FAUCHER Vinh

TD3 :

1) Le programme affiche bien 100 lignes

```
user@PNS-VirtualBox:~/Documents/ProgSys/td03$ ./multiple_fork.exe | wc -l
100
```

2) Avec l'ajout de la fonction sleep(1), le programme met bien 10 secondes à s'exécuter.

La commande renvoie les pid des 10 processus fils (7305 … 7314) puis du processus père (7337).

On peut voir que les pid des processus fils se suivent tous, ce qui est logique sachant qu'il n'y a pas d'autres programmes lancés en même temps que celui demandé.

```
user@PNS-VirtualBox:~/Documents/ProgSys/td03$ ps aux | grep multiple_fork.exe
user        7305  0.0  0.0   2496    80 pts/0    S    09:56   0:00 ./multiple_fork.exe
user        7306  0.0  0.0   2496    80 pts/0    S    09:56   0:00 ./multiple_fork.exe
user        7307  0.0  0.0   2496    80 pts/0    S    09:56   0:00 ./multiple_fork.exe
user        7308  0.0  0.0   2496    80 pts/0    S    09:56   0:00 ./multiple_fork.exe
user        7309  0.0  0.0   2496    80 pts/0    S    09:56   0:00 ./multiple_fork.exe
user        7310  0.0  0.0   2496    80 pts/0    S    09:56   0:00 ./multiple_fork.exe
user        7311  0.0  0.0   2496    80 pts/0    S    09:56   0:00 ./multiple_fork.exe
user        7312  0.0  0.0   2496    80 pts/0    S    09:56   0:00 ./multiple_fork.exe
user        7313  0.0  0.0   2496    80 pts/0    S    09:56   0:00 ./multiple_fork.exe
user        7314  0.0  0.0   2496    80 pts/0    S    09:56   0:00 ./multiple_fork.exe
user        7337  0.0  0.0  17704   740 pts/1    S+   09:57   0:00 grep --color=auto multiple
_fork.exe
```

3) On peut voir que le fils (8008) devient un zombie (8009).

```
user@PNS-VirtualBox:~/Documents/ProgSys/td03$ ps aux | grep zombie.exe
user        8008  0.0  0.0   2364   592 pts/0    S+   10:18   0:00 ./zombie.exe
user        8009  0.0  0.0      0     0 pts/0    Z+   10:18   0:00 [zombie.exe] <defunct>
user        8011  0.0  0.0  17696   740 pts/1    S+   10:18   0:00 grep --color=auto zombie.e
xe
```

4) On voit que le programme finit prématurément, et que le fils se fait adopter par un autre processus.

```
user@PNS-VirtualBox:~/Documents/ProgSys/td03$ ./orphelin.exe
Père avec pid : 8353
user@PNS-VirtualBox:~/Documents/ProgSys/td03$ Fils dont le père est : 1111
```

5) On voit que le pid ne change pas et le code écrit après le execlp ne s'exécute pas.

```
user@PNS-VirtualBox:~/Documents/ProgSys/td03$ ./exec_prop.exe
Début 9239
Deuxième programme
pid : 9239 et arg exec_prop-aux
```

6)

```
user@PNS-VirtualBox:~/Documents/ProgSys/td03$ ./shell_exec.exe
total 176
 4 -rw-rw-r-- 1 user user   289 févr. 23 11:11 exec_prop-aux.c
20 -rwxrwxr-x 1 user user 19320 févr. 23 11:11 exec_prop-aux.exe
 4 -rw-rw-r-- 1 user user   319 févr. 23 11:11 exec_prop.c
20 -rwxrwxr-x 1 user user 19296 févr. 23 11:11 exec_prop.exe
 4 -rwxrwx--- 1 user user  3547 févr. 23 09:22 fcat.c
20 -rwsrwxr-x 1 user user 20024 févr. 23 09:31 fcat.exe
 4 -rwxrwx--- 1 user user  1725 févr. 23 09:22 Makefile
 4 -rw-rw-r-- 1 user user   819 févr. 23 10:12 multiple_fork.c
20 -rwxrwxr-x 1 user user 19640 févr. 23 10:16 multiple_fork.exe
 4 -rw-rw-r-- 1 user user   565 févr. 23 11:26 orphelin.c
20 -rwxrwxr-x 1 user user 19432 févr. 23 11:27 orphelin.exe
 4 -rw------- 1 user user    39 févr. 23 11:31 secret_file.txt
 4 -rw-rw-r-- 1 user user   431 févr. 23 11:31 shell_exec.c
20 -rwxrwxr-x 1 user user 19344 févr. 23 11:31 shell_exec.exe
 4 -rw-rw-r-- 1 user user   432 févr. 23 10:54 zombie.c
20 -rwxrwxr-x 1 user user 19376 févr. 23 11:08 zombie.exe
user     :0           2022-02-23 09:22 (:0)
```

7)Après exécution du programme, il se trouve dans le dossier du premier processus lancé, contrairement à la console qui est dans le répertoire initial “Home“. La commande ls va ensuite lister les fichiers du répertoire.

8) Code

9) Lorsque l’on utilse system() et qu’on arrête le programme, system ne tue que le processus fils mais n’empêche pas le programme de continuer de fonctionner.

Lorsqu’on utilise my_system, c’est le processus père qui est tué à la place, ce qui met ainsi fin au programme.