Les primitives fork et wait

exercice.c

```
 1 : #include <stdlib.h>
 2 : #include <stdio.h>
 3 : #include <unistd.h>
 4 : #include <sys/types.h>
 5 : #include <sys/wait.h>
 6 : #include <time.h>
 7 : #define NB_PROCESSUS 3
 8 :
 9 : void filio(int delay)
10 : {
11 :     unsigned char n;
12 :     srand(getpid() * time(NULL));
13 :     n = rand() & 0xff;
14 :     printf("%d dit : j'ai choisi le numero %d\n", getpid(), n);
15 :     sleep(delay);
16 :     exit(n);
17 : }
18 :
19 : void padre(void)
20 : {
21 :     int i, status, pid;
22 :     for (i = 0; i < NB_PROCESSUS; i++)
23 :     {
24 :         pid = wait(&status);
25 :         if (WIFEXITED(status))
26 :             printf("%d dit : le fils %d a choisi le numero %d\n", getpid(),
27 :                    pid, WEXITSTATUS(status));
28 :     }
29 : }
30 :
31 : int main(void)
32 : {
33 :     int i;
34 :
35 :     for (i = 0; i < NB_PROCESSUS; i++)
36 :     {
37 :         switch (fork())
38 :         {
39 :         case -1:
40 :             exit(EXIT_FAILURE);
41 :         case 0:
42 :             filio(2 * NB_PROCESSUS - i);
43 :         default:;
44 :         }
45 :     }
46 :     padre();
47 :     exit(EXIT_SUCCESS);
48 : }
```

Exercice 1. Que se passe-t'il si lorsque l'on exécute le programme précédent ?

Exercice 2. Que se passe-t'il si l'on déplace la ligne 12 en 34 ?

Exercice 3. Écrire une version équivalente du programme précédent sans utiliser la fonction `wait`.

Exercice 4 : course de processus. Écrire un programme qui lance dix fils qui effectuent une « course ». A la fin du programme, l'ordre des fils est affiché (chaque fils effectuera, par exemple une boucle vide de 10000 tours). Modifier le programme pour que les processus fils attendent le signal `SIGUSR1` avant commencer la course.

Exercice 5. Écrire un programme qui crée trois fils puis envois dix fois le signal `SIGUSR1` aléatoirement à l'un des fils. Chaque fils comptabilise le nombre de signaux reçus. Le père affiche le score de chaque fils.

Exercice 6. Écrire un programme qui lance deux fils, qui eux même lancent un fils. L'arbre généalogique (avec le numéro de processus) est finalement affiché. On n'utilisera pas la valeur de retour de `fork` ou `wait` : chaque processus affiche son propre numéro.

Exercice 7. Écrire une version plus générale ou le programme lance n_1 fils qui eux-mêmes lancent n_2 fils, ..., qui eux mêmes lancent n_k fils. On supposera, par exemple, que l'on a une variable globale `nofils` de type tableau d'entiers contenant le nombre de fils $n_1, n_2, \ldots, n_k$ pour chaque génération.