Le but de ce TD est de permettre à un processus de multiplexer des entrées/sorties; c'est-à-dire, dans le cas qui nous intéresse, d'"écouter" plusieurs descripteurs de fichiers sur lesquels des données sont susceptibles d'arriver. Pour celà, on mettra en place un ensemble de processus similaire aux co-processus du TD 6 ("arithmétique d'école primaire, en version distribuée") à une différence près: au lieu d'avoir un programme de dispatch qui envoie des ordres et récupère des résultats, le programme "maître" sera en attente et, à tout moment, un de ses fils pourra lui envoyer des données.

Exercice 1 - Déblatérateur

Rapidement [1], écrire le programme `repeat`, qui prend deux arguments: un message et un délai (exprimé en microsecondes). Le programme devra afficher le message à intervalles réguliers (dans une boucle sans fin).

Afin d'éviter tout problème de bufferisation, utilisez `write()` pour écrire le message et `usleep()` pour la temporisation.

Exercice 2 - Le Daily Marne diffuse l'info

Écrire un programme `listen` qui doit lancer plusieurs instances de `repeat`, avec des paramètres différents, en redirigeant la sortie de chacun de ces programmes vers un tube (un tube par instance). Le programme `listen` doit "écouter" tous les tubes en parallèle, et lorsqu'une donnée apparaît sur un tube, il doit l'afficher sur la sortie standard (en la préfixant du PID du processus, par exemple).

Afin de multiplexer les entrées, utiliser l'appel système `select()`.

Faire la même question avec l'appel système `poll()`.

Exercice 3 - L3 messenger

Écrire un serveur de chat: chaque client devra se connecter sur un port donné, donner un pseudonyme. Il pourra alors envoyer des messages qui seront diffusés à tous les autres connectés. Lorsqu'il diffuse un message, le serveur le fait précéder du pseudonyme de son auteur.

[1]: Si un tel programme vous prend plus de 10 minutes à écrire, c'est qu'il y a un problème. Normalement, ça ne prend pas plus d'1 minute.