TP4 : Clonage et mutations

Matthieu Boutier *(boutier@pps.univ-paris-diderot.fr)*

1 Compter en parallèle

1. Créez un programme `compte.c` qui lit 4096 octets [1] (`read(2)`) du fichier passé en paramètre (`argv[1]`), et affiche le nombre de 'e'.
2. Comptons en parallèle : créez un 2e processus (`fork(2)`), et utilisez le code de retour du processus (`wait(2)`) pour transmettre au père le nombre d'occurrences comptées par le fils.
3. Evidemment, le code de retour d'un processus est sur 8 bits. C'est ennuyeux si on veut compter plus de 255 occurrences. Avant le `fork()`, créez un fichier (`O_RDWR`) `/tmp/compte_tmp` dans lequel le fils écrira le résultat.
4. Supprimez le fichier temporaire juste après sa création (`unlink(2)`), et comprenez pourquoi cela est juste et bon.

2 Un mini-shell

Le but de cet exercice est d'écrire un shell minimaliste.

1. Créez un programme `my_shell.c`, qui affiche un prompt "> ", puis attends une commande et affiche la commande reçue (`printf(3), fflush(3)`).
2. Répétez cette opération jusqu'à ce que la commande "exit" soit tapée.
3. Faites en sorte que l'affichage de la commande se fasse dans un processus fils (`fork(2)`) [2].
4. A la place de l'affichage, on exécutera la commande passée en paramètre, avec `execve(2)` ou `execvp(3)` [3].
5. Étendez le shell de sorte que l'on puisse passer des arguments aux commandes (on pourra utiliser `strtok(3)`).

1. Ce programme ne considère que les 4096 premiers octets, mais peut être étendu pour traiter l'ensemble du fichier.
2. Attention, un shell attend la fin de la commande avant de redonner un prompt à l'utilisateur !
3. Les commandes passées sont considérées sans arguments.