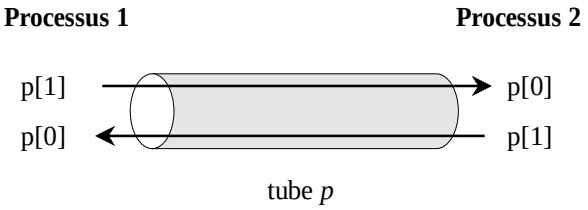
III. TUBES

Un tube est un canal unidirectionnel qui fonctionne en mode flot d'octets. Mécanisme d'échange bien adapté à l'envoi de caractères. Les tubes sont utilisés pour la communication entre processus.

1. Les Tubes Anonymes



Un tube est implémenté, sous Unix, comme un fichier. Ainsi, la plupart des fonctions sur les fichiers sont utilisables sur les tubes. Cependant, un tube n'a pas de nom physique. Il n'existe donc que pendant le temps d'exécution du processus.

Propriétés :

- Les processus communicant avec un tube doivent avoir un lien de *parenté*
- Un tube est une file de type FIFO
- La communication est unidirectionnelle
  Ce qui est lu quitte définitivement le tube et ne peut être relu

1.1 Création d'un tube

La création d'un tube se fait à l'aide de la fonction `pipe` :

```
int p[2];
pipe(p)
```

`p[0]` est le descripteur par lequel on peut lire dans le tube
`p[1]` est le descripteur par lequel on peut écrire dans le tube

Remarque : si le tube a été créé par un processus avant la création de ses fils, tous ces fils héritent alors des descripteurs du père.

1.2 Écriture/Lecture dans un tube

char buf[TAILLE_BUF]
Nb_écrit = write(p[1], buf, n);

L'appel *write* demande l'écriture dans le tube de descripteur *p[1]* des n caractères accessibles par l'adresse *buf* dans l'espace d'adressage du processus.

char buf[TAILLE_BUF]
Nb_lu = read(p[0], buf ,n*)*

La commande read lit *n* octets du tube *p* référencé par l'identificateur *p[0]* et transfert le contenu lu dans *buf*.

1.3 Fermeture d'un flux associé à un descripteur de tube

close(P[0]) ;
……
……
close(P[1]) ;

La commande close permet de fermer le descripteur d'un tube. L'accès à la partie entrante (resp. sortante) du tube se fait par la commande *close(p[1]) (resp. close(p[0]))*. Il est fortement conseillé de fermer les descripteurs non utilisés au niveau d'un processus, si celui ci ne compte pas les utilisés.

2. Redirection des Entrée et Sorties standard

La fonction système *dup* permet de rediriger les entrées/sorties standards dans un tube.

Les descripteurs suivants sont ouverts automatiquement pour chaque processus :
0 : entrée standard (clavier)
1 : sortie standard des résultats (écran)
2 : sortie standard des erreurs (écran)

Il est possible d'effectuer des redirections entre les entrées/sorties standards et un tube.

2.1 Redirection de l'entrée standard (0) vers le tube p :

```
close(0);
close(p[1]);
dup(p[0]);
close(p[0]);
```

2.2 Redirection de la sortie standard (1) vers le tube p :

```
close(1);
close(p[0]);
dup(p[1]);
close(p[1]);
```

Exemple : la commande ps –e | wc –l

```
main()
{ int p[2] ;

if  (pipe(p)==-1)
          { perror("creation du tube");
           exit(2);
          }
switch(fork()){

case –1 : /* Erreur */
           perror("Fork");
           exit();
case 0 : close(1);
          close(p[0]);
           dup(p[1]);
           close(p[1]);
           execlp("ps", "ps", "-e", NULL);
           exit();
case 1 : close(0);
          close(p[1]);
           dup(p[0]);
           close(p[0]);
           execlp("wc", "wc", "-l", NULL);
           exit();
}
```

3. Les Tubes Nommées

La définition d'un tube nommé vise à permettre à des processus sans lien de parenté particulier dans le système, de communiquer en mode flot. Ils ont toutes les caractéristiques des tubes que nous avons mentionnées et ont la propriété supplémentaire de posséder des références dans le système de gestion de fichiers.

3.1 Création

Pour créer un tube nommé, il faut créer une référence dans le système de fichiers. Depuis le *shell*, on peut utiliser la commande *mkfifo*. Il existe cependant, la primitive *mkfifo()* qu'il est possible d'appeler depuis un programme. Le prototype de cette fonction se trouve dans le fichier d'entête <sys/stat.h>

```
int mkfifo(char *ref, mode_t mode);
```

`ref` : nom physique du tube sur le disque
`mode` : Ce paramètre permet d'indiquer les permission du mode de création (Lecture, Ecriture et Exécution). Mode de création qui est conjugué avec le `umask` et les informations sur le propriétaire

3.2 Ouverture d'un tube nommé

La primitive open utilisée pour l'ouverture des fichiers, permet aussi d'ouvrir un tube de communication avec des modes en écriture/lecture. La fonction open retourne un descripteur de fichier. Cette primitive a la propriété d'être bloquante lorsque elle est appliquée à un tube nommé.

```
int open(const char *pathname, int flags);
```

open() renvoie le nouveau descripteur de fichier , ou -1 s'il échouent, auquel cas errno contient le code d'erreur.

pathname : le chemin complet d'accès au tube(fichier)
fllags : Le paramètre flags doit inclure l'un des mode d'accès suivants :
O_RDONLY, O_WRONLY : l'ouverture en lecture seule ou écriture seule.

3.3 Écriture/Lecture dans un tube nommé

La lecture et l'écriture sur le tube nommé s'effectuent de la même manière en utilisant les primitives read() et write() utilisées pour les pipes et les fichiers.

Exemple : Deux processus P1, P2
P1 crée un tube nommé et l'ouvre en écriture
P2 utilise le tube crée par P1 pour lire

```
main(){
int id_Ecriture ;
mode_t  mode ;

/* lecture, écriture pour tous */
 mode =0666

/*CREATION*/
if (mkfifo(″TubeN″, mode) == -1)
          {
        perror(″mkfifo″) ;
        exit(0);
          }

else  /* Ouverture du tube nommé TubeN en écriture */
  id_Ecriture = open(″TubeN ″, O_WRONLY);

 ..................
……..
……..
```

Processus P1

```
main()
{
int id_Lecture ;
mode_t  mode ;

/* Ouverture du tube nommé TubeN en Lecture */
id_Lecture = open(″TubeN ″, O_RDONLY);

if (id_Lecture == -1)
          {
        perror(″open″) ;
        exit(0);
          }

 .................
……..
……..
```

Processus P2