LIAM PAUL
10.000U +

Examen Intra

IFT 2245

23 avril 2024

Directives

— Vous avez droit à une page de notes (recto verso) écrite à la main.
— Calculatrice est autorisée.
— Répondez dans le cahier fourni **à l'exception de Question 1** que vous pouvez répondre sur cette feuille.
— Le pointage pour chaque question est entre parenthèses (total = 20).
— Les traductions en anglais sont en *italics*.
— Vous pouvez répondre aux questions en anglais ou en français.
— Notez clairement toutes les suppositions que vous faites.

Question 0. *Nom et matricule (1 point de bonus)*

Écrivez votre nom et votre matricule sur ce papier et sur votre cahier d'examen.

Question 1. *Questions à choix multiple (5 points)*

(i) (0.5 points) Dans un système de stockage RAID, quel est l'avantage de la *striping* de disque (cochez tout ce qui s'applique) ?

- ☐ Il augmente la fiabilité
- ☒ Il augmente le débit (*throughput*) de lecture
- ☒ Il augmente le débit (*throughput*) d'écriture
- ☐ Il économise de l'espace disque

(ii) (0.5 points) Combien de bits sont utilisés pour afficher les informations concernant les autorisations (*permissions*) sur un fichier sous Linux ?

A. 6
B. 10
C. 9
D. 64
E. Aucune de ces réponses

(iii) (0.5 points) Parmi les algorithmes de remplacement de page suivant, lesquels souffrent de l'anomalie de Balady (cochez tout ce qui s'applique) ?

- ☐ Optimal
- ☒ Aléatoire (*random*)
- ☒ *First-In-First-Out* (FIFO)
- ☐ *Least Recently Used* (LRU)

(iv) (0.5 points) Avec Docker, quelle est la différence entre une image et son conteneur associé (cochez tout ce qui s'applique) ?

- ☒ Le conteneur est actif (en cours d'exécution) mais l'image n'est pas
- ☐ L'image est actif (en cours d'exécution) mais le conteneur n'est pas
- ☒ Le couche sur le dessus du conteneur est inscriptible mais toutes les couches de l'image sont en lecture seule
- ☐ Le couche sur le dessus d'une image est inscriptible mais toutes les couches du conteneur sont en lecture seule

(v) (0.5 points) Mettez les étapes suivantes nécessaires pour lire un fichier avec l'instruction `read(index)` dans le bon ordre :

4 ☐ Lire les blocs de données
2 ☐ Trouver la bonne entrée dans la table des fichiers ouverts du système (*system-wide open-file table*)
3 ☐ Charger le bloc de contrôle de fichier (*file-control block*)
1 ☐ Trouver la bonne entrée dans la table des fichiers ouverts par processus (*per-process open-file table*)

(vi) (0.5 points) Parmi les options suivantes, quels sont les **avantages** d'avoir une **grande** taille de page (cochez tout ce qui s'applique) ?

- ☐ Moins de fragmentation
- ☒ Portée TLB plus grande (*larger TLB reach*)
- ☒ Table de pages plus petit
- ☐ Meilleure résolution d'E/S (Il nous suffit de transmettre que les données nécessaires)

(vii) (1 point) Vous recevez le code de Hamming (à parité paire calculée de gauche à droite) suivant :

1 0 1 1 0 1 0 0 1 0 1 0

Il y a une erreur dans l'un des bits. Lequel ? (Quelle est sa position dans la châine de bits ?)

A. 2
B. 10
C. 3
D. 5
E. Aucune de ces réponses

P1 : 1 1 0 0 1 1 → 0
P2 : 0 1 1 0 0 1 → 1
P4 : 1 0 1 0 0 → 0
P8 0 1 0 1 0 → 0

12 pointeurs directs : 12 × 1KB

simple indirect : 2^8 1KB

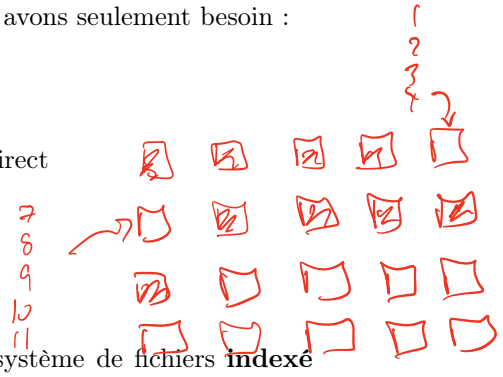
(viii) (1 point) Considérons un système de fichiers **ext2** avec des inodes qui utilise un adressage **32-bit** et une taille de bloc de **1KB**. Pour accéder à un fichier d'une taille de **1MB**, nous avons seulement besoin :

A. des pointeurs directs

B. des pointeurs directs et le pointeur simple indirect

C. des pointeurs directs, le pointeur simple indirect, et le pointeur double indirect

D. de tous les pointeurs directs et indirects



Question 2. *Systèmes de fichiers (2 points)*

Dessine un système de fichiers (répertoire + disque de 20 blocs) pour un système de fichiers **indexé** contenant 2 fichiers :

— `foo.txt` besoin de 4 blocs

— `bar.txt` besoin de 5 blocs

rep : foo.txt ... 5

bar.txt ... 6

Indiquer le structure du répertoire, quels blocs sont libres sur le disque et lesquels sont attribués à chaque fichier et montrer le contenu de chaque bloc d'index.

Question 3. *Mémoire en masse (5 points au total)*

(a) (2 points) Un disque a 120 pistes. Actuellement, la tête de lecture est sur la piste 70 en se déplaçant dans le sens de l'augmentation des numéros de piste (et faire des lectures dans ce sens). La séquence des requêtes de lecture de piste à venir est :

50, 20, 10, 90

Quel est **l'ordre des pistes visitées** et **la distance totale parcourue** par la tête de lecture pour les algorithmes d'ordonnancement de disques suivants :

i. C-LOOK 70 → 90 → 10 → 20 → 50 distance 140

ii. SCAN 70 → 90 → (120) → 50 → 20 → 10 distance 160

(b) (3 points) Compte tenu des informations suivantes :

— Taille d'un bloc est **4KB**

— Le disque tourne a une vitesse de **7200 révolutions par minute**

— Le taut de transfert est **1Mb/s** (N.B. b≠B)

— La tête de lecture se déplace à une vitesse de **20 pistes par ms**

Quel est le temps **total maximum** pour que les 4 demandes d'E/S soient complétées dans les cas i. et ii. dans la question (a) ?

$T_{io} = T_s + T_a + T_t$

$T_s^{max} = \frac{1}{7200} \cdot 60 \times 1000 = 8.3\,ms.$

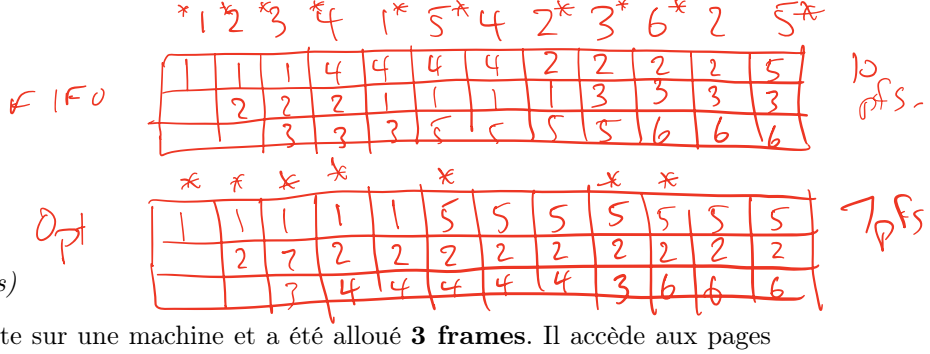
$T_a(i) = 7\,ms$

$T_a(ii) = 8\,ms.$

$T_t = \frac{4KB/bloc}{2^{10} KB/s \cdot \frac{1B}{8b}} \cdot \frac{1000\,ms}{s} = 31.25\,ms.$

$T_{io}(i) = 4(8.3) + 4(31.25) + 7 = 165.3\,ms$

$T_{io}(ii) = 4(8.3) + 4(31.25) + 8 = 166.3\,ms.$



Question 4. *Mémoire virtuelle (4 points)*

(a) (3 points) Un programme s'exécute sur une machine et a été alloué **3 frames**. Il accède aux pages suivantes dans l'ordre indiqué :

1 2 3 4 1 5 4 2 3 6 2 5

En présumant que toutes les *frames* sont vides au départ, indique le contenu des *frames* et le nombre total de *page faults* pour chacun des algorithmes de remplacement de page suivants :

i. FIFO (*first in first out*)

ii. Optimal

(b) (2 points) Considérons maintenant la même séquence et le même nombre de *frames* allouées (3) mais où tous les accès à l'intérieur d'un box sont des accès en écriture et le reste sont des accès en lecture :

[1] 2 3 [4] 1 5 4 [2] 3 [6] 2 5

(i) Nous allons concevoir un nouvel algorithme de remplacement de page dans lequel nous utilisons une politique FIFO mais donnons toujours la priorité au remplacement des pages qui ne sont pas marquées comme sales (*dirty*) (nous ne devrions jamais remplacer une page sale s'il y a une page qui ne l'est pas). Quel est le contenu des *frames* mémoire résultant de cet algorithme ?

(ii) Quel est le nombre d'E/S nécessaires à la suite de cet algorithme (où une E/S est un transfert d'un bloc de données entre la frame en mémoire vers le disque ou vice versa) ?

Question 5. *Memoire centrale (4 points)*

(i) (1 point) Considérez un système de pagination avec un *translational look-aside buffer* (TLB) où la table de pages est stockée dans la mémoire principale. Étant donné qu'un temps de recherche dans le TLB est $\epsilon = 20ns$, et que le temps d'accès à la mémoire principale est $m = 400ns$, trouvez le taux de réussite du TLB (*TLB hit ratio*) α ($0 \leq \alpha \leq 1$) pour que le temps d'accès effectif (*EAT*) pour accéder à la mémoire soit réduit à 80% par rapport au cas où il n'y a pas de TLB (EAT $= 0.8 \times 2m$).

Les détails suivants s'appliquent aux questions (b) et (d) : Étant donné un système avec un espace d'adressage de **32 bits**, une taille de page de **32KB** (2^{15}B) et avec la taille d'une entrée dans la table de pages de 32b :

(b) (1 point) Quelle est la taille (en *Bytes*) de la table de page à un niveau ?

(c) (1 point) Considérant que nous voulons que chaque table de pages dans un système de pagination hiérarchique s'inscrive dans un *frame* de mémoire (c'est-à-dire ne soit pas plus grande qu'une page), combien de niveaux de pagination sont nécessaires et comment l'espace d'adressage est-il divisé ?

(d) (1 point)) Considérant qu'un programme a besoin de **1 GB** (2^{30}B) d'espace mémoire virtuel, combien de tables de pages totales sont nécessaires pour la configuration hiérarchique décrite en (c).