TD 3 Gestion des processus en python

Q1) Quel est le code de sortie émis par un processus quand tout c'est bien passé ?

Q2) Avec geany Écrire un programme en python (importer la bibliothèque os) qui affiche bonjour et qui donne comme code de sortie 0 (l'aide de l'instruction os._exit(0)). Vérifier.
Q3) Avec geany , écrire un programme qui ne s'arrête pas (nommé Q3.py).
Ouvrir un terminal, repérer le PPID et le PID du processus gérant l'exécution de Q3.py
ps -ef |grep Q3
En déduire comment geany lance l'exécution d'un programme python.

Q4) Donc , pour la suite nous pouvons écrire le codes avec geany, mas nous l'exécuterons directement depuis une console . Chaque code devar donc commencer par la ligne :

#!/usr/bin/python3

ici le code

puis enregistrer le fichier, ouvrir une console là ou le ficher est enregistré, rendre le fichier exécutable , puis lancer le fichier avec la commande ./nomfichier & . Essayer avec Q3.py.

Q5) Modifier Q2.py, et ajouter le code de sortie 12 , exécuter avec ./Q2.py et récupérer le code de sortie. Pourquoi n'ajoute-t-on pas le &.

--NOTES ---
Python permet à un processus d'en lancer un autre qui est un clone de lui même il s'agit de la fonction os.fork() . Après un appel à cette fonction, deux processus existent donc dans la mémoire et s'exécutent en parallèle. Cet appel retourne une valeur de l' identifiant du fils au père et une valeur nulle pour le processus fils.
En python, l'appel time.sleep(x) (importer la bibliothèque time) met le processus en sommeil pendant x seconde.

Q6) Écrire un programme Q6.py qui écrit toutes les secondes "alive since x s", x variant de 1 à 10, puis s'arrête.

Q7) Écrire le code suivant nommé Q7.py

```
#!/usr/bin/python3
import os

ret=os.fork()

if (ret==0):
    # si je suis le fils
    print("je suis le fils")
else:
    # si je suis le pere
    print("je suis le pere et mon fils a pour pid : ",ret)
```

Que fait -il ? Vérifier .

Q8) Améliorer Q7.py en utilisant en utilisant os.getpid() (qui donne le pid du processus) et os.getppid() (qui donne le pid du processus pere) pour avoir un affichage de cette forme :

je suis le pere et mon fils a pour pid : 4708 et moi 4707
je suis le fils 4708 mon pere est 4707

Q9) Dans une console, lancer top -o PID

puis écrire un programme Q9.py qui crée un nouveau processus (avec fork()).
•Le père affiche « je suis le père et j'attends 10 secondes ». Puis il attend 10s. Puis il affiche « le père est mort » et il se termine
•Le fils affiche « je suis le fils et j'attends 15 secondes ». Puis il attend 15s.Puis il affiche « le fils est mort »

Recommencer en réglant le temps du fils à 5s.
Exécuter le programme, quels est le statu du fils après qu'il soit mort et avant la mort du père.

Q10) Échange de signaux . la fonction os.kill(toto, SIGKILL) lance un signal de terminaison au processus dont le pid est toto.
Écrire un programme qui créé un fils puis :
◦ le père affiche toutes les secondes « je suis le père » pendant un temps infini
◦ le fils attend 5 secondes "j'attends 5s " puis tue le père, attend 5s et affiche "papa j'ai pas voulu ça".

Q11) Envoyer des signaux en python.
Pour trouver la liste des signaux, taper le commande kill -l, et repérer les signaux contenant **USR** qui sont utilisables par les processus utilisateurs.
Commenter le code ci-dessous, puis l'exécuter (Q11.py) .

```
import signal
# quel signal va être traité (SIGUSR1) et par qui (traitement)
signal.signal(signal.SIGUSR1, traitement)
print(os.getpid())
# boucle infinie
while(True) :
    a=0
```

Depuis un terminal , envoyer la commande kill -USR1 au processus lancer avec Q11.py

Q12) Modifier le programme pour que il affiche deux messages différents en fonction des signaux usr1 et usr2.