3iLINGENIEURS
</les sciences informatiques>

TP1 – Les Processus

Exercice 1 : fork et processus zombie.

Un processus entre dans l'état zombie lorsqu'il se termine alors que son père n'a pas encore pris connaissance de sa terminaison.

- Ecrire un programme qui crée un processus fils. Le père affiche le son pid ainsi que celui de son fils. Le fils affiche son pid ainsi que celui du père.
- Ajouter des temporisations dans les deux processus et retrouver à l'aide de la commande **ps -la** l'identifiant du processus père et celui du processus fils.
- Ajouter une temporisation de 20 secondes, seulement dans le processus père. Expliquer l'affichage **de ps -la** avant et après 20 secondes d'exécution.

Expliquer l'état du processus fils avant 20 secondes. Proposer une solution pour éviter cet état en utilisant la fonction wait().

- Ajouter une temporisation de 20 secondes, seulement dans le processus fils.

Expliquer l'affichage de **ps -la** avant et après 20 secondes d'exécution. Quel est le processus parent du processus fils ?

Remarque : vous pouvez aussi utiliser un utilitaires **htop** sympathique pour suivre l'évolution des processus et les différentes ressources utilisées. Voici un lien intéressant pour prendre en main cet outil :
https://christiansueur.com/htop-visualiser-et-gerer-facilement-les-processus-lances-sous-linux/

Exercice 2 : synchronisation avec exit et wait

Ecrire un programme en C qui crée deux processus fils. Le processus père est chargé d'afficher les 5 premiers multiples de 4. Un des processus fils est chargé d'afficher les 5 premiers multiples de 3. L'autre processus fils est chargé d'afficher les 5 premiers multiples de 2.

Réaliser une synchronisation de telle façon que tous les multiples de 2 s'affichent en premier, puis tous les multiples de 3, puis tous les multiples de 4.

Exercice 3 : Communication entre processus

Sous Unix un processus fils indique sa fin à l'aide de l'appel système void exit(int status) en renvoyant la variable status au processus parent.

Un processus père peut en prendre connaissance à l'aide de l'appel système pid t wait(int* status). La variable status ne peut pas être lue directement. On utilisera la macro WEXITSTATUS pour la lire (voir les pages de man).

Ecrire un programme qui calcule la factorielle de 10, c'est-à-dire sous la forme de deux processus. Le processus fils calculera le premier en allant de n = 1 à 5. Le processus père parcourra n de 6 à 10 en utilisant le résultat du calcul que lui aura renvoyé son fils.

Voir : manuel de la fonction wait http://manpagesfr.free.fr/man/man2/wait.2.html

Exercice 4 : utilisation de la fonction execl

Ecrire un programme C équivalent à la commande shell suivante :

- `who & ps & ls -l`
- `who; ps; ls -l`

Voir : http://manpages.ubuntu.com/manpages/bionic/fr/man3/execl.3.html ou http://manpagesfr.free.fr/man/man3/exec.3.html

Exercice 5 : utilisation des signaux SIGUSR1

Ecrire un programme prenant en paramètre un exécutable avec ses paramètres et qui démarre 5 processus en affichant leurs numéros de PID. Une fois que les processus sont créés, le père doit attendre une entrée clavier **avec getchar(),** et envoyer ensuite simultanément un signal **SIGUSR1** à tous ses enfants. De leur côté, les fils doivent dormir avec pause() jusqu'à ce que le père les réveille avec un signal SIGUSR1. Et c'est à ce moment-là qu'ils doivent exécuter la commande passée en argument. Une fois que tous les fils sont terminés, le père doit afficher l'ordre de terminaison des fils, qu'il aura stocké dans un tableau.

Exemple:
$>./death_race ls l
3453
3454
3457
3462
3475
Attente du signal de départ...
[frappe de la touche entrée]
[affichages des ls l des fils]
Classement de la course: 3454 3575 3453 3462 3457

Exercice 6 : Bonus - recouvrement d'un processus : execl

Ecrire un programme C qui prend en paramètre une série de fichiers source .c, les compile chacun séparément et simultanément puis édite les liens pour produire un exécutable. Ce programme doit :

- lancer un processus fils pour chacun des noms de fichiers passés en paramètre;
- chaque fils doit exécuter le programme gcc -c sur le fichier dont il s'occupe;
- le père doit attendre la terminaison de tous ses fils;
- si l'ensemble des fils ont terminé sans erreur, le père réalise l'édition de liens en exécutant gcc sur les fichiers .o produits par les fils.

Voir la compilation avec gcc :

https://tdinfo.phelma.grenoble-inp.fr/2Aproj/fiches/compilation_gcc.pdf