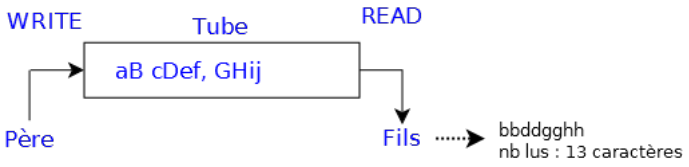
TP 4 – Tubes

- ☞ L'exo principal est l'exercice 1. Lire entièrement l'exo 1 avant de le commencer - d'ailleurs les indices sont à la fin.
- ☞ Suivez aussi les recommandations de l'enseignant, données sur le moyen de communication choisi (Discord ou Framateam)
- ☞ L'exo 1 peut très bien se passer, ce qui vous amènerez à pouvoir faire l'exo2 et ainsi jouer avec des descripteurs et avoir la possibilité d'atteindre le Niveau 4 de *Mage-Guerrier(ère)* en Système. Mais l'important est de bien maitriser l'exercice 1 avant les TPs suivant (le Niveau 3 avec maîtrise de *l'incantation du tube de feu* est bien aussi) [a].

a. Utiliser Discord pour les cours entraine certains effets secondaires ;-)

1 Communication par tube anonyme

On souhaite tester la communication par tube (anonyme) entre un père et son fils.
— Le père envoie des caractères dans le tube.
— Le fils lit le tube au fur-et-à-mesure et si la lettre est une majuscule, il l'affiche 2 fois en minuscule (et à la fin il affiche le nombre total de caractères reçus)



Mettez en place ce mécanisme. **On écrira 2 programmes**, l'un pour le père et l'autre qui sera utilisé par le fils (il y a aura donc un `fork` dans le père et un `execl` dans la partie fils appelant le deuxième programme.

- La chaine de caractères sera donnée en argument au père. Le père traitera cette chaine (argv[1]) caractère par caractère jusqu'à trouver '\0' (argv[1][0], argv[1][1], ...).
- À la fin le père indiquera combien de caractère il a envoyés.
- Dès que le fils lit un caractère (c) dans le tube, il vérifie si c'est une lettre en majuscule ('A'$\leq c \leq$'Z') et l'affiche 2 fois (en faisant $c+32$, pour avoir la lettre ASCII correspondante en minuscule). Si ce n'est pas une lettre en majuscule, il n'affiche rien.
- À la fin du processus fils (lorsqu'il n'y a plus rien à lire dans le tube), ce dernier devra afficher le nombre total de caractères lus (lettres en majuscule ou non).

- Le fils ne sait pas à l'avance le nombre de caractères qu'il va recevoir. La gestion de la terminaison du fils, après la fin (d'écriture) du père, utilise les propriétés d'un tube anonyme (voir les propriétés de la fonction *read* lorsqu'il n'y a plus d'écrivain).

Indices

- ☞ Regarder le cours ;-) [les descripteurs sont hérités automatiquement par le fils lors du fork, cependant il faut juste les transmettre par execl - attention à bien transmettre une chaine de caractère pas directement le `int` (le numéro du descripteur)]
- ☞ Il faut penser à fermer les bouts de tube qui ne servent pas (pour éviter une attente infinie, notamment du lecteur qui croit qu'il y a encore un écrivain à l'autre bout alors que c'est lui-même... il a oublié de fermer son descripteur pour l'écriture).
- ☞ Pour passer une chaine de caractères contenant des espaces en argument d'un programme, il faut la mettre entre guillemets. Exemple : `prog "Texte à Traiter"`

2 La redirection

Objectifs sous-jacents de l'exercice

- ☞ Comprendre le principe de descripteur de "fichiers" ouverts
- ☞ Illustrer l'utilisation des fonctions dup et dup2 (duplication de descripteurs) avec les tubes.
- ☞ Au final, réaliser ce que fait l'opérateur *pipe* | sous Linux : enchainer 2 programmes avec la sortie de du premier devient l'entrée du second.

Le but de cet exercice est d'écrire un programme *bypipe* qui réalise le mécanisme de redirection. Ce programme prend en argument deux chaînes de caractères contenant chacune les deux commandes à enchainer. La deuxième commande utilise la sortie de la première comme entrée.

ex : *bypipe* "ls -la" "wc -l" ⇒ compte le nombre "d'objets" contenus dans le répertoire courant (*ls* liste les objets et *wc* les compte).

Pour réaliser les commandes, il est possible de le faire par la création de 2 processus fils [1] et d'un tube de communication :

- Il faut donc rediriger la sortie (standard) du premier processus (devant réaliser la première commande) dans un tube.
- Et remplacer l'entrée (standard) du deuxième processus (devant réaliser la deuxième commande) par le tube.

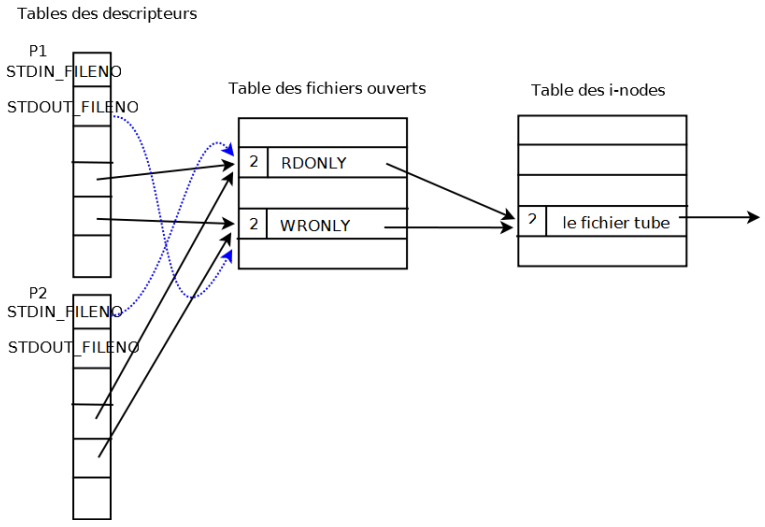
1. Il n'y a pas à écrire les programmes fils puisque l'on recouvre tout simplement le code du fils par les programmes passés en paramètres avec execl. Il faut juste les créer et les recouvrir.

Le principe repose sur la duplication des descripteurs pour associer la sortie et l'entrée standard à respectivement l'entrée et la sortie d'un tube : on utilisera la fonction *dup2*. Rappelez-vous le principe des descripteurs et de la table de fichiers ouverts. La sortie standard a pour descripteur STDOUT_FILENO et l'entrée standard STDIN_FILENO. La fonction **dup2(descr1, descr2)** permet de recopier/dupliquer le descripteur descr1 à l'emplacement du descripteur descr2. Vous pouvez vous aider du schéma (de la page suivante) pour comprendre ce qu'il faut faire. Le schéma représente les fichiers ouverts juste après la création du tube et des deux fils.

Il faut ainsi faire croire au processus 1 qu'il écrit comme d'habitude sur le descripteur de la sortie standard. Mais en fait vous avez modifié le descripteur de la sortie standard pour qu'il pointe maintenant sur l'entrée du tube... oh les malins !

fermeture des descripteurs inutiles

Il ne faut pas oublier de fermer les descripteurs qui pointent entrées/sorties du tube qui ne servent pas (comme pour l'exercice précédent).



Aide en C :

- Dans un premier temps les commandes seront inscrites directement dans les *execl*. Ex : `execl("/bin/ls", "ls", -la", null);)`
- Dans un second temps, pour récupérer les paramètres des arguments de manière automatique, on pourra utiliser la fonction *char *strsep (char **stringp, const char *delim)*. La fonction *strsep()* renvoie l'élément lexical (token) suivant contenu dans la chaîne *stringp*, délimitée par *delim*. Le mot renvoyé est terminé par un caractère nul '\0', et le pointeur *stringp* est mis à jour pour pointer après le mot. La fonction *strsep()* renvoie un pointeur sur l'élément lexical extrait, ou NULL si le séparateur *delim* n'est pas trouvé dans *stringp*.

```
char *lgcom, *comm1, *comm2;
```

```
char *args1[30], *args2[30] ;
/* découpage de la ligne de commande donnée en paramètre */
lgcom = argv[1] ;

i = 0 ;
while ((args1[i] = (char *)strsep(&lgcom, " ")) != NULL) {
        i++ ;
}
comm1 = args1[0] ;
```

On pourra ainsi utiliser la fonction *execvp(commande, pointeur_tableau_arguments)* pour le recouvrement : `execvp(comm1, args1)` ;