Examen Systèmes - Master 1 Informatique
Année 2008-2009
Juin 2009

Durée 3 heures, tous documents autorisés

Exercice 1 (Mémoire Virtuelle)

1) Succinctement, expliquez en quoi consiste un mécanisme de mémoire virtuelle reposant sur la pagination.
2) Sauriez-vous donner une définition concise des principes de localité temporelle et localité spatiale ?
3) Écrivez un petit programme qui viole manifestement le principe de localité spatiale.
4) En quelques mots, expliquez en quoi consiste l'anomalie de Belady.

Exercice 2 (Processus)

1) Écrivez une fonction *void fils(int n,char *cmds[])* permettant de créer *n* processus fils s'exécutant en parallèle. Chaque fils exécutera la fonction (que l'on suppose donnée) *void calcul(char *commande)* où *commande* sera la *i*-ème chaîne de caractères du tableau *cmds*, et où *i* sera le numéro d'ordre de création du processus en question (*i.e.* le premier recevra *cmds[0]*, le second *cmds[1]*, etc.) De plus, on supposera que la terminaison normale d'un fils se fera en renvoyant au système une valeur. Le père devra impérativement attendre la terminaison de tous ces fils et calculer la somme de tous les résultats corrects renvoyés. (Un résultat correct est une valeur renvoyée dans le cadre d'une terminaison normale.)
2) Écrivez le code de la fonction *void calcul(char *commande)* permettant à un processus fils d'exécuter la *commande* passée en argument. Si cette exécution se révèle impossible on prendra soin de terminer le processus de sorte que le père en soit prévenu. (voir *résultat correct* si dessus).
3) Écrivez le code de la fonction *main* permettant de lancer l'exécution de *n* processus chacun exécutant la commande dont le nom aura été passé en argument de la façon suivante: *execute n* cmd_1 cmd_2... cmd_n. On prendra soin de vérifier la cohérence des arguments (nombre correct).
4) Modifiez la fonction *main* ainsi que le reste du programme de sorte qu'à la réception du signal *SIGUSR1* le programme s'arrête correctement (tous les processus doivent se terminer, aucun zombi ni orphelin ne doit subsister, etc.)

Exercice 3 (Processus et tubes)

1) Écrire une commande permettant de créer un anneau de *n* processus communiquant entre eux par tube anonyme. Par anneau on entend un ensemble de processus et de tubes structurés de sorte que le *i*-ème processus de l'anneau écrive au processus *i+1* à travers le tube *i*; la boucle étant bouclée sachant que le *n*-ième processus pourra communiquer avec le premier processus à l'aide du *n*-ième tube. Chaque membre de l'anneau connaîtra son numéro dans l'anneau (identifiant)
2) Écrire une fonction permettant à un processus d'adresser un message à n'importe quel autre processus de l'anneau.

3) Écrire une fonction permettant à un processus d'extraire un message de son tube d'entrée et de réaliser un traitement quelconque s'il lui est adressé, ou une retransmission s'il ne lui est pas adressé.

Exercice 4 (Projection)

Écrire un programme permettant de renverser le contenu d'un fichier dont le nom sera passé en argument. Le renversement consiste à échanger le premier et le dernier caractère, le second et l'avant-dernier, ainsi de suite. Pour éviter les problèmes de performance on vous propose de le faire en utilisant la projection en mémoire du fichier.

Exercice 5 (Concurrence)

Dans certains systèmes, les entrées/sorties vers un périphérique sont réalisées en lisant/écrivant à certaines adresses mémoire (memory mapped devices). Ici nous nous proposons de réaliser un système d'impression utilisant un mécanisme de même nature.
L'imprimante (commande *imprimante)* sera représentée par un processus dont le travail sera essentiellement de lire le contenu d'un segment de mémoire partagé dans lequel des données auront été écrites et de les afficher à l'écran.
La commande *imprime fichier* permettra de réaliser l'impression du *fichier* en question en écrivant morceau par morceau son contenu dans le segment de mémoire partagé.

1) Décrire les mécanismes à mettre en œuvre pour que tout fonctionne bien (en particulier que l'imprimante ne puisse être utilisée que par un seul processus à un moment donné, que l'imprimante et le processus imprimant se coordonnent pour que les écritures et les lectures soient correctement entrelacées)
2) Écrire le code de la commande *imprimante.*
3) Écrire le code de la commande *imprime fichier.*