# ­TP5 (3h) : Substitution de descripteurs de fichiers 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. **Substitution de descripteurs de fichiers**

Ecrire un programme qui :

1. Ouvre un fichier nommé *f* .*txt* en écriture/création
2. Substitue le descripteur de *f.txt* au descripteur de sortie standard (voir fonctions **dup** et/ou **dup2**)
3. Lance avec **exec** la commande **ls –l**

Exécuter le programme, examiner le contenu de *f.txt* et expliquer le résultat.

1. **Substitution de descripteurs de fichiers – bis**

La fonction **traceback.print\_stack()**écrit à la sortie d’erreurs (**stderr**). Nous souhaitons écrire un programme qui appelle cette fonction, mais, au lieu d’afficher son résultat sur **stderr**, il le récupère dans une variable sous la forme d’une liste de chaîne de caractères, une chaîne par ligne.

Pour faire cela, le programme doit :

1. Ouvrir un fichier temporaire (cf. **tempfile.TemporaryFile**).

*Attention : le descripteur retourné par cette fonction n’est pas un descripteur bas-niveau (numérique) mais un objet de type* ***io.BufferedRandom****. A ce titre, il dispose de méthodes intéressantes comme* ***readlines****; pour obtenir le descripteur bas-niveau correspondant, utiliser la méthode* ***fileno****.*

1. Substituer le descripteur du fichier temporaire au descripteur de la sortie d’erreurs stderr (avec **dup2**). Attention à sauvegarder l’ancien descripteur (avec **dup**) pour le remettre en place à la fin.
2. Appeler la fonction **traceback.print\_stack()**
3. Récupérer le contenu du fichier temporaire avec **readlines**. Il faut au préalable déplacer le curseur de lecture du fichier au début (fonction **seek**)
4. Remettre les descripteurs de fichiers du processus dans l’état où ils étaient au début du programme
5. **Communication par tube anonyme**

Écrire un programme qui crée un tube de communication (avec la fonction **pipe**) puis crée un processus fils et lui communique un entier aléatoire N entre 0 et 0xFFFFFFFF (cf. **random.randint**) à travers le tube. Le processus fils lit l’entier depuis le tube et l’écrit à la sortie standard.

***A noter :***

1. *La fonction* ***pipe*** *crée un tube permettant de communiquer des données entre processus. Le tube est “anonyme” dans le sens où seul le processus qui le crée et ses descendants (créés avec* ***fork*** *après l’appel à* ***pipe****) peuvent y accéder.*
2. *L’entier N sera communiqué via le tube avec* ***read/write*** *en format binaire. Vous devez pour cela utiliser la méthode* ***N.to\_bytes(4, sys.byteorder)*** *pour l’encoder avant le* ***write*** *et la fonction* ***int.from\_bytes*** *pour le décoder après le* ***read****.*
3. *Vous aurez besoin de Python v.3 pour les fonctions d’encodage/décodage*
4. *Ne pas oublier de fermer les descripteurs de tube inutilisés dans chacun des processus.*

*Le schéma suivant décrit la solution et les principaux pas exécutés par les deux processus :*

**