TP3 Gestion de processus - Système d'exploitation

Sereysethy Touch

16 octobre 2018

Nota :
— Que les exercices marqués * sont obligatoires, vous devez rendre vos travaux avant le **26/10/2018**.
— Pour chaque exercice, il faut toujours un fichier Makefile qui permet de compiler vos programmes et éventuellement un fichier `README.md` indiquant comment exécuter votre programme.
— S'il s'agit de documenter un programme, il faut simplement un fichier `README.md`.

1 Processus, fork et wait

Exercice 1*

Écrivez un programme qui crée un fils. Le père doit afficher "je suis père" et le pid de son processus fils, et le fils doit afficher "je suis fils" son pid et son ppid.

Exercice 2*

Écrivez un programme qui crée 4 processus fils choisissant chacun un nombre aléatoire. Pour chaque fils, affichez son PID ainsi que le numéro choisi. À la fin de votre programme le père doit afficher un message pour chacun de ses fils : le message doit comporter son PID, le PID de fils et le numéro choisi par son fils.

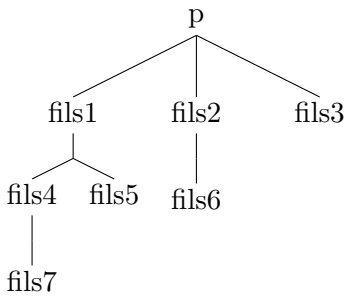
Consultez le manuel en ligne de l'appel-système `wait()`.

Exercice 3*

Écrivez un programme C qui crée deux fils, l'un affiche les entiers de 1 à 25, l'autre de 26 à 50. Si l'affichage des nombres n'est pas dans l'ordre, veuillez modifier votre programme pour que ce soit le cas.

Exercice 4*

Écrivez un programme C qui crée une famille de processus comme dans le graphe ci dessous :



Le programme doit gérer correctement la terminaison de tous les processus fils créés.

2 Famille d'exec

Exercice 1*

Écrivez un programme C **monshell** qui se comporte comme le shell, c'est-à-dire une fois lancé, il affiche une invite, prend le nom d'une commande, l'exécute, affiche le résultat d'exécution et puis ré-affiche une invite. Pour complexifier un peu le problème, **monshell** peut exécuter la commande avec des arguments mais sans redirection. Le programme se termine quand l'utilisateur saisit le mot clé `quit`.

Exercice 2* - Path et variables d'environnement

1. Écrivez un programme `affichez` qui affiche une chaîne de caractères passée en argument.
2. Écrivez un programme qui crée un processus fils qui exécute un autre programme `affichez` avec l'argument `bonjour`.

   Pour vous aider, demandez vous :
   — Que donne la commande shell `which ls` ?
   — Que donne la commande shell `echo $PATH` ?