Exercices de TP sur le shell et la mémoire

Fabien Rico

1 Ressources d'un processus

Pour cet exercice, utilisez le code : `code.py`.

1) Lancez le programme ; pendant qu'il tourne trouvez son pid grâce à la commande (`ps auxf`).

2) Utilisez les commandes `fuser` ou `lsof` pour connaître les fichiers utilisés dans le répertoire courant.

3) Dans quel cas cela peut-il vous servir ?

4) Lorsqu'un processus est lancé vous pouvez retrouver les informations le concernant dans `/proc/< pid >/...`. Où se trouvent les informations sur les fichiers ouverts ? Le répertoire courant ? Les variables d'environnement ?

5) Lancez la commande `ls -R / | less` ; retrouvez leur pid et afficher les répertoires contenant les descripteur de fichiers : `ls -lh /proc/<pid>/fd/`. À quoi correspondent les descripteurs 0 et 1 ?

1.1 Mémoire :

Pour cet exercice, utilisez les codes : `code1.c`, `code2.c`, `code3.c` et `code4.c`. Ce sont des codes en C car python avec son interpreteur ajoute trops de chose pour que l'étude de la mémoire soit clair.

Pour compiler ces codes, vous pouvez utiliser `gcc` grâce à la commande suivante :

```
gcc -g -Wall -O0 nomdufichier.c -o nomexecutable
```

— Grâce à la commande `pmap`, affichez les segments utilisés par les programmes `code1` et `code2`. Quelles sont les différences ? À quoi correspondent-elles ?
— Faites de même pour les codes code3 et code4. Que devient le tableau tab ?
— Donnez pour `code2` le contenu des différents segments.
— Peut-on retrouver ces informations dans le répertoire `/proc/< pid >/` ?

1.2 Variables d'environnement

Nous allons voir les variables d'environnement et leur effet à travers la notion d'environnement virtuel python. Ces environements permettent de définir un ensemble de dossiers contenant tout le necessaire au fonctionnement d'un projet avec les fichiers python, mais aussi les librairie nécessaire et en dehors de celle du système hote.

Pour gérer un envirronement vous disposez des commandes suivantes :

```
# creation de l'envirronement
python3 -m venv ./nomDuRepertoireDeLEnv
# "entrée dans l'environnement (en bash)
source <repEnv>/bin/activate
# "sortie de l'environnement
deactivate
```

1) Créez l'environnement `testTP`

2) Dans ce répertoire, que contient le sous répertoire `./lib/python???/` et le sous répertoire `./bin/`

3) Ouvrez le fichier `./bin/activate`, que fait ce code pour l'activation, la desactivation ?

4) Entrez dans l'environnement et comparez les variables `PATH`, `PYTHONHOME` `VIRTUAL_ENV` et `PS1` avec un shell qui n'y est pas. A quoi servent ces 3 variables ?

5) Installer la librairie `pandas` grâce à `pip`. Quel est le résultat dans le répertoire `./lib/` ?

6) Lancer un processus python et vérifiez que le chemin de recherche des bibliothèque est bien modifié par l'environnement (le contenur de `sys.path`)

7) Comment lancer un processus python dans l'environnement sans entrer dedans ?

```
  [frico@wu-pers-00516 testTP]$ ls -l ./bin/
total 48
-rw-r--r--. 1 frico frico 2036 12 mai 11:41 activate
-rw-r--r--. 1 frico frico 962 12 mai 11:41 activate.csh
-rw-r--r--. 1 frico frico 2104 12 mai 11:41 activate.fish
-rw-r--r--. 1 frico frico 9033 12 mai 11:41 Activate.ps1
```

```
-rwxrwxr-x. 1 frico frico 282 12 mai 11:41 pip
-rwxrwxr-x. 1 frico frico 282 12 mai 11:41 pip3
-rwxrwxr-x. 1 frico frico 282 12 mai 11:41 pip3.10
lrwxrwxrwx. 1 frico frico 7 12 mai  11:41 python -> python3
lrwxrwxrwx. 1 frico frico 16 12 mai 11:41 python3 -> /usr/bin/python3
lrwxrwxrwx. 1 frico frico 7 12 mai  11:41 python3.10 -> python3
[frico@wu-pers-00516 testTP]$ ls -l lib/python3.10/site-packages/
total 4
drwxrwxr-x. 1 frico frico 66 12 mai 11:41 _distutils_hack
-rw-rw-r--. 1 frico frico 152 12 mai 11:41 distutils-precedence.pth
drwxrwxr-x. 1 frico frico 114 12 mai 11:41 pip
drwxrwxr-x. 1 frico frico 154 12 mai 11:41 pip-21.2.3.dist-info
drwxrwxr-x. 1 frico frico 80 12 mai 11:41 pkg_resources
drwxrwxr-x. 1 frico frico 714 12 mai 11:41 setuptools
drwxrwxr-x. 1 frico frico 146 12 mai 11:41 setuptools-57.4.0.dist-info
```

Le fichiers activate est un script qui modifie juste les variables `PATH`, `PS1` et `VIRTUAL_ENV`. De plus il supprime `PYTHONHOME`

```
## Position de l'envirronement
VIRTUAL_ENV="/home/frico/Cours/SystemCapes/cours-capes/tp-sys-linux/src/testTP"
export VIRTUAL_ENV

# sauvegarde du PATH
_OLD_VIRTUAL_PATH="$PATH"

# Modification du PATH
PATH="$VIRTUAL_ENV/bin:$PATH"
export PATH

# sauvegarde et suppression de PYTHONHOME
if [ -n "${PYTHONHOME:-}" ] ; then
    _OLD_VIRTUAL_PYTHONHOME="${PYTHONHOME:-}"
    unset PYTHONHOME
fi

# Changement du prompt
if [ -z "${VIRTUAL_ENV_DISABLE_PROMPT:-}" ] ; then
    _OLD_VIRTUAL_PS1="${PS1:-}"
    PS1="(testTP) ${PS1:-}"
    export PS1
    VIRTUAL_ENV_PROMPT="(testTP) "
    export VIRTUAL_ENV_PROMPT
fi
```

Après l'installation, les bibliothèques pythons sont installées dans le répertoire `lib` ou `lib64` de l'environnement, les eventuels exécutables sont installés dans le sous répertoire `bin`

Pour lancer un processus dans l'environnement, il faut le lancer avec les bonnes valeurs de `PATH` et `VIRTUAL_ENV` :

```
PATH=./nomDuRep/:$PATH VIRTUAL_ENV=./nomDuRep/ python3 ...
```

Cela fonctionne aussi avec un exécutable installé dans l'environnement.

Attention, `pyzo` utilisé officielement pour le CAPES impose de fixer les valeurs correctement pour qu'elles soient prise en compte. Sinon, cela echoue silencieusement. Il faut entrer les valeurs dans les configurations du shell avant de lancer le shell. Contrairement à ce qui est dit ici `https://pyzo.org/faq.html` il n'est pas utile d'après mes test de donner une valeur à EXE. PAr contre, les variables `VIRTUAL_ENV` et `PATH` doivent être entrées comme dans l'exemple

```
VIRTUAL_ENV=/path/to/venv
PATH=/path/to/venv/bin:$PATH
# et pas comme dans activate
VIRTUAL_ENV="/path/to/venv"
# ou
PATH=$VIRTUAL_ENV/bin:$PATH
```