TD Gestion de processus

Exercice 1

Ecrire un programme C qui permet à un processus de créer 2 processus fils. Les processus fils effectuent le traitement suivant :

- affichage d'un message indiquant qu'ils sont les fils et précisant leur PID,
- temporisation avec la fonction `unsigned int sleep(unsigned int seconds)` de la bibliothèque `unistd.h`, puis fin.

Le processus père effectue le traitement suivant :

- affichage d'un message indiquant qu'il est le père et précisant son PID,
- attente de la terminaison du premier fils crée,
- affichage d'un message indiquant le réveil du père après la terminaison de son fils, puis fin.

Exercice 2

Donner un exemple d'exécution du programme C suivant :

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

int main (void)
{
int n = 100 ;

printf(«  bonjour --> ») ;
n *= 2 ;
if (fork() == 0) {
     sleep(1) ;
     printf(« dans le fils, adresse de n = %p\n » , &n) ;
     n += 10 ;
     sleep(1) ;
     printf(« n = %d\n », n) ;
     }
else {
     printf(« dans le pere, adresse de n = %p\n », &n);
     n += 20 ;
     sleep(3) ;
     printf(« n = %d\n », n) ;
     }
exit(0) ;
}
```

Exercice 3

Soit le programme suivant :

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

```
int n = 1000;
int main (void) {
int m = 500 ;

/* processus P1 */
printf(« Adresse de n dans le processus P1 :%p\n », &n) ;
printf(« Adresse de m dans le processus P1 :%p\n », &m) ;
printf(« 1 : Valeur de m et n dans le processus P1  → %d %d \n », m,
n) ;
     switch (fork ()) {
case (pid_t)-1 :
   printf(" Erreur ") ;
   exit(2) ;
case (pid_t) 0 : /* processus P2*/
   printf(" Adresse de n dans le processus P2 :%p\n ", &n) ;
   printf(" Adresse de m dans le processus P2 :%p\n ", &m) ;
   printf(" 2 : Valeur de m et n dans le processus P2 → %d %d \n ",
   m, n) ;
   m *= 2 ; n *= 2 ;
   printf(" 3 : Valeur de m et n dans le processus P2 → %d %d \n ",
   m, n) ;
   sleep(3) ;
   printf(" 6 : Valeur de m et n dans le processus P2 → %d %d \n ",
   m, n) ;
   exit(0) ;
default : /* processus Px */
   sleep(2) ;
   printf(" 4 : Valeur de m et n dans le processus Px → %d %d \n ",
   m, n) ;
   m *= 3 ; n *= 3 ;
   printf(" 5 : Valeur de m et n dans le processus Px → %d %d \n ",
   m, n) ;
   sleep(2) ;
   exit(0) ;
     }
}
```

1. A quel processus correspond le processus Px : P1 ou P2 ? Justifier la réponse.
2. Donner un exemple d'exécution de ce programme dans le cas où l'appel système `fork()` réussit.

Exercice 4

Ecrire un programme C qui permet à un processus de créer dix processus fils. Chaque processus fils affiche le double de la valeur de son PID à l'écran et se termine.