Cadereau Sarah

INFO632 TP2 – Compte rendu

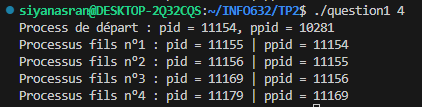
Lien GitHub vers le code : <https://github.com/srh1001/INFO632.git>

Question 1 : Création d’une chaîne de processus.

Le programme réalise une boucle de *n* itérations pour créer les *n* processus fils, en faisant un *fork()* à chaque itération.

Si l’on se trouve dans le processus fils, on affiche son PID et celui de son parent. Si l’on se trouve dans le processus parent, on attend 10 secondes (temps pendant lequel on peut taper *pstree -p* *<pid du process de départ>* dans la console) puis la fin du processus fils et on termine le processus.

Voici un exemple de sortie du code et le pstree associé pour n = 4 :





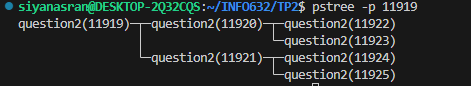
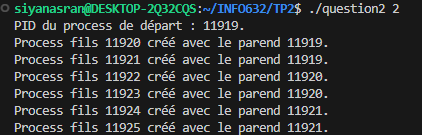
Question 2 : Création d’un arbre binaire de processus à *n* niveaux.

La fonction *create\_two\_children* prend en paramètre le PID d’un processus et le niveau dans l’arbre de ce processus. Cette fonction crée à l’aide d’une boucle de deux itérations, deux processus fils à celui passé en paramètre, puis utilise la récursivité pour créer de nouveau deux fils à chacun des fils créés avant. La récursivité s’arrête lorsque le dernier niveau de l’arbre a été atteint (i.e. que la valeur de n passée en entrée de la fonction est 0).

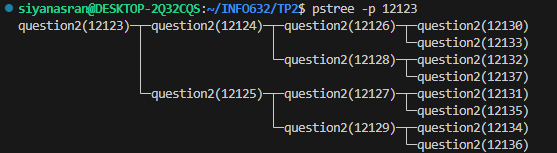
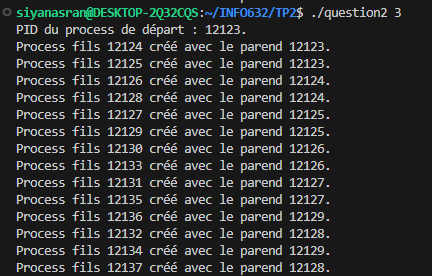
Enfin, chaque processus parent attend la fin de ses deux processus fils avec un *sleep(5)*, ce qui laisse aussi le temps de taper *pstree -p <pid du process de départ>* dans la console pour observer l’arbre.

Voici un exemple de sortie du code et le pstree associé :

Pour n = 2 :



Pour n = 3 :

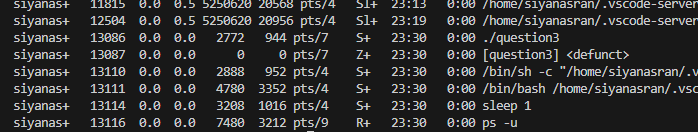


Question 3 : Création d’un processus zombie.

Le programme crée un processus fils avec un *fork()* puis si on est dans le processus parent, alors celui-ci dort pendant 10 secondes, c’est pendant ces 10 secondes là que le processus fils devient zombie.

On peut le voir à l’aide de la commande *ps -u* comme ci-dessous :







Question 4 : Création d’un processus qui retourne une valeur au parent.

Le programme crée un nouveau processus, puis si l’on est dans ce processus fils, alors l’utilisateur doit saisir une valeur qui sera stockée dans la variable *child\_return\_value* et passée en argument de la fonction *exit*.

Si l’on est dans le processus parent, alors on stocke le retour du processus fils avec *wait(&child\_return\_value)* et on accède à cette valeur avec *WEXITSTATUS*.

Exemple de sortie :

