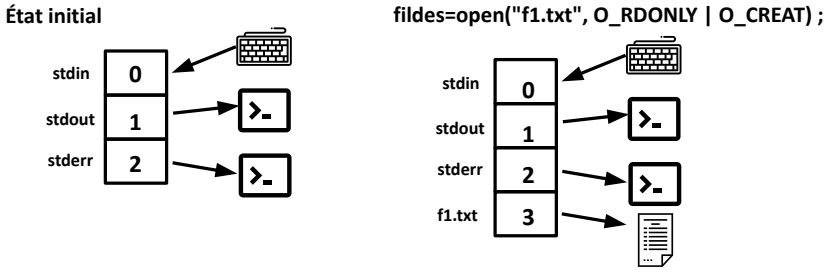
TP 4 : redirections, fichiers et pipes

Rappel sur les descripteurs de fichiers :

Tous les processus Linux possédent automatiquement 3 descripteurs, stdin (entrée standard clavier), stdout (sortie standard sur le terminal), stderr (sortie d'erreur standard sur le terminal, non bufferisée), de numéros respectifs 0,1 et 2. Ils sont manipulables par les primitvies classique des fichiers (`read`, `write`, `close`...), et il n'est pas nécessaire de les ouvrir par `open` (c'est déjà fait !). Quand un nouveau fichier est ouvert par `open`, il obtient le premier descripteur libre (par exemple, 3 au lancement du processus).



Exercice 1. Redirection

- Ecrivez le programme *mycat* qui, appelé sur le terminal sans arguments, reproduit le comportement de la commande shell`cat`. C'est-à-dire qu'il duplique sur la *sortie standard* (identifié par la constante `STDOUT_FILENO`, (égale en fait à 0)) les caractères qui sont entrés au clavier (*entrée standard* `STDIN_FILENO`) :

```
% cat
je copie
je copie
^D
```

Vous utiliserez les primitives systèmes suivantes : `open`, `read`, `write`, `close` (et `perror` pour gérer les erreurs).

- Modifiez le programme pour qu'il accepte de façon optionnelle le nom d'un fichier (ex : *mycat fichier.txt*) en paramètre. Cette fois, le comportement sera celui de *cat > fichier.txt*.

```
% cat > fichier.txt
je copie
^D
```

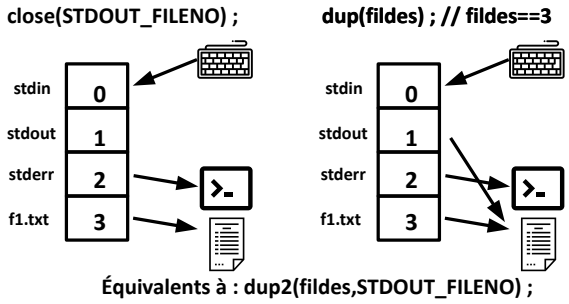
```
4
5 % more fichier.txt
6 je copie
```

Vous devrez obligatoirement utiliser pour l'écriture (sur fichier comme sur la sortie standard) l'instruction suivante :

`write(STDOUT_FILENO,buf,n)`

Donc, dans le cas où un nom de fichier en argument est donné, il vous faudra **rediriger la sortie standard vers ce fichier.**

Vous utiliserez les primitives systèmes suivantes : `open`, `read`, `write`, `close` (et `perror` pour gérer les erreurs) ainsi que *dup*. On rappelle que cette instruction duplique un descripteur passé en paramètre dans le descripteur libre de plus petit numéro (*i.e.* le premier en partant de 0).



- Modifiez le programme pour utiliser ***dup2*** à la place de *dup*.

Exercice 2. Pipe

Ecrire un programme principal *pipe.c* qui prend en argument sur la ligne commande, deux noms de fichiers. Ce programme crée deux processus fils qui communiquent par tube anonyme de la façon suivante :

- Le premier processus fils ouvre le premier fichier donné en argument du programme principal et transmet le contenu de ce fichier au second processus fils via un tube de communication.
- Le second processus fils écrit le contenu du tube dans le deuxième fichier.

Exercice 3. Pipe, fork et exec

Ecrire un programme C équivalent à la commande shell :

```
% ps -uax | grep root | wc -l
```

Vous utiliserez les notions vues dans ce TP : redirections, exec, fork…