TP1 - Les processus (1)

Préambule

- Récupérer sur la plateforme Pléiad le dossier …/`sources`.
- Fonctions utiles : fork(), execl(), exit(), wait()

1 Création

a) Examinez `prg1.c`, tapez `echo $$` pour avoir le `pid` du shell puis exécutez plusieurs fois `prg1`.

- Que déduisez-vous de l'ordre d'apparition des messages ?

b) Afin de provoquer la terminaison du processus fils après celle de son père, tapez maintenant `prg1 -1`.

- Que pensez-vous du `ppid` du processus fils ?

c) Tapez :

```
prg1 -2
prg1 2
```

Expliquez la raison pour laquelle le shell vous rend la main tout de suite dans un cas et pas dans l'autre ?

d) Modifiez `prg1.c` pour que le processus père n'écrive sur la sortie standard qu'après la terminaison du processus fils quelle que soit la valeur du paramètre temps.

e) Lancez le script `essaiSaisie` en premier et en arrière-plan.

- Qu'en déduisez-vous ? (pensez à `ps` puis éventuellement `kill`).

2 Destruction

a) Lancez `dormeur 60` depuis un `xterm` puis tuez cet `xterm.` Tapez ensuite `ps` (selon la commande ps utilisée, par exemple `/usr/bin/ps -lu $USER`) depuis un autre `xterm`.

b) Recommencez la même opération avec `dormeur 60` en arrière-plan. Qu'en déduisez-vous ?

3 Héritage

3.1 Contexte d'exécution

a) Attribut "propriétaire", droits associés. Suite à :

```
prg1 30 &
ps lx
id
```

Que déduisez-vous sur le propriétaire d'un processus fils ?

b) Attribut "répertoire de travail". Exécutez :

```
essaiRepertoire
pwd
```

- Dans quel(s) sens s'effectue la transmission du contexte ?
- Pourquoi `cd` n'est-elle pas une commande externe ?

c) Espace d'adressage. Exécutez `prg3`.

- Que déduisez-vous du résultat ?

3.2 Recouvrement

a) Examinez `prg5_avant.c` et `prg5_apres.c` puis exécutez `prg5_avant.`

- Pourquoi le message "`ne passe pas ici`" n'apparaît-il pas ?

b) Afin de comprendre l'association `fork + exec` utilisée lors du lancement de commandes Unix par le shell, créez deux exécutables `pere` et `fils` qui affichent respectivement "`je suis le pere`", "`je suis le fils`" tels que le processus père n'affiche son message qu'après celui du processus fils.

3.3 Redirections

a) Expliquez le résultat de `prg1 > toto`.

b) Ecrivez une fonction `redirige(int descripteur, char * fich)` qui redirige l'entrée standard (descripteur 0) ou la sortie standard (descripteur 1) sur `fich`.

- Testez-là avec `printf` et `scanf` (`dup2`).