IMA 4 - T.P. Système
Utilisation d'appels système (`fork, wait, exec, ...`)

Pour réaliser ce TP, vous n'utiliserez pas l'interface graphique mais uniquement les terminaux. Pour basculer de l'interface graphique vers le premier terminal, vous devez utiliser la combinaison de touches Ctrl + Alt + F1 . Une fois dans un terminal, vous pouvez basculer d'un terminal à l'autre en utilisant la combinaison Alt + Fn , où $1 <= n <= 6$.

Exercice 1

Écrire un programme dans lequel le processus père crée deux processus fils. Le père et ses deux fils affichent leur PID et PPID. Synchroniser correctement les processus pour empêcher le père de mourir avant ses fils. Vérifier que cette synchronisation est correctement effectuée en retardant la mort des 2 fils (placer l'instruction `sleep(2)` au début du code de chaque fils).

Exercice 2

Modifier le programme précédent de façon à avoir un père, un fils, et un petit-fils (i.e., un fils du processus fils). Synchroniser correctement les processus : un père ne doit pas mourir avant ses fils. Vous afficherez les valeurs de retours de chacun des processus fils.

Exercice 3

Écrire le code d'un processus père qui crée un processus fils tels que :

- le processus père lit sur l'entrée standard un nom de fichier qu'il stocke dans la variable `fichier`
- le processus fils exécute la commande `wc -l fichier > /tmp/result`, qui permet d'obtenir le nombre de lignes du fichier `fichier` et de rediriger la sortie standard dans un fichier.
- le processus père s'arrête après le processus fils.

Exercice 4

1. Écrire, en C, un programme qui réalise l'équivalent de l'enchainement de commandes suivant (en se basant sur le même principe qu'un shell UNIX) :

   ```
   grep ima4 /etc/passwd | cut -f5 -d: > /tmp/liste
   ```

2. Quelles sont les modifications à apporter au programme précédent pour qu'il réalise la commande :

   ```
   grep ima4 /etc/passwd | cut -f5 -d: >> /tmp/liste
   ```

Exercice 5

Écrire, en C, le code d'un processus père qui crée deux processus fils désignés fils1 et fils2. Le processus fils1 lit sur l'entrée standard une suite d'entiers terminée par 0 et transmet les entiers pairs au processus fils2 via un tube. Le processus fils2 récupère les entiers pairs sur le tube et affiche ceux qui sont supérieurs à un seuil S donné. En fin d'exécution des fils, le père affiche un message indiquant la fin du programme.

Exercice 6

Écrire un interpréteur shell complet qui permet :

- de naviguer dans l'arborescence des fichiers (utilisation de l'appel système `chdir`)
- de créer et supprimer des répertoires (utilisation des appels système `mkdir` et `rmdir`)
- de renommer un fichier (utilisation de l'appel système `rename`)
- de lister le contenu d'un répertoire (utilisation des appels sytème `opendir`, `readdir`)
- d'afficher les informations liées à un fichier (droit, uid, gid, ...) (utilisation de l'appel système `stat`)
- de lancer tous les exécutables de la machine (utilisation des appels système `fork` et `execlp`)
- de rediriger les entrées/sorties (utilisation des appels système `pipe`, `dup2`, `open`)

Pour rappel, un shell consiste en un boucle infinie qui lit sur l'entrée standard les commandes utilisateurs, puis les interprète et enfin les exécute.