

TP N°2 : Appels système & processus – 3 exercices – Durée approx. ~ 1h45min.

A l'aide du support de cours et des éventuels mémentos réalisez les exercices suivants :

(Tout code ou implémentation compilant ou non sera étudié)

Le présent sujet de TP comporte 2 pages

I – « **Fork yourself** »

Écrivez un programme générant un processus fils avec la primitive système **fork**.

- Le processus fils doit afficher son numéro (**PID**) ainsi que celui de son père à l'aide méthodes **getpid** et **getppid**, puis sort **exit** avec un code de retour égal au dernier chiffre du **PID**.
- Le processus père, quant à lui, affiche le **PID** du fils, puis attend sa terminaison via **wait** et affiche le code de retour son fils.

II – **Redirection de flux standard**

Réalisez l'exercice suivant en réalisant les étapes dans l'ordre spécifié :

1. Écrivez un programme qui affiche un message sur la sortie standard, dans ce message doit figurer le premier mot passé sur la ligne de commande après le nom du programme) (*argv[1]*) .
On s'assurera que le bon nombre de paramètres est passé sur la ligne de commande.
2. Faites évoluer le programme de sorte à ce qu'il crée un fils qui :
 - Affiche son **PID**.
 - Ferme le descripteur **1 (-STDOUT)** (*close(1)*).
 - Ouvre en création et écriture le fichier temporaire **/tmp/proc-exercise** (cf **mkstemp**).
 - Affiche le numéro du descripteur du fichier ouvert. (*Possible avec la méthode dup2*)
 - Exécute le programme affiché par son père lors de l'étape N°1 (*argv[1]*) (cf **execXXX**).
 - Le père quant à lui :
 - Affiche son **PID**.
 - Attend la fin du fils.
 - Affiche un message quelconque avant de terminer (*ie : « That's All Folks ! »*).

3. * Examinez le contenu de **/tmp/proc-exercise**, recommencer en fermant cette fois le descripteur 2.
(*Laissant le descripteur 1 ouvert*)

**facultatif*

III – **Redirection de flux via « pipe »**

Écrivez un programme C équivalent au script shell suivant :

```
ps eaux | grep "^root " > /dev/null && echo "root est connecté"
```

Votre programme devra ainsi prendre en charge les actions suivantes :

- Réaliser les exécutions de **ps** et **grep** en utilisant la primitive **execvp** dans l'ordre des priorités usuelles.
- Mettre en place les redirections des entrées/sorties nécessaires grâce à la primitive **dup** (ou **dup2**).
- Réaliser l'affichage final avec la primitive **write**.

Vous chercherez à simplifier le code pour ne prendre en charge que cette séquence de commandes.

IV – Rappels

Approche incrémentale du développement

Pour obtenir les résultats attendus aux différents exercices, veuillez toujours appliquer une approche incrémentale en termes d'ajout de code/fonctionnalité.

Par exemple: une approche incrémentale pour ce type d'exercice "ls-like" serait :

1. La récupération des paramètres.
2. Tester fichier/répertoire.
3. Parcourir les éléments du répertoire en affichant leur nom.
4. Alimenter chaque fichier avec une information supplémentaire : permission / taille / propriétaire...

Documentation

Pour obtenir des informations ou de la documentation ayez le réflexe d'utiliser les pages du manuel.

Par exemple:

- man 3 stat / man 2 open / man 2 readdir / man errno

Gestion des erreurs

Afin d'avoir une gestion des erreurs la plus précise possible ayez le réflexe d'utiliser les codes retours **ERRNO** spécifiés dans les pages de manuel

Par exemple:

- | | |
|------------------|---|
| ➤ EEXIST | File exists (POSIX.1) |
| ➤ EFAULT | Bad address (POSIX.1) |
| ➤ EISDIR | Is a directory (POSIX.1) |
| ➤ ENOTDIR | Not a directory (POSIX.1) |
| ➤ ELOOP | Too many levels of symbolic links (POSIX.1) |