TD1 : Création de processus

Objectifs :

Utilisation des fonctions ***fork( ), exec ( ), wait( ) et waitpid( )***

Présentation :

Il s'agit se familiariser avec les fonctions ***fork( ), exec( ), wait() et waitpid()***

Exercice1 :

Complétez (père/fils) l'affichage dans le programme suivant :

```
int main()
{
  int pid;
  pid = fork();
  if (pid == -1)
  {
    perror("fork"); exit(-1);
  }
  if (pid == 0) printf("Je suis le ...\n");
  if (pid != 0) printf("Je suis le ...\n");
}
```

Améliorez le programme en affichant les PID/PPID de chaque processus (voir les appels getpid() et getppid()) avec les messages suivants : "Je suis le père, mon PID est x et j'ai créé un fils dont le PID est x" et "Je suis le fils, mon PID est x et mon père a le PID x".

➔ *Analysez les résultats obtenus*

Exercice2 :

Écrire un programme qui crée **trois processus fils**. Chaque fils doit afficher son PID et ″dormir″ pendant un temps différent.

➔ *Analysez les résultats obtenus*

Exercice3 :

Cet exercice va consister à créer plusieurs processus fils, mémoriser leurs PID dans le père. Chaque processus fils devra exécuter le code suivant :

```
int fils(int valeur)
{
        int max = 1000 * valeur;
        int i;
        for (i = 0; i < max; i++)
```

```
                {
                }
        printf("\tFils N° %d : fin max = 0x%x\n", valeur, max);
        return max;
}
```

Une fois que le père a créé tous ses fils, il devra attendre la fin de chaque fils, déterminer si ce fils s'est bien terminé puis afficher la valeur retournée par ce fils. Le père devra attendre la fin de tous ses fils.

Exercice 4 : ***Le recouvrement :***

À partir d'un programme en C, il est possible d'exécuter des processus de plusieurs manières avec soit l'appel system :

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
  printf("Résultat de la commande ps fl :\n");
  system("ps fl");
  printf("Fin\n");
  return 0;
}
```

soit les appels exec :

```
#include <stdio.h>
#include <unistd.h>
int main()
{
  printf("Résultat de la commande ps fl :\n");
  execlp("ps", "ps", "fl", NULL); // cf. man execlp
  printf("Fin\n");
}
```

- Comparez au niveau des PID/PPID l'exécution des deux exemples ci-dessus. Quelle conclusion faites-vous ?
- Dans l'exemple 2, le message "Fin\n" ne s'affiche pas. Pourquoi ?

Exercice 5 : Compléter de manières différentes le programme ci-dessous pour faire en sorte que le processus administrateur se termine obligatoirement après :

1. La fin de l'une des deux tâches, principale ou secondaire.
2. La fin de la tâche principale.
3. La fin de la tâche secondaire.
4. La fin des deux tâches, principale et secondaire

```
int main(){
        int pp, ps;
        printf("Je suis le processus administrateur de PID %d\n", getpid());
        pp=fork();
        switch(pp) {
                case -1: printf("echec du fork\n"); exit(-1);
                case 0: printf("Je suis le processus fils de PID %d qui exécute la tâche principale\n",
                                                                                        getpid());
                                sleep(2);//exécution de la tâche principale
                                printf("Fin de la tâche principale\n");
                                exit(1);
                default: ps=fork();
                                switch(ps) {
                                        case -1: printf("echec de fork"); exit(-2);
                                        case 0: printf("Je suis le processus fils de PID %d qui exécute la
                                                                                tâche secondaire\n", getpid());
                                        sleep(10);// exécution de la tâche secondaire
                                        printf("Fin de la tâche secondaire\n");
                                        exit(2);
                                        default: printf("Je suis le processus administrateur, j'attends la
                                                                                fin de .....\n");
                                                .....
                                                .....
                                                .....
                                                printf("Fin du processus administrateur\n");
                                }
        }
return(0);
```