UNIX - TD n^o 2 : Fork

1 Programme de base

Pour mettre en œuvre la fonction fork() et comprendre son exécution, écrivez un programme C qui contient :

1. une section de code qui affiche le PID (getpid()), le PPID (getppid()), le numéro de propriétaire et le numéro de groupe propriétaire du processus,
2. la création d'un processus fils avec fork() et le stockage de la valeur de retour dans val_fork.
3. une section de code identique à la section 1,
4. une branche conditionnelle (if) qui sépare les processus père et fils, grâce à la variable val_fork.
5. dans chaque partie de cette conditionnelle, on exécute du code identique à la section 1.

Expérimentez autour de ce programme pour comprendre le fonctionnement de fork(). Tirez une structure algorithmique standard d'utilisation de l'appel système fork(). C'est cette structure que vous utiliserez systématiquement par la suite.

2 Maîtrise du multitâche

Montrez par un programme adapté :

1. que les processus père et fils s'exécutent en temps partagé et qu'il peut y avoir des affichages imbriqués à l'écran,
2. que les processus père et fils ne se terminent pas en même temps (voir la fonction sleep()),
3. que vous pouvez synchroniser la fin du père avec la mort du fils avec la fonction wait().

3 Plusieurs fils

Synchronisez la fin du processus père avec la fin des plusieurs processus fils sur le calcul suivant : **(a + b) * (c + d) / (e + f)**

- **a + b** est réalisé par le fils 1,
- **c + d** est réalisé par le fils 2,
- **e + f** est réalisé par le fils 3,
- *** et /** sont réalisées par le processus père.

Les valeurs sont toutes entières.

Le père peut passer les valeurs des paramètres à ses fils de différentes manières. Quelles sont-elles ? Choisissez la plus simple à mettre en œuvre.

On utilisera les valeurs de retour des fils (return(12)) pour passer les informations des fils vers le père (par exemple 12).

Il n'y a pas de méthode de communication particulier à mettre en oeuvre dans ce TD.