TD2 – Compte rendu

1) Fork

En augmentant MAX_COUNT d'un facteur de 10 dans le premier exercice, l'affichage issu du fils et l'affichage issu du père s'entremêlent sur le terminal.

```
This line is from pid 9147, value = 774
This line is from pid 9148, value = 743
This line is from pid 9147, value = 775
This line is from pid 9148, value = 744
This line is from pid 9147, value = 776
This line is from pid 9148, value = 745
This line is from pid 9147, value = 777
This line is from pid 9148, value = 746
```

Nous pouvons garantir qu'on obtiendra toujours un résultat entremêlé puisque le scheduler prend ses décisions selon le temps d'exécution des processus et leur priorité.

2) Orphelin

Il y a effectivement création d'un processus orphelin parce que le fils est mis en sommeil pendant 40 secondes, tandis que le père est tué après seulement 15 secondes via la fonction exit(0). Le fils est toujours en sommeil, mais n'a plus de père.

```
ll503889  9763  9738  0 16:07 pts/9    00:00:00 usershell
ll503889  9769  9748  0 16:07 pts/0    00:00:00 ./orphelin
ll503889  9770  9769  0 16:07 pts/0    00:00:00 ./orphelin
ll503889  9770  7387  0 16:07 pts/0    00:00:00 ./orphelin
ll503889  9773  9748  0 16:08 pts/0    00:00:00 ps -e -f
ll503889@henson ~/Systeme/TP2/orphelin>  Je suis donc devenu orphelin !
```

Le screenshot précédent montre le résultat de la commande « ps » avant la fin de l'exécution du programme (en haut) puis après la fin d'exécution (en bas). On voit que le processus père de PID 9769 n'est plus présent sur la fin, alors que son processus fils de PID 9770 lui est encore là. Il est bel et bien devenu un processus orphelin.

3) Zombie

Création d'un processus zombie puisque le père est mis en sommeil pendant 30 secondes, tandis que le fils est rapidement tué par exit(0). Le père étant toujours en sommeil, il n'a pas encore lu le code de retour de son fils. Ignorant la disparition de son fils, ce dernier devient un processus zombie, identifié comme processus « defunct » sur le terminal.

Le screenshot suivant montre le début de l'exécution du programme (en haut), suivi de l'affichage de la commande « ps » lorsque le programme a pris fin (en bas). On voit que le processus fils de PID 9936 est identifié comme « defunct » tandis que son pére de PID 9935 est toujours présent dans l'arbre des processus !

```
ll503889@henson ~> cd Systeme/TP2/zombie
ll503889@henson ~/Systeme/TP2/zombie> ./zombie

Je suis le pere : PID  9935
 Qu ' est devenu mon fils 9936 ?
 Vous avez 30 sec pour lancer un  ps -e -f  et constater qu ' il est zombi !
Je suis le fils : PID 9936 et le PID de mon pere est 9935
 Je suis le fils et je meurs : PID 9936
```

```
ll503889  9747  9738  0 16:07 ?        00:00:00 gnome-pty-helper
ll503889  9748  9738  0 16:07 pts/0    00:00:00 usershell
ll503889  9850  9738  0 16:10 pts/10   00:00:00 usershell
ll503889  9912  7387  0 16:13 ?        00:00:00 /usr/bin/python3 /usr/share/unit
ll503889  9931  9738  0 16:13 pts/9    00:00:00 usershell
ll503889  9935  9850  0 16:13 pts/10   00:00:00 ./zombie
ll503889  9936  9935  0 16:13 pts/10   00:00:00 [zombie] <defunct>
ll503889  9938  9931  0 16:13 pts/9    00:00:00 ps -e -f
ll503889@henson ~>
```

La commande « kill -9 » suivi du PID du zombie ne produit pas de résultat. On ne peut se débarrasser du fils qu'en lançant un « kill -9 » sur le PID du père.

4) Wait

Le screenshot suivant illustre le résultat de l'exécution du programme lorsqu'on laisse le processus fils mourir de façon naturelle.

```
ll503889@henson ~/Systeme/TP2/wait> ./wait
Pid du pere = 9623
Attente de la terminaison du fils ...
Pid du fils = 9624
 Le fils 9624 s'est termine correctement : 0
ll503889@henson ~/Systeme/TP2/wait>
```

Ce second screenshot montre le résultat de cette même exécution lorsqu'on envoie un signal 9 au processus fils via la commande « kill » pour arrêter son exécution. Le processus père attend le code de retour de son fils, puis affiche un message sur le terminal.

```
ll503889@henson ~/Systeme/TP2/wait> ./wait
Pid du pere = 9670
Attente de la terminaison du fils ...
Pid du fils = 9671
Le fils 9671 s'est mal termine : 9
ll503889@henson ~/Systeme/TP2/wait>
```

```
ll503889@henson ~> kill -9 9671
ll503889@henson ~>
```

5) SIGCHLD

Quand un processus fils s'arrête ou est interrompu, le signal SIGCHLD est envoyé au processus père. La réponse par défaut est d'ignorer ce signal, cependant, il peut être réceptionné et traité par le père pour récupérer le code de retour du processus fils grâce à la fonction wait(). En créant une fonction prenant un signal (type int) en paramètre, on peut ainsi surveiller les processus fils et supprimer les zombies aussi efficacement que possible. C'est le cas de ce programme, avec la fonction eliminer_zombie(int sig).