UNIVERSITÉ DE ROUEN	ANNÉE 2007-2008
FACULTÉ DES SCIENCES	SYSTÈME D'EXPLOITATION
LICENCE INFORMATIQUE 3ème ANNÉE	EXAMEN
16 JANVIER 2008	3H00

Documents manuscrits autorisés + polycopié de cours.

Exercice 1 On suppose un système de fichiers de type inode (ext2) dont les blocs sont de 512 octets et les adresses de blocs sont codées sur 16 bits. Ce système de fichiers contient une bitmap de représentation des blocs libres et une pour les inodes libres.

1) Quelle est la taille du plus gros système de gestion de fichiers que vous pouvez construire avec ces données

2) Quelle est la taille maximale de la bitmap ?

Le modèle manipulé est le suivant ; chaque fichier est représenté par un nœud d'information ayant un tableau d'adresse.

- 7 adresses d'accès direct ;
- 1 adresse d'indirection simple ;
- 1 adresse d'indirection double.

Ce nœud d'information contient également un nombre de liens qui est le nombre de fois où l'inœud est référencé dans les répertoires.

3) Quelle est la taille maximale d'un fichier n'utilisant que des blocs d'accès directs, d'indirection simple, double ?

4) Pourquoi n'y a-t-il aucun intérêt à avoir une adresse d'indirection triple ?

On souhaite maintenant manipuler des fichiers beaucoup plus gros. Pour ce faire, nous allons passer la taille des adresses de blocs de 16 à 32 bits.

5) Quelle est maintenant la taille du plus gros SGF possible ? Du plus gros fichier ?

La bitmap des blocs libres a été endommagé.

6) Existe t-il un moyen de la reconstituer ? (Donner un algorithme)

Considérons maintenant que nous avons un champ nombre de liens (`nb_ln`) dans la structure d'inode.

7) Quel est le principal intérêt du nombre de liens ?

8) Pourquoi le nombre de liens d'un répertoire est-il au moins de 2 ?

9) Donnez l'algorithme de la fonction de suppression de lien. Vous détaillerez le cas d'un lien sur un fichier et sur un répertoire.

Exercice 2 Tous les moyens de synchronisation sont équivalents en ce sens qu'ils peuvent tous être implantés les uns au moyen des autres. On rappelle ici le fonctionnement des moniteurs. Un moniteur est une structure de haut niveau regroupant des variables partagées ainsi que des fonctions permettant de les manipuler. Seul un processus à la fois peut être actif dans le code d'un moniteur. Un processus souhaitant acceder au moniteur alors qu'il existe déjà un processus en cours d'exécution de celui-ci sera bloqué. Le moniteur peut utiliser 2 fonctions spécifiques sur des variables partagées.

- la fonction **wait(variable)** permet de bloquer un processus et provoque la libéreration de l'accès au moniteur,
- la fonction **signal(variable)** permet de réactiver un processus bloqué par un **wait**.

1) Expliquez comment le compilateur devra implanter les moniteurs au moyen des sémaphores. Vous expliciterez comment le compilateur garantit l'exclusion mutuelle du code d'un moniteur spécifique. Vous donnerez une spécification pour les fonctions **wait** et **signal**.

Exercice 3 On souhaite écrire un serveur d'anagrammes. Celui-ci fonctionnera avec deux tubes nommés. Le serveur bouclera sur la lecture d'un mot dans un tube **QUESTION**. Il se dupliquera et le sous-serveur sera chargé de créer l'anagramme et de l'écrire dans le tube **REPONSE** avant de se terminer. Le serveur quant à lui reprendra son écoute sur le tube **QUESTION**. On supposera par la suite que la taille du mot dont on souhaite un anagramme n'est pas trop grand (il correspond à une seule écriture dans le tube). On suppose que le client connait la taille du message à récupérer. Ce modèle fonctionne très bien si les différents clients sont lancés séquentiellement.

1) Quelles sont les problèmes qui peuvent se poser si deux clients sont lancés en parallèle ?

2) Décrivez les différents scénarii possibles sur un exemple en fonction du choix de l'ordonnanceur.

Pour solutionner une partie du problème, le client et le serveur s'accordent sur une structure de données à échanger. En effet, le client commence par envoyer un entier qui correspond à la longueur du mot qu'il écrira dans le tube.

3) Proposez une solution (toujours avec des tubes) pour que le client récupère bien l'anagramme du mot qu'il a envoyé.

On modifie le modèle de manière à remplacer nos tubes par une file de messages unique.

4) Donnez la structure de données qui sera échangée dans cette file et explicitez le fonctionnement de ce client-serveur avec une file de message unique.

On suppose que l'on dispose d'une fonction

```
char *anagramme(char *);
```

qui retourne l'anagramme d'un mot. Nous nous plaçons dans le cas d'une seule file de messages.

5) Donnez un algorithme pour le client.

6) Donnez un algorithme pour le serveur.