

## 1 Introduction

Dans ce TD/TP, nous allons voir comment sont gérés les signaux, quels sont les problèmes qu'ils apportent et comment les résoudre.

## 2 Exercice 1

L'objectif de ce premier exercice est de comprendre certains mécanismes.

1. Soit le code suivant dont le résultat est donné ci-contre,

```
/* fichier mon_fils_est_fini.c */
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <signal.h>
#include <sys/types.h>
#include <sys/wait.h>

void mon_fils_est_fini(int sig){
    printf("mon fils est terminé... \n");
}

int main(){
    pid_t pid;
    system("clear"); // une belle console...
    if( (pid = fork())==0){ // Dans le fils
        printf("fils : je me met en pause, mon PID est: %d\n", getpid());
        for(;;); // boucle infinie
    }
    /* le père stoppe son fils */
    printf("père, mon PID : %d\n",getpid());
    system("ps -lH");
    (void) signal(SIGCHLD, mon_fils_est_fini);
    printf("appuyez sur une touche pour stopper le fils\n");
    getchar();
    kill(pid, SIGINT); // le père interrompt le fils pid
    system("ps -lH");
    waitpid(pid, NULL, 0);
    system("ps -lH");
    return EXIT_SUCCESS;
}
```

```

Fichier  Édition  Affichage  Terminal  Onglets  Aide

papa, PID : 20646
fils : je suis en pause, PID : 20649
F S  UID  PID  PPID  C  PRI  NI  ADDR  SZ  WCHAN  TTY          TIME CMD
0 S  1000  18738  18733  0  80   0  -  1597  wait   pts/0      00:00:00 bash
0 S  1000  20646  18738  0  80   0  -  406  wait   pts/0      00:00:00 mon_fils_est_fi
1 R  1000  20649  20646  0  80   0  -  406  -      pts/0      00:00:00 mon_fils_est_fi
0 S  1000  20650  20646  0  80   0  -  461  wait   pts/0      00:00:00 sh
0 R  1000  20651  20650  0  80   0  -  628  -      pts/0      00:00:00 ps
appuyez sur une touche pour stopper le fils

F S  UID  PID  PPID  C  PRI  NI  ADDR  SZ  WCHAN  TTY          TIME CMD
0 S  1000  18738  18733  0  80   0  -  1597  wait   pts/0      00:00:00 bash
0 S  1000  20646  18738  0  80   0  -  407  wait   pts/0      00:00:00 mon_fils_est_fi
1 Z  1000  20649  20646  99  80   0  -  0  exit   pts/0      00:00:05 mon_fils_est_fi <defunct>
0 R  1000  20656  20646  0  80   0  -  461  -      pts/0      00:00:00 sh
0 R  1000  20657  20656  0  80   0  -  628  -      pts/0      00:00:00 ps
mon fils est fini...
F S  UID  PID  PPID  C  PRI  NI  ADDR  SZ  WCHAN  TTY          TIME CMD
0 S  1000  18738  18733  0  80   0  -  1597  wait   pts/0      00:00:00 bash
0 R  1000  20646  18738  0  80   0  -  407  -      pts/0      00:00:00 mon_fils_est_fi
0 S  1000  20658  20646  0  80   0  -  461  wait   pts/0      00:00:00 sh
0 R  1000  20659  20658  0  80   0  -  628  -      pts/0      00:00:00 ps
mon fils est fini...

```

1. Pourquoi y a-t-il 2 fois le message “*mon fils est terminé...*” d’affiché ?
2. **Modifiez** le programme, pour que cela n’arrive pas.
3. **Concluez**.

### 3 Exercice 2

Vous utilisez la primitive **signal** pour cet exercice.

**Écrivez** un programme C dont le comportement est une boucle infinie et qui envoie sur STDOUT le message BONJOUR lorsqu’il reçoit le signal SIGUSR1 et le message BONSOIR lorsqu’il reçoit le signal SIGUSR2.

Pour SIGUSR1 cela ne doit se produire qu’une seule fois par contre il doit réagir autant de fois que le signal SIGUSR2 lui arrive.

Pour lui envoyer les signaux utilisez la commande :

**# kill -10 *nřpid* pour SIGUSR1**

**# kill -12 *nřpid* pour SIGUSR2**

### 4 Exercice 3

Vous utilisez la primitive **sigaction** pour cet exercice.

1. **Écrivez** un programme C permettant d’empêcher qu’un processus soit tué lorsqu’il exécute une division par zéro.

Ce programme explique ce qui s’est passé.

Dans un premier temps on autorisera qu’une seule fois cette erreur.

2. **Modifiez** le code afin que si l’**exception** se reproduit, on explique à nouveau et en permanence.

### 5 Concluez