Université IBN-ZOHR
Faculté Polydisciplinaire-Ouarzazate
Département de Mathématiques, Informatique et Gestion

Année Universitaire : 2019-2020
Filière : SMI
Semestre : S4
Module : Systèmes d'exploitation II

TP1

Exercice 1 : Création de processus fils avec la primitive *fork()*

La primitive ***fork()*** permet la création d'un nouveau processus qui s'exécute de façon concurrente avec le processus qui l'a crée.

```
# include <unistd.h>
#include <stdio.h>
int main()
{
int valeur_fork;
valeur_fork = fork();   //Création du processus fils
switch (valeur_fork) {
case -1 :
        printf ("Problème lors de la création du processus \n");
        break;
case 0 : /* on est dans le processus fils*/
        printf("Je suis le processus fils, valeur de la fonction fork = %d \n",valeur_fork);
        break;
default : /* on est dans le processus père*/
        printf ("Je suis le processus père\n");
        printf ("Je viens de créer le processus fils dont le PID est %d \n",valeur_fork);
        break;
}
return valeur_fork;
}
```

Solution :

- Le programme ci-dessous permet la création d'un processus fils.
- La primitive *fork* retourne :
  - -1 : s'il y a une erreur de création
  - 0 : si on est dans le processus fils
  - PID du processus fils > 0 : si on est dans le processus père
- La primitive *fork* crée une copie du processus appelant.
- Les processus père et fils ont alors la même image mémoire et les mêmes fichiers ouverts.
- Selon la valeur retournée par la primitive fork, on exécute le code convenable (la structure *switch*).

Exercice 2 :

Réécrire le programme de l'exercice précédant tout en affichant le PID (Identifiant du processus) et PPID (Identifiant du processus père) du processus créé. Pour cela on utilise les primitives suivantes :

- *getpid()* : Retourne l'identifiant du processus.
- *getppid()* : Retourne l'identifiant du processus père.

Exercice 3 : **Processus concurrents**

Le programme suivant créera deux processus qui vont modifier la variable ***a***.

```
# include <unistd.h>
# include <stdio.h>
int main ()
{
int valeur_ fork;
int a=20;
valeur_ fork = fork(); // création d'un processus fils
switch (valeur_ fork)
{
case -1:
        printf ("Problème lors de la création du processus \n");
        break;
case 0: /* on est dans le processus fils*/
        printf ("Je suis le processus fils, le PID %d.\n",getpid()) ;
        a+=10;
        break ;
default : /* on est dans le processus père*/
        printf ("Je suis le processus père, le PID %d.\n",getpid());
        a+=100;
}
// les deux processus exécutent cette instruction
printf ("Fin du processus %d avec a = %d.\n", getpid(),a) ;
return 0 ;
}
```