Compte rendu

TP6

Communication entre processus par tubes

Mail : maxencedeliot@gmail.com

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

Principe : On souhaite faire communiquer deux processus père et deux processus fils grâce à un tube de communication. Le premier fils doit exécuter la commande ***ls –l*** tandis que le second fils doit lui récupérer le résultat de cette première commande et exécuter un ***grep \\.py***. Ce programme permettra de reproduire l'équivalent de la commande ***ls -l | grep \\.py*** .

Vérification :

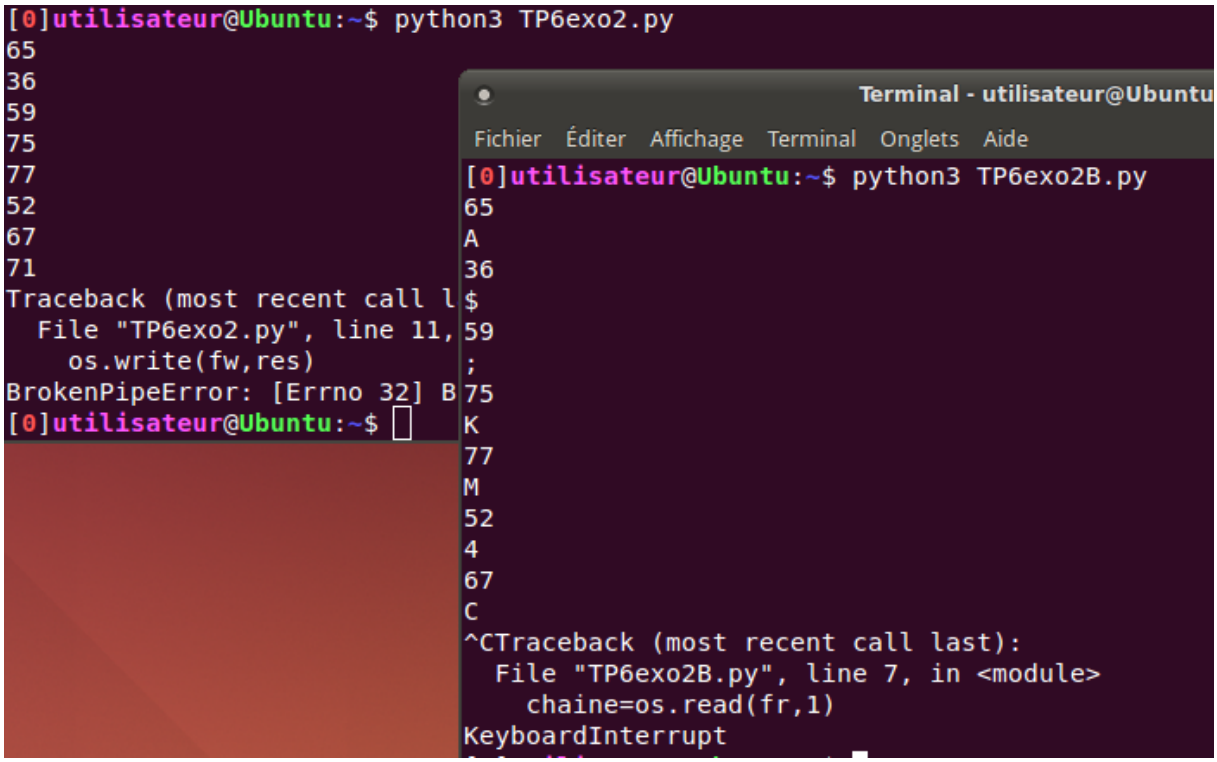
```
[0]utilisateur@Ubuntu:~$ python3 TP6exo1.py
-rw-rw-r-- 1 utilisateur utilisateur  136 nov.  10 08:32 TP5exo1.py
-rw-rw-r-- 1 utilisateur utilisateur  134 nov.  10 08:31 TP5exo1.py~
-rw-rw-r-- 1 utilisateur utilisateur  260 nov.  10 09:13 TP5exo2.py
-rw-rw-r-- 1 utilisateur utilisateur  258 nov.  10 09:12 TP5exo2.py~
-rw-rw-r-- 1 utilisateur utilisateur  319 nov.  10 10:12 TP5exo3.py
-rw-rw-r-- 1 utilisateur utilisateur  320 nov.  10 10:10 TP5exo3.py~
-rw-rw-r-- 1 utilisateur utilisateur  300 nov.  10 10:49 TP6exo1.py
-rw-rw-r-- 1 utilisateur utilisateur  297 nov.  10 10:44 TP6exo1.py~
```

On obtient bien ce que l'on souhaite cad uniquement les fichiers .py .

Ex. 2. Communication par tube nommé

Principe : Un tube nommé (créé avec la commande shell ***mkfifo***) permet à deux processus quelconques (sans lien de parenté) de communiques des données. On souhaite faire communiquer deux processus sans lien de parenté grâce à un tube nommé : « tmp ». Le premier doit écrire dans le tube des nombres aléatoires entre 32 et 99 toutes les secondes, tandis que le second doit lui récupérer ce nombre et écrire sur la sortie standard le caractère ASCII correspondant**.**

Vérification :



On arrive donc à faire communiquer les deux processus grâce à un tube nommé.

Ex. 3. Communication bidirectionnelle et tubes

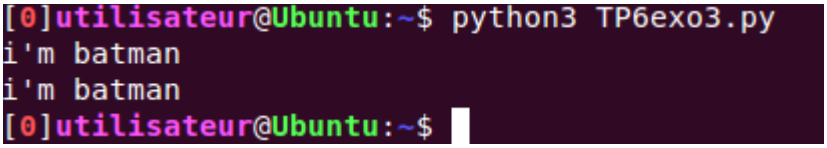
Principe : On souhaite faire communiquer deux processus père et fils grâce à un tube de communication. Ici, la difficulté est que l'on souhaite que la communication soit bidirectionnelle. On va donc utiliser deux ***pipe***.

Premièrement : le premier (P1) doit lire le contenu du fichier ***ff.txt*** et le communiquer au second, tandis que le second (P2) doit afficher le contenu reçu à la sortie d'erreurs.

Deuxièmement : c'est le même principe mais inversé. P2 devra envoyé le contenu du fichier ***gg.txt*** à P1 qui l'affichera à la sortie standard.

Vérification :

Dans un premier temps, j'ai écrit le même texte « i 'm batman » dans les deux fichiers (sans vraiment réfléchir) :



On obtient bien l'affichage des deux contenus. Puis j'ai ensuite écrit « i'm bat » dans ***gg.txt :***

On a donc bien ce que l'on souhaite cad l'affichage du contenu de ff.txt (« i'm batman ») et celui de gg.txt(« i'm bat »).