Programmation système avancée
TP n° 1 : création de processus et exécution de commande

Rappelons les appels systèmes utiles :

- `fork()` pour la création d'un nouveau processus
- `wait()` pour attendre la terminaison d'un fils (et connaître les causes du décès)
- `execve()` (et ses différentes variantes de librairie : `execvp(), execl()` etc.) pour remplacer l'image mémoire du processus courant par une nouvelle

Exercice 1 : arbre de processus

Combien de processus sont créés par le code suivant ? Qu'est-ce qui est affiché ?

```
int main() {
    for(int i=0; i<10; i++) {
        fork();
        printf("%i\n",i);
    }
}
```

Exercice 2 : un père avait dix fils...

1. Écrivez un programme qui lance dix fils. À la fin du main, chaque processus (père ou fils) doit afficher
   - son *pid* et celui de son père (`getpid` et `getppid`) ;
   - un nombre aléatoire obtenu avec `rand`.

   Vérifiez que votre programme affiche 11 lignes. Qu'observe-t'on ? Comment l'éviter ?
2. Même question mais le père lance un seul fils ; ce fils lance un fils ; etc jusqu'à la dixième génération (l'ancêtre aura 10 descendants en tout)

Exercice 3 : ls -l

1. Expliquez l'effet de la commande shell `exec ls`.
2. Écrivez un programme qui exécute la commande `ls -l`. Votre programme devra seulement exécuter `/bin/ls`, sans créer de nouveau processus.

Exercice 4 : please

1. Écrivez un programme « `please` » prenant en argument un nom de programme `cmd` éventuellement suivi d'arguments (n'importe quels mots), et tel que `./please cmd [args]` affiche, dans une première ligne, le mot `Please,` suivi du texte de la commande ; puis le résultat de la commande ; puis une dernière ligne `"You are welcome!"`. Par exemple :

```
$ ./please /usr/bin/echo Hello, world!
Please, /usr/bin/echo Hello, world!
Hello, world!
You are welcome!
```

Vérifiez que la sortie du programme apparaît toujours entre la ligne « `Please...` » et la ligne « `You are...` ».

2. Modifiez votre programme pour qu'il affiche également

- `"Terminaison normale avec résultat n."`, où *n* est la valeur utilisée par `_exit(n)`,
- ou (exclusif, normalement) `"Tué par le signal s."`.

Testez votre programme en utilisant la commande shell `kill`.

Exemple :

```
$ ./please /usr/bin/xeyes
Please, /usr/bin/xeyes
You are welcome!
Tué par le signal 15.
```

3. Modifiez votre programme pour que, toutes les secondes, le père affiche (au milieu des éventuels affichage du fils) le temps écoulé (en secondes) jusqu'à la mort du fils, par exemple :

```
$ ./please /usr/bin/xeyes
Please, /usr/bin/xeyes
1s and running...
2s and running...
3s and running...
4s and running...
You are welcome!
Tué par le signal 15.
```

Le père attendra en utilisant la fonction `sleep` et non en faisant une attente active. Votre programme attendra probablement jusqu'à une seconde de trop. (Comment pourrait-on éviter ce problème ?)