ISIMM 2023/2024	1 Linfo

TP 1 Système exploitation 2
Fork() , exit(), wait()

Exercice 1 :

Pour chaque exemple, tester le code et noter le nombre de processus générés en expliquant le résultat obtenu (que fait chaque processus).

Exemple 1 :

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

int main() {

printf("Coucou %d \n", getpid());
fork();
printf("Mon PID %d : PID de mon père %d\n", getpid(), getppid());

return 0;

}
```

Exemple 2 :

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

int main() {

printf("Coucou %d \n", getpid());
fork();
fork();
printf("Mon PID %d : PID de mon père %d\n", getpid(), getppid());

return 0;

}
```

Exemple 3 :

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

int main() {

pid_t pid;

pid = fork();

if (pid == -1)
   printf("Echec de création\n");
else if (pid == 0)
   printf("je suis le fils et je dis bonjour à mon père, mon PID est %d \n", getpid());
else
   printf("je suis le père et je dis bonjour à mon fils, mon PID est %d \n", getpid());

return 0;

}
```

Exemple 4 :

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

int main() {

pid_t pid;

int i=5, j=2;

pid = fork();

if (pid == -1)
   printf("Echec de création\n");
else if (pid == 0)
   {
   printf("je suis le fils, mon PID est %d \n", getpid());
   j--; i=5 ; j=4
   }
else
   {
   printf("je suis le père, mon PID est %d \n", getpid());
   i++;
   }

 printf("PID est %d , i= %d , j= %d \n", getpid(), i, j);

return 0;

}
```

Exercice 2 :

Combien de « hello » affiche chaque programme. Préciser le nombre de processus générés.

Programme 1 :

```
int main() {
  int i;

  for (i=0; i<2; i++)
    fork();
  printf("hello!\n");
  exit(0);
}
```

Programme 2 :

```
void doit() {
  fork();
  fork();
  printf("hello!\n");
}
int main() {
  doit();
  printf("hello!\n");
  exit(0);
}
```

Programme 3 :

```
int main() {
  if (fork())
    fork();
  printf("hello!\n");
  exit(0);
}
```

Programme 4 :

```
int main() {
  if (fork()==0) {
    if (fork()) {
      printf("hello!\n");
    }
  }
}
```

Exercice 3 :

Soit l'algorithme suivant:

```
(1)  #include <stdlib.h>
(2)  #include <unistd.h>
(3)  #include <errno.h>
(4)  #include <stdio.h>
(5)  int main() {
(6)  int compt=0 ;
(7)  pid_t   pid_fils1, pid_fils2 ;
(8)  printf("1-Compteur= %d \n",compt) ;
(9)  pid_fils1 = fork() ;
(10) if (pid_fils1 == -1) {
(11)        printf("Erreur") ;
(12)        return(1) ;
(13) }
(14) if (pid_fils1 == 0) {
(15)        compt++ ;
(16)        printf("2-Compteur %d\n",compt) ;
(17)        pid_fils2 = fork() ;
(18)        if (pid_fils2 == -1) {
(19)             printf(" Erreur ") ;
(20)             return(1) ;
(21)        }
(22)        if (pid_fils2 == 0) {
(23)             compt++ ;
(24)             printf("3-Compteur %d\n",compt) ;
(25)        }
(26) }else {
(27)        compt-- ;
(28)        printf("4-Compteur %d\n",compt) ;
(29)     }
(30) }
```

1. Donner le nombre de processus créés par ce programme
2. Donner, pour chaque processus, l'affichage effectué.

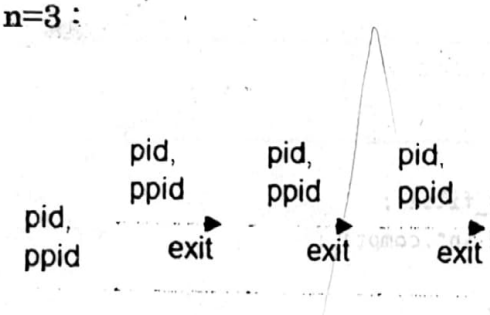
Exercice 4 :

Écrire un programme qui crée 10 processus fils. Chacun d'entre eux devra afficher dix fois d'affilié son numéro d'ordre entre 0 et 9. Vérifiez que votre programme affiche 100 caractères.

Exercice 5 :

On considère la structure de filiation chaîne représentée ci-dessous. Écrire un programme qui réalise une chaîne de n processus, où n est passé en paramètre de l'exécution de la commande (par exemple, n = 3 sur la figure ci-dessus). Afficher le numéro de chaque processus et celui de son père.

Exemple : chaîne avec n=3 :



Exercice 6 :

Même question de l'exercice 2 mais avec la structure arbre. Exemple un arbre avec n=3 :