# L3 Système TP6

**Exercice 1 - A-La, A-La-Queue-Leu-Leu**
Ecrire un programme qui prend en argument deux noms de commandes.
Il doit ensuite organiser un "pipe" de ces deux commandes, la sortie de la première alimentant l'entrée de la seconde.
Exemple : ./queueleuleu ls more ==> lance un ls, dont la sortie alimente le more.

**Exercice 2 : Réunion de famille**
En utilisant *fork()*, créer deux processus communiquant par un tube (lui-même créé avec l'appel système *pipe()*)
Le père lira depuis l'entrée standard et écrira dans le tube.
Le fils lira depuis le tube et écrira sur la sortie standard en mettant tout en majuscules (en utilisant la fonction *toupper()*, par exemple).

**Exercice 3 : A death in the family**
Reprendre l'exercice précédent.
Cette fois, après avoir 10 caractères, le fils termine son exécution.
On souhaite aussi que le père termine lorsque le fils termine.

Plusieurs solutions sont possibles :

- Utiliser **SIGCHLD** afin que le père réalise la terminaison du fils.
- Fermer soigneusement les descripteurs de fichiers inutilisés (par exemple, après le *fork()*, chaque processus ferme la moitié du tube qu'il n'utilise pas). La prochaine écriture dans le tube provoquera un **SIGPIPE** dans le père.

**Exercice 4 : Le tube ne sonne qu'une fois**
Reprendre l'exercice précédent en utilisant un tube nommé (créé par *mkfifo*). Cette fois, il n' y a ni père ni fils, mais deux processus lancés à partir du shell (ou des shells), un écrivant et l'autre lisant. Étudier l'ordre dans lequel on doit lancer ces deux processus selon le mode d'ouverture (lire le man de *open*).

**Exercice 5 - Arithmétique d'école primaire, en version distribuée**
Dans cet exercice, on appelle **co-processus** un programme lancé par un autre programme, et contrôlé par ce dernier par l'intermédiaire de ses entrée et sortie standards.

On veut ici écrire un programme *dispatch*, et des programmes *addition, multiplication, soustraction* (par exemple).

Lorsqu'on lance *addition*, ce programme attend 2 nombres sur son entrée standard, séparés par des '\n', et renvoie leur somme sur la sortie standard (puis il attend à nouveau deux nombres).
Ensuite, on lance *dispatch*, qui doit effectuer des opérations en utilisant les co-processus.

Écrire les programmes *addition, multiplication, soustraction*. Utiliser scanf pour lire les nombres, printf pour les afficher (en clair: faire simple, rapide, concis)
Écrire le programme *dispatch*, qui doit commencer par lancer les co-processus, puis attend sur l'entrée standard un ordre du style "*addition 2 4*". En utilisant *strcmp*, trouver quel est le co-processus à utiliser, lui envoyer l'ordre, puis afficher le résultat.