Gestion des Signaux

Exercice 1 : Ecrire un programme qui réalise un affichage dans une boucle infinie, mais qui prévoit de s'arrêter à la réception du signal **SIGINT**. La fonction d'interception affichera un message signalant la réception du signal avant de terminer le programme par un appel **exit( )**.

Exercice 2 : Modifier le programme précédent pour qu'il ignore le signal **SIGINT**. Lancez-le en tâche de fond (**./exo2 &**). Essayez de l'interrompre avec un **^C**. Que constatez-vous ?
Pour l'arrêter, il faut taper **ps –l** pour obtenir son numéro de processus (**pid**) et lancer la commande :

kill -9 pid (*où le signal **9** est le **SIGKILL***)

Exercice 3: Modifiez le programme précédent pour qu'il fasse son affichage dans une boucle conditionnée par une variable booléenne **fin** initialisée à **faux** et qui sera mise à **vrai** par la fonction d'interception du signal.

Exercice 4 : Ecrire un programme composé de 2 processus : Le père fait des affichages toutes les secondes dans une boucle **for** et le fils fait des affichages toutes les secondes aussi mais dans une boucle infinie, quand le compteur de boucle du père arrive à **3**, le père envoie un signal **SIGKILL** au fils. On a constaté dans l'exercice 2, l'impossibilité d'ignorer ce signal.

Exercice 5 : Recopier le programme précédent et le modifier pour que le père n'envoie plus de signal au fils mais que le fils intercepte tous les **SIGINT** en affichant un message d'interception.
Exécutez alors le programme et interrompez le par un **^C** : Que constatez-vous? Expliquez pourquoi. (Pour vous aider, utilisez la commande **ps –l**).

Exercice 6 : Reprendre le programme de l'exercice précédent et le modifier pour que le fils ne fasse ses affichages qu'à la réception du signal **SIGUSR1**. Le père envoie ce signal dans son itération 3 et dans son itération 5. Il signale la fin du traitement au fils par envoi du signal **SIGUSR2**.
Il n'y aura qu'une seule fonction d'interception dans le fils, elle recevra le numéro du signal déclencheur en paramètre.

Exercice 7 : Ecrire un programme qui demande à l'utilisateur de taper au clavier un entier en moins de **5** secondes. Pour cela, votre programme ne doit pas "planter" si ce qu'il lit n'est pas un entier. Il devra donc lire une chaîne et tenter de la convertir en un entier, si c'est bon il se terminera après avoir désarmé le **time out**, sinon il recommencera éventuellement jusqu'aux 5 secondes où il affichera "**trop tard**" avant de s'arrêter.

Exemple de sortie écran possible:

```
$./exercice8
Entrez un entier en moins de 5 secondes
Svp un entier : a
Svp un entier : x
Svp un entier : 12
Ok merci !!
```

```
$./exercice8
Entrez un entier en moins de 5 secondes
Svp un entier : E
Svp un entier : f
Svp un entier : D
Trop tard !!
```