# ­TP6 (3h) : Communication entre processus par tubes

**Rappel sur la documentation.** Les fonctions du module **os** en Python sont des copies conformes des fonctions système en langage C, qui disposent de pages manuel UNIX (**man**). La documentation du module Python **os** étant souvent assez succincte, il est utile de consulter également les pages **man** des fonctions C utilisées.

1. **Combiner pipe et dup pour créer un tube entre deux commandes**

Écrire un programme qui crée un tube de communication (avec **pipe**) puis crée 2 processus fils, le premier exécutant la commande **ls -l**, le second exécutant la commande **grep \\.py**. Le tube doit préalablement être raccordé (avec **dup2**) de manière à reproduire l’équivalent de la commande shell **ls -l | grep \\.py**.

*Le schéma suivant décrit la solution et les principaux pas exécutés par les trois processus. Ce schéma n’est cependant pas complet, notamment il faudra fermer les extrémités du tube inutilisées dans chacun des trois processus pour que tout fonctionne bien…*

**

*A noter :* ***grep*** *continuera à tourner tant qu’il peut y avoir des données qui arrivent par le tube, c’est à dire tant que le côté* ***wd*** *reste ouvert dans l’un des processus. Pour que le programme s’arrête correctement, il faut veiller, dans chacun des processus à toujours fermer les descripteurs non-utilisés avec la fonction* ***close.***

1. **Communication par tube nommé**

*Un tube nommé permet à deux processus quelconques (sans lien de parenté) de communiques des données. Il est créé avec la* ***commande shell mkfifo*** *et il peut ensuite être utilisé comme un fichier ordinaire (il apparaît dans l’arborescence des fichiers, on l’ouvre avec* ***open****,…), avec la différence que les données ne sont pas stockées sur le disque, qu’un* ***read*** *effectué**dessus**est bloquant tant qu’il n’y a pas de données envoyées dans le tube, et que les données disparaissent du tube après la lecture.*

Essayer la séquence de commandes shell suivante:

mkfifo tmp

ls -l tmp

cat tmp

Dans un autre terminal, exécuter la commande "cat > tmp" et saisir quelques lignes de texte. Regarder le premier terminal et expliquer le résultat (en faisant appel à "**man mkfifo**").

Écrire **deux** **programmes** qui communiquent par le tube "tmp" crée précédemment. Le premier programme doit écrire dans le tube, en boucle toutes les secondes, des nombres aléatoires entre 32 et 99 (l’encodage sur 1 octet des nombres envoyés se fait avec les fonctions vues en TP5, Ex.3). Le deuxième programme doit récupérer les nombres et écrire sur la sortie standard les caractères ASCII correspondants aux codes numériques reçus.

Tester le comportement du système si l’un ou l’autre des deux programmes est interrompu par CTRL-C. Que peut-on en déduire ?

1. **Communication bidirectionnelle et tubes**

On souhaite implémenter un programme permettant de réaliser le comportement suivant :

* Il lance deux processus s’échangeant des données à l’aide de **pipe**
* la communication entre les deux processus doit être *bidirectionnelle*
* le processus P1 doit lire le contenu d’un fichier *ff.txt* et le communiquer au processus P2
* le processus P2 doit afficher à la *sortie d’erreurs* le contenu reçu (en provenance de P1)
* le processus P2 doit lire le contenu d’un fichier *gg.txt* et le communiquer au processus P1
* le processus P1 doit afficher à la *sortie standard* le contenu reçu (en provenance de P2)