Dépt. Informatique 2eme année, S4# système, TP1

Thème : Duplication d'un processus : Fork

Exercice 1 : Exemple du fork

```
#include <stdio.h>
#include <errno.h>
#include <sys/types.h>
#include<string.h>
#include<stdlib.h>

extern int errno;

main()
{
        pid_t pid;
        pid = getpid();
        fprintf(stdout,"Avant le fork, pid = %d\n",pid);

       pid = fork();
    /* En cas de reussite du fork, le pere et le fils poursuivent
leur execution a partir d'ici.
      La valeur de pid permet de distinguer le pere du fils. */
    switch(pid) {
        case -1 :   /* erreur dans fork() */
                    fprintf(stderr,"error %d in fork:
%s\n",errno,strerror(errno));
                    exit(errno);

      case 0 :    /* on est dans le fils */
                   sleep(6);
                /* On suspend le processus pendant 6 secondes. Cela
permet d'utiliser
                 la commande ps (p.ex. ps -gux) pour visualiser la
liste des processus. */

                    fprintf(stdout,"Dans le fils, pid = %d \t pid
du pere = %d\n",getpid(),getppid());
                    break;

      default :   /* on est dans le pere */
                 fprintf(stdout,"Dans le pere, pid = %d \t pid du
pere = %d \n",getpid(),getppid());
     }
}
```

Ce programme (et les autres) se trouve sous **corton sous /tmp/info2#/tp0**

Questions

1°) Exécuter ce programme, identifier les différents processus crées et expliquer le résultat.

2°) Modifier le programme pour que le processus père il affiche le pid de son fils.

3°) Modifier le programme en utilisant la fonction wait ci-dessous, pour que le père attend la terminaison du processus fils

```
#include <sys/types.h>
#include <sys/wait.h>
 pid_t wait (int *status) ;
```

La fonction `wait` suspend l'exécution du processus appelant jusqu'à ce qu'un de ses processus fils se termine. Si un processus fils s'est déjà terminé, `wait()` retourne le résultat immédiatement. `wait()` retourne le pid du processus fils si le retour est déjà la terminaison d'un processus fils ; -1 en cas d'erreur.

Exercice 2 : la file de processus

a) Écrire un programme "*file.c*" permettant de créer une file de 8 processus P1, P2, P3,P4, P5, P6, P7 et P8 telle que P2 est le fils de P1, P3 est le fils de P2, P4 est le fils de P3, ainsi de suite jusqu'à enfin P8 est le fils de P7. Chaque processus doit afficher son nom (P1, P2, P3, P4, P5, P6, P7, P8), son PID et son PPID (sauf P1 qui n'affiche que son nom et son PID).
b) Écrire un programme "*fileRecurs.c*" permettant de créer récursivement une file de processus dont la longueur est demandée à l'utilisateur. Vous écrirez pour cela une fonction récursive *creerProc(int nb, int num)* prenant en paramètres,
nb : le nombre de processus à créer dans la file et *num* : le numéro du processus à créer.
Chaque processus affichera les mêmes informations que dans la question précédente.

Exercice 3 : les variables dans une duplication de processus

Écrire un programme "*calcul.c*" qui demande 2 entiers a et b à l'utilisateur puis crée un processus fils. Ce processus fils doit afficher le résultat de la division de a par b si b n'est pas nul. Dans ce cas, le fils renvoie un code de retour égal à 0. Si b est nul, le fils renvoie un code de retour à 1(erreur). Dans ce dernier cas, le processus père devra redemander la valeur de b à l'utilisateur et créer un nouveau processus fils jusqu'à ce que le processus fils renvoie un code de retour nul. Dans ce cas, le processus père affiche "*au revoir*" et se termine.
Indication : dans le processus fils, on utilisera la commande ***exit(int code)*** pour renvoyer un code de retour.

Exercice 4

a) Écrire un programme qui crée un processus fils dont le pid sera affiché par le père. Le processus père attend la fin du fils. Le processus fils doit renvoyer un code de retour (grâce à exit) pour terminer normalement. Dans ce cas, le processus père affichera le message "fils terminé normalement" et la valeur du code de retour du fils. Dans le cas contraire, le père affichera le message "fils terminé anormalement".
b) Modifier le programme pour traiter le cas où le fils est tué. Le processus père affichera le message "fils tué " et le numéro du signal qui a provoqué la terminaison du processus fils.

Exercice 5

Écrire un programme qui affiche, dans l'ordre, les entiers de 0 à 3 par 4 processus.