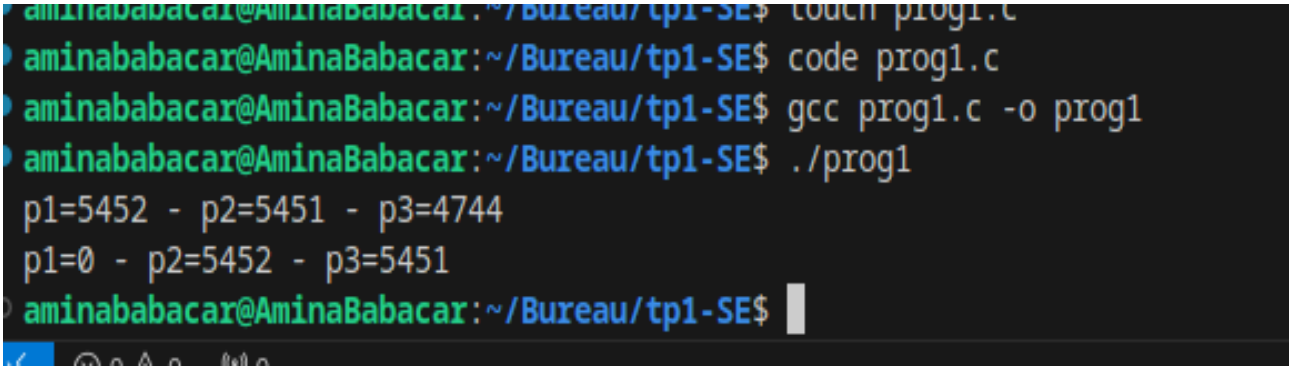
EXERCIE 1 Analysons et interprétation le code source

Après avoir complété les programmes suivants, dites ce qu'ils réalisent . Combien de processus sont crées?

Indiquons le résultat affiché à l'ecran.
Programme 1



Il crée un premier processus (p1) en appelant la fonction `fork()`. Après l'appel à `fork()`, deux processus sont créés : le processus parent et le processus enfant. La variable `p1` contient le PID (Process ID) du processus enfant dans le processus parent, et 0 dans le processus enfant.

Il obtient le PID du processus en cours à l'aide de la fonction `getpid()` et stocke la valeur dans la variable `p2`.

Il obtient le PID du processus parent à l'aide de la fonction `getppid()` et stocke la valeur dans la variable `p3`.

Il affiche les valeurs de `p1` (résultat de `fork()`), `p2` (PID du processus en cours), et `p3` (PID du processus parent).

En termes de processus créés :
- Un processus est créé par le programme lui-même au début.
- Ensuite, la fonction `fork()` crée un deuxième processus, faisant un total de deux processus : le processus parent et le processus enfant.

Ainsi, le programme crée deux processus. Notez que ces processus s'exécuteront de manière indépendante après la bifurcation (`fork()`), et chacun d'eux affichera les valeurs de `p1`, `p2`, et `p3`.

Programme 2
Au départ, un processus est créé.

- À chaque itération de la boucle, un nouveau processus est créé, jusqu'à ce que `i` dépasse N ou que `fork()` échoue.
- Chaque processus affiche la valeur finale de `i`.

Donc, le nombre total de processus créés dépend de la valeur de N et de combien de fois `fork()` réussit. Dans ce cas, avec N égal à 10, le programme pourrait créer jusqu'à 11 processus (1 initial + 10 itérations de la boucle). Cependant, le nombre réel peut être moindre si `fork()` échoue à un moment donné.

```
aminababacar@AminaBabacar:~/Bureau/tp1-SE$ gcc prog2.c -o prog2
aminababacar@AminaBabacar:~/Bureau/tp1-SE$ ./prog2
1
2
3
4
5
6
7
8
9
10
11
```

Programme 3
Au départ, un processus est créé.

Ensuite, trois processus fils sont créés successivement, formant une hiérarchie.

Par conséquent, le programme crée un total de quatre processus (1 initial + 3 itérations de la boucle `for`). Le tableau `pid` contiendra les ID des processus créés, et ces ID seront affichés à la fin du programme.

```
aminababacar@AminaBabacar:~/Bureau/tp1-SE$ gcc prog3r.c -o progr3
cc1: fatal error: prog3r.c: Aucun fichier ou dossier de ce type
compilation terminated.
aminababacar@AminaBabacar:~/Bureau/tp1-SE$ gcc progr3.c -o progr3
aminababacar@AminaBabacar:~/Bureau/tp1-SE$ ./progr3
32637 9046 9047
32637 9046 0
32637 0 9048
32637 0 0
aminababacar@AminaBabacar:~/Bureau/tp1-SE$
```

Programme 4
Le programme commence par afficher des informations sur le processus père, y compris son PID (getpid()) et le PID de son parent (getppid()).

Ensuite, il appelle `fork()` pour créer un premier processus fils. Le retour de `fork()` est stocké dans la variable `valeur`.

Le programme affiche des informations sur le processus fils nouvellement créé, y compris son PID et le PID de son parent.

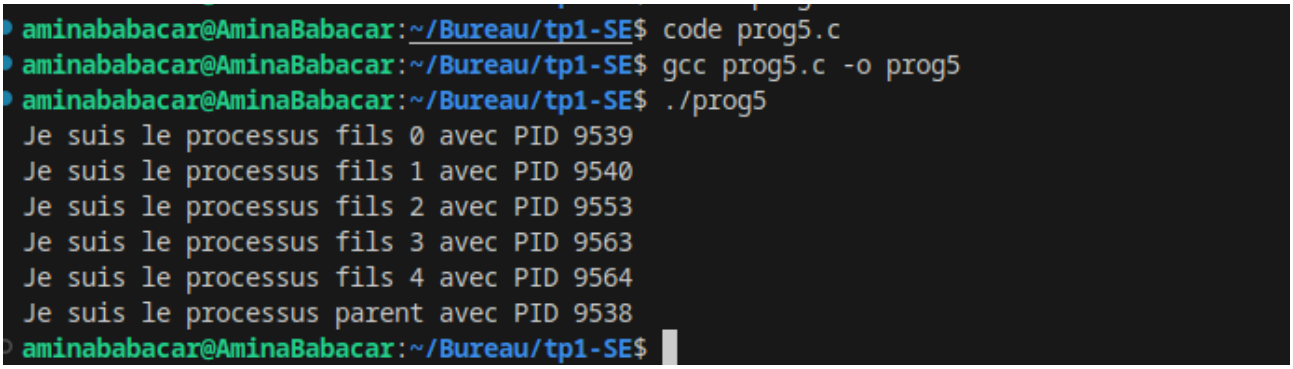
Ensuite, il appelle à nouveau `fork()` pour créer un deuxième processus fils à partir du premier processus fils. Le retour de cette deuxième `fork()` est stocké dans la variable `valeur1`.

Le programme affiche des informations sur le deuxième processus fils, y compris son PID et le PID de son parent.

On peut dire que , le programme crée une hiérarchie de processus. Chaque processus père crée deux processus fils successifs. Le nombre total de processus créés est de 2^2, soit 4. Chaque processus affiche des informations sur son PID, le PID de son parent, et le retour de la fonction `fork()`.

```
aminababacar@AminaBabacar:~/Bureau/tp1-SE$ gcc prog3.c -o prog3
aminababacar@AminaBabacar:~/Bureau/tp1-SE$ ./prog3
Affichage 1 --> Processus pere [7789] ; mon pere à moi est [4744]
Affichage 2 --> retour fork [7790] - Processus fils [7789] ; mon pere est [4744]
Affichage 2 --> retour fork [0] - Processus fils [7790] ; mon pere est [7789]
Affichage 3 --> retour fork [7791] - Processus fils [7789] ; mon pere est [4744]
Affichage 3 --> retour fork [7792] - Processus fils [7790] ; mon pere est [7789]
Affichage 3 --> retour fork [0] - Processus fils [7792] ; mon pere est [1900]
Affichage 3 --> retour fork [0] - Processus fils [7791] ; mon pere est [1900]
aminababacar@AminaBabacar:~/Bureau/tp1-SE$
```

Exercice 2


aminababacar@AminaBabacar:~/Bureau/tp1-SE$ code prog5.c
aminababacar@AminaBabacar:~/Bureau/tp1-SE$ gcc prog5.c -o prog5
aminababacar@AminaBabacar:~/Bureau/tp1-SE$ ./prog5
Je suis le processus fils 0 avec PID 9539
Je suis le processus fils 1 avec PID 9540
Je suis le processus fils 2 avec PID 9553
Je suis le processus fils 3 avec PID 9563
Je suis le processus fils 4 avec PID 9564
Je suis le processus parent avec PID 9538
aminababacar@AminaBabacar:~/Bureau/tp1-SE$