tubes nommés
mkfifo()

anonymous

Tubes anonymes vs. Tubes nommés

- Un problème se pose avec l'utilisation des tubes classiques :
  - Il faut obligatoirement que les processus connaissent le processus qui a créé le tube.
  - Avec les tubes tout simples, il n'est pas possible de lancer des programmes indépendants, puis qu'ils établissent un dialogue.
- Les tubes nommés une "extension" aux tubes classiques.
- Le tube dispose d'un nom dans le système de fichier. Il suffit qu'un processus l'appelle par son nom, il accourt pour laisser le processus lire ou écrire en son intérieur.

Créer tube nommé
int mkfifo (const char* nom, mode_t mode);

- Pour créer un tube nommé, on utilise la fonction mkfifo :

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

  - Le premier argument est le nom du tube nommé. Ex : essai.fifo
  - Le deuxième : il s'agit concrètement des droits d'accès du tube.
    - ➢ La première, c'est de lire la documentation du fichier sys/stat.h dans la section « File mode bits ». Vous y trouverez des constantes correspondant aux droits d'accès (S_IRUSR,S_IWUSR...). Vous pouvez combiner ces constantes avec le symbole « | ».
    - ➢ Deuxième solution : fabriquer des valeurs de droit en octal 0755 0750 etc...

Créer tube nommé
int mkfifo (const char* nom, mode_t mode);

- Pour créer un tube nommé, on utilise la fonction mkfifo :

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

Le tube crée aura les autorisation suivantes :

(mode & ~umask)

Créer tube nommé
int mkfifo (const char* nom, mode_t mode);

- La fonction renvoie 0 si elle réussit, ou -1 en cas d'erreur.
- Vous pouvez aussi consulter la variable errno, qui peut contenir :
  - EACCES : le programme n'a pas les droits suffisants pour accéder au chemin de création du tube nommé ;
  - EEXIST : le tube nommé existe déjà ;
  - ENAMETOOLONG : dépassement de la limitation en taille du nom de fichier (assez rare) ;
  - ENOENT : le chemin du tube nommé n'existe pas ;
  - ENOSPC : il n'y a plus assez de place sur le système de fichiers.
- Il y a beaucoup de constantes possibles, donc généralement on ne les utilise pas dans un programme classique. Mais cela peut servir si votre programme ne marche pas très bien.

Créer tube nommé
int mkfifo (const char* nom, mode_t mode);

Exercice : Créer un tube nommé « essai » : u:rwx / g:rw / o:---

Deux solutions de possibles, pour chacune présentées pour les droits d'accès :

```
if (mkfifo(("essai.fifo"), S_IRWXU | S_IRGRP | S_IWGRP) == -1){
    fprintf(stderr, "Erreur de création du tube");
    exit(EXIT_FAILURE);
}

if (mkfifo(("essai.fifo"), 0760) == -1){
    fprintf(stderr, "Erreur de création du tube");
    exit(EXIT_FAILURE);
}
```

Ouvrir

- Ensuite, il faut ouvrir l'entrée/la sortie du tube avec la fonction **open**:

```
int open (const char* cheminFichier, int options);
```

  - La fonction renvoie une valeur de type **int** que l'on attribue à l'extrémité du tube en question.
  - Le premier argument est le nom du fichier (on mettra le nom du tube nommé).
  - Le second argument indique si c'est l'entrée ou la sortie du tube. Il existe deux constantes pour cela, déclarées dans **fcntl.h** :
    - O_WRONLY: pour l'entrée ;
    - O_RDONLY: pour la sortie.

Ouvrir

- Exemple : Pour ouvrir l'entrée d'un tube « essai.fifo » :

```
descripteur[1] = open("essai.fifo", O_WRONLY);
```

- Ensuite, vous pouvez écrire et lire avec **`write`** et **`read`** comme si c'était des tubes classiques.

Exercice

- Exercice : Écrivez deux programmes indépendants : un écrit un message dans un tube nommé, et l'autre le lit, puis l'affiche. Exécutez ces deux programmes en même temps.

Ecrivain.c :

```
#include <fcntl.h> <stdio.h> <stdlib.h> <unistd.h> <string.h>
#define TAILLE_MESSAGE  256

int main(void){
      int inTube;
      char nomTube[] = "tube1";
      char chaine[TAILLE_MESSAGE];
      printf("\n Saisir une chaine %d : ",sizeof(chaine));
      fgets(chaine,TAILLE_MESSAGE,stdin);
      mkfifo(nomTube, 0644);
      inTube = open(nomTube, O_WRONLY);
      write(inTube, chaine, TAILLE_MESSAGE);
      return EXIT_SUCCESS;
}
```

Lecteur.c :

```
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#define TAILLE_MESSAGE 256
int main(void){
     int outTube;
     char nomTube[] = "tube1";
     char chaineLue[TAILLE_MESSAGE];
     outTube = open ("tube1", O_RDONLY);
     read(outTube, chaineLue, TAILLE_MESSAGE);
     printf("%s", chaineLue);
     return EXIT_SUCCESS;
}
```

Ecrivain.c : code complet

```
#include <fcntl.h> <stdio.h> <stdlib.h> <unistd.h>
#define TAILLE_MESSAGE  256
int main(void){
   int entreeTube;
   char nomTube[] = "essai.fifo";
   char chaineAEcrire[TAILLE_MESSAGE] = "Bonjour";
   if(mkfifo(nomTube, 0644) != 0) {
      fprintf(stderr, "Impossible de créer le tube nommé.\n");
      exit(EXIT_FAILURE);
    }
   if((entreeTube = open(nomTube, O_WRONLY)) == -1) {
      fprintf(stderr, "Impossible d'ouvrir l'entrée du tube nommé.\n");
      exit(EXIT_FAILURE);
   }
   write(entreeTube, chaineAEcrire, TAILLE_MESSAGE);
   return EXIT_SUCCESS;
}
```

Lecteur.c code complet

```
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#define TAILLE_MESSAGE 256
int main(void)
{
int sortieTube;
char nomTube[] = "essai.fifo";
char chaineALire[TAILLE_MESSAGE];
if((sortieTube = open ("essai.fifo", O_RDONLY)) == -1)
{
fprintf(stderr, "Impossible d'ouvrir la sortie du tube nommé.\n");
exit(EXIT_FAILURE);
}
read(sortieTube, chaineALire, TAILLE_MESSAGE);
printf("%s", chaineALire);
return EXIT_SUCCESS;
}
```