UNIVERSITÉ DE TOULOUSE

Licence Informatique

Université Paul Sabatier
118 route de Narbonne
31062 Toulouse Cedex 09

SR2 – Partie « Systèmes » - Travaux pratiques n°1
Processus Unix – Signaux

Exercice 0 – Main paramétré

Écrire une application (qui sera nommée *boucler*) qui affiche un message, toutes les NBS secondes (utiliser la fonction *sleep()* pour temporiser), un certain nombre NBF de fois. NBS et NBF sont les **paramètres** de cette application.

Exécuter cette application pour vérifier son comportement y compris le fait que les paramètres sont fournis à l'appel.

Exemple d'exécution : ./boucler 3 5

/ Afficher 5 fois toutes les 3 secondes */*

```
Mon numero est 572, il est Wed Mar 19 15:36:55 2025
Mon numero est 572, il est Wed Mar 19 15:36:57 2025
Mon numero est 572, il est Wed Mar 19 15:36:59 2025
Mon numero est 572, il est Wed Mar 19 15:37:01 2025
Mon numero est 572, il est Wed Mar 19 15:37:03 2025
```

Exercice 1 – Se protéger

Écrire une application dans laquelle un processus **boucle indéfiniment** en affichant, toutes les secondes, un message contenant son identifiant (utiliser la fonction *sleep()* pour temporiser). Lorsque le signal SIGUSR1 lui **parviendra**, il affichera le message « **SIGUSR1** reçu par le processus de n° <pid> » avant de **continuer** sa boucle d'affichage normale.

Exécuter cette application plusieurs fois pour **vérifier** ce qu'il se passe en envoyant de temps en temps le signal SIGUSR1 à l'aide de la commande shell *kill* depuis un autre terminal (ou depuis le même terminal si vous exécutez l'application en arrière-plan).

Exemple d'exécution : ./exo1

```
Je suis le processus 587 Wed Mar 19 15:38:52 2025
>> SIGUSR1 recu par 587
Je suis le processus 587 Wed Mar 19 15:38:52 2025
Je suis le processus 587 Wed Mar 19 15:38:53 2025
Je suis le processus 587 Wed Mar 19 15:38:54 2025
>> SIGUSR1 recu par 587
Je suis le processus 587 Wed Mar 19 15:38:54 2025
Je suis le processus 587 Wed Mar 19 15:38:55 2025
>> SIGUSR1 recu par 587
Je suis le processus 587 Wed Mar 19 15:38:56 2025
Je suis le processus 587 Wed Mar 19 15:38:57 2025
Je suis le processus 587 Wed Mar 19 15:38:58 2025
^C
```

Remarque : L'affichage de la date n'est pas demandé, cela est destiné à vous montrer le respect de la périodicité du traitement.

Exercice 2 – Protéger sa descendance

En vous inspirant de l'exercice précédent, écrire une application dans laquelle le père crée un processus **fils** qui **bouclera indéfiniment** en affichant, toutes les secondes, un message l'identifiant et en affichant un message spécifique lorsque le signal **SIGINT** lui parvient.

Faire en sorte que le père **se termine lorsque** son fils se termine effectivement (et non avant).

Exécuter l'application plusieurs fois – en envoyant SIGINT par appui des touches Ctrl-C – pour vérifier son bon fonctionnement.

Exercice 3 – Signaux et primitives de recouvrement

Écrire une application dans laquelle le processus principal **se protège contre** le signal SIGINT avant de **remplacer** son code (par recouvrement ou commutation d'image – voir rappels de cours) par celui de l'exécutable *boucler* créé à l'exercice 0.

Cette application sera **paramétrée** par le nombre de messages à afficher et leur périodicité.

Exécuter cette application pour **vérifier** ce qu'il se passe lorsqu'on envoie le signal SIGINT – par appui des touches Ctrl-C au clavier – après la commutation d'image.

Exemple d'exécution : ./exo3 2 10

/ Afficher 10 fois toutes les 2 secondes */*

```
Processus de pid 665 : Je suis protege contre SIGINT
Et je vais devenir l'executable boucler pour afficher 10 fois toutes les 2 secondes
Mon numero est 665, il est Wed Mar 19 15:52:56 2025
Mon numero est 665, il est Wed Mar 19 15:52:58 2025
Mon numero est 665, il est Wed Mar 19 15:53:00 2025
```

. . . . à vous de tester l'effet du Ctrl-C . . .

Exercice 4 – Envois de signaux entre processus

Écrire une application dans laquelle un processus père et **deux** processus fils s'exécutent en **parallèle**.

Chaque processus fils compte les caractères tapés au clavier. À chaque fois que son compte atteint un multiple de NB, il envoie un **signal** à son père pour lui indiquer que NB caractères **de plus** ont été lus. Un fils se termine lorsqu'il a lu NBMAX caractères.

Le processus père affiche le nombre de caractères qui ont été lus par chaque fils **au fur et à mesure** de la réception des informations. Le père se **termine lorsque ses fils se sont terminés**.

NB et NBMAX sont les **paramètres** de l'application.

Exécutez votre programme **plusieurs fois**, que constatez-vous parfois ?

Attention : Est-ce **normal** ? Et si oui, **pourquoi** ?

Exemple d'exécution : ./exo4 2 6

/ Les fils envoient la mise à jour au père tous les 2 caractères lus – Ici, on a tapé les caractères 12ab<entrée>34cd<entrée>56 - et se terminent quand ils ont compté 6 caractères (chacun) */*

```
12ab
        Fils 0 (673) : 2 caracteres de plus => 2
Pere (672) - Fils 2 : nombre de caracteres = 2
        Fils 1 (674) : 2 caracteres de plus => 2
Pere (672) - Fils 1 : nombre de caracteres = 2
34cd
        Fils 0 (673) : 2 caracteres de plus => 4
Pere (672) - Fils 1 : nombre de caracteres = 4
        Fils 0 (673) : 2 caracteres de plus => 6
        Fils 0 (673) : Termine
Pere (672) - Fils 1 : nombre de caracteres = 6
        Fils 1 (674) : 2 caracteres de plus => 4
Pere (672) - Fils 2 : nombre de caracteres = 4
56
        Fils 1 (674) : 2 caracteres de plus => 6
        Fils 1 (674) : Termine
Pere (672) - Fils 2 : nombre de caracteres = 6
        Pere (672) - Je me termine en dernier
```

Exercice 5 – Affichages périodiques

On veut modifier l'application précédente pour que le père affiche la valeur des compteurs à intervalles de temps **réguliers** et non pas à chaque réception de l'information.

Le comportement des fils reste identique.

Le père se termine quand les compteurs des fils ont atteint la valeur limite NBL.

L'application doit **pouvoir** être arrêtée par un signal SIGINT.

Dans cet exercice, il est **interdit** d'utiliser la fonction *sleep()*.

Exemple d'exécution : ./exo5 2 6 2

/ Les fils envoient la mise à jour au père tous les 2 caractères comptés et se terminent quand ils ont compté 6 caractères (chacun). Le père affiche les compteurs toutes les 2 secondes – On a tapé les caractères ab12<entrée>cd34<entrée>e5<entrée> */*

```
Pere (680) - Wed Mar 19 15:55:26 2025 : Fils 1 : nombre de caracteres = 0
Pere (680) - Wed Mar 19 15:55:26 2025 : Fils 2 : nombre de caracteres = 0
aPere (680) - Wed Mar 19 15:55:28 2025 : Fils 1 : nombre de caracteres = 0
Pere (680) - Wed Mar 19 15:55:28 2025 : Fils 2 : nombre de caracteres = 0
b12
        Fils 0 (681) : 2 caracteres de plus => 2
        Fils 1 (682) : 2 caracteres de plus => 2
Pere (680) - Wed Mar 19 15:55:30 2025 : Fils 1 : nombre de caracteres = 2
Pere (680) - Wed Mar 19 15:55:30 2025 : Fils 2 : nombre de caracteres = 2
cdPere (680) - Wed Mar 19 15:55:32 2025 : Fils 1 : nombre de caracteres = 2
Pere (680) - Wed Mar 19 15:55:32 2025 : Fils 2 : nombre de caracteres = 2
34
        Fils 1 (682) : 2 caracteres de plus => 4
        Fils 0 (681) : 2 caracteres de plus => 4
ePere (680) - Wed Mar 19 15:55:34 2025 : Fils 1 : nombre de caracteres = 4
Pere (680) - Wed Mar 19 15:55:34 2025 : Fils 2 : nombre de caracteres = 4
5
        Fils 0 (681) : 2 caracteres de plus => 6
        Fils 1 (682) : 2 caracteres de plus => 6
        Fils 1 (682) : Termine
        Fils 0 (681) : Termine
Pere (680) - Wed Mar 19 15:55:36 2025 : Fils 1 : nombre de caracteres = 6
Pere (680) - Wed Mar 19 15:55:36 2025 : Fils 2 : nombre de caracteres = 6
        Pere (680) - Je me termine (en dernier)
```

Remarque : Selon les entrelacements des exécutions, il est normal que vous n'obteniez pas **exactement** cet affichage.

Exemple d'exécution : ./exo5 2 6 1

/*L*e père affiche toutes les secondes. On a tapé les caractères ab et arrêté l'exécution par Ctrl-C */*