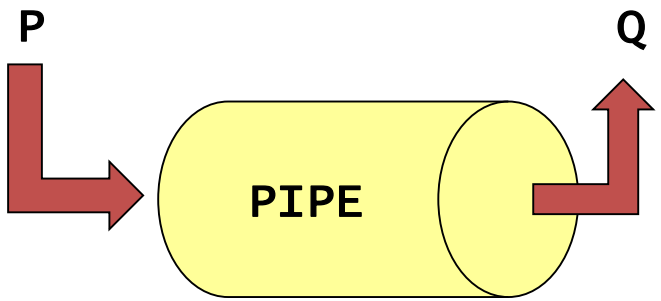
LINUX – LE NOYAU
LES TUBES

Les tubes

- Les **tubes** (pipe) sont utilisés pour la **communication entre processus**
- Un **tube** peut être représenté comme un **tuyau** dans lequel circulent des informations
  - Un tube fonctionne à **sens unique** d'où son nom
  - L'information disparaît après lecture

- Les processus P et Q communiquent via un tube de la façon suivante :
  - Le processus P écrit des informations dans le tube (entrée du tube)
  - Le processus Q lit les informations dans le tube (sortie du tube)



La fonction pipe()

```
include <limits.h>

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

pipe() crée une paire de descripteurs de fichiers pointant sur les deux « bouts » du tube et les place dans un tableau fd :
- fd[0] pour la lecture
- fd[1] pour l'écriture

pipe() renvoie 0 s'il réussit, ou -1 s'il échoue

Le tube est utilisé comme un fichier normal à l'aide des fonctions **read()** et **write()** :

- une lecture sur un tube vide est bloquante
- une écriture sur un tube n'ayant plus de lecteur provoque l'envoie du signal **SIGPIPE**
- une lecture sur un tube n'ayant plus d'écrivain rend la valeur **EOF**

Exemple avec les tubes

```
#include <sys/types.h>
#include <limits.h>
#include <string.h>
#include <stdlib.h>
int tubpf[2];
int tubfp[2];

pere( ) {
        char *chaineLue = (char*)malloc(80);
        char *chaine="Salut oh toi fils !!!";
        read( tubfp[0], chaineLue, 14 );
        printf("(du fils) : %s\n", chaineLue);
        write( tubpf[1], chaine, strlen(chaine) );
        close( tubfp[0] );
        close( tubpf[1] );
}

fils( ) {
        char *chaineLue = (char*)malloc(80);
        char *chaine="Salut père !!!";
        write( tubfp[1], chaine, strlen(chaine) );
        read( tubpf[0], chaineLue, 21 );
        printf("(du père) : %s\n", chaineLue);
        close( tubpf[0] );
        close( tubfp[1] );
        exit(0);
}
```

```
main()
{
        pid_t pid;
        pipe(tubpf);
        pipe(tubfp);
        pid=fork();
        if (pid == 0)
                fils();
        else
                pere();
}
```

Exemple avec les tubes

```
fils( ) {
        char *chaineLue = (char*)malloc(80);
        char *chaine="Salut père !!!";
        write( tubfp[1], chaine, strlen(chaine) );









        read( tubpf[0], chaineLue, 21 );


        printf("(du père) : %s\n", chaineLue);
        close( tubpf[0] );
        close( tubfp[1] );
        exit(0);
}
```

```
pere( ) {
        char *chaineLue = (char*)malloc(80);
        char *chaine="Salut oh toi fils !!!";
        read( tubfp[0], chaineLue, 14 );

        printf("(du fils) : %s\n", chaineLue);


        write( tubpf[1], chaine, strlen(chaine) );
        close( tubfp[0] );
        close( tubpf[1] );
}
```

Exemple avec les tubes

Création et destruction de processus

```
#include <stdio.h>

FILE *popen(command, type)
    char *command, *type;



pclose(stream)
    FILE *stream;
```

popen() lance l'exécution du processus (fils) dont le nom est passé en argument : ***command*** et ouvre un tube entre le père et le fils

Comme les fichiers ouverts sont partagés ***type*** précise si la commande peut être utilisée :

- en lecture "r"
- ou en écriture "w"

La fonction **pclose()** ferme le tube

Exemple avec popen() et pclose()

```
#include <stdio.h>
#define BUFSIZE 4096

main() {
        FILE *fp;
        char *buf;
        char *date = "date";
        buf = (char*)( malloc(sizeof(BUFSIZE) ) );
        if ( ( fp = popen("date", "r") ) == NULL )  {
                printf("Erreur creation pipe\n");
                exit(-1);
        }
        while ( fgets(buf, sizeof(buf), fp ) != NULL )
                fprintf(stdout, "%s", buf);
        pclose(fp);
}
```

Exemple avec popen() et pclose()

Les tubes en shell

- L'emploi des tubes permet de réaliser des enchaînements de commandes dans les shells

- Le tube (ou pipe) est symbolisé par le caractère **|**
  - Exemple :

```
# ls –l | more
```