Les primitives fork et wait

Exercice 1. Considérer le programme suivant :

ex2.c

```
 1 : #include <stdio.h>
 2 : #include <unistd.h>
 3 :
 4 : int i = 0;
 5 :
 6 : int main(void)
 7 : {
 8 :   if (fork())
 9 :     printf("Je suis le père, i vaut %d\n", ++i);
10 :   else
11 :     printf("Je suis le fils, i vaut %d\n", ++i);
12 :
13 :   return 0;
14 : }
```

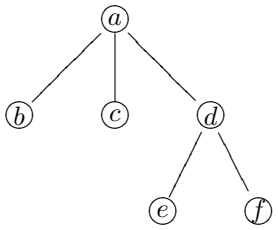
Dire ce que est affiché par ce programme, et dans quel ordre.

Exercice 2. Dessiner l'arbre généalogique des processus engendrées par le programme suivant.

ex1.c

```
1 : #include <unistd.h>
2 :
3 : int main(void)
4 : {
5 :   fork() && (fork() || fork());
6 :   return 0;
7 : }
```

Exercice 3. Écrire un programme qui engendre l'arbre généalogique suivant :



Exercice 4.

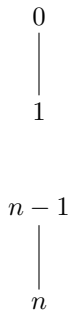
exercice.c

```
 1 : #include <stdlib.h>             /* exit, srand, rand */
 2 : #include <stdio.h>              /* printf */
 3 : #include <unistd.h>             /* fork, getpid, sleep */
 4 : #include <sys/wait.h>           /* wait, WIFEXITED, WEXITSTATUS */
 5 : #include <sys/types.h>          /* pid_t */
 6 : #include <time.h>               /* time */
 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", (int) getpid(), n);
15 :   sleep(delay);
16 :   exit(n);
17 : }
18 :
19 : void padre(void)
20 : {
21 :   int i, status;
22 :   pid_t pid;
23 :   for (i = 0; i < NB_PROCESSUS; i++)
24 :   {
25 :     pid = wait(&status);
26 :     if (WIFEXITED(status))
27 :       printf("%d dit : le fils %d a choisi le numero %d\n", getpid(), pid,
28 :              WEXITSTATUS(status));
29 :   }
30 : }
31 :
32 : int main(void)
33 : {
34 :   int i;
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 : }
```

– Que se passe-t'il lorsque l'on exécute le programme précédent ?
– Que se passe-t'il si l'on déplace la ligne 12 en 34 ?

Exercice 5 : 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 n tours, avec n choisit à hasard entre 5000 et 10000).

Exercice 6. Écrire un programme qui, pour un $n \geq 0$ donné, engendre l'arbre généalogique suivant :



Chaque processus choisit un nombre à hasard entre 0 et 127.
Le processus 0 affichera la plus grand valeur choisie par les processus $0, \ldots, n$.

Exercice 7. 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 8. Écrire une version plus générale où 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.