IFT320 Intra 2019

(...) = to be double checked or completed. **BOLD** = questions

1 a)

Nommez les modifications matérielles qui ont été apportées aux ordinateurs pour qu'un système d'exploitatoin (de type traitement par lots) puisse correctement contrôler l'accès aux unités d'entrée/sotries.

- 2 modes d'execution
- Intructions prévilégières
- Protection mémoire
- Erreurs de programation (interuption logicielle) [2e et 2e points]
- interuption (...) (interuption matérielle)
- Appels systeme (interruption logicielle)

1 b)

Pourquoi a-t'on un probleme de synchronisation entre les entrées/sorties et le traitement de l'UTC?

- Il y a un programme qui s'execute plus rapidement qu'un autre.
-

1 c)

Faites la distinction entre les termes suivants: *Tampon*, *Cache* et '*Spool*'.

cache

- hash map
- Conserve les donnees pou8r accelerer les usages futures.

tampon

- une file
- Accumule des donnes pour etre consommé par un procédé plus lent.

spool

- Permet de faire les i/o (entrées/sorties) d'un programme avant sont execution. Pour des i/o très lent. Si lent qu'on choisi de ne pas consomé de mémoire pour ca, on le laisse sur le disque dur. C'est un tampon à très longue durés.
- La sorti des i/o d'un programme peut être à postériorit de son execution.

1 d)

Pourquoi est-ce qu'un programme utilisateur doit se terminer par l'appel système *Exit()*?

- C'est une facon pour redonner le controle au UTC pour dire qu'on a terminé.

- pourquoi? Car le O.S. aimerait savoir quand est-ce qu'on a terminé.
- Erreurs system fait quelque chose de similaire.

1 e)

Éxpliquer la différence entre la gestion asynchrone des entrées/sorties par le système d'exploitation et l'implantation de services asynchrones pour les applications.

Dessin?? awww maaan..

Gestion asynchrone des entré sorties par le O.S.

1. Si vous faites un WRITE du coté application, les données vont êtres copiées vers le tampon qui est du coté systeme.
2. Si tampon n'est pas plein, Redonner le controle au user (App)
3. Si tampon est plein, mettre en

Implantation de Services Asynchrone pour app:

1. Effectuer une lecture.
2. Donne le controle au OS
3. Redonner controle au user (app), what so ever
4. Execute du code indépendant de cette demande.
5. Arrivé du resultat de la lecture
6. Faire un deuxieme appel systeme a la fin de la lecture.
7. Faire explicitement une demande d'acces à la donner pour mettre fin a la lecture.

cin>>x; se doit d'etre synchrone

cout<<x; ne peut somplement pas être synchrone.

Résumé:

> Gestion asynchrone des E/S par OS: Utilisation d'un tampon par le OS pour ne pas attendre après ce dernier. Doit bien etre géré par l'app

> Déclencher une demande de E/S meme si la demande précédente n'est pas compliquer. Ne fais pas atttendre l'app après l'UTC pour une réponse. Est géré par le tampon, du systeme.

Avoir une maitrise du concepte de tab des fichiers ouvert, tel l'exemple dans nachos.

2

*Gros bout de code **sale***

2 a)

Indiquez quels segments de code se trouvent dans l'espace du noyeau du O.S. et du programme utilisateur (application), respectivement.

Noyeau:

- Copy_to_user
- Copy_from_user
- inb
- outb
- wait_event_interuptible
- wake_up_call

Application:

- printf
-

2 b)

Quel est l'utilité des tampons cbuf_in et cbuf_out?

2 c)

Expliquez brièvement les deux situations principales dans lesquelles il faut utilisé l'énnoncé *wait_event_interruptible* Par ce que il n'y a pas de données qui sont arrivé dans le tampon. Alors on bloque l'app meanwhile.

Car le tampon est plein et on dit d'attendre en bloquant l'app.

2 d)

-- Gestiond es interruptions et routine d'interruption sont synonymes. -- **Définissez, en quelques mots, le rôle du segment de code E.** (...)

2 e)

Pourquoi doit-on absolument faire usage des fonctions *copy_to_user* et *copy_from_user* dans les segments de code A et B respectivement, alors que l'on dispose directement de l'adresse de la chaine de caractères à copier (le paramètre *userbuffer*)? L'ensemble d'addresses utilisé par le user ne sont pas les memes que l'ensembles d'addresse que le O.S. utilise. On doit faire la traduction. Ne sont pas dans le même espace d'addresse. Si ce ne sont pas les mêmes espaces, les addresses n'ont absolument pas la meme valeurs des deux cotés. C'est ***AUSSI*** pour la protection de la mémoire, pour ne pas rien fuckup.

2 f)

YAOO le wake_up cest l'inverse du wait_event_interruptible

Ordonnancez les segments de code (A à E inclusivement) de facon à reproduire le chemin d'une ligne de caracteres entrée au clavier d'un des ordinateurs et destinée à être envoyée sur le port série, pour être ensuite affichée sur l'autre ordinateur.

Ordonnancement:

1. writer (endehors du noyeau)
2. rs232_tut_write
3. rs232_isr
4. rs232_read
5. reader (hors du noyeau)

3

Faire l'allocation la plus contigu possible, à l'octet près. Le fichier sera en un seul morceau au début, mais on peut lui en ajouter d'autres si nécéssaire. Son emplacement phtsique sera donc représenté par une liste de descripteurs d'extension (des couples <position, taille>).

3 a)

Quelle technique de gestion de l'espace libre lui conseillerez-vous d'utiliser dans son systeme et pourquoi? -- Ouin, pas un *bitmap*

3 b)

Vous décidez, par le fait même, de lui rappeler qu'il ferait bien de penser à minimiser la fragmentation sur son système. Quel algorithme allez-vous lui suggérer pour choisir les morceaux d'espace libre qui seront alloués aux fichiers? Pourquoi? Worst-Fit est idéale -- Squeeze? --. Best-Fit est plus complexe à justifié.

3 c)

Si vous stockez les informations sur les morceaux alloués au fichier (la liste des descripteurs d'extension) dans l'entrée de répertoire, et que celui-ci est implanté sous forme de liste linéaire, avec des entrées de répertoire de taille fixe, quel problème allez-vous rencontrer? Suggérez une facon de corriger ce problème.

Visualiser la question. Largeur fixe

index	nom	empl. phys
0	maName	<pos, taille>
...	...	...

- Solution ghetto:
  - ...

probleme: t'as nul pars ou mettre l'information supplementaire. Rajouter un attribut pour l'information supplementaire dans le taableau. On perd de l'espace car la on a un espace dedier a l'info supplementaire. Si non utilis/, ca sert a rien. (-A revoir-)

4

Les méthodes d'allocation contigu,

4 a)

Nommez le type de fragmentation associé à chacune de ces méthodes et expliquez en quoi il consiste.
(...)

4 b)

Suggérez et détaillez une méthode pour tenter de minimiser la fragmentation pour chaque méthode d'allocation.

- Squeeze
- Tail-packing
- Stocker des donnees (compactes) dans l'en-tete ou, pire encore, dans l'entré de repertoire.

4 c)

Normalement, l'allocation par liste chainée est moins performante que l'allocation par index pour implanter l'accès direct. Décrivez une situation ou l'allocation par index souffrira d'un accès plus lent ainsi que de plus en plus de fragmentation que l'allocation chainée. *Dessin, ffs ma dude...* En fonction de la taille du fichier. Les très petit fichiers risquent de souffrirent lors de l'acces au ficheir. **O(n)** est meilleur pour les smoll files. **$O(Log_{TI}n)$** est meilleur pour les phat files.

5

Soit le O.S. TACOS dont le systeme de fichier utilise la structure suivante.

- un bloc est d'une longueur de 512o
- un numero de bloc tient sur 16 bits
- une entrée de répertoire contient, entre autre, six numéros de blocs, dont:
    - quatre poitent vers des données
    - un pointe vers un bloc d'index à un niveau
    - un pointe vers un bloc d'index à deux niveaux

5 a)

Calculez la taille maximale que peut atteindre un fichier dans ce système de fichiers..

T.I. = TB / TNB = 512o / 2o = 2^9 / 2^1 = 2^8 TNB = 2o (Taille du numero de bloc)

(..)

5 b)

Calculez la taille minimale des blocs qu'on obtiendra si on conserve des numéros de blocs sur 16 bits et on installe ce système de fichiers sur un discue de 4 pétaoctets (2^{52} octets). Suggérez une taille de numéros de blocs qui serait mieux adaptée à lasituation. On pourrait prendre 32 ou 64 bits, mais à 32 bits, ca fait des blocs de 1mo et 64 bits ca fait *fucking* petits blocs. Alors 32bits cest bien, mais sinon entre les deux c,Est gennant a jouer avec numero de 39 bits, ce qui fait des blocks de 8 ko....

6

Dans le systeme d'exploitation NACHOS, l'entrée d'un répertoire contient le nom du fichier et un pointeur vers l'en-tête. L'en-tête contien quelques informations sur le fichier (attributs).
Si on ne tient pas compte des limites imposées sur la taille de l'en-tête, comment peut-on adapter le système de fichiers de NACHOS (en-tête, répertoires, ...) pour ajouter la protection par **les liste d'accès**? Doit-on ajouter ou modifier d'autres structures que celles associées aux fichiers?
Mentionnez les structures de donées, les fonctions et les services du système que vous devrez modifier et expliquez brièvement ces modifications.

- Ajouter la liste d'accès dans l'en-tête de fichier
- Dans *create*, initialiser la liste d'accès avec le créateur
- Dans *thread*, ID user
- Ajouter un système d'utilisateur dans NACHOS
- Ajouter un service qui est une fonction de *filesys*, nommé CHMOD -- w/e it means
- Dans *open*, vérifier si le user est dans la liste *Grosso modo*...

Ici, il est important de prendre en note que cette question couvre de la matière non vue en classe. Il faut alors s'interressé à la facon de répondre à ce type de question. Pas de code, mais axée sur la compréhension.