IN405 – Système d'exploitation
TD 4 – Processus

S. Gougeaud

2017/2018

Exercice 1 – Création simple de processus

En utilisant la fonction `fork`, écrivez les programmes correspondant aux comportements suivants pour un processus père et son processus fils :

1. Affichage de 'Hello World!' pour les deux processus.
2. Affichage de 'Mon PID est ... et celui de mon père/fils est ... !'.
3. Le processus fils choisit aléatoirement un nombre entre 1 et 50, l'affiche, puis le communique à son père qui l'affiche à son tour.

Exercice 2 – `sleep()` & `wait()`

Écrivez le programme correspondant à l'énoncé suivant : le processus père crée 10 processus fils et attend qu'ils se terminent. Chaque fils attend un nombre de secondes choisi aléatoirement entre 1 et 10, affiche son PID puis se termine. Le processus père affiche à chaque terminaison, le PID du processus fils qui a terminé son exécution.

Exercice 3 – Création multiple de processus

Écrivez les programmes correspondants aux énoncés suivants, avec m et n, deux entiers donnés au lancement du programme :

1. Le processus père crée m fois n processus fils.
2. Le processus père crée m processus fils, puis chaque processus fils crée n processus petit-fils.
3. Le processus père crée n processus fils, puis chaque processus fils crée n processus petit-fils, puis chaque processus petit-fils etc., ceci m fois.

Pour chacun des énoncés, calculez, à l'aide du programme, le nombre total de processus créés.

Exercice 4 – Temps d'exécution

A l'aide de la fonction `times()`, écrivez le programme correspondant à l'énoncé suivant : le processus père crée un processus fils qui liste le contenu d'un répertoire donné en argument (à l'aide de la commande ls). Une fois l'exécution du fils terminée, le père affiche le temps d'exécution du processus fils (et donc de la commande ls). **Attention :** pour éviter un temps d'exécution quasi nul, n'hésitez pas à lister **récursivement** le répertoire.

Exercice 5 – Envoi de signal

A l'aide de la fonction `kill()` et des signaux `SIGSTOP` et `SIGCONT`, écrivez le programme correspondant à l'énoncé suivant : le processus père crée un processus fils qui compte de 1 à 5 (un affichage par seconde). Trois secondes après avoir créé son processus fils, le père met en pause le fils, attend cinq secondes puis le relance. Que se passe-t-il si le signal `SIGINT` est envoyé au lieu de `SIGSTOP` ?

Exercice 6 – Optionnel : Gestion de signal

A l'aide de la fonction `sigaction()`, écrivez le programme correspondant à l'énoncé suivant : le processus père crée un processus fils qui compte de 1 à 12 (un affichage par seconde). Un signal `SIGUSR1` est envoyé par le père au fils à 3, 5 et 8 secondes. A la réception de ce signal, le processus fils affiche la phrase 'debug: x' avec x la valeur du compteur.

Indications :

- `sigaction()` demande en argument une structure `sigaction` composé des champs `sa_handler` et `sa_flags`.
- Le premier champ est un pointeur de fonction décrivant le comportement à adopter lors de la réception du signal (aussi appelé gestionnaire de signal).
- Le second champ **doit** être initialisé à `SA_ONSTACK` pour éviter la terminaison du processus (comportement par défaut de `SIGUSR1`).
- Pour obtenir la valeur du compteur à partir du gestionnaire de signal, il faut que ce compteur soit déclaré en variable globale.