Programmation des signaux (ANSI C)

Ce document comporte 4 pages.

1. Exemple de gestion de signal

Le programme suivant boucle jusqu'à recevoir un signal d'alarme.

Listing 1 – Programme boucle-until-alarme.c

```
1  #include <signal.h>
2  #include <stdio.h>
3  #include <stdlib.h>
4
5  volatile sig_atomic_t keep_going = 1;
6
7  void
8  catch_alarm (int sig) {
9          keep_going = 0;
10         signal (sig, catch_alarm);
11 }
12
13 void
14 do_stuff (void) {
15         puts ("Faire quelque chose en attendant l'alarme...");
16 }
17
18 int
19 main (void) {
20         signal (SIGALRM, catch_alarm);
21         alarm (2);
22         while (keep_going)
23                 do_stuff ();
24         return EXIT_SUCCESS;
25 }
```

2. Handler

Exercice 2.1. Le listing suivant montre le programme d'un chronomètre, qui utilise la fonction `alarm`, en supposant que :

- la résolution du chronomètre est de 1 seconde,
- le signal SIGINT servira à démarrer et arrêter le chronomètre
- le signal SIGUSR2 servira à provoquer l'affichage de la durée écoulée sans arrêter le chronomètre.

Listing 2 – Programme chronometre.c

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <unistd.h>
4  #include <signal.h>
5
6  int secondes;
7
8  void
```

```
9  handler_alarm() {
10         alarm(1);
11         secondes++;
12 }
13
14 void
15 handler_int_fin() {
16         printf("Le nombre de secondes est %d\n", secondes);
17         exit(EXIT_SUCCESS);
18 }
19
20 void
21 handler_int_dep() {
22         printf("Debut\n");
23         alarm(1);
24         signal(SIGALRM, handler_alarm);
25         signal(SIGINT, handler_int_fin);
26 }
27
28 void
29 handler_usr() {
30         printf("Le nombre de secondes est %d\n", secondes);
31 }
32
33 int
34 main() {
35         secondes = 0;
36         printf("Moi %d\n", getpid());
37         fflush(stdout);
38         signal(SIGINT, handler_int_dep);
39         signal(SIGUSR1, handler_usr);
40         signal(SIGUSR2, handler_usr);
41
42         while(1);
43         return EXIT_SUCCESS;
44 }
```

Exercice 2.2. En vous inspirant du programme précédent, programmez un chronomètre avec *stop-resume*, qui arrête le temps, (toujours en utilisant la fonction `alarm`), en supposant que :

- la résolution du chronomètre est de 1 seconde,
- le signal SIGINT servira à démarrer et arrêter le chronomètre,
- le signal SIGUSR1 servira à arrêter temporairement le chronomètre et afficher le temps intermédiaire,
- le signal SIGUSR2 servira à redémarrer le chronomètre à partir du temps précédemment affiché.

Exercice 2.3. Ecrivez un programme qui permet d'attraper les signaux SIGINT et SIGQUIT :

- à chaque réception de SIGINT, on affiche seulement un message indiquant la réception de ce signal ;
- à la réception de SIGQUIT, on quitte le programme.

3. Délivrance de signaux entre père et fils

Exercice 3.1. Reprenez et programmez le dernier programme du cours, tel que le processus père roupille pendant 3 secondes avant d'envoyer SIGUSR1 au fils.

Exercice 3.2. Écrivez une variante du programme précédent où :

- le père envoie le signal SIGUSR1 de manière répétitive pendant un certain nombre de fois N,
- le fils traite le signal pendant un certain nombre de fois M.

Faites varier N et M tel que $N \neq M$ et expliquez ce que vous observez.

Exercice 3.3. Écrivez un programme dans lequel le père envoie un nombre de fois N le signal SIGUSR1 au fils. Le fils affiche à la fin le nombre de signaux reçus du père.

Qu'observez-vous ? Proposez une solution pour synchroniser tous les signaux entre père et fils.

4. Accès atomique à des données

Exercice 4.1. Écrivez un programme qui remplit dans une boucle infinie une structure de données composée de deux champs de type entier. Le programme remplit alternativement cette structure de données de zéro et de un. Initialement la zone mémoire est remplie de zéro.

La structure de données est définie de la manière suivante :
- `struct deux_mots {long a, long b; } memoire;`

Ce programme doit gérer deux signaux :
- SIGALRM, dont le handler affiche le contenu de la structure et repositionne l'alarme. La résolution de l'alarme est 1.
- SIGINT pour arrêter et quitter le programme.

Que constatez-vous ? Pouvez-vous expliquer ce phénomène ?

5. Utilisation de pause et sleep pour attendre un signal

Exercice 5.1. Avant de l'écrire, compiler et exécuter, expliquez le comportement du programme suivant.

Listing 3 – Programme à expliquer

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <unistd.h>
4  #include <sys/types.h>
5  #include <sys/time.h>
6  #include <signal.h>
7
8  pid_t pid = 0;
9
10 void
11 handler(int s) {
12   int heures, minutes, secondes;
13   struct timeval t;
14
15   gettimeofday(&t,NULL);
16   heures = (t.tv_sec/60/60)%24;
17   minutes = (t.tv_sec/60)%60;
18   secondes = t.tv_sec%60;
19
20   fprintf(stderr,
21           "Processus %d a recu un signal    %dh%d:%d\n",
22           pid, heures, minutes, secondes);
23   /* Peut etre necessaire selon les systemes. */
24   signal(SIGUSR1, handler);
25 }
26
27 int
28 main() {
29   pid = getpid();
30   fprintf(stderr, "Lancement du processus parent %d\n",pid);
31   /* signaler au systeme que la fonction handler() gere les
32      interruptions de type SIGUSR1. */
33   signal(SIGUSR1, handler);
34   switch ((pid = fork())) {
35         case -1:
36                 perror("fork()");
37                 exit(EXIT_FAILURE);
38         case 0:
39                 pid = getpid();
40         for(;;) pause();
41                 break;
42         default:
43         while (1) {
```

```
44                      sleep(1); /* Toutes les secondes */
45                      kill(pid, SIGUSR1);
46              }
47      }
48      printf("fin\n");
49      return EXIT_SUCCESS;
50  }
```

Exercice 5.2. Écrire un programme C qui crée deux processus à l'aide de l'appel système fork(). Le père affichera les entiers pairs compris entre 1 et 100, le fils affichera les entiers impairs compris dans le même intervalle. Synchroniser les processus à l'aide des signaux (e.g., SIGUSR1), de pause() et d'un drapeau positionné par le handler, pour que l'affichage soit 1 2 3 ... 100.

Quel problème potentiel pose l'utilisation de pause() pour l'attente d'un signal ?

Exercice 5.3. Écrire une deuxième version du programme ci-dessus en remplaçant l'appel à pause par un appel à sleep au sein d'une boucle vérifiant la valeur du drapeau.

Exercice 5.4. Écrire un programme qui utilise 3 processus H, M, S qui incrémentent les 3 « aiguilles » d'une horloge. S reçoit un signal SIGALRM chaque seconde et émet un signal SIGUSR1 à M quand son compteur passe de 59 à 0. Quand M reçoit un signal, il incrémente son compteur. Quand son compteur passe de 59 à 0, M envoie un signal SIGUSR2 à H. Quand le compteur du processus H atteint 24, il repasse à 0.

Un exemple de trace d'exécution est fourni dans le fichier heuresMinutesSecondes.trace.

Vous pouvez établir la hiérarchie suivante : le processus H est le père de M, qui est le père de S.