Programmation Système
TP noté du 10/04/2013 (durée : 2h)
Groupe 1

Tri aléatoire en moins de 5 secondes

Cet exercice a pour but de réaliser le tri par ordre croissant d'un tableau d'entiers. L'algorithme de tri utilisé est le suivant : Tirer au hasard 2 indices de tableau, si les éléments correspondant sont dans le désordre, alors les mettre dans le bon ordre, répéter cette *opération élémentaire* autant de fois que nécessaire.

Pour réaliser ce tri, le père crée 2 fils et un segment de mémoire partagée. Le tableau est stocké en mémoire partagée et le travail est partagé entre les 2 fils : Dès qu'un des fils est disponible, il réalise une opération élémentaire et redevient disponible. (Un fils ne réalise qu'une seule opération élémentaire à chaque fois.) Au bout de 5 secondes, le père arrête ses 2 fils (en leur envoyant le signal `SIGKILL`, attend leur terminaison, affiche le tableau, détruit le segment de mémoire partagée et termine.

Attention, les fils doivent se synchroniser car ils leur est interdit d'accéder en même temps au tableau. Le père joue le rôle de *contrôleur* et réalise la synchronisation des fils en utilisant les signaux. Voici en résumé le protocole de synchronisation utilisé par les fils et le père :

```
Code d'un fils ::

tant que vrai
  émettre(demande,contrôleur);
  recevoir(acquittement,contrôleur);
  accéder au tableau (réaliser une
   opération élémentaire)
  émettre(fin,contrôleur);
```

```
Code du père (contrôleur) ::

tant que vrai
  recevoir(demande,fils);
  émettre(acquittement,fils);
  recevoir(fin,fils);
```

Voici les signaux à utiliser pour implémenter ce protocole :
- `demande` envoyé par le premier fils = `SIGUSR1`
- `demande` envoyé par le deuxième fils = `SIGUSR2`
- `acquittement` envoyé par le père = `SIGCONT`
- `fin` envoyé par un des fils = `SIGINT`

Noter que la répartition du travail entre les 2 fils n'est pas forcément équitable : Les fils n'agissent pas forcément l'un après l'autre : Si un fils est plus rapide que l'autre, il peut réaliser plusieurs opérations élémentaires de suite (voire la totalité du travail) sans que l'autre fils ne puisse agir.

1. Commencer par écrire un programme `tri1.c` qui crée un segment de mémoire partagée pour un tableau de `N` entiers (où `N` est une constante définie par un `#define`), et 2 fils qui réalisent une boucle sans fin. Le code doit s'arrêter automatiquement au bout de 5 secondes (utiliser la primitive `alarm()` et le signal `SIGALRM`). Le code devra être propre, c'est-à-dire qu'au bout des 5 secondes, le père doit envoyer un signal `SIGKILL` aux fils, attendre la terminaison des fils (primitive `wait()`) et libérer la mémoire partagée avant de terminer.
2. Écrire un programme `tri2.c` obtenu à partir de `tri1.c` en ajoutant :
   - une fonction `init()` qui initialise le tableau en le remplissant de nombres aléatoires (utiliser les fonctions `srand()` et `rand()`).
   - une fonction `print()` qui affiche le tableau.
   - une fonction `hasard()` qui tire au hasard 2 indices de tableau.
   - une fonction `echange()` qui prend 2 indices en argument et échange les 2 éléments du tableau si ces éléments sont dans le désordre.
3. Écrire un programme `tri3.c` obtenu à partir de `tri2.c` en ajoutant le protocole de synchronisation coté contrôleur (père) et pour le premier fils, et faire tourner avec un seul fils. (Utiliser `sigsuspend()` pour attendre l'arrivée d'un signal.)
4. Écrire un programme `tri4.c` obtenu à partir de `tri3.c` en ajoutant le protocole de synchronisation pour le deuxième fils. (Si tout va bien, le tri devrait être fini au bout des 5 secondes.) (Penser à tous les cas de figure et penser à utiliser `sigprocmask()` pour bloquer les signaux indésirables dans certains cas.)
5. Questions subsidiaires (répondre en ajoutant des commentaires à la fin de `tri4.c`) :
   - peut-on éviter qu'un fils fasse tout le travail si il est très rapide ? Si oui, comment ? (Citer les primitives à utiliser le cas échéant.)
   - est-ce que les deux fils pourraient être autorisés à accéder en même temps au tableau ? Si oui, à quelles conditions et que faudrait-il changer dans le code ?