

# Programmation Système

## TP 3 : gestion des processus sous Unix

### Exercice 1

Ecrire un programme C qui crée deux fils, l'un affiche les entiers de 1 à 50, l'autre de 51 à 100. Il faut que l'affichage soit 1 2...100.

### Exercice 2

A l'aide de la procédure `fork()` et en utilisant les primitives `getuid`, `getgid`,... fournies en annexe, écrire un programme qui crée un nouveau processus et qui affiche pour les deux processus (père et fils) les caractéristiques générales d'un processus :

- Identifiant du processus.
- Identifiant du père du processus.
- Le propriétaire réel.
- Le propriétaire effectif.
- Le groupe propriétaire réel.
- Le groupe propriétaire effectif.

Avant d'afficher ces informations, on prendra bien soin de préciser si l'on est dans le processus père ou fils.

### Exercice 3

- Ecrire un programme C qui crée un fils qui choisit un nombre aléatoire entre 0 et 100 et le retourne comme un code de retour. Le processus père doit attendre la terminaison de son fils et affiche le code retour de celui-ci.
- Écrire une fonction `is_in_tab(int* tab, int length, int val)` ; qui vérifie si la valeur `val` se trouve dans `tab` de longueur `length`. La valeur de retour est une coordonnée de `val` si `tab` contient `val` et -1 sinon. On effectue une recherche en parallèle en utilisant deux processus : le père recherche dans la première partie du tableau, le fils dans la seconde.

### Exercice 4

a) A l'aide des primitives `fork` et `execl`, écrire le programme qui permet de faire exécuter la commande `ps` avec l'option `-l` depuis un processus existant. Ce dernier devra se mettre en attente du code retour de la commande `ps`.

b) Reprendre le programme précédent mais en utilisant la commande `execv`.

### Exercice 5

Ecrire un programme qui crée 2 processus l'un faisant la commande `ls -l`, l'autre `ps -l`. Le père devra attendre la fin de ses deux fils et afficher quel a été le premier processus à terminer.

### Exercice 6

En utilisant la fonction `system()`, écrivez un programme permettant d'ouvrir avec la commande `cat` un fichier dont le nom a été passé en argument d'entrée de l'exécutable et qui rend ensuite la main à l'utilisateur.

### Exercice 7

Ecrire un programme C permettant de lancer la commande passée en argument.

### Exercice 8

Ecrire un programme C équivalent à la commande shell suivante :

1. `who & ps & ls -l`
2. `who; ps; ls -l`

### Exercice 9

Ecrire un programme `monshell` qui simule le fonctionnement d'un terminal de commande. Le programme doit lire les commandes tapées au clavier, créer un nouveau processus, et le recouvrir par la commande correspondante.

```
$/monshell.exe
```

```
monshell> ls
```

```
monshell.c monshell.exe toto.c titi.doc
```

```
monshell> ps
```

```
PID PPID PGID WINPID TTY UID STIME COMMAND
```

```
4660 1 4660 4660 con 12177 Oct 17 /usr/bin/bash
```

```
6040 4660 6040 5128 con 12177 Oct 17 /usr/bin/bash
```

```
5208 6040 6040 5112 con 12177 Oct 17 /usr/bin/ipc-daemon2
```

```
5868 6040 6040 5188 con 12177 Oct 17 /usr/X11R6/bin/xinit
```

```
monshell> exit
```

```
$
```