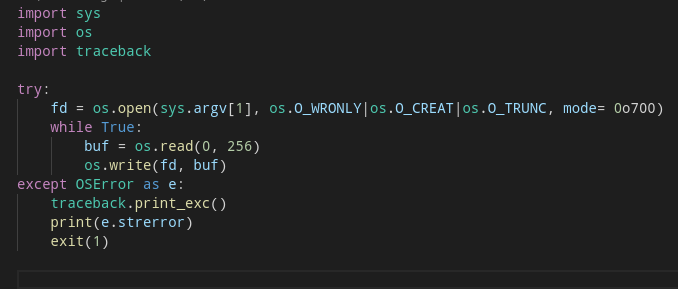
# TP 2.1 (3h): Fichiers

**Objectif.** *L’objectif de ce TP est de vous familiariser avec les fonctionnalités de base de l’API système POSIX pour la lecture/écriture des fichiers*

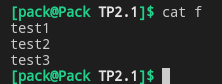
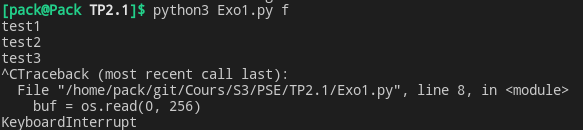
**Rappel sur la documentation.** Les fonctions du module **os** en Python sont des copies conformes des fonctions système en langage C, qui disposent de pages manuel UNIX (**man**). La documentation du module Python **os** étant souvent assez succincte, il est utile de consulter également les pages **man**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

1. **Variation sur le thème du cat**Modifier le programme de l’exercice 1 du TD2 pour que maintenant il reproduise le fonctionnement de la commande  
    **cat > fichier**  
   i.e., il lit des caractères au clavier et les écrit dans un fichier dont le nom est passé en argument.  
   ***A noter :*** 
   1. *Le programme doit être lancé sans redirection (>) et il récupère le nom du fichier passé par la ligne de commande toujours au moyen* ***sys.argv****).*
   2. *Vous devez utiliser* ***os.read*** *pour lire l’entrée standard (et non pas* ***input****).*

***Code de l’Exercice 1 :***



Résultats :



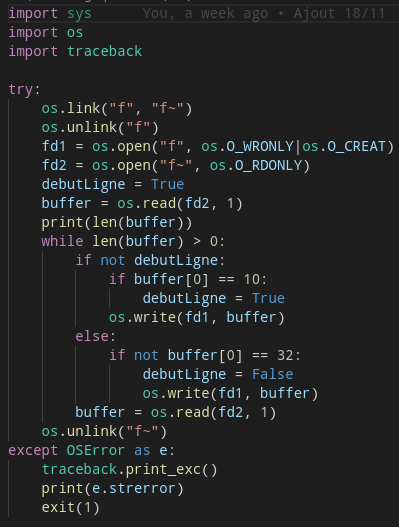
1. **Sauvegarde et modification d’un fichier**

Ecrire un programme qui prend en argument le nom d’un fichier texte **f** et :

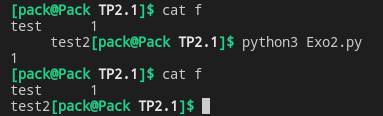
* Crée une copie de sauvegarde de **f**, intitulée **f~** (écrase le fichier **f~** si celui-ci existe)
* Supprime les espaces se trouvant *au début* de chaque ligne du fichier **f**

***Indication :*** *la modification du fichier* ***f*** *étant difficile dans ces conditions, un moyen plus simple d’arriver au résultat est de changer le nom de* ***f*** *en* ***f~*** *(en utilisant les fonctions* ***link*** *et* ***unlink****) et de recréer* ***f*** *à partir de* ***f~*** *(en utilisant* ***read/write****) et en supprimant les espaces au passage.****A noter :*** *lors de la recréation de* ***f****, il faut veiller à garder les mêmes droits d’accès que sur l’ancien fichier. Utiliser la fonction* ***os.stat*** *pour les obtenir avant de l’effacer.*

*Code de l’exercice 2 :*



Résultats :



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

La fonction **traceback.print\_stack()**écrit des caractères à la sortie d’erreurs **stderr** (il s’agit d’un descriptif de la pile d’appels, similaire à ce qui est affiché par l'interprète python lors d’une erreur).

Nous souhaitons écrire un programme qui appelle cette fonction, mais qui, au lieu d’afficher son résultat sur **stderr**, le récupère dans une variable sous la forme d’une liste de chaînes de caractères, une chaîne par ligne.

Pour faire cela, le programme doit :

1. Créer un fichier temporaire   
   (avec **tempfile.TemporaryFile**, cf : <https://docs.python.org/3/library/tempfile.html> ).

*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. Remettre les descripteur de la sortie d’erreurs dans l’état où il était au début du programme
4. Déplacer le curseur de lecture du fichier au début (fonction **os.lseek**)
5. Récupérer le contenu du fichier temporaire avec la méthode **readlines** (attention c’est une méthode de l’objet fichier -- ce n’est pas un appel système) et l’afficher avec **print**.

Code de l’exercice 3 :



Résultat :

