Chapitre 10

Tubes et Tubes Nommés

Les tubes sont un mécanisme de communication qui permet de réaliser des communications entre processus sous forme d'un flot continu d'octets. Les tubes sont un des éléments de l'agrément d'utilisation d'UNIX. C'est ce mécanisme qui permet l'approche filtre de la conception sous UNIX.

Mécanisme de communication lié au système de gestion de fichier, les tubes nommés ou non sont des paires d'entrées de la table des fichiers ouverts, associées à une inode en mémoire gérée par un driver spécifique. Une entrée est utilisée par les processus qui écrivent dans le tube, une entrée pour les lecteurs du tube.
L'opération de lecture y est destructive !

L'ordre des caractères en entrée est conservé en sortie (premier entré premier sorti).

Un tube a une capacité finie : en général le nombre d'adresses directes des inodes du SGF (ce qui peut varier de 5 à 80 Ko).

10.1 Les tubes ordinaires (*pipe*)

Un tube est matérialisé par deux entrées de la table des ouvertures de fichiers, une de ces entrées est ouverte en écriture (l'entrée du tube), l'autre en lecture (la sortie du tube). Ces deux entrées de la table des fichiers ouverts nous donnent le nombre de descripteurs qui pointent sur elles. Ces valeurs peuvent être traduites comme :

nombre de lecteurs = nombre de descripteurs associés à l'entrée ouverte en lecture. *On ne peut pas écrire dans un tube sans lecteur.*

nombre d'écrivains = nombre de descripteurs associés à l'entrée ouverte en écriture. La nullité de ce nombre définit le comportement de la primitive `read` lorsque le tube est vide.

10.2 Création de tubes ordinaires

Un processus ne peut utiliser que les tubes qu'il a créés lui-même par la primitive `pipe` ou qu'il a hérités de son père grâce à l'héritage des descripteurs à travers `fork` et `exec`.

```
#include <unistd.h>
int pipe(int  p[2]);
```

On ne peut pas manipuler les descripteurs de tubes avec les fonctions et primitives : `lseek, ioctl, tcsetattr` et `tcgetattr`, comme il n'y a pas de périphérique associé au tube (tout est

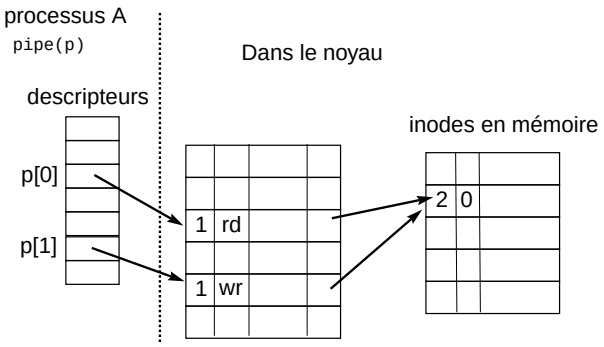

FIG. 10.1 – Ouverture d'un tube

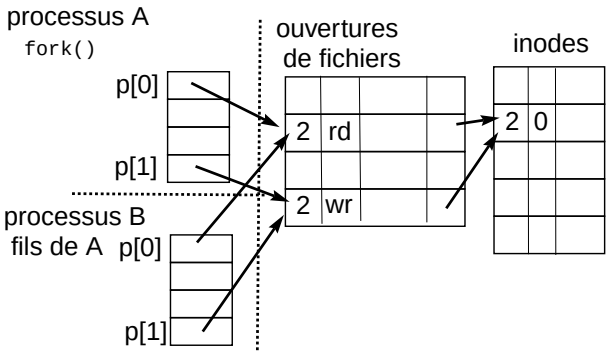

FIG. 10.2 – Héritage d'un tube

fait en mémoire).

Héritage d'un tube dans la figure 10.2 : le processus B hérite des descripteurs ouverts par son père A et donc, ici, du tube.

Dans la Figure 10.3, les descripteurs associés aux tubes sont placés comme descripteurs 0 et 1 des processus A et B, c'est à dire la sortie de A et l'entrée de B. Les autres descripteurs sont fermés pour assurer l'unicité du nombre de lecteurs et d'écrivains dans le tube.

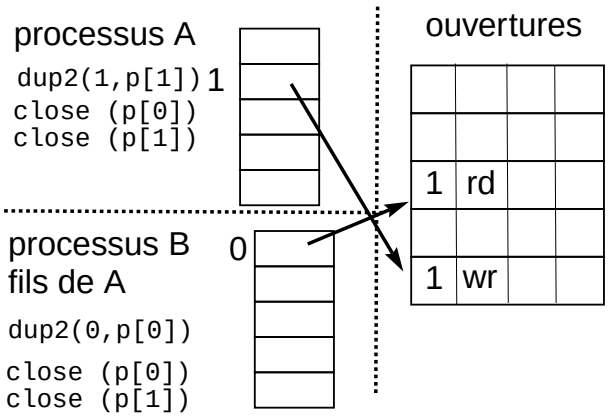

FIG. 10.3 – Redirection de la sortie standard de A dans le tube et de l'entrée standard de B dans le tube, et fermeture des descripteurs inutiles

10.3 Lecture dans un tube

On utilise l'appel système `read`.

```
int nb_lu;
nb_lu = read(p[0], buffer, TAILLE_READ);
```

Remarquer que la lecture se fait dans le descripteur p[0].

Comportement de l'appel :

Si le tube n'est **pas vide** et contient **taille** caractères :
lecture de nb_lu = min(taille, TAILLE_READ) caractères.
Si le tube est **vide**
Si le nombre d'écrivains est *nul*
alors c'est la fin de fichier et nb_lu est nul.
Si le nombre d'écrivains est *non nul*
Si lecture bloquante alors sommeil
Si lecture non bloquante alors en fonction de l'indicateur
`O_NONBLOCK` nb_lu= -1 et errno=EAGAIN.
`O_NDELAY` nb_lu = 0.

10.4 Ecriture dans un tube

```
nb_ecrit = write(p[1], buf, n);
```

L'écriture est atomique si le nombre de caractères à écrire est inférieur à PIPE_BUF, la taille du tube sur le système. (cf `<limits.h>`).

Si le nombre de lecteurs est nul
    envoi du signal SIGPIPE à l'écrivain.
Sinon
    Si l'écriture est bloquante, il n'y a retour que quand
        les n caractères ont été écrits dans le tube.
    Si écriture non bloquante
        Si n > PIPE_BUF, retour avec un nombre inférieur à n
            éventuellement -1 !
        Si n ≤ PIPE_BUF
            et si n emplacements libres, écriture nb_ecrit = n
        sinon retour -1 ou 0.

Comment rendre un read ou write non bloquant ? en utilisant fctnl sur le descripteur du tube. `F_SETFL` fixe de nouveaux attributs pour le descripteur de fichier fd. Les nouveaux attributs sont contenus dans arg. Seuls O_APPEND, O_NONBLOCK et O_ASYNC peuvent être modifiés ainsi, les autres attributs ne sont pas affectés.

10.5 Interblocage avec des tubes

Un même processus a deux accès à un tube, un accès en lecture, un accès en écriture et essaie de lire sur le tube vide en mode bloquant ⟶ le processus est bloqué indéfiniment dans la primitive `read`.

Avec deux processus :
deux tubes entre les deux processus, tous les deux bloqués en lecture ou tous les deux bloqués en écriture, tous les deux en attente d'une action de l'autre processus.

10.6 Les tubes nommés

Les tube nommés sont des tubes (`pipe`) qui existent dans le système de fichiers, et donc peuvent être ouverts grâce à une référence.
Il faut préalablement créer le tube nommé dans le système de fichiers, grâce à la primitive `mknod` (`mkfifo`), avant de pouvoir l'ouvrir avec la primitive `open`.

```
int mknod(reference, mode | S_IFIFO,0);
```

`mode` est construit comme le paramètre de mode de la fonction `open`.

En POSIX, un appel simplifié :

```
#include <sys/types.h>
#include <sys/stat.h>
int mkfifo(const char *ref, mode_t mode);
```

On peut créer des FIFOs à partir du shell grâce à

```
mkfifo [-p] [-m mode] ref ...
```

L'ouverture d'un tube nommé se fait exclusivement soit en mode O_RDONLY soit en mode O_WRONLY, ainsi le nombre de lecteur et d'écrivain peut être comptabilisé.

10.6.1 Ouverture et synchronisation des ouvertures de tubes nommés

Il y a automatiquement synchronisation des processus qui ouvrent en mode bloquant un tube nommé.

L'opération d'ouverture sur un tube nommé est bloquante en lecture.
Le processus attend qu'un autre processus ouvre la fifo en écriture.
L'ouverture en écriture est aussi bloquante, avec attente qu'un autre processus ouvre la fifo en lecture. L'ouverture bloquante se termine de façons synchrone pour les deux processus.

Ainsi un unique processus ne peut ouvrire à la fois en lecture et écriture un tube nommé.

En mode non bloquant (O_NONBLOCK, O_NDELAY), seule l'ouverture en lecture réussit dans tous les cas. L'ouverture en écriture en mode non bloquant d'un tube nommé ne fonctionne que si un autre processus a déjà ouvert en mode non bloquant le tube en lecture, ou bien qu'il est bloqué dans l'appel d'une ouverture en lecture en mode bloquant. Ceci pour éviter que le processus qui vient d'ouvrir le tube nommé, n'écrive dans le tube avant qu'il n'y ait de lecteur (qu'un processus ait ouvert le tube en lecture) et ce qui engendrerait un signal SIGPIPE (tube détruit), ce qui n'est pas vrai car le tube n'a pas encore été utilisé.

10.6.2 Suppression d'un tube nommé

L'utilisation de `rm` ou `unlink` ne fait que détruire la référence, le tube n'est réellement détruit que lorsque son compteur de liens internes et externes est nul.
Une fois que tous les liens par référence sont détruits, le tube nommé devient un tube ordinaire.

10.6.3 les appels popen et pclose

Une interface plus facile pour lancer un coprocessus est proposé avec les primitives popen et pclose.