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.

Ex. 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 :

a. *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**).*

b. *Vous devez utiliser **os.read** pour lire l'entrée standard (et non pas **input**).*

Code de l'Exercice 1 :

```
import sys
import os
import traceback

try:
    fd = os.open(sys.argv[1], os.O_WRONLY|os.O_CREAT|os.O_TRUNC, mode= 0o700)
    while True:
        buf = os.read(0, 256)
        os.write(fd, buf)
except OSError as e:
    traceback.print_exc()
    print(e.strerror)
    exit(1)
```

Résultats :

```
[pack@Pack TP2.1]$ python3 Exo1.py f
test1
test2
test3
^CTraceback (most recent call last):
  File "/home/pack/git/Cours/S3/PSE/TP2.1/Exo1.py", line 8, in <module>
    buf = os.read(0, 256)
KeyboardInterrupt
```

```
[pack@Pack TP2.1]$ cat f
test1
test2
test3
[pack@Pack TP2.1]$
```

Ex. 2. 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 :

```python
import sys      You, a week ago • Ajout 18/11
import os
import traceback

try:
    os.link("f", "f~")
    os.unlink("f")
    fd1 = os.open("f", os.O_WRONLY|os.O_CREAT)
    fd2 = os.open("f~", os.O_RDONLY)
    debutLigne = True
    buffer = os.read(fd2, 1)
    print(len(buffer))
    while len(buffer) > 0:
        if not debutLigne:
            if buffer[0] == 10:
                debutLigne = True
            os.write(fd1, buffer)
        else:
            if not buffer[0] == 32:
                debutLigne = False
                os.write(fd1, buffer)
        buffer = os.read(fd2, 1)
    os.unlink("f~")
except OSError as e:
    traceback.print_exc()
    print(e.strerror)
    exit(1)
```

Résultats :

```
[pack@Pack TP2.1]$ cat f
test      1
     test2[pack@Pack TP2.1]$ python3 Exo2.py
1
[pack@Pack TP2.1]$ cat f
test      1
test2[pack@Pack TP2.1]$
```

Ex. 3. 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**.*
2. 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.
3. Appeler la fonction **traceback.print_stack()**
4. Remettre les descripteur de la sortie d'erreurs dans l'état où il était au début du programme
5. Déplacer le curseur de lecture du fichier au début (fonction **os.lseek**)
6. 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 :

```python
import os      You, a week ago • Ajout 19/11
import sys
import traceback
import tempfile

try:
    tmpfile = tempfile.TemporaryFile()
    descStd = os.dup(2)
    os.dup2(tmpfile.fileno(), 2)
    traceback.print_stack()
    os.dup2(descStd, 2)
    os.lseek(tmpfile.fileno(), 0, os.SEEK_SET)
    for li in tmpfile.readlines():
        print(li)
except OSError as e:
    traceback.print_exc()
    print(e.strerror)
    exit(1)
```

Résultat :

```
[pack@Pack TP2.1]$ python3 Exo3.py
b'  File "/home/pack/git/Cours/S3/PSE/TP2.1/Exo3.py", line 10, in <module>\n'
b'    traceback.print_stack()\n'
[pack@Pack TP2.1]$
```