Arguments en ligne de commande

```
#Exemple_argv.py
import os , sys
if len(sys.argv) < 2 :
    print ("Ce script est sans paramètres")
print (sys.argv )
for argument in sys.argv[1 : ] :
    print(argument )
os._exit(0)
```

```
$ python test.py a bc "def ghi" \"ild mno\"
['test.py', 'a', 'bc', 'def ghi', '"ild', 'mno"']
a
bc
def ghi
"ild
mno"
$
```

Les Processus sous Linux

Les fonctions fondamentales

Duplication (Création) de Processus

os.fork()

Tout processus a un seul père.

Tout processus peut avoir zéro ou plusieurs processus fils.

Recouvrement (Chargement) de Processus

os.execl(path , arg0, arg1, ...)

os.execv(path, args)

os.execle(path , arg0 , arg1 , ... , env)

os.execlp(file , arg0 , arg1 , ...)

os.execvp(file, args)

os.execve(path, args, env)

os.execv(path, args)

...

```
args = ("ls"  , "-l")
os.execv("/bin/ls"  , args)
```

os.execlp(file, arg0, arg1, ...)

```
os.execlp("ls"  , "ls"  , "-l"  )
```

Il n'est pas nécessaire ici de spécifier le chemin d'accès (/bin/ls) pour l'exécutable.

Pour recouvrir un processus avec la commande ps -aux
nous pouvons utiliser le code suivant :

```
args = ("ps"  , "-aux")
execv("/bin/ps" , args)      /* l'éxécutable ps se trouve dans /bin */
print ("erreur dans execv")
```

Un processus se termine lorsqu'il n'a plus d'instructions
ou lorsqu'il exécute la fonction
sys.exit()

L'élimination d'un processus terminé [*de la table des processus*]
ne peut se faire que par son père, grâce à l'appel :
os.wait()

Grâce à 3 fonctions système : os.fork( ) , os.exec( ) et os.wait( )

on peut écrire un interpréteur de commandes simplifié.
Il prend la forme suivante :

```
while True :
        lire_commande(commande, parametres)     /* attend commande*/
        if (os.fork() != 0) :                   /* processus père */
                os.wait()
        else :
                execv(commande , parametres)    /* processus fils */
```

Le père attend son fils

```
import os , sys
import time

if os.fork( ) == 0 :
        print ("Le processus fils = %d" %( os.getpid( ) ) )
        sys.exit(10)
pid , status = os.wait( )
print ( "Le processus PERE = %d" %(os.getpid( ) ) )
print ( "Sortie du wait()"  )
time.sleep(15)
print ( "pid = %d status = %d" %(pid, os.WEXITSTATUS(status) ) )
sys.exit(0)
```

Le père n'attend pas son fils et est toujours en vie après la terminaison de son fils

```
if os.fork() == 0 :
        print ( "Le processus fils = %d" %( os.getpid( ) ) )
        sys.exit(10)
print ( "Le processus PERE = %d" %( os.getpid() ) )
while True :
        print ( "ok" )
```

Le père reçoit le signal de terminaison de son fils et n'exécute le os.wait() qu'après
Le fils reste zombie momentanément

```
if os.fork() == 0 :
        print ("Le processus fils = %d" %(os.getpid()) )
        sys.exit(3)
print ("Le processus PERE = %d" %(os.getpid()) )
time.sleep(10)
print ("Sortie après 10 secondes")
pid, status = os.wait()
print ("sortie du wait()" )
time.sleep(10)
print ("pid = %d - status = %d" %( pid , os.WEXITSTATUS(status) ) )
sys.exit(0)
```

Le père n'attend pas son fils
mais se termine avant celui ci.
Le fils devient orphelin

```
if os.fork() == 0 :
        print("Le processus fils = %d " %(os.getpid()) )
        time.sleep(60)
        sys.exit(10)
print("Le processus PERE terminé %d " %(os.getpid()) )
sys.exit(0)
```

Commenter ce programme
(préciser la fonctionnalité réalisée par ce programme).

```
if len(sys.argv) < 2 :
        print ("Préciser les programmes à exécuter en ligne de commande" )
        sys.exit(1)
print (sys.argv)
liste = [ ]
for cmd in sys.argv[1 : ] :
        pid = os.fork()
        if pid == 0 :
                try :
                        os.execlp(cmd , cmd)
                except :
                        print ("echec de execlp" )
                        sys.exit(3)
        p , s = os.wait()
        if os.WIFEXITED(s) :
                print ("Terminaison normale du processus fils : " , p)
                if os.WEXITSTATUS(s) == 3 :
                        liste.append(cmd)
if len(liste) > 0 :
        print ("Commandes non exécutées = ", liste)
sys.exit(0)
```

Détailler (ligne par ligne) le comportement de ce script.

```
n=0
pid = os.fork()
if pid != 0 :
    n += 1
    p , s = os.wait()
    if os.WIFEXITED(s) :
        n = n + os.WEXITSTATUS(s)
else :
    n += 10
print (os.getpid() , n)
sys.exit(n)
```