Systèmes - M1 - Janvier 2011

Exercice 1 (merci Christian)

Écrire en C le code d'une commande de nom `confinement` prenant en argument :
1. une combinaison d'arguments de contrôle comme `-t entier -o entier -output fichier`
2. un nom d'exécutable suivi de ses propres arguments

La commande `confinement` permettra de lancer l'exécution de l'exécutable et de ses arguments de façon contrôlée. Ce contrôle permettra :
1. d'empêcher l'exécution de dépasser le nombre de secondes indiqué par l'option `-t`
2. d'empêcher la commande d'écrire en sortie standard plus de caractères qu'indiqué à l'option `-o`
3. de capturer la sortie standard dans le fichier indiqué à l'option `-output`

Si une option n'est pas spécifiée, aucun contrôle correspondant n'est réalisé.

On prendra soin de faire en sorte que le processus de confinement force la terminaison du processus contrôlé (gracieusement pour commencer en employant `SIGTERM` puis `SIGKILL`). D'autre part la valeur de retour du processus contrôlé par le confinement sera stockée sous forme ASCII dans un fichier de nom `exitStatus`. Ce code devra prendre la valeur 200 si le processus s'est terminé en violant la contrainte de temps, la valeur 201 si le processus s'est terminé en violant la contrainte d'écriture. Par défaut ce code sera celui renvoyé par le processus contrôlé lorsqu'il se termine normalement.

Exercice 2

1. Écrire en C le code d'une commande qui :
   a. à la réception d'un exemplaire du signal SIGUSR1 affiche à l'écran le contenu d'un fichier de nom f1
   b. à la réception d'un exemplaire du signal SIGUSR2 affiche à l'écran le contenu d'un autre fichier de nom f2
2. Est-il possible que l'écriture d'un des fichiers s'entrelace avec elle-même ? Si oui, comment l'empêcher ?
3. Est-il possible que l'écriture d'un des fichiers s'entrelace avec l'écriture de l'autre ? Si oui, comment l'empêcher ?

Exercice 3

Soit un tableau à une dimension contenant des valeurs entières. On souhaite calculer la moyenne de ces valeurs.
1. Écrire en C le code d'un programme permettant :
- d'initialiser un tableau d'entiers dont la dimension est reçue en argument de la commande et les valeurs tirées au hasard (`void srandom(long racine_generateur)`, `long random()`)
- calculer la valeur moyenne des valeurs du tableau
- afficher le résultat et le temps CPU calculé par `clock_t times(NULL)` qui renvoie une durée écoulée depuis un instant du passé et où `clock_t` est une valeur exprimée en unité de mesure de temps égale à `1/CLK_TCK` secondes

On constate que le système dont on dispose autorise la création de segments de mémoire partagée, et dispose de plusieurs unités de calcul. On espère donc beaucoup d'une parallélisation du calcul car on sait que si l'on partitionne l'ensemble de départ en sous-ensemble disjoints, alors la moyenne de l'ensemble est égale à la moyenne des moyennes de ses sous-ensembles.

2. Écrire en C le code d'un programme :

- qui reçoit en argument un nombre n. Ce nombre représentera le degré de parallélisation, *i.e.* le nombre de processus qui réaliseront le calcul en même temps
- qui crée autant de processus que demandé par l'utilisateur (n) et fournit à ceux-ci de quoi calculer la moyenne de l'un des sous-ensembles (on documentera bien quelles données sont échangées/transmises)
- qui se synchronise de sorte que les sous-calculs soient tous terminés et calcule le résultat final en affichant le temps de calcul.
- qui prend soin de relancer les calculs ratés si l'un des sous-calculs se termine prématurément

Exercice 4

Écrire un programme qui permet d'obtenir ce qu'un shell réalise lorsqu'on lui donne la commande suivante :

```
grep bonjour < f | wc -l > r
```

Problème

On suppose que la position des comptes client d'un établissement bancaire réside dans un fichier (compte n°i, i-ième entier du fichier, les comptes sont numérotés à partir de 0, le compte 0 est celui de la banque elle-même). On souhaite écrire différentes commandes permettant de réaliser des opérations sur ces comptes en banque. À chaque type d'opération souhaitée, un programme correspondra.

1. Écrire une commande permettant d'effectuer un dépôt ou un retrait sur un compte donné. La commande sera de la forme `operation compte valeur`, si la valeur est positive il s'agira d'un dépôt, négative d'un retrait et nulle d'une simple consultation
2. Écrire une commande permettant de réaliser un virement entre comptes : `virement compte_origine compte_destination somme`
3. Écrire une commande permettant au banquier de calculer la somme des dépôts à un instant donné (le compte de la banque ne compte pas dans cette somme) : `combien`
4. Écrire une commande permettant au banquier de prélever sur tous les comptes (sauf le sien) des frais de gestion et de déposer le tout sur son propre compte. On dit que les frais s'élèvent à 5 patates; la patate étant la monnaie de notre banquier, en dessous rien : des nèfles. Le programme s'appelle `ramasse`

En y réfléchissant on distingue plusieurs problèmes qui pourraient se produire :

a. on tente deux opérations sur un même compte à deux instants très proches dans le temps
b. on tente une opération pendant que le banquier calcule la somme des dépôts
c. on tente un virement pendant que le banquier prélève des frais de gestion

5. Pour chacun de ses problèmes, donner un exemple permettant d'être convaincu que de l'argent se perd (le comble pour un banquier)
6. Pour chacun ces problèmes trouver une solution (votre banquier vous en saura gré)
7. Écrire le code correspondant pour chacune des commandes