Série TP 0 : Rappels sur les processus

Exercice 1 :

1- Écrire un programme "C" qui crée deux processus (un père et son fils).
Le processus père affichera : "je suis le processus père, mon PID est:.... Et le PID de mon fils est : …."
Le processus fils affichera : " je suis le processus fils, mon PID est :...Et le PID de mon père est :..."

```
# include <unistd.h>
# include <stdlib.h>
# include <stdio.h>
void main(){
int pid,i;
pid = fork ( ) ;

if (pid == - 1)
        { /* code si échec : */
        printf ( " le fork ( ) a échoué \n " )
         exit(0) ;
       }
if (pid == 0)
        { /* code correspondant à l'exécution du processus fils */

        }
else
        { /* code correspondant à l'exécution du processus père */ }

         }
}
```

2- Modifier le programme 1 pour que le processus père crée n processus fils. Chaque fils affiche son pid et celui de son père (fonctions **getpid** et **getppid)** et exécute l'instruction sleep(8) avant de quitter.
3- Exécuter la commande shell ps aux a partir d'une autre fenêtre du terminal pour visualiser les processus crées.
4- **Processus orphelin :** modifier le programme 1 (un père et un seul fils) en rajoutant la boucle suivante au niveau du processus fils, observer les **pid** et **ppid** affichés par le fils et expliquer la différence avec le programme 1:

```
for(int i=0 ; i<5 ; i++) { sleep(2) ;
 printf( " je suis le processus fils N %d,  mon PID est : %d, Et le PID de mon père est : %d" , i,
getpid(), getppid() ) ;
}
exit(0) ;
```

5- Que ce passe-t-il quand un processus devient orphelin ? Que faut il faire pour éviter cette situation ?

}

Exercice 2 :

Écrire un programme "C" qui crée deux processus fils (un père et deux fils). Chaque processus fils affiche et retourne une valeur aléatoire. Réalisez les scénarios suivants :

- Le processus père attendra la fin du premier processus fils qui se termine et affichera le pid du fils et sa valeur.
- récupère la valeur aléatoire retournée par chaque processus fils et affichera la plus grande valeur.

- Modifier le programme précédant en déclarant une variable int a initialisée a zéro, chacun des fils modifie la variable a (a+valeur aléatoire) et l'affiche et le père attend la fin des deux fils puis affiche la valeur de a.
- La variable a est elle partagée entre les 3 processus ? Essayez avec un pointeur int * a.

Exercice 3 (a faire et a rendre au prochain TP):

Écrire un programme qui crée l'arborescence suivante de processus : Le processus P0 crée deux fils p1 et p2, attend leur terminaison puis crée le processus p5. Le processus p1 crée deux processus fils p3 et p4.

- Chaque processus attend la fin de ses fils avant de se terminer.
- Chaque processus exécute sleep(10) avant de quitter par un exit(0).
- Utiliser les commandes **ps** et **pstree** (à partir d'un autre terminal) pour visualiser l'arbre réalisé (`ps -aef –forest, pstree -pl`)

Exercice 4 : utilisation de fork combinée avec la famille de fonctions exec (a faire et a rendre au prochain TP):

Le but de cet exercice est d'écrire un mini shell qui fonctionne comme suit :

- Le processus (ou programme) principal, qu'on appellera **pshell**, va créer un fils différent pour executer chacune des commandes passées en argument (utiliser argv, argc, getopt()).

- Chaque fils affichera d'abord un message de présentation : je suis le fils N i, de pid getpid(), mon pere est le processus de pid getppid, j'exécute la commande argv[i]. Puis il appel une des fonctions exec avec la commande.

- Le père attend la fin d'exécution de chacun de ses fils et affiche pour chacun le pid, le nom de la commande exécutée et le type de terminaison : arrêt normal ? Erreur ? Interruption ? Utiliser la variable **status** ***wait(int *status)*** ou dans ***waitpid(pid_t pid, int * status....)***

Questions supplémentaires:

- Ajouter un message « au revoir du fils de pid getpid() après l'appel de la fonction ***exec*** au niveau des fils. Est ce que les fils affichent ce message ? Pourquoi ?

- Réaliser l'extension suivante :

- A la fin d'exécution des fils, le père affiche le message : « je suis le processus pshell, vous avez la main pour taper d'autres commandes a exécuter séparées par des espaces et délimitées par un retour chariot. Si plus de commandes a exécuter, taper q pour quitter le pshell. »