Programmation Système
TP 3 : exécution de programmes

Juliusz Chroboczek
Khouloud Zine Elabidine

8 octobre 2012

L'utilisation de la fonction `system` n'est pas autorisée dans ce TP.

Exercice 1 (Fork dans une boucle)**.** Écrivez un programme qui crée treize fils dont chacun affiche son *pid*. Vérifiez que votre programme affiche bien treize lignes.

Exercice 2 (Premiers pas avec exec)**.**

1. Écrivez un programme qui exécute la commande `ls`. Votre programme devra seulement exécuter `/bin/ls` — il n'est pas nécessaire de créer un nouveau processus.
2. Modifiez votre programme pour qu'il exécute la commande `ls -l`.
3. Modifiez votre programme pour qu'il exécute la commande dans un processus fils (n'oubliez pas d'attendre la fin de l'exécution du fils).

Exercice 3 (Exec avec des vecteurs)**.**

1. Écrivez un programme `please` qui exécute la commande passée en paramètre. Par exemple, `please ls -l` exécutera la commande `ls -l`.
2. Modifiez votre programme pour que, une fois la commande exécutée, il affiche comment elle s'est terminée — soit *Terminaison normale avec résultat n*, soit *Tuée par le signal n*, soit *Bizarre*.
3. Écrivez un programme `please5` qui exécute la commande passée en paramètre cinq fois d'affilée — `please5 ls -l` exécutera la commande `ls -l` cinq fois.

Exercice 4. Écrivez un programme `si` qui se comporte (presque) comme la commande `if` du *shell*. Votre programme prendra deux ou trois paramètres : un test, une conséquence et optionnellement une alternative. Il commencera par exécuter le test ; si le test retourne 0 (succès), il exécutera la conséquence ; sinon, il exécutera l'alternative si elle est présente. Votre programme ne devra créer qu'un seul processus.

Par exemple, la commande

```
./si true emacs
```

exécutera `emacs`, tandis que

```
./si false emacs vi
```

exécutera vi. (On rappelle que les commandes `true` et `false` ne font rien et retournent 0 et 1 respectivement.)