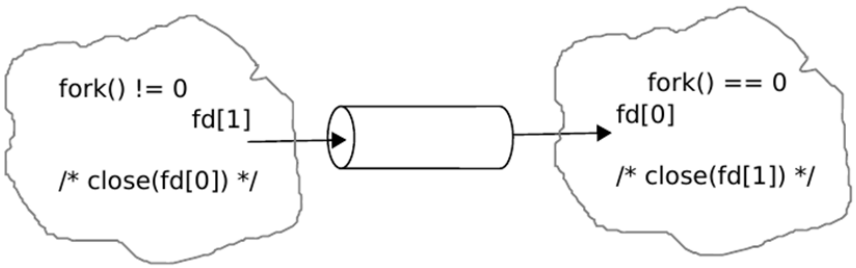
Communication par tube

Un tube est un moyen de communication unidirectionnelle entre deux processus issus de la même application (père et fils, ou frères), ou entre des processus indépendants (en utilisant des tubes nommés).

1 Communication entre processus père et fils et redirection des E/S

La fonction `int pipe(int fd[2])` permet de créer un tube sans nom, c'est-à-dire, une paire de deux descripteurs de fichiers : l'un `fd[0]` ouvert en lecture seule, l'autre `fd[1]` ouvert en écriture seule. Les données écrites à l'entrée du tube par un processus pourront être lues sur la sortie du tube par un autre processus.



EXERCICE 1 :

Écrire un programme qui prend en argument deux commandes (`commande1` et `commande2`) et réalise une exécution équivalente à l'exécution de « `commande1 | commande2` » dans un shell.

Nous faisons appel à la fonction `system(...)` pour l'exécution des commandes.
Nous utilisons l'appel système `dup2(...)` pour remplacer les flux standards `stdin` et `stdout` des processus

par les extrémités du tube. Par exemple, pour rediriger l'entrée standard d'un processus vers un descripteur de fichier `fd`, on utilise la fonction `dup2` comme suit :

```
dup2 (fd[0], STDIN_FILENO);
```

2 Tubes nommés (FIFO)

Un tube nommé (ou FIFO) possède un nom dans le système. S'il n'est pas détruit, il persiste dans le système après la fin du processus qui l'a créé. Tout processus possédant les autorisations appropriées peut l'ouvrir en lecture ou en écriture. Un tube nommé conserve toutes les propriétés d'un tube : taille limitée, lecture destructive, lecture/écriture réalisées en mode FIFO. Un tube nommé permet à des processus sans lien de parenté dans le système de communiquer.

Manipulation 1 :

On peut créer un tube nommé dans le shell avec la commande `mkfifo`. Par exemple :

```
$ mkfifo /tmp/fifo
$ ls -l /tmp/fifo
prw-rw-rw- 1 debbabi users 0 Nov 20 08:03 /tmp/fifo
```

Notez que le premier caractère du résultat de la commande `ls` est un `p`, indiquant que ce fichier est un tube nommé (ou *named pipe* en anglais).

Dans un terminal, lisez depuis ce tube nommé on invoquant la commande suivante :

```
cat < /tmp/fifo
```

Sur un deuxième terminal, tapez la commande suivante afin d'écrire sur le même tube :

```
cat > /tmp/fifo
```

Ensuite, tapez quelques lignes de texte dans ce deuxième terminal et validez en appuyant sur la touche *Entrer*. Que constatez-vous ?

EXERCICE 2 :

On vous demande de développer en langage C deux programmes comme suit :

- Un programme serveur qui crée un tube dans le système de fichiers, nommé `/tmp/fifo`. Il ouvre ensuite ce tube en lecture sous forme de flux en attendant des textes envoyés par le client.
- Un programme client qui essaye d'ouvrir en écriture seule le tube crée par le serveur. S'il y a échec d'ouverture, le processus client se termine.
- Le client envoie au serveur un texte entré par l'utilisateur comme argument au programme, et il se termine.
- Le serveur affiche à l'écran le texte envoyé par le client, et il reste en attente d'autres messages.

	$ ./client “ bonjour ”
	$ ./client “ comment allez-vous ? ”
	$ ./client “ y a-t-il quelqu’ un ? ”
	
		$ ./serveur
		>>> bonjour
		>>> comment allez-vous ?
		>>> y a-t-il quelqu'un?

On utilise les fonctions suivantes (voir les pages du manuel pour chaque fonction) :

`int mkfifo(const char *pathname, mode_t mode);`

Cette fonction crée un fichier spécial représentant le tube nommé. Son nom (chemin) est donné comme paramètre (`pathname`). Le paramètre `mode` spécifie les droits d'accès (par exemple, `0666`).

`int unlink(const char *pathname);`

Permet de supprimer le tube correspondant.

`FILE *fopen(const char *path, const char *mode);`

Cette fonction ouvre le fichier spécifié dans `path` et lui associe un flux.

`int fputs(const char *s, FILE *stream);`

Ecrit la chaîne de caractères `s` dans le flux `stream`.

`char *fgets(char *s, int size, FILE *stream);`

Lit au maximum un nombre `size` de caractères depuis le flux `stream` et les sauvegarde dans le buffer pointé par `s`.

`int fclose(FILE *fp);`

Cette fonction ferme le descripteur de fichier `fp`.