Programmation Système TD : Complément Signaux Synchronisation Père/Fils

Département Informatique
Licence S6

Notions Abordées :

- **Signaux** : *kill(), pause(), fork(), sigaction(), kill(), pause(), wait(), waitpid(), sigpending(), sigprocmask, signal sets*

Exercice 1 : Synchronisation Père/Fils (Dernier exercice de la Série 2)

Ecrire le programme C sous Unix qui permet à un processus père de créer un processus Fils. Le père et le Fils doivent s'exécuter en parallèle et permettent l'affichage suivant :
Fils: 2 4 6 8 10
Père: 3 6 9 12 15
Fils: 12 14 16 18 20
Père: 18 21 24 27 30
.....
.....
Père: 99

Appels Systèmes : fork(), signal(), kill(), pause()

Indication :
Chaque fois qu'un processus affiche un multiple de 5 il se bloque et envoi un signal à l'autre processus pour le réveiller.

Exercice 2 :

Quel est la valeur du compteur *count* ?

```
#include <sys/types.h>
#include <unistd.h>
#include<stdio.h>
#include <sys/wait.h>

#define N 3

int count = 0;

int main(int argc, char* argv[]){

 int i = 0, pid;

 while(i < N){

    pid = fork();
    if(pid == 0){
       count++;
        printf(" count Fils : %d \n", count);
        break;
    printf("if \n");
 }
  i++;

 }

if(pid != 0){
   for(i = 0; i<N; i++)
      wait(NULL);
 printf("count PERE : %d \n", count);
 }
return 0;
}
```

Exercice 3 :

Ecrire un programme qui crée 5 fils. Chaque fils envoie la valeur de retour au processus père tel que :

Le processuss fils n° 1 renvoie la valeur 1
Le processuss fils n° 2 renvoie la valeur 4
Le processuss fils n° 3 renvoie la valeur 9
...

Le père doit également afficher les PID de ses fils.

Appels Systèmes : fork(), waitpid()

Exercice 4: (Signaux pendants)

Vérifier quels sont les signaux qui ont été envoyés à un processus mais qui n'ont pas encore été pris en compte par le processus.

Appel système : *sigaddset, sigemptyset, sigprocmask, sigismember, sigpending.*

1. Ecrivez un programme en C qui masque les signaux SIGINT et SIGQUIT. Le programme doit alors s'endormir (fonction sleep).
   Pendant qu'il dort, envoyez-lui à partir d'un terminal (kill –SIG $< pid_processus >$) un SIGINT et un SIGQUIT. Lorsque le processus se réveille, il affichera la liste des signaux qui se trouvent pendants.

2. Modifiez le programme de la question précédente de telle sorte à effacer le masque des signaux pour SIGINT et SIGQUIT après l'affichage de la liste des signaux pendants.

   Est-ce que le processus s'exécutera jusqu'à la fin ? Pourquoi ? Pour vérifier ce comportement ajoutez à la fin de votre programme l'affichage d'un message quelconque.

Exercice 5 :

1. Commenter, ligne par ligne, le programme ci-dessous
2. Dire ce qui est affiché à l'écran pendant l'exécution du programme *signaux.c*. Expliquez votre réponse.
3. Modifier votre programme de tel façon à obliger le fils par commencer le premier.

```
#include<unistd.h>
#include<stdio.h>
#include<signal.h>
#include<stdlib.h>

void message(int signum){
   printf(" Message reçu %d , %d \n", getpid(), signum);
}

int main(void){
   int c_pid, sig;
   struct sigaction action;
   action.sa_handler = message;
   action.sa_flags = 0;

   sigemptyset(&action.sa_mask);
   sigaction(SIGUSR1, &action, NULL);
   sigaction(SIGINT, &action, NULL);

   if( c_pid = fork()){
     sigset_t ens1, ens;
     printf("\n Valeur de retour du Père: %d son PID %d. \n", c_pid, getpid());

     sigemptyset(&ens);
     sigaddset(&ens, SIGUSR1);
     sigaddset(&ens, SIGINT);
     sigprocmask(SIG_SETMASK, &ens, NULL);

    kill(getpid(), SIGUSR1);

    sigpending(&ens1);
      for(sig = 1; sig < NSIG; sig++)
        if(sigismember(&ens1, sig))
          printf(" \n message pendant %d \n", sig);
        putchar('\n');

    sleep(5);

    sigemptyset(&ens)
    sigprocmask(SIG_SETMASK, &ens, NULL);
   }
   else{
     printf("Valeur du retour du fork %d\n ", c_pid);
     printf("Son PID : %d, ", getpid());
     printf(" Le PID de son père : %d \n",getppid());
     sleep(1);

 kill(getppid(), SIGINT);

     sleep(1);
 }
     return 0;
  }
```