TD 1 : Notions de processus

Exercice 1.

a- Compléter (père/fils) dans l'affichage

```
int main()
{
   int pid;
   pid = fork();
   if (pid == -1)
   {
      perror("fork"); exit(-1);
   }
   if (pid == 0) printf("Je suis le ...\n");
   if (pid != 0) printf("Je suis le ...\n");
}
```

b- Améliorer le programme en affichant les PID/PPID de chaque processus (voir les appels getpid() et getppid()) avec les messages suivants : "*Je suis le père, mon PID est x et j'ai créé un fils dont le PID est x*" et "J*e suis le fls, mon PID est x et mon père a le PID x*".
Donner l'affichage obtenu.

Exercice 2.

Compléter le programme suivant pour qu'il lance une calculatrice (kcalc) et un éditeur de texte (kwrite).

```
if (! fork()) { printf("Fils 1 : je lance kcalc !\n"); ... }
if (! fork()) { printf("Fils 2 : je lance kwrite !\n"); ... }
```

Nota bene : il est possible que sur vos machines les noms des logiciels soient différents.

Exercice 3.

Tester et analyser le programme *status.c* fourni.

Exercice 4.

Compiler et tester les deux fichiers suivant :

- *tube1.c* : communication bidirectionnelle (pèrefils) avec un seul tube
- *tube2.c* : communication bidirectionnelle (pèrefils) avec deux tubes

Exercice 5.

On crée deux fonctions : une affichera des étoiles '*' et l'autre des dièses '#' (cf. fichier *threads.1.c)*.

a- Que constatez-vous ?

b- Proposer une solution à base de *threads* posix permettant de palier à ce problème.

Exercice 6.

Le fichier *threads.2.c* contient un code qui doit *incrémenter* puis *décrémenter* cinq fois un compteur, pour revenir au point de départ.

a- Compilez ce code et commentez le résultat de son exécution.

b- Pouvez-vous résoudre ce problème ?