Programmation Système

Fiche 2 : Création de processus

Cette fiche ainsi que les fichiers à télécharger sont disponibles sur le site
https://gforgeron.gitlab.io/progsys/

Exercice 1.1 Écrire un programme `pere-et-fils` qui lance un processus fils, père et fils déclineront ensuite leur identité.

```
je m'appelle 5585 et je suis le père de 5586
je m'appelle 5586 et je suis le fils de 5585
```

Comment faire en sorte que le processus père affiche son message toujours après son fils ? Modifier le programme de façon à ce que le fils devienne un zombie et, dans une autre version, un orphelin. On pourra utiliser la commande `ps -forest` pour voir l'arborescence des processus.

Exercice 1.2 Écrire un programme qui crée N processus fils (N étant passé en paramètre sur la ligne de commande) et qui attend la fin de tous les fils avant de se terminer à son tour. Les N processus fils se contenteront simplement d'afficher leur numéro de rang (de 0 à $N-1$) sur la sortie standard.

Exercice 1.3 Écrire une fonction `System(char *commande)` qui lance un processus exécutant un shell pour interpréter la commande passée en paramètre puis attends la fin de cette interprétation pour retourner. Il s'agit de reprogrammer la fonction `system()` sans faire appelle à celle-ci.

Vérifier que votre code exécute correctement les deux codes suivants :

```
{ System("echo bonjour"); System("echo au revoir"); }
{ if (!fork()) exit(0); System("sleep 1 ; echo bonjour") ; System("echo au revoir"); }
```

NB. Das les exercices suivants on évitera de lancer un shell pour exécuter une commande : cette technique n'est optimale ni en matière de temps d'exécution, ni en terme de programmation.

Exercice 1.4 Il s'agit de programmer un lanceur de commandes.

1. En utilisant une des fonctions de la famille `exec`, écrire la commande `execute` *commande liste-de-parametres*.

   ```
   > execute ls -l
   *** execution
   total 4
   -rw-r--r--    1 paw      paw           240 Apr  2 10:37 #shell.c#
   *** code de retour : 0
   ```

2. Comparer la valeur de retour retournée par votre shell et celle produite par bash lorsque le programme est interrompu en utilisant `ctrl-c` pour terminer le programme. Corriger au besoin votre programme [1].

1. Indice : rechercher 128 dans le manuel de bash.

Exercice 1.5 Il s'agit d'écrire une commande `apply cmd [liste d'arguments] motif` qui exécute la commande passée en paramètre sur tous les fichiers du répertoire courant répondant au motif passé en argument. Par exemple la commande `./apply echo "*"` listera tous les fichiers du répertoire courant à raison d'un nom de fichier par ligne. La commande stoppera son exécution dès qu'une exécution se termine anormalement.

On utilisera les fonctions de la famille `opendir()` pour parcourir le répertoire et la fonction `fnmatch()` pour vérifier la correspondance entre le nom de fichier et le motif donné.