Examen Final

IFT 2245

25 avril 2023

Directives

— Vous avez droit à une page de notes (recto-verso) écrite à la main.
— Calculatrice est autorisée.
— Répondez dans le cahier d'examen 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 prénom (1 point de bonus)*

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

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

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

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

Quel est le vrai byte (sans bits de parité) qui a été transmis ?

A. 1 0 0 0 1 0 0 0
B. 0 0 1 1 0 0 0 1
C. 1 0 0 1 1 0 0 1
D. 1 0 1 1 0 1 0 1
E. aucune de ces réponses

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

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

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

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

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

(iv) (0.5 points) De quelles manières pouvons-nous créer une nouvelle image Docker (cochez tout ce qui s'applique) ?

- [ ] Nous pouvons exécuter l'image (avec *docker run*)
- [ ] Nous pouvons le télécharger à partir d'un registre (avec *docker pull*)
- [x] Nous pouvons commettre un conteneur en cours d'exécution (avec *docker commit*)
- [x] Nous pouvons construire l'image à partir d'un Dockerfile (avec *docker build*)

(v) (0.5 points) De quelles manières pouvons-nous rendre les données disponibles à partir d'un conteneur Docker en cours d'exécution (cochez tout ce qui s'applique) ?

- [ ] On peut y accéder avec un lien symbolique
- [ ] Nous pouvons utiliser les *Linux CGroups*
- [x] Nous pouvons les monter (*mount*) lorsque nous exécutons le conteneur
- [x] Nous pouvons les copier dans l'image Docker lorsque nous créons l'image

(vi) (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*)

(vii) (0.5 points) Quelles sont les raisons possibles pour lesquelles une entrée dans la table des pages pourrait avoir un bit "invalide" (cochez tout ce qui s'applique) ?

- ☑ La page n'est pas en mémoire
- ☑ La page n'est pas dans l'espace d'adressage logique du processus
- ☑ La page a été "*swapped out*"
- ☐ La page a été "*swapped in*"

(viii) (1 point) Comment l'utilisation du bit sale (*dirty bit*) réduit-elle potentiellement le temps d'accès effectif (EAT) dans un système de pagination ?

A. Il réduit le nombre de *page faults*
B. Nous n'avons pas besoin de faire un *trap* vers le système d'exploitation lorsque le bit sale (*dirty bit*) est 1
C. En supprimant la nécessité de copier les pages non modifiées sur le disque si le bit sale (*dirty bit*) est 0
D. En rendant plus probable qu'il y aura un "*TLB hit*"

Question 2. *Systèmes de fichiers (3 points)*

Pour les trois questions suivantes, considérez que la taille de bloc est de **2KB** et que l'espace d'adressage est **32 bits**.

(i) (1 point) Quelle est la taille de fichier maximale accessible par un système de fichiers avec une **table FAT** avec 100 lignes ?

(ii) (1 point) Quelle est la taille de fichier maximale accessible par un système de fichiers **indexé** en supposant un seul bloc d'index ?

(iii) (1 point) Quelle est la taille de fichier maximale accessible par un **inode** dans un système de fichiers **ext2** si on n'utilise pas les pointeurs doubles ou triples indirects ?

Question 3. *Accès à la mémoire de masse (2 points)*

Compte tenu les informations suivantes :

— Taille d'un bloc est **2KB**
— Le disque tourne a une vitesse de **7200** révolutions par minute
— Le taux de transfert est **2Gb/s**
— Le seek time moyen est **6ms**

(i) (1 point) Quel est le temps de transfert, T_t, pour un bloc (en ms) ?
(ii) (1 point) Quel est le temps moyen d'entrée/sortie, T_{io}, pour un bloc (en ms) ?

Question 4. *Ordonnancement de disque (2 points)*

Un disque a **200** pistes. À l'instant présent, la tête de lecture est sur la piste 50 en se déplaçant dans le sens de l'augmentation des numéros de piste. La séquence des requêtes de lecture de piste à venir est :

[80, 10, 35, 140]

(i) (1 point) Quel est la distance totale parcourue par la tête de lecture avec l'algorithm C-LOOK ? 245
(ii) (1 point) Quel est la distance totale parcourue par la tête de lecture avec l'algorithm SCAN ? 340

Question 5. *Mémoire virtuelle (3 points)*

Un processus qui s'exécute sur une machine a été alloué **3 frames** en mémoire. Il accède aux pages suivantes dans l'ordre indiqué :

1 4 3 2 4 1 4 5 2 5 3

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

(i) (1 point) Deuxième Chance (*Second Chance* ou "horloge")

(ii) (1 point) Optimal

(iii) (1 point) Supposons la même séquence d'accès aux pages comme indiqué ci-dessus. Si la valeur de $\Delta = 5$ frames, quelle est la taille **maximale** de la *working set* pour la durée de la séquence.

(i)

	1	4	3	2	4	1	4	5	2	5	3
	1	1	1	2	2	2	2	5	5	5	5
-		4	4	4	4	4	4	4	2	2	2
-	-		3	3	3	1	1	1	1	1	3

8

(ii)

	1	4	3	2	4	1	4	5	2	5	3
	1	1	1	1	1	1	1	5	5	5	3
-		4	4	4	4	4	4	4	4	4	4
.	-		3	2	2	2	2	2	2	2	2

6

(iii) 4

Question 6. *Memoire centrale (5 points)*

(i) (2 points) En supposant un espace d'adressage **32-bits**, quelle est la taille du frame (page) pour que la table de page dans un système de pagination à 1 niveau tienne exactement dans un frame dans la mémoire principale (Réponse en KB.) ? Vous pouvez supposer que chaque entrée de la table des pages stocke une adresse (donc 32 bits).

En considerant le Fig.1 :

(ii) (1 point) Comment l'espace d'adressage logique est-il divisé (combien de bits pour le numéro de page et combien de bits pour le décalage) ?

(iii) (2 points) Dessinez la table des pages pour ce processus et incluez les bits valides/non valides dans chaque ligne.

(i) $2^{32-d} \cdot 2^2 = 2^d$

$2^{34-d} = 2^d$

$2d = 34$

$d = 17$

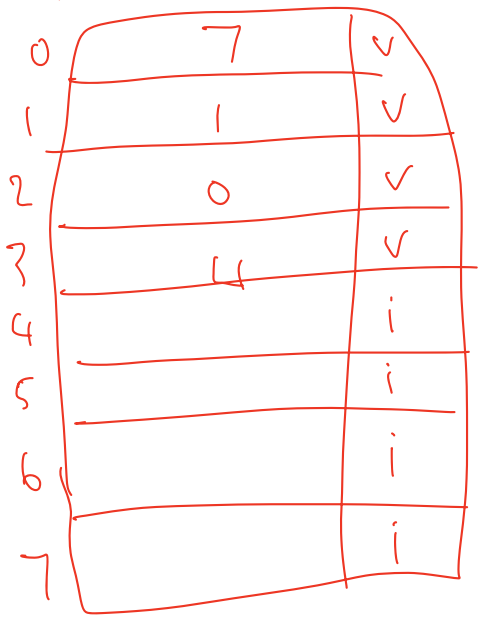
2^{17} B

$= 2^7$ KB

$= 128$ KB.

(ii) 1 3 1 2

(iii)



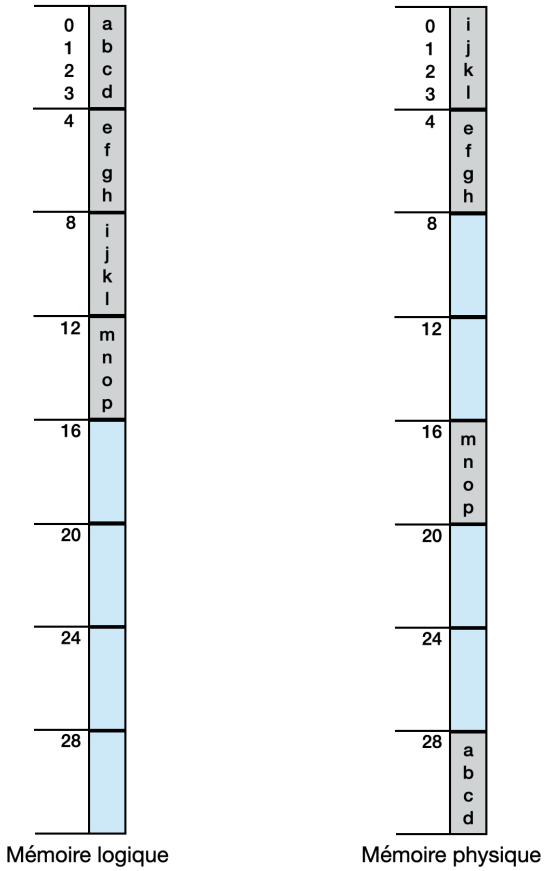


FIGURE 1 – Question 6 (ii) et (iii)