TD 1
l’appel système fork()

Exercice 1

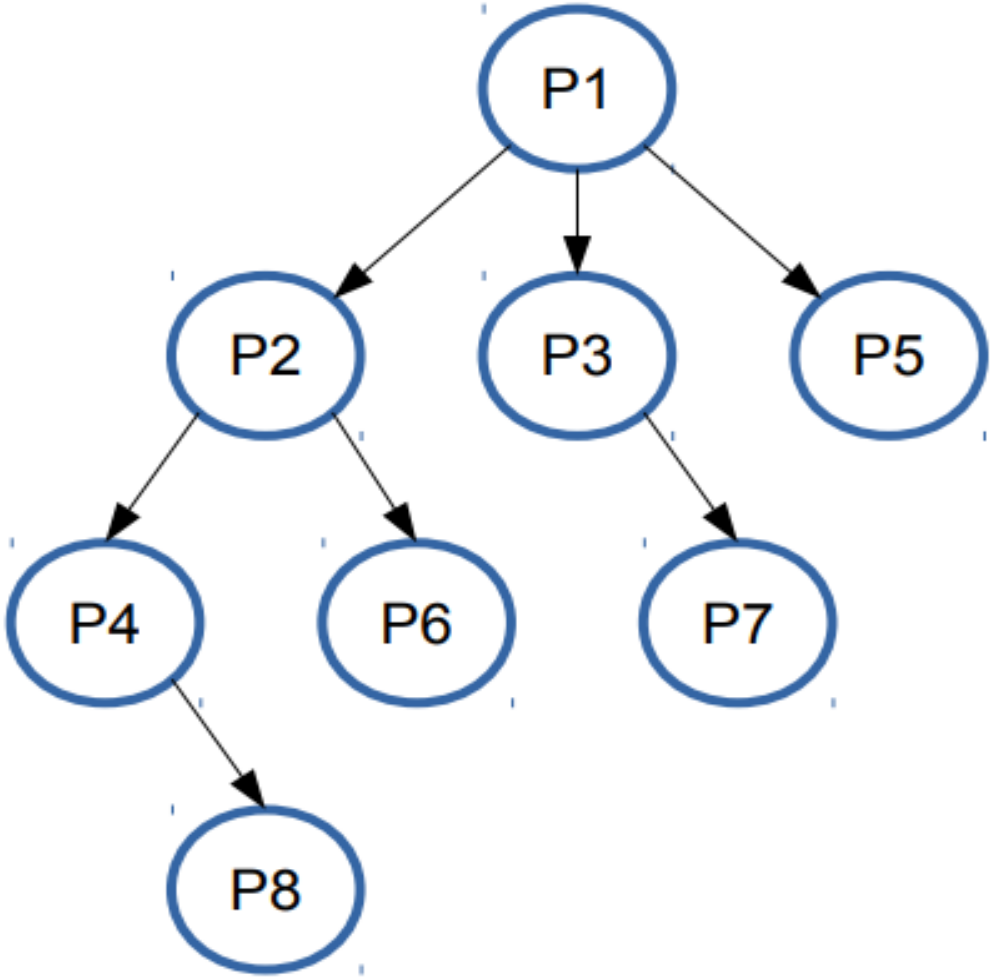
- Préciser le nombre de processus créer par les programmes ci-dessous :

Code1	Code 2	Code 3

`int main() {`
`  fork();`
`  fork();`
`  fork();`
`  }`
	`int main() {`
	`  if (fork() > 0)`
	`  fork();`
	`}`
		`int main() {`
		`  int cpt=0;`
		`  while (cpt < 3) {`
		`  if (fork() > 0)`
		`  cpt++;`
		`  else`
		`  cpt=3;`
		`  }`

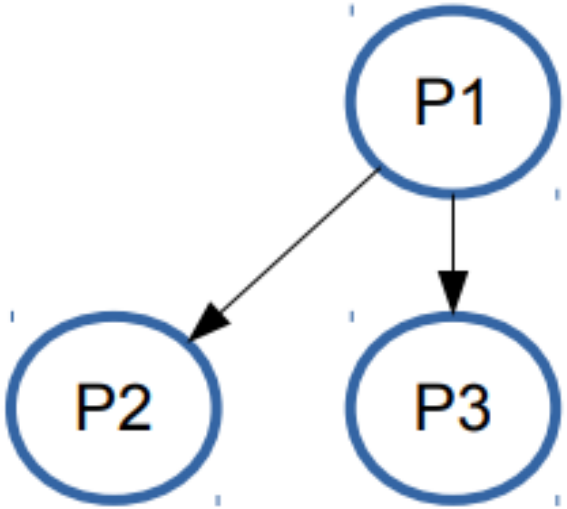
Correction : code 1

- 8 processus sont crées :
  - L'exécution du programme crée un processus P1.
  - A la lecture de la première instruction fork() , P1 se duplique et crée alors P2. Les deux processus continuent l'exécution à partir de la ligne incluse ;
  - A la lecture de la seconde instruction fork(), P1 se duplique et crée P3 tandis que P2 crée P4.
  - Les quatre processus continuent l'exécution a partir de la ligne incluse ;
  - A la lecture de la troisième instruction fork(), P1 se duplique et crée P5, P2 crée P6, P3 crée P7 et P4 crée P8



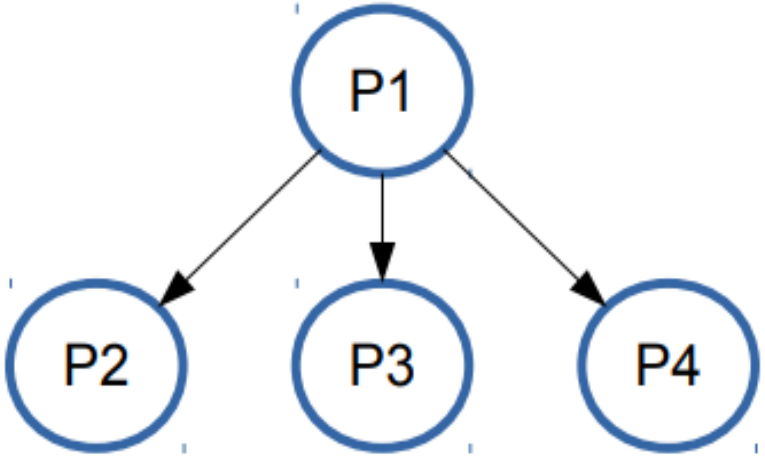
Correction : code 2

- 3 processus sont crées:
  - L'exécution du programme crée un processus P1.
  - A la lecture de la première instruction fork(), P1 se duplique et crée alors P2. P1 est le processus parent, P2 le processus enfant.
  - Les deux processus continuent l'exécution a partir de la ligne incluse ; Le résultat de l'appel précédent est supérieur à 0 pour P1. Ce dernier rentre donc dans la suite d'instructions conditionnée et exécute l'instruction fork().
  - P1 se duplique et crée donc P3.
  - En revanche, le résultat de l'appel précédent est égale a 0 pour P2, qui ne rentre donc pas dans la suite d'instructions conditionnée.



Correction : code 3

- 4 processus sont crées :
  - L'exécution du programme crée un processus P1, qui initialise la variable cpt a 0.
  - P1 rentre dans la boucle while() et se duplique lors de l'exécution de fork(). Il crée alors P2.
  - Le résultat de l'appel précédent est supérieur à 0 pour P1. Ce dernier rentre donc dans la suite d'instructions conditionné par "if" et incrémente son compteur cpt qui passe a 1.
  - En revanche, le résultat de l'appel précédent est égale a 0 pour P2, qui rentre donc dans la suite d'instructions conditionnée par else et affecte cpt à 3. Des lors P2 sort de la boucle et n'exécutera plus d'instruction.
  - Seul P1 ré-exécute la séquence d'instruction de la boucle while(), et le même schéma ce reproduit : a chaque entrée dans la boucle, P1 se duplique, tandis que le processus dupliqué n'exécute aucune instruction.
  - P1 aura ainsi exécuté 3 fois l'instruction fork() jusqu'a ce que sa variable cpt atteigne 3.
  - Il aura donc engendréé P2, P3 et P4



Exercice 2

```c
int main(int argc, char * argv[]) {
     pid_t pid;
     int attente_fils,attente_pere;

     if(argc != 3)
     {
          printf("usage: ex1 'attente_pere' 'attente_fils'\n");
          exit(-1);
     }

     attente_pere = atoi(argv[1]);
     attente_fils = atoi(argv[2]);

     switch(pid=fork()) {
          case -1:
               perror("fork error");
               break;
          case 0:
               sleep(attente_fils);
               printf("fils attente finie\n");
               break;
          default:
               sleep(attente_pere);
               printf("pere attente finie\n");
               break;
     }
     return 0;
}
```

Exercice 2

- 1- Lancer le programme ci-dessous avec les arguments 10 20. Que constatez-vous ? Donnez une explication.

- 2- Lancer le programme ci-dessous avec les arguments 10 0. Que constatez-vous ? Donnez une explication.

Correction : Exercice 2

- 1- Le père meurt avant son fils, le fils devient orphelin

- 2- Le fils meurt avant son père, le père est en sommeil, il ne lit pas le code de retour de son fils. Le fils devient zombie.