ED SYSTEME DE GESTION DE FICHIERS

CORRIGE

Gestion de fichiers Windows

1	L (libre)
2	7
3	L (libre)
4	L (libre)
5	10
6	11
7	6
8	L (libre)
9	13
10	15
11	EOF
12	L (libre)
13	17
14	EOF
15	14
16	L (libre)
17	18
18	20
19	L (libre)
20	EOF

Les numéros de chaque premier bloc de fichiers (fichier 1 : bloc n°2, fichier 2 : bloc n°5, fichier 3 : bloc n°9) sont mémorisés dans l'entrée de répertoire correspondant au fichier.

Politique de gestion des requêtes disque

On considère un disque composé de 300 pistes numérotées de 0 à 299. Le bras est couramment positionnée sur la piste 50.
La liste des requêtes (n°de piste cherchée) à servir donnée selon l'ordre d'arrivée est la suivante :

62, 200, 150, 60, 12, 120, 250, 45, 10, 100

FCFS
ordre de service : 62, 200, 150, 60, 12, 120, 250, 45, 10, 100
déplacement du bras : 12 + 138 + 50 + 90 + 48 + 108 + 130 + 205 + 35 + 90 = 906

SSTF
ordre de service : 45, 60, 62, 100, 120, 150, 200, 250, 12, 10
déplacement du bras : 5 + 15 + 2 + 38 + 20 + 30 + 50 + 50 + 238 + 2 = 450

SCAN sens initial montant
ordre de service : 60, 62, 100, 120, 150, 200, 250, 45, 12, 10
déplacement du bras : 10 + 2 + 38 + 20 + 30 + 50 + 50 + 205 + 33 + 2 = 440

Gestion de fichiers UNIX

Question A

Question A.1

Le processus lit 256 octets à la fois, donc il faut 4 demandes pour lire un bloc de 1024 octets en entier.
Lors de la première demande de lecture, le descripteur de fichier contient en `TABDIRECT[0]` le numéro du premier bloc de données. Il faut donc le lire, et transférer les 256 premiers octets (disons ceux de numéro 0 à 255) de ce bloc dans une zone du programme. Lors de la deuxième demande, il s'agit du même bloc, mais comme il n'y a pas de conservation dans un tampon des blocs, il faut le relire, et délivrer les octets 256 à 511. Pour la cinquième demande, il s'agit des octets 0 à 255 du deuxième bloc de données, c'est-à-dire, le bloc de numéro `TABDIRECT[1]`.

Question A.2

Les octets de la 41ième demande sont situés dans le 11ième bloc de données (car 41/4=10,25 donc on désire lire le premier quart de 256 octets du 11ième bloc de données). Il faut donc lire le bloc `INDIRECT_1` pour connaître le numéro du bloc de données qui nous intéresse (premier numéro de bloc dans ce bloc), et pouvoir lire celui-ci. Les octets de la 45ième demande sont situés dans le 12ième bloc de données. Il faut donc aussi lire le bloc `INDIRECT_1` pour connaître le numéro du bloc de données qui nous intéresse (deuxième numéro de bloc dans ce bloc), et pouvoir lire celui-ci. Il s'ensuit que chaque demande de lecture, à partir de maintenant entraînera deux accès disque (1 premier accès disque pour lire le bloc d'index INDIRECT_1, et second accès pour lire le bloc de données désiré).

Question A.3

Les octets de la 1065ième demande sont situés dans le 267ième bloc de données (car 1065/4=266,25, fait référence au 1er quart de 256 octets du 267ième bloc de données). Le 267ième bloc de données est le premier bloc de données du niveau INDIRECT_2. Il faut donc lire le bloc `INDIRECT_2`, puis celui dont le numéro est le premier dans le bloc d'index venant d'être lu, pour avoir le numéro du bloc de données qui nous intéresse (premier numéro de bloc dans celui venant d'être lu), et lire enfin le bloc de données. Les octets de la 1066ième demande sont situés également dans le 267ième bloc de données (car 1066/4=266,5, fait référence au 2e quart de 256 octets du 267ième bloc de données). En l'absence de mémorisation, il faudra relire les mêmes blocs que précédemment pour satisfaire la demande. On peut donc en conclure que dorénavant, 3 accès disque seront nécessaires pour satisfaire chaque demande (1er accès pour le bloc d'index INDIRECT_2, un 2ième accès pour le bloc INDIRECT_2_i et un dernier accès pour le bloc de données désiré).

Question A.4

Les octets de la 2089ième sont situés dans le 523ième bloc de données (car 2089/4=522,25, fait référence au 1er quart de 256 octets du 523ième bloc de données). Il faut donc lire le bloc `INDIRECT_2`, puis celui dont le numéro est le *deuxième* dans le bloc d'index venant d'être lu, pour avoir le numéro du bloc de données qui nous intéresse (premier numéro de bloc dans celui venant d'être lu), et lire enfin le bloc de données. Les octets de la 2090ième demande sont situés également dans le 523ième bloc de données (car 2090/4=522,5, fait référence au 2e quart de 256 octets du 523ième bloc de données). En l'absence de mémorisation, il faudra relire les

mêmes blocs que précédemment pour satisfaire la demande. On peut donc en conclure que 3 accès disque sont toujours nécessaires pour satisfaire chaque demande.

Question A.5

Les 40 premières demandes de lecture (correspondant au 10 premiers blocs de données) demanderont chacune un accès disque. Les 1024 suivantes (correspondant au 256 blocs de données du niveau INDIRECT_1) en demanderont chacune deux. Les demandes restantes (32768 - 1024 - 40 = 31704, correspondant au 7926 derniers blocs de données au niveau INIDRECT_2) en demanderont chacune trois. On a donc un total de 97200 accès disque (4*10 + 2*1024 + 3*31704), et un temps d'attente d'entrées-sorties de 3888 secondes (97200*0,040 s), soit un peu plus d'une heure. Notons que l'indirection de niveau 3 n'est pas utilisée puisque 2 niveaux permettent de mémoriser 65 Mo, alors que nous n'avons que 8 Mo.

Question B.1

Plusieurs raisons justifient que les tampons soient dans d'espace système, et soient ainsi partagés par tous les processus. Tout d'abord, mentionnons une raison d'économie: s'il fallait réserver des tampons pour chaque processus, il y aurait perte de place pour ceux qui n'accèdent à aucun fichier, et manque de place pour les autres. Le partage de l'espace permet d'attribuer les tampons lorsque nécessaires.

Cependant, la raison essentielle est liée aux accès simultanés à un même fichier par plusieurs processus. Si plusieurs processus cherchent à lire dans le même fichier, le partage permet de diminuer les accès disque, puisque une lecture d'un bloc dans un tampon partagé pour le compte d'un processus évitera la lecture de ce même bloc pour le compte d'un autre processus. Par ailleurs, si les tampons étaient propres aux processus, une écriture (opération `write`) par un processus sur le fichier, serait en fait une écriture dans ses propres tampons et son effet ne serait visible aux autres processus que plus tard, à un moment imprévisible. La technique de cache des blocs disques doit en fait être «transparente» aux utilisateurs. Les processus manipulent des flots par les opérations `read` et `write`. L'implantation de ces opérations doit respecter la «sémantique» attendue par leur spécification. L'effet sur le contenu du fichier doit être immédiatement visible par les autres processus (la cohérence des données est du ressort de la synchronisation entre les processus).

Question B.2

L'utilisation d'un algorithme LRU pour la gestion des tampons signifie que chaque accès à un tampon place celui-ci en tête de liste. Contrairement à la question A, le besoin d'une information située dans un bloc disque ne demande pas d'accès disque, si ce bloc est déjà présent dans un tampon. En reprenant le raisonnement de la question A, on constate que si la première demande, comme la cinquième, demande un accès disque pour obtenir le bloc, par contre la seconde n'en demande pas, puisque le bloc a été conservé dans un tampon. Plus généralement, les 40 premières demandes ne demandent que 10 accès disque pour obtenir les 10 blocs concernés qui tiennent amplement dans les 100 tampons.

Lors de la 41^{ième}demande, il faut lire le bloc pointeur `INDIRECT_1` dans un tampon, puisqu'une information qui y est contenue (le numéro du 1er bloc de données de ce niveau d'indirection) est nécessaire. Il faut ensuite lire le bloc de données lui-même. Notons que pour les 3 demandes suivantes, aucun accès n'est nécessaire, puisque les informations sont dans les tampons. Lors de la 45^{ième} demande, le bloc pointeur `INDIRECT_1` est déjà dans le tampon, et il est alors placé en tête de liste LRU. Le bloc de données est alors lu et placé devant lui dans cette liste. Plus généralement, après chaque demande de lecture (jusqu'à la 1064^{ième}

comprise), le bloc de données qui était concerné se trouve en tête de liste LRU, suivi immédiatement par le bloc `INDIRECT_1`. On peut donc en conclure que les 1024 demandes (41 à 1064) nécessitent 257 accès disque (1 accès pour le bloc `INDIRECT_1` et 256 accès pour les 256 blocs de données).
Lors de la 1065ième demande, il faut lire le bloc `INDIRECT_2`, puis le bloc pointeur dont le numéro est le premier dans le bloc `INDIRECT_2`, et enfin le bloc de données. Lors de la 1066ième demande, les mêmes blocs sont nécessaires, mais aucun accès n'a lieu, puisqu'ils sont présents dans les tampons. Ici encore, après chaque demande, on constate que le bloc de données est en tête de la liste LRU, suivi du bloc pointeur `INDIRECT_2_i`, suivi du bloc `INDIRECT_2`. Ces blocs pointeurs resteront en mémoire au moins jusqu'à la 2089ième demande. Les demandes 1065 à 2088 demanderont donc 258 accès disques (1 accès pour le bloc `INDIRECT_2`, 1 accès pour le bloc d'index INDIRECT_2_1 et 256 accès disque pour les 256 blocs de données).
Lors de la 2089ième demande, on a besoin du bloc `INDIRECT_2`, toujours présent en mémoire, puis du bloc pointeur INDIRECT_2_2 (deuxième référence indiqué dans le bloc `INDIRECT_2`), qui doit donc être lu, et enfin du bloc de données. Ces mêmes blocs seront nécessaires pour la demande 2090. Ici encore, on constate qu'après chaque demande, le bloc de données est en tête de la liste LRU, suivi par le bloc INDIRECT_2_2, suivi par le bloc `INDIRECT_2`. Les 1024 demandes suivantes (2089 à 3112) demanderont donc 257 accès disques (1 accès disque pour le bloc INDIRECT_2, 1 accès pour le bloc INIDRECT_2_2 et 256 accès pour les 256 blocs de données), et ainsi de suite.
En résumé, nous avons:

- 40 demandes avec 10 accès disques (10 premiers blocs de données au niveau direct)
- 1024 demandes avec 1+256 accès disques (256 blocs de données au niveau INDIRECT_2)
- 30720 (30*1024) demandes avec 1+30*(1+256) accès disques (7680 blocs de données au niveau INDIRECT_2 référencés par les blocs d'index INDIRECT_2_1 à INDIRECT_2_30 pleinement utilisés)
- 984 (32768-40-1024-30720) demandes avec 1+246 accès disques (246 derniers blocs de données du fichier au niveau INDIRECT_2 référencés par le bloc d'index INDIRECT_2_31 dont seules les 246 premières entrées sont utilisées).

Soit un total de 8225 accès disque, qui se décomposent en 8192 lectures de blocs de données et 33 lectures de blocs pointeurs de blocs (10 + (1 + 256) + (1 + (30*(1 + 256)) + 247 = 8225 accès disque). Ceci donne un temps d'attente de 329 secondes (8225*0,040s), ou 5.5 minutes. Par rapport à la question A, le nombre d'accès disque a été divisé par environ 12! On pouvait s'y attendre: en A, chaque demande en double indirection, c'est-à-dire, les plus nombreuses, demandait la lecture de 3 blocs, alors que maintenant il y a environ une lecture toutes les 4 demandes.
Notons que LRU permet de replacer en début de liste les blocs pointeurs utilisés, ce qui entraîne sur notre exemple que ces blocs ne sont lus qu'une seule fois. Avec l'algorithme FIFO, après avoir lu environ 100 blocs de données repérés par le même bloc pointeur, ce dernier serait chassé de la mémoire pour y être rappelé lors de la demande suivante. Il y aurait donc un peu plus d'accès disque.

Gestion de fichiers UNIX

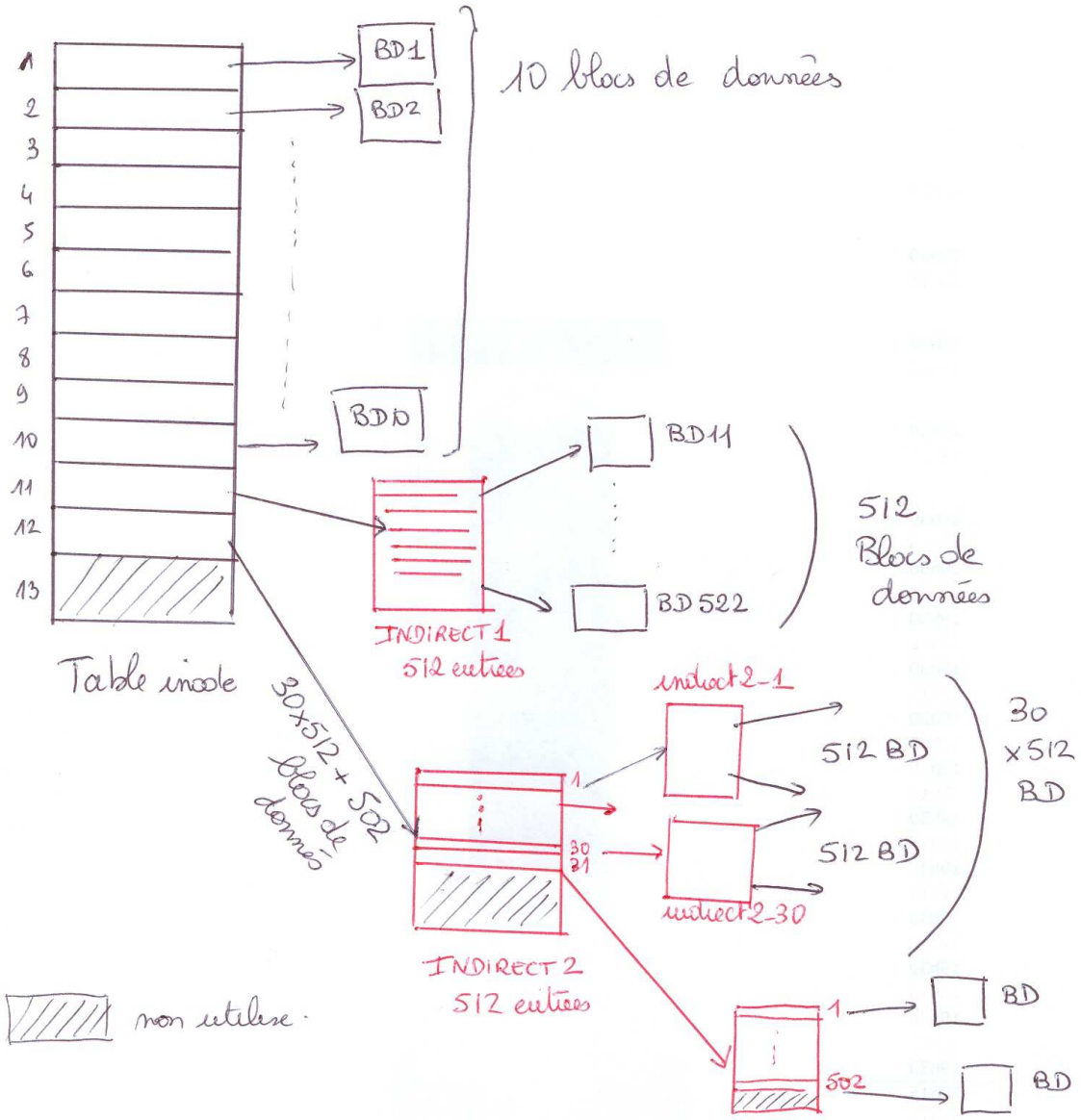
On considère un fichier de type UNIX. Le fichier a une taille de 16 Moctets. Les blocs disque sont de 1024 octets. Un numéro de bloc occupe 2 octets.

Question 1 — Quel est le nombre de blocs de données du fichier ?

Taille_fichier / Taille_bloc_disque = $2^4 * 2^{20} / 2^{10} = 2^{14}$ = 16384 blocs de données.

Question 2 — Combien comporte-t-il de blocs d'adresses ? Représentez sur un schéma l'allocation des blocs de données du fichier.

Un bloc d'adresse comporte : Taille_bloc_disque / Taille_numero_bloc = 1024 / 2 = 512 entrées.
Selon le dessin ci-dessous, l'allocation est réalisée comme suit :
10 blocs de données sont en accès direct pointés par les 10 premières entrées de la table de l'inode.
Il reste à allouer 16384 – 10 = 16374 blocs de données.
Le niveau INDIRECT_1 permet d'allouer 512 blocs. Il reste 16374 – 512 = 15862 blocs de données. On a à ce niveau un premier bloc d'adresses (ou d'index).
Le niveau INDIRECT_2 permet d'allouer au maximum 512^2 blocs de données ce qui est largement supérieur à 15862. Tous les blocs restants sont donc repérés à partir de ce niveau. Il faut déterminer jusqu'à quelle profondeur.
On peut écrire que 15862 = 512 * 30 + 502, nécessite donc 31 blocs d'index INDIRECT_2_i (30 blocs d'index INDIRECT_2_i pleinement utilisés et 1 dernier bloc d'index INDIRECT_2_31 avec uniquement les 502 premières entrées d'utilisées). Nous avons donc à ce niveau 32 blocs d'index (31 blocs d'index INDIRECT_2_i + 1 bloc d'index INDIRECT_2).
Au total 33 blocs d'adresses (ou d'index) sont donc nécessaires.



Question 3 — Si le système dispose d'un cache disque, combien d'accès disque sont nécessaires pour lire les 600 premiers blocs de données du fichier.
En présence du cache, seule la première lecture d'un bloc donne lieu à un accès disque.

Comme indiqué dans la question précédente, les 600 premiers blocs de données du fichier sont répartis en : 10 blocs au niveau direct, 512 blocs au niveau INDIRECT_1 et 78 blocs au niveau INDIRECT_2.
Il faut donc 10 + (1 + 512) + (1 + 1 + 78) = 603 entrées sorties pour lire les 600 premiers blocs de données du fichier.