

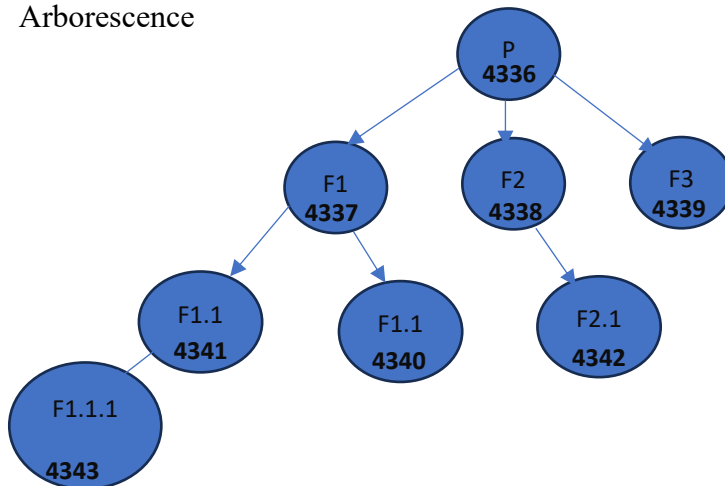
# Correction TD1

## Exercice 1

### P1 :

<pre>Open  [icon]  *P1.c ~/Desktop  1 #include &lt;stdio.h&gt; 2 #include &lt;unistd.h&gt; 3 #include &lt;stdlib.h&gt; 4 int main() { 5     int i; 6     for (i = 0; i &lt; 3; i++) 7         fork(); 8     printf("plop %d \n", getpid()); 9     exit(0); 10 } 11</pre>	<pre>aymen@aymen-virtual-machine:~/Desktop\$ gcc P1.c aymen@aymen-virtual-machine:~/Desktop\$ ./a.out plop 4336 plop 4337 plop 4338 plop 4339 plop 4341 plop 4340 plop 4342 plop 4343 aymen@aymen-virtual-machine:~/Desktop\$</pre>
--	---

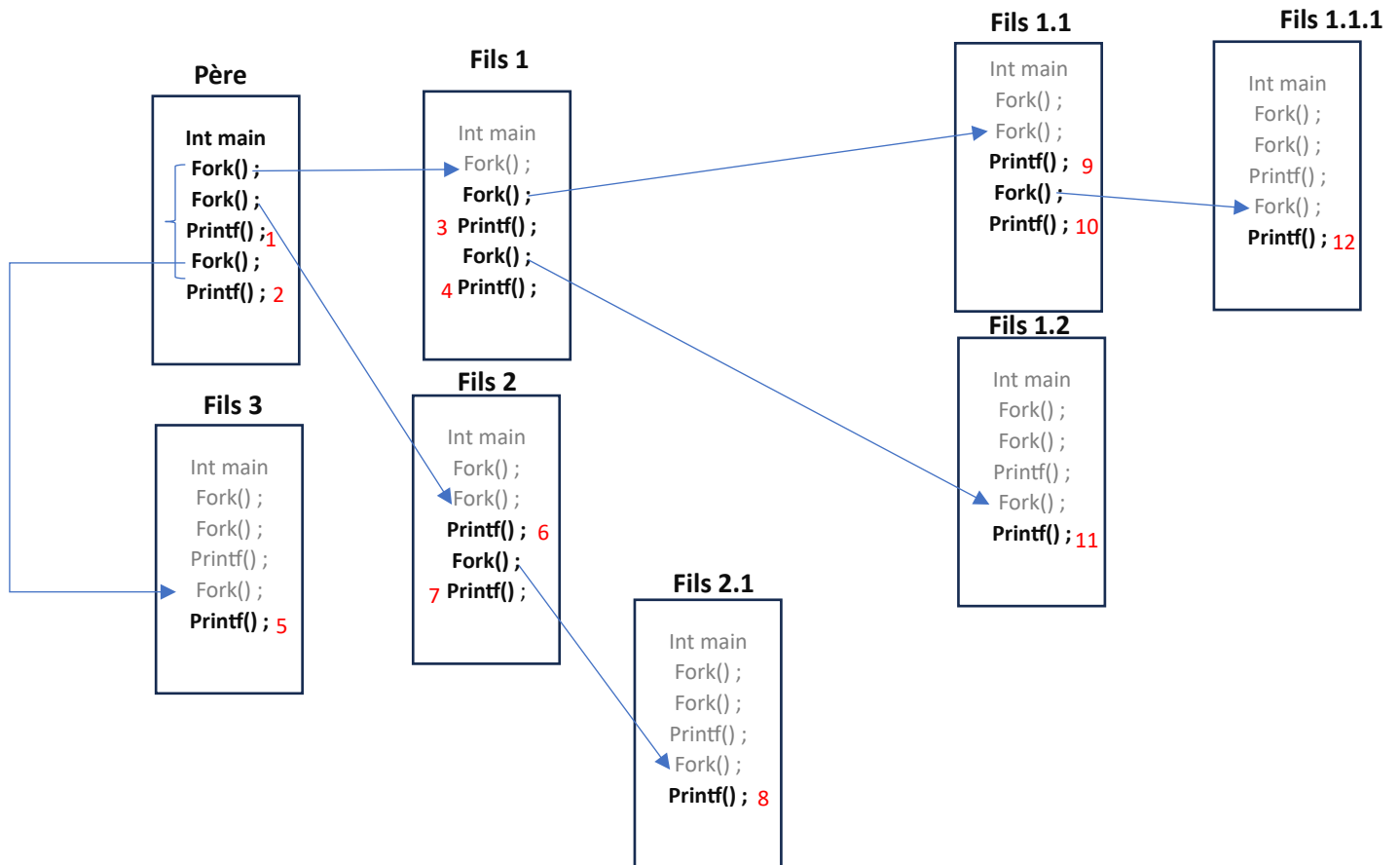
1. Nombre d'affichages : 8 affichages
2. Arborescence



### P2 :

<pre>Open  [icon]  P2.c ~/Desktop  1 #include &lt;stdio.h&gt; 2 #include &lt;unistd.h&gt; 3 #include &lt;stdlib.h&gt; 4 void doit(){ 5     fork(); 6     fork(); 7     printf("plop %d \n", getpid()); 8     fork(); 9 } 10 int main() { 11     doit(); 12     printf("plop %d \n", getpid()); 13     exit(0); 14 }</pre>	<pre>aymen@aymen-virtual-machine:~/Desktop\$ gcc P2.c aymen@aymen-virtual-machine:~/Desktop\$ ./a.out plop 4641 plop 4642 plop 4643 plop 4641 plop 4642 plop 4643 plop 4646 plop 4645 plop 4644 plop 4647 plop 4644 plop 4648 aymen@aymen-virtual-machine:~/Desktop\$</pre>
---	---

1. Nombre d'affichages : 12 affichages
2. Arborescence

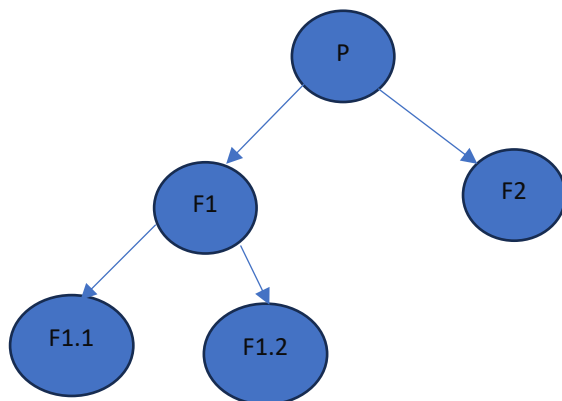
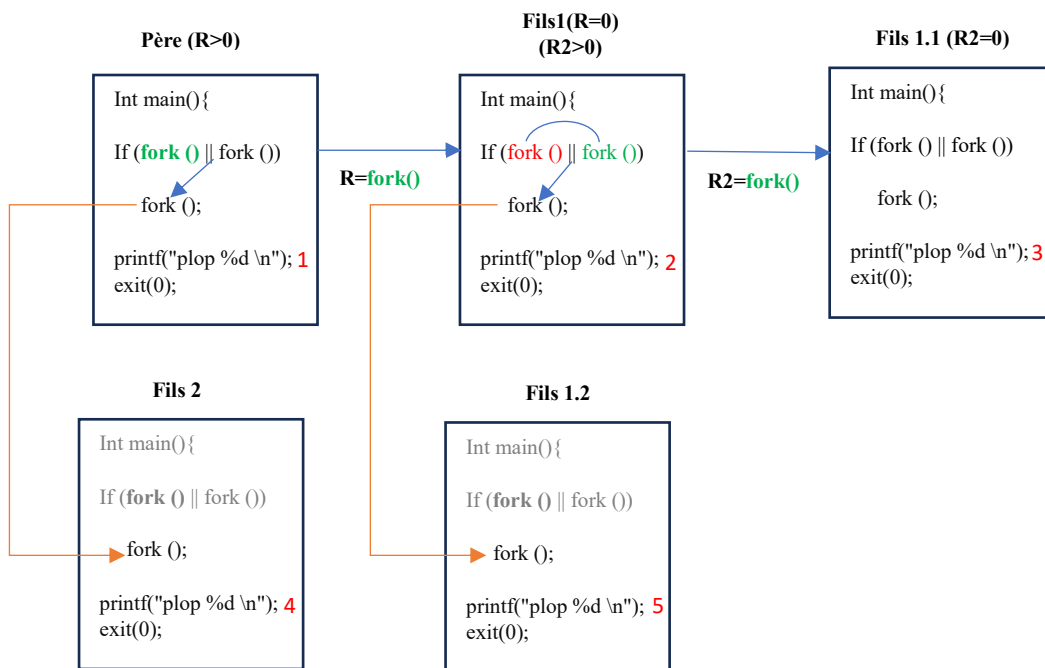


### P3 :

<pre> 1 #include &lt;stdio.h&gt; 2 #include &lt;unistd.h&gt; 3 #include &lt;stdlib.h&gt; 4 5 int main() { 6     if(fork()  fork()) 7         fork(); 8     printf("plop %d \n", getpid()); 9     exit(0); 10 } </pre>	<pre> aymen@aymen-virtual-machine:~/Desktop\$ gedit P3.c aymen@aymen-virtual-machine:~/Desktop\$ gcc P3.c aymen@aymen-virtual-machine:~/Desktop\$ ./a.out plop 5125 plop 5127 plop 5126 plop 5128 plop 5129 aymen@aymen-virtual-machine:~/Desktop\$ </pre>
---	--

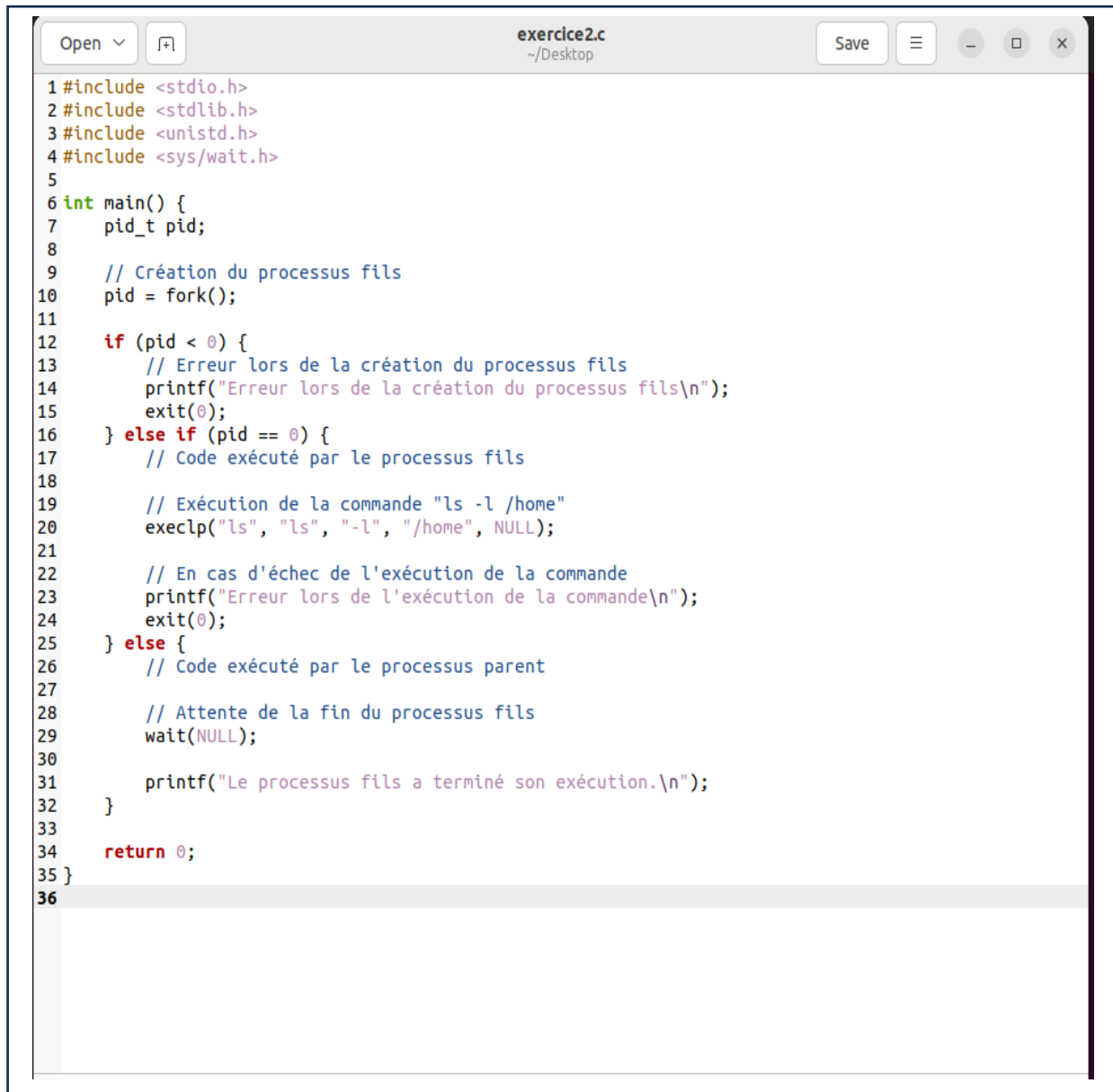
1. Nombre d'affichages : **5 affichages**
2. Arborescence

La fonction OR (||) fonctionne si au moins une condition est vraie (la valeur de retour de fork>0)



## Exercice 2 :

1.



```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <unistd.h>
4 #include <sys/wait.h>
5
6 int main() {
7     pid_t pid;
8
9     // Création du processus fils
10    pid = fork();
11
12    if (pid < 0) {
13        // Erreur lors de la création du processus fils
14        printf("Erreur lors de la création du processus fils\n");
15        exit(0);
16    } else if (pid == 0) {
17        // Code exécuté par le processus fils
18
19        // Exécution de la commande "ls -l /home"
20        execlp("ls", "ls", "-l", "/home", NULL);
21
22        // En cas d'échec de l'exécution de la commande
23        printf("Erreur lors de l'exécution de la commande\n");
24        exit(0);
25    } else {
26        // Code exécuté par le processus parent
27
28        // Attente de la fin du processus fils
29        wait(NULL);
30
31        printf("Le processus fils a terminé son exécution.\n");
32    }
33
34    return 0;
35 }
36
```

```
aymen@aymen-virtual-machine:~/Desktop$ gedit exercice2.c
aymen@aymen-virtual-machine:~/Desktop$ gcc exercice2.c
aymen@aymen-virtual-machine:~/Desktop$ ./a.out
total 8
drwxr-x--- 18 aymen aymen 4096 21:50 23 نوفمبر aymen
drwxr-x--- 15 stack stack 4096 09:23 23 نوفمبر stack
Le processus fils a terminé son exécution.
aymen@aymen-virtual-machine:~/Desktop$
```

## 2.

```
exercice2.c
~/Desktop

1#include <stdio.h>
2#include <stdlib.h>
3#include <unistd.h>
4#include <sys/wait.h>
5
6int main(int argc, char *argv[]) {
7    pid_t pid;
8
9    // Vérifier qu'au moins un argument (la commande à exécuter) est fourni
10   if (argc < 2) {
11       printf("Usage: %s <command> [args...]\n", argv[0]);
12       exit(0);
13   }
14
15   // Création du processus fils
16   pid = fork();
17
18   if (pid < 0) {
19       // Erreur lors de la création du processus fils
20       printf("Erreur lors de la création du processus fils\n");
21       exit(0);
22   } else if (pid == 0) {
23       // Code exécuté par le processus fils
24
25       // Exécution de la commande avec ses arguments
26       execvp(argv[1], &argv[1]);
27
28       // En cas d'échec de l'exécution de la commande
29       printf("Erreur lors de l'exécution de la commande\n");
30       exit(0);
31   } else {
32       // Code exécuté par le processus parent
33
34       // Attente de la fin du processus fils
35       wait(NULL);
36
37       printf("Le processus fils a terminé son exécution.\n");
38   }
39
40   return 0;
41 }
42
```

```
aymen@aymen-virtual-machine: ~/Desktop

aymen@aymen-virtual-machine:~/Desktop$ gcc exercice2.c -o exercice2
aymen@aymen-virtual-machine:~/Desktop$ ./exercice2
Usage: ./exercice2 <command> [args...]
aymen@aymen-virtual-machine:~/Desktop$ ./exercice2 ls -l /home
total 8
drwxr-x--- 18 aymen aymen 4096 21:50 23 نوفمبر aymen
drwxr-x--- 15 stack stack 4096 09:23 23 نوفمبر stack
Le processus fils a terminé son exécution.
aymen@aymen-virtual-machine:~/Desktop$
```

## Exercice 3 :

### P1 :

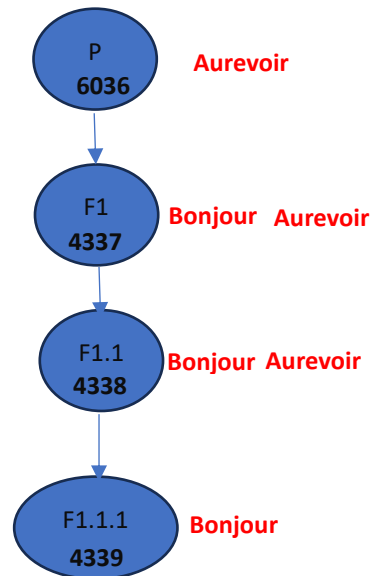
```
Open  [icon] *exercice3P1.c
~/Desktop

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <unistd.h>
4 #include <sys/wait.h>
5 #include <sys/types.h>
6 void traitement_bis(char *chaine){
7     printf ("%s\n", chaine);
8 }
9 int main(){
10     int pid=0 ,i;
11     for (i=0 ;i < 3 && !pid ; i++){
12         switch(pid=fork()){
13             case -1:
14                 printf (" Erreur de creation fork\n"); exit(-1);
15             case 0:
16                 traitement_bis("bonjour");
17             default:
18                 wait(NULL);
19         }
20     }
21     if (pid != 0)
22         printf ("Aurevoir\n");
23     return 0;
24 }
25
```

```
aymen@aymen-virtual-machine:~/Desktop$ gedit exercice3P1.c
aymen@aymen-virtual-machine:~/Desktop$ gcc exercice3P1.c
aymen@aymen-virtual-machine:~/Desktop$ ./a.out
bonjour :PID6037
bonjour :PID6038
bonjour :PID6039
Aurevoir :PID6038
Aurevoir :PID6037
Aurevoir :PID6036
aymen@aymen-virtual-machine:~/Desktop$
```

1. Le nombre de processus créés : **3 processus**
2. Le nombre total de processus : **4 processus**

3. La relation de parenté (l'arborescence des processus).



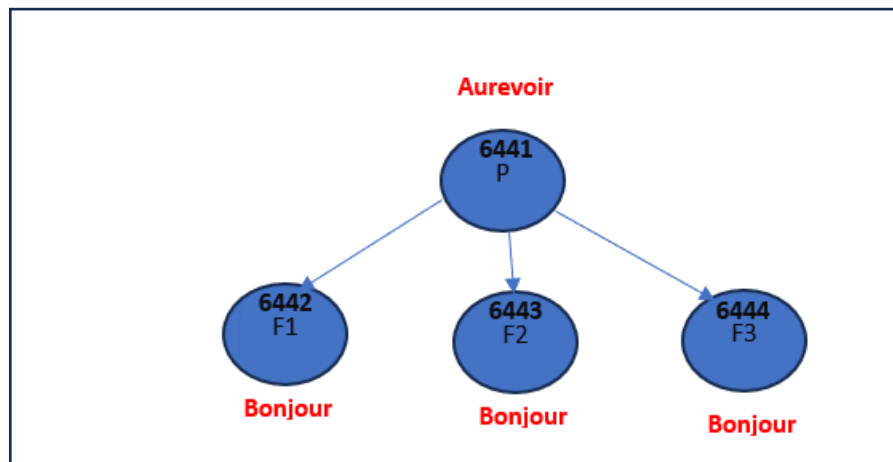
4. L'affichage obtenu : 3 « Aurevoir » et 3 « Bonjour ».

```
exercice3P2.c
~/Desktop

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <unistd.h>
4 #include <sys/wait.h>
5 #include <sys/types.h>
6 void traitement_bis(char *chaine){
7 printf ("%s\n", chaine);
8 exit(0);
9 }
10 int main(){
11 int pid=0 ,i;
12 for (i=0 ;i < 3 ; i++){
13 switch(pid=fork()){
14 case -1:
15 printf (" Erreur de creation fork\n"); exit(-1);
16 case 0:
17 printf("PID%d\n",getpid());
18 traitement_bis ("bonjour");
19
20 default:
21 wait(NULL);
22 }
23 }
24 printf ("Aurevoir :PID%d\n",getpid());
25 return 0;
26 }
27
```

```
aymen@aymen-virtual-machine:~/Desktop$ gedit exercice3P2.c
aymen@aymen-virtual-machine:~/Desktop$ gcc exercice3P2.c
aymen@aymen-virtual-machine:~/Desktop$ ./a.out
PID6442
bonjour
PID6443
bonjour
PID6444
bonjour
Aurevoir :PID6441
aymen@aymen-virtual-machine:~/Desktop$
```

1. Le nombre de processus créés : **3 processus**
2. Le nombre total de processus : **4 processus**
3. La relation de parenté (l'arborescence des processus).



4. L'affichage obtenu : **1 « Aurevoir »** et **3 « Bonjour »**.
5. Dans un algorithme d'ordonnancement non préemptif, chaque tâche est autorisée à s'exécuter sans être interrompue jusqu'à sa complétion. Par conséquent, les transitions qui impliquent l'interruption d'une tâche en cours d'exécution pour démarrer une autre tâche deviennent superflues. Ainsi, dans ce contexte, **la transition numéro 2** du diagramme d'états/transitions devrait être supprimée, car elle représente une interruption de la tâche en cours pour démarrer une autre tâche, ce qui n'est pas pertinent dans un système non préemptif.

