Système d'exploitation

Examen – 26 mai 2010
2h

Le barème est donné à titre indicatif. Les différentes parties sont totalement indépendantes.
L'examen est volontairement trop long, passez à une autre partie si vous êtes bloqués.

I. Questions d'échauffement *7 points*

Les sous-questions suivantes sont indépendantes.

a) Le noyau est-il la même chose que le système d'exploitation ? *1 pt*

b) Quelle est la différence entre l'administrateur (root) et le noyau ? Quels privilèges ont-ils ? Qui met en place ces privilèges ? *2 pts*

c) Le noyau connaît les traductions d'adresses virtuelles en adresses physiques. Mais pourquoi doit-il tout de même manipuler des adresses virtuelles lorsqu'il veut accéder à une donnée en mémoire (déréférencer un pointeur) ? *1 pt*

d) Comparez les appels-système et les interruptions aux niveaux matériel et logiciel. Quels contextes d'exécution sont mis en jeu ? Qui, quand et comment est programmée la gestion de ces événements ? Doit-on les traiter immédiatement ou peut-on déferrer le traitement à plus tard, et pourquoi ? *3 pts*

II. Anomalie de Belady *6 points*

On considère un programme qui manipule 5 pages de mémoire virtuelle dans l'ordre suivant: 3 2 1 0 3 2 4 3 2 1 0 4. Ces 5 pages virtuelles sont initialement stockées sur le disque. Elles vont pouvoir être chargées (individuellement) dans les pages de mémoire physique lors d'un accès par le programme, puis éventuellement renvoyées (individuellement) vers le disque plus tard si nécessaire.

1) Déroulez l'exécution du programme lorsque la mémoire virtuelle charge les pages en mémoire physique quand elles sont réellement nécessaires, et évince (*swappe*) les pages sur le disque selon l'algorithme FIFO. A chaque accès mémoire, dites quelle page virtuelle va éventuellement être chargée dans quelle page physique. Combien de défauts de pages vont se produire ? *2 pts*

2) On considère maintenant une machine avec 4 pages physiques au lieu de 3. Déroulez à nouveau l'exécution du programme. Que constate-t-on ? Qu'en pensez-vous ? *2 pts*

3) Que dire de l'efficacité de l'algorithme FIFO ? Pourquoi LRU est-il souvent meilleur ? Qu'en est-il de l'algorithme optimal ? *2 pts*

III. Concurrence et synchronisation *6 points*

a) Quelle est la différence entre de la concurrence logique et de la concurrence physique ? Comment les machines multiprocesseurs et/ou multicoeurs influent-elles sur ces deux notions ? *2 pts*

b) Sur une machine monoprocesseur monocoeur, quel événement matériel peut causer de la concurrence ? Cette concurrence est-elle logique ou physique ? Donner un exemple illustré avec une carte réseau. Comment peut-on s'en prémunir ? *2 pts*

c) Quelles sont les points communs et différences entre la synchronisation en espace utilisateur et en espace noyau ? Comment les événements matériels influent-ils sur l'espace noyau et utilisateur ? Qu'est-ce qu'un développeur peut faire dans un espace et pas dans l'autre ? *2 pts*

IV. Gestion mémoire *12 points*

On considère une architecture où la taille des pages est 1ko. La mémoire physique sera considérée comme infinie.

a) On lance un programme qui alloue 1Mo (mapping privé anonyme) puis les remplit de zéros. On remarque que le lancement de notre programme provoque environ mille défauts de page. Expliquez à quoi sont dûs ces défauts de page. Expliquez ensuite pourquoi ces défauts de page disparaissent si le programme alloue la mémoire mais ne l'utilise pas. *2 pts*

b) On remarque ensuite que le lancement du processus, même sans allouer de la mémoire ou la modifier, provoque tout de même quelques défauts de page. A quoi sont-ils dûs ? Que peut-on observer si on lance plusieurs fois le processus d'affilée peu après le démarrage de la machine ? *2 pts*

c) On modifie maintenant notre programme pour se dupliquer par *fork* après avoir alloué les 1Mo de mémoire et les avoir remplis de zéros. Le fils créé par *fork* remplit alors les 1Mo avec d'autres valeurs. Qu'observe-t-on en terme de défauts de page ? Expliquez pourquoi et l'intérêt du modèle. *3 pts*

d) On modifie enfin le programme pour que le processus père remette les 1Mo de mémoire à zéro après que le fils a terminé. Qu'observe-t-on en terme de défauts de page ? Expliquez pourquoi. *1 pt*

e) Expliquez rapidement ce qui change dans les résultats précédents si on remplace les 1Mo de mapping privé par un mapping public (partagé). *2 pts*

f) On considère maintenant que la mémoire physique n'est plus infinie. Expliquez rapidement comment les résultats précédents pourraient évoluer en cas de pression mémoire, c'est-à-dire si la mémoire physique est presque totalement remplie. *2 pts*

V. Table de pages *9 points*

On suppose disposer d'une architecture 64bits avec 3 niveaux de table de pages matériels et des pages de 8ko (2^{13} octets). La table de pages est constituée de tableaux de la taille d'une page. Ces tableaux peuvent être remplis avec des pointeurs vers les sous-niveaux (pointeurs de taille 4 octets) ou avec des PTE (de taille 16 octets).

Pour les deux questions suivantes, on pourra arrondir les calculs.

Rappel: 1 k = 10^3 = 2^{10} 1 M = 10^6 = 2^{20} 1 G = 10^9 = 2^{30} 1 T = 10^{12} = 2^{40}

a) Rappelez rapidement ce qui est stocké dans chaque niveau de la table de pages. Pourquoi utilise-t-on en général des tableaux de la taille d'une page ? *1 pt*

b) Combien de pointeurs ou PTE peut-on mettre par page ? En déduire le nombre de bits utilisés pour chaque niveau de la table de pages. Combien de bits sont nécessaires pour décrire un décalage dans une page ? *2 pts*

c) En déduire la taille maximale des adresses virtuelles manipulées sur cette architecture. Comment relier ce résultat au fait que l'architecture est dite « 64bits » ? *2 pts*

d) Si l'intégralité de la table de pages est allouée, quelle place occupe-t-elle en mémoire ? Quand peut-on éviter de tout allouer ? Donner un exemple où la réduction de place occupée par la table est significative. *4 pts*