Exercice 1

1. Exécutez la commande « **ps** » et interprétez le résultat obtenu.
2. Créer un script qui affiche bonjour toutes les 15 secondes. Lancer ce script en arrière-plan (bonjour.sh &), et afficher son PID (**ps**), puis son numéro de job (**jobs -l, jobs -p**).
3. Pour remettre le script au **premier plan**, on peut utiliser la commande **fg** ("foreground"). Pour basculer le script en **tâche de fond** tapez, **CTRL+Z** puis tapez **jobs -l** pour afficher l'état du job.
4. Tapez ensuite **bg** (background), vous voyez s'afficher „bonjour".
5. Tuer ce processus en utilisant son PID ou son numéro de job.
6. Lancer (**nohup**) à nouveau ce script en détaché du terminal (insensible à la déconnexion).
7. Déconnectez-vous, et connectez-vous. Affichez vos processus en tapant: **ps** Le processus **bonjour** n'apparait pas, pourquoi?
8. Tuez le processus « bonjour » et détruisez son fichier de sortie.

Exercice 2

Donner le nombre des messages « Hello » affichées.

```
int main() {
   fork();
   printf("hello!\n");
   exit(0);
}
```

```
int main() {
   fork();
   fork();
   printf("hello!\n");
   exit(0);
}
```

Exercice 3

Écrire un programme créant un processus fils. Le processus père affichera son identifiant ainsi que celui de son fils, le processus fils affichera son identifiant ainsi que celui de son père. Quel est le PID du père affiché par le fils ?

Exercice 4

Afin de ralentir l'exécution du père pour que le fils termine son exécution avant le père, ajouter un appel de la fonction sleep() de façon à afficher le PID du père par le fils.

Exercice 5

Ajouter un appel de la fonction wait(NULL) pour éviter que le père ne se termine pas avant le fils.

Exercice 6

Dans l'exercice précédent, utiliser wait() pour obtenir le code de retour du fils et l'afficher.

Exercice 7

Ecrire un programme qui crée 10 processus fils. Chacun d'entre eux devra afficher dix fois d'affilée son numéro d'ordre entre 0 et 9.