TP7

La page web du cours :

`http://www.lsv.ens-cachan.fr/~schwoon/enseignement/systemes/ws1415/`.

1 Ping-pong

Sur la page web vous trouverez un programme (`pingpong.c`) qui crÃ©e deux processus, père et fils. L'objectif est de compléter le programme ; lorsque l'utilisateur saisit 0 ou 1, le père devrait communiquer ce fait au fils en lui envoyant SIGUSR1 ou SIGUSR2. Le fils devrait alors reconstruire la suite des 0 et 1 correctement et l'afficher sur l'écran.

Le défi dans cette exercice est de rendre cette communication fiable dans toutes les circonstances. Par exemple, un processus ne peut avoir deux signaux d'un même type au même temps et l'ordre d'exécution des gestionnaires n'est pas garanti lorsque plusieurs signaux sont en attente, du coup après avoir envoyé un signal le père doit attendre une confirmation du fils avant d'en envoyer un autre. De plus, avant d'envoyer un premier signal un processus doit être sûr que le gestionnaire de signal a bien être installé par son partenaire.

Une bonne solution doit bloquer des signaux lorsqu'un processus n'est pas prêt a en recevoir (voir `sigprocmask`). Un autre appel système très utile est `sigsuspend` qui attend un signal (avec un masque donné) et restaure immédiatement l'ancien masque après en avoir reçu un.

2 Lire et écrire dans des fichiers

L'objectif de cette exercice est d'écrire une version simple du programme `cp` qui recopie des fichiers. Un programme squelette se trouve sur la page web du cours.

Appels système utiles: `open`, `creat`, `read`, `write`, `close` (tous dans la section 2 du manuel).

Pour obtenir la taille (en octets) d'un fichier, il y a plusieurs possibilités: (i) tester les valeurs renvoyés par `read`; (ii) utiliser `lseek`, (iii) utiliser `stat`.

1. Écrivez d'abord une version simple du programme et rajoutez des tests pour vérifier si les opérations réussissent. Utiliser la fonction `syserr` du squelette en cas d'erreur.

2. Testez votre programme avec des grands fichiers et mésurez le temps en fonction du nombre d'octets traités dans les appels de `read` et `write`. (Utilisez `time` dans le shell pour mésurer.)

3. Pour ceux qui avancent plus rapidement, étendez le programme pour recopier les droits d'accès et l'heure de dernière modification du fichier source au fichier destination (voir `stat`, `chmod`, `utime`).

3 Premier essai avec les pipes

Créons deux processus qui communiquent par un *pipe*.

On peut repartir du squelette pour `pingpong` pour cette exercice, sauf que cette fois, père et fils vont utiliser un pipe pour communiquer.

Assurez que le fils termine correctement. Plus précisement, lorsque l'utilisateur tape Ctrl+D (ce qui signale *end-of-file* dans *stdin*), le père devrait terminer ce qui va fermer son accès au pipe. Le fils devrait remarquer cette condition et terminer lui aussi. Pour cela, il faudra évaluer les valeurs renvoyés par `read` et fermer les accès non utilisés aux deux accès du pipe.

Exercice avancée : Faites en sort que le programme affiche `zzzz...` quand l'utilisateur ne saisit rien pendant cinq secondes. (Plusieurs solutions sont possibles.)