TP0: *Gestion des processus / signaux*

Remarque : Rendre un rapport TP contenant tout les exercices sur la plateforme google classroom

Les TPs système sont en monome.

Exercice 1

1. A partir d'une console, lancer un éditeur de texte et saisir le code du programme suivant.

```
// fils.c
#include <stdio.h>
#include <unistd.h>  // fork
#include <stdlib.h>  // exit

void main()
{
  int pid= fork();

  if (pid == - 1)
  { /* code si échec : */
     perror ( "fork " ) ;
    exit(1) ;    //sortir sur un code d'erreur
  }
  if (pid==0)
     {
       // Code du fils
       printf("Début fils \n");
       printf("Processus fils de pid=%d, ppid=%d\n", getpid(), getppid());
       sleep(6);
       exit(0);
       // Fin du processus fils
     }
  // Suite code du père, si pid > 0
  sleep(2);
  printf("Processus père de pid=%d, ppid=%d \n", getpid(), getppid());
}
```

2. Compiler et exécuter le programme généré. Que font les fonctions getpid(), getppid () et exit() ? modifier pour que le père affiche aussi le pid du fils crée. Utiliser man

3. Exécuter la commande shell ***ps aux*** à partir d'une autre fenêtre du terminal pour visualiser les processus crées. Trouver qui est le processus père de votre programme (le père ici).

4. Modifier le programme précédant pour que le père crée **n** fils. Chaque fils devra afficher un message du type *"fils de pid x : bonjour, le pid de mon pére est y"* au départ de son exécution,

puis dormir 6 secondes et afficher un message du type *"fils de pid x : au revoir, pid de mon père est y"* à la fin de son exécution.

5. Que remarquez-vous ? NB. vérifier les valeurs du pid du père au bonjour et au revoir de chaque fils.

6. **Processus orphelins :** Que ce passe-t-il quand un processus devient orphelin ? Que faut-il faire pour éviter cette situation ?

Exercice 2 : Les fonctions wait(..) et waitpid(..)

Écrire un programme dont le père, après avoir créé trois fils (*f*1, *f*2, *f*3), attend le retour de ces trois fils pour réaliser le calcul 3 × 10 + 5.
Les données :
- le fils *f*1 retourne la valeur 5;
- le fils *f*2 retourne la valeur 10;
- le fils *f*3 retourne la valeur 3.

Exercice 3

Écrire un programme qui crée l'arborescence suivante de processus : Le processus P0 crée deux fils p1 et p2, il 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** ou **pstree** (à partir d'un autre terminal) pour visualiser l'arbre réalisé (**ps -aef --forest, pstree -pl)** pour afficher uniquement l'arbre du processus P0, ajouter l'option –p pid à la commande ps.

Exercice 4 (Signaux)

NB. La **commande** kill -l permet d'obtenir la **liste** complète des **signaux** sous linux.

1. Ecrire un programme « infini.c » qui réalise une boucle infinie. Lancer le programme puis taper les touches "Ctrl+c". Que se passe-t-il ? quel est le signal associé à l'action Ctrl+C ?

2. Relancer le programme **infini**. Sur un autre terminal, déterminer le pid du processus correspondant au programme **infini** en utilisant "ps –A". Ensuite taper "kill -9 pid". Que se passe-t-il ? quel est le signal envoyé par kill -9 ?
3. Créer un programme « assassin.c ». Ce programme demande la saisie d'un pid, puis fait appel à la fonction kill(pid,SIGINT). Lancer le programme infini. Sur un autre terminal, déterminer le pid du processus correspondant à **infini**. Lancer **assassin** et saisir le pid de infini. Que se passe-t-il ?
4. Quel est le comportement par défaut d'un processus a la réception d'un signal (selon le signal reçu)?
5. Modifier le programme **infini** de telle sorte qu'il affiche un message « Hello ! I am here » au lieu de se terminer lorsqu'il reçoit un signal **SIGINT** (par Ctrl+c ou par kill).

Exercice5 (SIGUSR1, SIGUSR2) (à rendre dans le rapport du TP)

Écrire un programme dans lequel le processus principal crée un processus fils. Les deux processus communiquent en utilisant le signal **SIGSUR1**. Le fils affiche les lettres miniscules de a à z, le père affiche des majuscules de A à Z. Ce programme doit être écrit de telle sorte que le texte affiché sera comme suit:
"AabcBCdefDEFghijGHIJklmnoKLMNOpqrstuPQRSTUvwxyzVWXYZ"

Exercice 6 :

Écrire un programme qui crée trois processus fils : H, M, S (Heure, Minute, Seconde). Chacun des trois processus possédera un compteur initialise a une heure quelconque.
Le processus S incrémente son compteur de 1 toutes les 1 seconde pendant que M et H sont bloqués avec la fonction pause (). Quand le compteur de S atteint 60, il le remet à zéro et envoie un signal SIGUSR1 au processus M qui va réagir en incrémentant à son tour son propre compteur et en vérifiant s'il atteint 60. Auquel cas, il remet à zéro et envoie un signal a H qui va incrémenter son propre compteur et appeler pause() encore une fois.