Programmation Système

# TP 4 : Communication et synchronisation entre les processus et threads sous Unix

**Exercice 1**

1) Ecrire un programme C qui crée deux processus à l'aide de l'appel système `fork()`. Le père affichera les entiers pairs compris entre 1 et 100, le fils affichera les entiers impairs compris dans le même intervalle. Synchroniser les processus à l'aide des signaux pour que l'affichage soit 1 2 3 ... 100.
2) Le signal SIGCHLD est un signal qui est automatiquement envoyé par le fils à son père lorsque le fils se termine (par un exit, un return, ou autre). **Ajoutez une fonction** et **le code nécessaire** afin que le père n'attende jamais son fils de façon bloquante et que le fils ne devienne pas zombie.

```
/*0*/
int main(int argc, char *argv[])
{
/*1*/
if (!fork())
{
/*2*/
for (int i = 0 ; i <10 ; i++) ; //simule un petit calcul
/*3*/
exit(1) ;
/*4*/
}
/*5*/
while(1) ; //Simule un calcul infini
/*6*/
}
```

**Exercice 2**

Ecrire un programme C équivalente à la commande shell

```
ps -uax | grep root | wc -l
```

**Exercice 3**

Ecrire une application client-serveur qui utilise une file de messages

- Un serveur qui lit des questions comportant deux nombres et envoie leur somme en réponse. Le type de chaque réponse est le pid du client correspondant.
- un client qui envoie des questions au serveur et lit ses réponses.

**Exercice 4**

Écrire un programme qui lance 2 threads. L'un écrira les 26 minuscules à l'écran et l'autre les 26 majuscules.

– Écrire un programme qui ouvre un ficher en écriture puis qui crée 2 threads, les attend et ferme le ficher. Le premier thread écrira les nombres de 0 à 100 000 dans le ficher et l'autre de 1 000 000 à 1 100 000. Consultez le fichier ainsi créé.
– Écrire un programme qui initialise une variable globale à 0 et crée 2 threads. Chacun des threads va incrémenter la variable N fois. Afficher la valeur de la variable à la fin de l'exécution de chacun des threads.
– Faites plusieurs exécutions et augmenter N jusqu'à remarquer que le résultat n'est plus 2xN. Pourquoi ?