du disque est-il définitivement perdu ? Développez votre réponse pour le système de fichiers UNIX d'une part, et pour celui d'un système FAT-16 d'autre part.

    Solution : Ce n'est pas un problème sérieux. La réparation est simple, mais prend du temps. L'algorithme de récupération consiste à créer une liste de tous les blocs qui se trouvent dans tous les fichiers et à utiliser le complément comme nouvelle liste libre. Sous UNIX, on obtient ce résultat en balayant tous les i-nodes. Dans le système de fichiers FAT, le problème ne peut se présenter, puisqu'il n'existe pas de liste libre. Toutefois, en existait une, il suffirait de parcourir la FAT à la recherche des entrées libres.

13. Nous avons étudié en détail les sauvegardes incrémentales. Sous Windows, il est facile de savoir quand sauvegarder un fichier, car chaque fichier possède un bit d'archivage. Mais ce bit est absent sous UNIX. De quelle manière les programmes de sauvegarde d'UNIX ont-ils connaissance des fichiers à sauvegarder ?

    Solution : Ils doivent conserver l'heure de la dernière sauvegarde dans un fichier sur le disque. À chaque Sauvegarde, on ajoute une entrée à ce fichier. Au moment de la sauvegarde, on lit le fichier et l'heure de la dernière entrée notée. Tout fichier modifié depuis cette heure est sauvegardé.

14. Un système de fichiers UNIX a des blocs de 1 KiB et des adresses disque sur 4 octets. Quelle est la taille maximale d'un fichier si les i-nodes contiennent 10 entrées directes et une simple indirection, une double indirection et une triple indirection pour chaque fichier ?

    Solution : L'i-node contient 10 pointeurs. La redirection simple contient 256 pointeurs, la redirection double contient 256^2 pointeurs et la redirection triple 256^3 pointeurs. En les additionnant, on obtient une taille de fichier maximale de 16 843 018 blocs, soit environ 16.06 GB.

15. Combien d'opérations disque sont nécessaires pour charger l'i-node du fichier `/usr/ast/courses/os/handout.t` ? Supposez que l'i-node du répertoire racine se trouve en mémoire, mais qu'aucun autre élément du chemin d'accès ne s'y trouve. Supposez aussi que tous les répertoires tiennent dans un bloc de disque.

Solution : Voici les lectures disque nécessaires :

- répertoire pour `/`
- i-node pour `/usr`
- répertoire pour `/usr`
- i-node pour `/usr/ast`
- répertoire pour `/usr/ast`
- i-node pour `/usr/ast/courses`
- répertoire pour `/usr/ast/courses`
- i-node pour `/usr/ast/courses/os`
- répertoire pour `/usr/ast/courses/os`
- i-node pour `/usr/ast/courses/os/handout.t`

Soit au total, 10 lectures disque.

16. Un RAID5 échoue si deux ou plus de ses disques tombent en panne dans un intervalle de temps limité. Disons que la probabilité qu'un disque tombe en panne à une heure donnée est de p ($0 \leq p \leq 1$. Quelle est la probabilité qu'un RAID5 avec k disques tombe en panne à une heure donnée ?

Solution : La probabilité de 0 panne, P_0, est de $(1-p)^k$.

La probabilité de 1 panne, P_1 est de $kp(1-p)^{k-1}$.

La probabilité d'une panne RAID est alors de $1 - P_0 - P_1$, soit $1 - (1-p)^k - kp(1-p)^{k-1}$.

		k				
		3	**4**	**5**	**6**	**7**
p	**0.001**	0.000003	0.000006	0.000010	0.000015	0.000021
	0.01	0.000298	0.000592	0.000980	0.001460	0.002031
	0.1	0.028000	0.052300	0.081460	0.114265	0.149694
	0.2	0.104000	0.180800	0.262720	0.344640	0.423283

17. Comparez les niveaux RAID de 0 jusqu'à 5 vis-à-vis des performances en lecture, des performances en écriture, de l'espace perdu et de la fiabilité.

Solution : Performances en lecture : les RAID de niveau 0, 2, 3, 4 et 5 permettent des lectures parallèles pour réaliser une demande de lecture. Mais le RAID de niveau 1 permet de réaliser simultanément deux demandes de lecture.

Performances en écriture : tous les niveaux ont des performances d'écriture similaires.

Espace perdu : il n'y a pas d'espace perdu au niveau 0, et 100% d'espace est perdu au niveau 1. Avec des mots de 32 bits et six disques de parité, l'espace perdu est d'environ 18.75% au niveau 2. Avec des mots de 32 bits, l'espace perdu est d'environ 3.13% au niveau 3. Finalement, en utilisant 33 disques aux niveaux 4 et 5, l'espace perdu est de 3.13%.

Fiabilité : elle est inexistante au niveau 0. Tous les autres niveaux peuvent accepter la panne d'un disque. De plus, aux niveaux 3, 4 et 5, une erreur aléatoire d'un bit par mot peut être détectée, alors qu'au niveau 2 une erreur aléatoire d'un bit par mot peut être détectée et corrigée.

18. Un fabricant de disques propose deux disques 5.25 pouces équipés de 10 000 cylindres. Le plus récent double la densité d'enregistrement linéaire de l'ancien. Quelles propriétés du disque sont meilleures sur le disque le plus récent et lesquelles sont identiques ?

    Solution : La capacité du disque et les taux de transfert sont doublés. Le temps de positionnement et le délai de rotation sont identiques.

19. Des requêtes de disque parviennent au pilote de disque pour les cylindres 10, 22, 20, 2, 40, 6 et 38, dans cet ordre. Un positionnement prend 6 ms par cylindre. Quel est le temps de positionnement nécessaire si l'on suit les méthodes suivantes :

    a) Premier arrivé, premier servi.

    b) Plus proche en premier.

    c) L'agorithme de l'ascenseur (déplacement vers le haut en premier).

    Dans tous les cas, le bras se trouve à l'origine au-dessus du cylindre 20.

    Solution :

    a) $10 + 12 + 2 + 18 + 38 + 34 + 32 = 146$ cylindres $= 876\,ms$

    b) $0 + 2 + 12 + 4 + 4 + 36 + 2 = 60$ cylindres $= 360\,ms$

    c) $0 + 2 + 16 + 2 + 30 + 4 + 4 = 58$ cylindres $= 348\,ms$