TRAVAUX PRATIQUE - PROGRAMMATION CONCURRENTE

CPE Lyon – 3ETI

TUBES ANONYMES & TUBES NOMMES

EXERCICE 1 – Introduction aux pipes anonymes

Réalisez un script montrant l'utilisation d'un tube anonyme (**pipe**) entre deux processus (voir cours).

EXERCICE 2 – Les redirections d'entrées/sorties

Testez une à une les commandes *shell* suivantes : **wc**, **sort**, **tail** (consultez le *man*).

Testez les redirections d'entrée (**<**), de sortie (**>**) et le pipe (**|**) à l'aide des commandes suivantes :

- ✓ `sort < fichier > fichier_trie`
- ✓ `sort fichier | grep toto | wc –l`

Ecrire 2 scripts Python réalisant respectivement les 2 commandes composées suivantes :

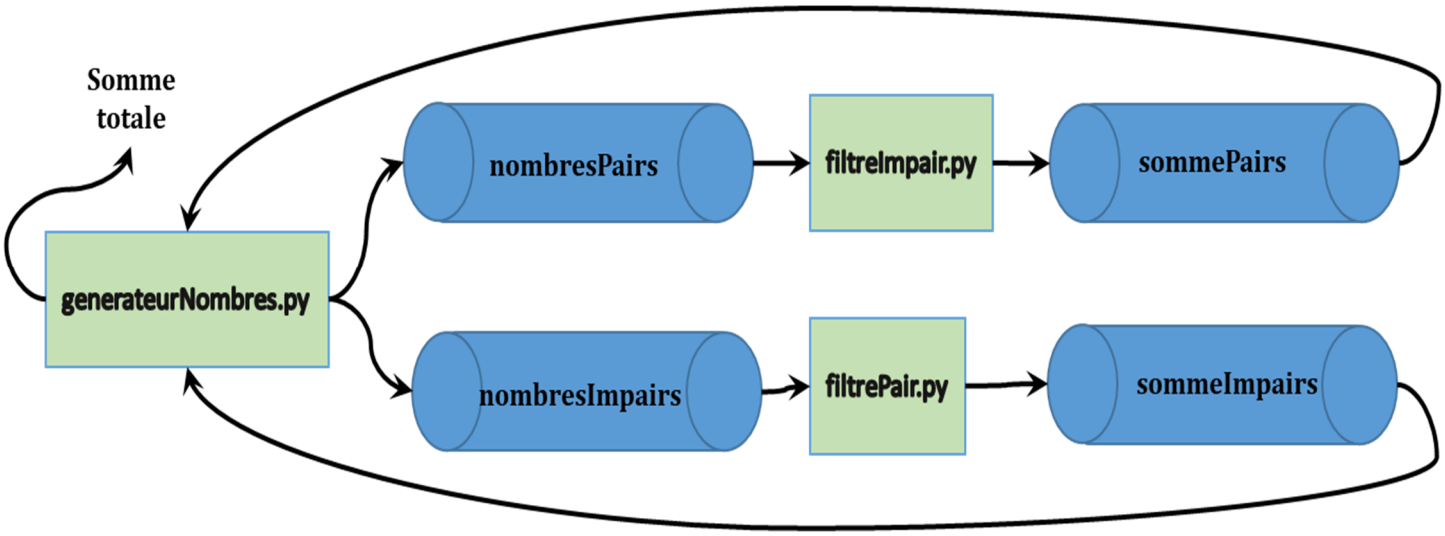
- ✓ `cat fichier | wc`
- ✓ `sort < fichier | grep chaine | tail –n 5 > sortie`

EXERCICE 3 – Utilisation des tubes nommés et pipes anonymes

On vous demander de réaliser les deux versions suivantes :

- Un script Python composé de **3** processus réalisant les traitements demandés en utilisant les tubes **anonymes**.
- Trois scripts Python correspondant à **3** processus indépendants (**generateurNombres.py**, **filtrePair.py** et **filtreImpair.py**) réalisant les traitements demandés en utilisant les tubes **nommés**.

1. Le premier processus génère **N** nombres aléatoires positifs ou nuls. Si le nombre généré est **pair** (resp. **impair**) alors il est déposé dans le tube **nombresPairs** (resp. **NnombresImpairs**) - à la fin de la génération, ce processus dépose la valeur **-1** dans les 2 tubes (pour indiquer la fin de la série). Ensuite il récupère les deux nombres stockés respectivement dans le tube **sommePairs** et **sommeImpairs**, réalise leur somme et affiche le résultat.
2. Un deuxième processus est chargé de récupérer les nombres stockés dans le tube **nombresPairs**, de réaliser la somme de ces nombres et de déposer le résultat dans le tube **SommePairs**.
3. Un troisième processus est chargé de récupérer les nombres stockés dans le tube **nombresImpairs**, de réaliser la somme de ces nombres et de déposer le résultat dans le tube **sommeImpairs**.



GESTION DES SIGNAUX

Exercice 1 : Ecrire un programme qui réalise un affichage dans une boucle infinie, mais qui prévoit de s'arrêter à la réception du signal **SIGINT**. La fonction d'interception [*de déroutement*] affichera un message signalant la réception du signal avant de terminer le programme par un appel à **sys.exit()**.

Exercice 2 : Modifier le programme précédent pour qu'il ignore le signal **SIGINT**. Lancez-le en tâche de fond (`python3 exo2.py &`). Essayez de l'interrompre avec un **^C**. Que constatez-vous ?
Pour le supprimer, il faut lancer la commande **ps –l** pour obtenir l'identifiant du Processus (PID) et lancer la commande : `kill -9 numProcessus` (Vous pouvez remplacer **9** par **SIGKILL**).

Exercice 3: Modifiez le programme précédent pour qu'il fasse son affichage dans une boucle conditionnée par une variable booléenne **fin** initialisée à **False** et qui sera mise à **True** par la fonction d'interception du signal.

Exercice 4 : Ecrire un programme composé de **2** processus : Le père fait des affichages toutes les secondes dans une boucle **for** et le fils fait des affichages toutes les secondes aussi mais dans une boucle infinie. Quand le compteur de boucle du père arrive à **3**, le père envoie un signal **SIGKILL** au fils. On a constaté dans l'exercice 2, l'impossibilité d'ignorer ce signal.

Exercice 5 : Recopier le script précédent et le modifier pour que le père n'envoie plus ce signal au fils mais que le fils intercepte tous les **SIGINT** en affichant un message d'interception.
Exécutez alors le programme et interrompez le par un **^C** : Que constatez-vous? Expliquez pourquoi. (Pour vous aider, utilisez la commande **ps –l**).

Exercice 6 : Reprendre le script de l'exercice précédent et le modifier pour que le fils ne fasse ses affichages qu'à la réception du signal **SIGUSR1**. Le père envoie ce signal dans sa boucle à l'itération **3** et à l'itération **5**. Il signale la fin du traitement au fils par envoi du signal **SIGUSR2**.
Il n'y aura qu'une seule fonction d'interception dans le fils, elle recevra le numéro du signal déclencheur en paramètre.

Exercice 7 : Ecrire un script qui demande à l'utilisateur de taper au clavier un entier en moins de **5 secondes**. Pour cela, votre programme ne doit pas "planter" si l'utilisateur n'a pas saisi un nombre entier. Il devra donc lire une chaîne de caractères et tenter de la convertir en un entier, si c'est bon il se terminera après avoir désarmé le **time Out**, sinon il recommencera éventuellement jusqu'aux **5** secondes où il affichera "trop tard" avant de s'arrêter.

Exemples :

```
$python3 exercice7.py
Entrez un entier en moins de 5 secondes
Svp un entier : a
Svp un entier : x
Svp un entier : 12
Ok merci !!
```

```
$ python3 exercice7.py
Entrez un entier en moins de 5 secondes
Svp un entier : E
Svp un entier : f
Svp un entier : D
Trop tard !!
```