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

Durée 3 heures, tous documents autorisés

Exercice 1

Soient les deux extraits de programmes suivant :
ex1:

```
int tableau[10000][20000]; int i; int j;
for (i=0; i<10000; i++)
        for (j=0; j<20000; j++)
                tableau[i][j] = 0;
```

ex2:

```
int tableau[10000][20000]; int i; int j;
for (j=0; j<20000; j++)
        for (i=0; i<10000; i++)
                tableau[i][j] = 0;
```

Une mesure de performance effectuée fait apparaître un biais important durant l'exécution des deux séquences précédentes. On notera que toutes les précautions ont été prises pour garantir que les exécutions ont été faites dans des conditions systèmes identiques (charge de la machine, etc.)
La mesure pour le premier est: `1.137u 1.279s 0:08.21 29.2% 0+0k 0+0io 0pf+0w`
La mesure pour le second est: `6.206u 3.043s 11:16.91 1.3% 0+0k 0+0io 9733pf+0w`
Sauriez-vous expliquer ce qu'il se passe ?

Exercice 2

Soit l'extrait de programme suivant :

```
void f(int i) {
        int j;
        for (j=0; j<i; j++)
                if (fork()==0) {
                        f(i-1);
                        exit(0);
                }
        // à remplir, lire la suite...
}
```

1) combien de processus sont-ils crées suite à un appel à `f(4)` ? On supposera que la fonction `fork()` n'échoue pas.
2) généraliser le résultat précédent, *i.e.* combien de processus sont-ils créés par un appel à `f(n)` en fonction de `n` ?
3) compléter le code (en lieu et place du commentaire) de sorte que la terminaison se produise sans processus zombi.

Exercice 3

Écrire le code de la commande

vivant délai commande [arguments...]

qui a pour effet de contrôler l'exécution de la commande avec ses arguments, de sorte que l'utilisateur soit obligé de frapper CTRL-C à intervalles au plus distants de délai secondes. Si l'utilisateur ne fournit pas de preuve de son attention (il oublie de frapper CTRL-C dans le délai imparti), l'exécution de la commande est alors brutalement interrompue de façon certaine.

On rappelle que la frappe du caractère CTRL-C au clavier provoque l'envoi du signal SIGINT au groupe de processus en avant-plan.

Exercice 4

On propose d'écrire une application parallélisée capable de compter le nombre d'occurences d'un caractère donné dans un fichier. Deux techniques sont à votre disposition: parallélisation par utilisation de processus concurrents ou par un processus multi-threadé. L'idée générale étant que pour les deux méthodes le fichier est partagé (soit entre processus, soit entre threads) en étant préalablement projeté en mémoire. Le compteur d'occurence aussi est partagé entre les différentes exécutions.

1) écrire le code de la commande occurproc n caractère fichier permettant de calculer puis afficher le nombre d'occurence du caractère donné dans le fichier projeté en mémoire puis découpé en n tronçons de taille approximativement égales, chaque morceau étant traité par un processus dédié
2) écrire le code de la commande occurthread n caractère fichier identique à la précédente mais utilisant des threads pour réaliser le calcul

Problème

On se propose de réaliser un serveur de documents.

Un client (programme de nom furetdesneiges) désireux d'obtenir un document se connecte au serveur en ouvrant en écriture un tube de nom /tmp/docserv puis y écrit un message constitué de deux parties: le nom d'un tube nommé permettant au serveur de lui répondre, et la commande du document qu'il désire recevoir. Ensuite, il attend la réponse du serveur sur le tube de nom indiqué.

Le serveur (programme de nom siouxd) attend sur son tube /tmp/docserv qu'un client daigne lui envoyer une commande. À la réception d'une commande, le serveur délègue son traitement à un thread (dit thread de service) et reprend immédiatement son attente sur le tube d'entrée. Le thread de service réalisera les vœux du client en traitant le message que le thread principal lui aura transmis.

1) décrire les formats des messages qui seront échangés entre les clients et le serveur.
2) écrire le code de furetdesneiges
3) décrire les formats des messages envoyés en réponse du serveur aux clients. On n'oubliera pas de prévoir des cas d'erreurs comme le cas d'un client ayant réclamé un document qui n'existe pas (une réponse doit être envoyée mais qui indique une erreur)...

4) écrire le code d'une fonction (de nom `service`) permettant de réaliser le service de réponse
5) décrire la communication interne nécessaire entre le thread principal et les threads de service
6) écrire le code de `siouxd`