

Séances 5 et 6 : Les signaux

Sommaire

- ☒ Exercice 1
- ☒ Exercice 2
- ☒ Exercice 3
- ☐ Exercice 4
- ☐ Exercice 5
- ☐ Exercice 6

Exercice 1

1. On écrit le code de notre timer :

```
#include <unistd.h> /* sleep */
#include <stdio.h> /* printf */

int main()
{
    printf("=====\n");
    printf("Lancement du compteur\n");
    printf("=====\n");
    for (int count = 1; count <= 30; count++)
    {
        printf("%d\n", count);
        sleep(1);
    }
    printf("=====\n");
    printf("Terminaison du compteur\n");
    printf("=====\n");
}
```

On obtient bien la sortie attendue :

```
=====
Lancement du compteur
=====
1
2
3
4
5
6
7
8
9
```

```
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
=====
Terminaison du compteur
=====
```

Pour envoyer le signal SIGINT au processus associé à l'exécution de ce programme, on peut utiliser la commande suivante :

```
kill -s SIGINT <PID>
```

ou simplement utiliser le raccourci **CTRL+C**. Cela provoque l'interruption de notre processus.

2. On ajoute à notre code la procédure suivante :

```
void SIGINT_handler(int sig)
{
    printf("Olé!\n");
}
```

Il ne nous reste plus qu'à appeler la procédure signal dans notre main :

```
int main()
{
    signal(SIGINT, SIGINT_handler);
    // ...
}
```

On obtient bien en sortie (à chaque envoi du signal SIGINT):

```
=====
Lancement du compteur
=====
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
Olé!
21
22
23
Olé!
24
25
26
Olé!
27
28
Olé!
29
Olé!
30
=====
Terminaison du compteur
=====
```

Exercice 2

On écrit le code suivant :

```
#include <stdio.h>
#include <stdlib.h>
```

```
#include <unistd.h>
#include <signal.h>

void Solutiend(int);

void initRand()
{
    FILE *urand = fopen("/dev/random", "r");
    unsigned int seed;
    fread(&seed, sizeof(int), 1, urand);
    fclose(urand);
    // fprintf(stderr, "seed : %u\n", seed);
    srand(seed);
}

int a;
int b;

int main(int argc, char *argv[])
{
    if (argc == 2 && atoi(argv[1]) >= 0)
    {
        initRand();
        signal(SIGALRM, Solutiend);
        signal(SIGINT, Solutiend);
        a = rand() % 11;
        b = rand() % 11;
        int answer;
        int time = atoi(argv[1]);
        printf("Vous avez %d seconde(s) pour résoudre la multiplication\n", time);
        printf("%d x %d = ", a, b);
        if (time != 0)
        {
            alarm(time);
            scanf("%d", &answer);
            if (answer == a * b)
            {
                printf("Bravo! \n");
            }
            else
            {
                printf("Dommage, mauvaise réponse\n");
            }
        }
        else
        {
            printf("\nLa solution était : %d\n", a * b);
        }
    }
    else
    {
        printf("/!\ Erreur : veuillez renseigner un unique paramètre /\n");
    }
}
```

```
    }

    return EXIT_SUCCESS;
}

void Solutiend(int sig)
{
    printf("\nLa solution était : %d\n", a * b);
    exit(EXIT_SUCCESS);
}
```

Exercice 3

1. Avec le code suivant, on appelle la primitive `signal` autant de fois qu'il y a de signaux disponible et pour chacun d'eux, notre procédure `sigCapture` récupère son identifiant et renvoie son nom grâce à la primitive `strsignal`.

```
void sigCapture(int sig)
{
    printf("Je viens de recevoir le signal : %s\n", strsignal(sig));
}

int main()
{
    for (int i = 1; i <= 31; i++)
    {
        signal(i, sigCapture);
    }
    while (1)
    {
    }
}
```

2. On rajoute le code du fils suivant après avoir appelé la primitive `fork`

```
if (pid_fils == 0)
{
    initRand();
    for (int i = 0; i < 50; i++)
    {
        int sig = 9;
        while (sig == 9 || sig == 19)
        {
            sig = (rand() % 31) + 1;
        }
        kill(getppid(), sig);
        printf("Fils : Je viens d'envoyer le signal : %s\n",
strsignal(sig));
        sleep(1);
    }
}
```

```
    }  
    int sig = 9;  
    kill(getppid(), sig);  
    printf("Fils : Je viens d'envoyer le signal : %s\n",  
strsignal(sig));  
    exit(EXIT_SUCCESS);  
}
```

Exercice 4