 الجامعة الدولية ٢٠٠٨/٢٠٠٩ : ٢٠٢٠/٢٠٢١ Université Internationale de Rabat L'UNIVERSITÉ INNOVANTE	Module : <i>Système d'exploitation et programmation système</i>	Année universitaire : 2019/2020
	<b><i>TP N°5 : Les signaux</i></b>	
	Intervenants : <i>Pr M. BAKHOUYA, A. Kharbouch, H. El Khoukhi</i>	

## Rapport

Fait Par : EL HANAFI Maha

### **Objectif :**

Création et communication entre processus

## Exercice1 :

```
1  #include <unistd.h>
2  #include <stdio.h>
3  #include <signal.h>
4  #include <stdlib.h>
5
6  void comportement(int sig)
7  {
8      /*traitement d'un signal de type SIGUSR1 */
9      if(sig== SIGUSR1)
10     {
11         /*##### COMPLETER #####*/
12         /*A la réception du signal SIGUSR1*/
13         printf("Fct Comportement : Processus [%d] a reçu SIGUSR1 \n",getpid());
14         printf("Fct Comportement : processus [%d] envoi SIGUSR2 au [%d]\n",getpid(),getppid());
15         /*le fils envoie à son père un signal SIGUSR2*/
16         kill(getppid(), SIGUSR2);
17         /* Remarque : SIGINT - Interruption du processus (stop) = Ctrl + c.
18          |         | La fonction comportement ne traitera pas SIGINT
19         */
20     /*##### COMPLETER #####*/
21     /*le fils se suicide en s'envoyant un signal SIGINT */
22     printf("Fct Comportement : Processus [%d] se suicide par SIGINT\n",getpid());
23     kill(getpid(), SIGINT);
24     }
25     /*Reception d'un signal de type SIGUSR1 */
26     else if(sig== SIGUSR2)
27     { /* A la réception du signal SIGUSR2, le
28      | père se suicide également de la même manière
29      */
30         printf("Fct Comportement : Processus [%d] a reçu SIGUSR2 \n",getpid());
31         printf("Fct Comportement : Processus [%d] se suicide par SIGINT\n",getpid());
32         /*##### COMPLETER #####*/
33         /*le père se suicide en s'envoyant un signal SIGINT */
```

```
32     /*##### COMPLETER #####*/
33     /*le père se suicide en s'envoyant un signal SIGINT */
34     kill(getpid(),SIGINT);
35 }
36 }
37 int main() {
38     int pid,ppid,statut;
39     /*pid du processus père*/
40     ppid=getpid();
41     printf("Processus Père [%d] lancé.\n",ppid);
42     //Redéfinir le comportement d'un processus à la reception
43     //d'un signal
44     struct sigaction action;
45     action.sa_handler=comportement;
46     /*##### COMPLETER #####*/
47     /*Redéfinir le comportement à la reception
48     du signal SIGUSR1. La fonction dans "action" (comportement)
49     va prendre en charge le traitement de ce signal
50     */
51     //Utiliser la fonction sigaction : int sigaction(int signum, const struct sigaction *act, struct sigaction *oldact);
52     sigaction(SIGUSR1, &action, NULL);
53     /*##### COMPLETER #####*/
54     /*
55     Redéfinir le comportement à la reception
56     du signal SIGUSR2. La fonction dans "action" (comportement)
57     va prendre en charge le traitement de ce signal
58     */
59     //Utiliser la fonction sigaction : int sigaction(int signum, const struct sigaction *act, struct sigaction *oldact);
60     sigaction(SIGUSR2, &action, NULL);
61     /*Creation du processus fils*/
62     pid=fork();
63     /* Erreur du fork */
```

```

63  /* Erreur du fork */
64  if (pid < 0)
65  {
66      perror("Erreur de creation du fils");
67      exit(1);
68  }
69  /*Processus fils*/
70  else if(pid==0)
71  {
72      /*##### COMPLETER #####*/
73      /*Le processus fils entre ensuite dans une boucle sans fin*/
74      printf("Processus Fils [%d] : lancé avec succès\n",getpid());
75      printf("Processus Fils [%d] : est entré en boucle infini\n",getpid());
76      while(1);
77  }
78  /*processus Père */
79  else
80  {
81      /*##### COMPLETER #####*/
82      /* Le processus père attend quelques secondes*/
83      printf("Processus Père [%d]: Création du processus fils [%d] est terminée avec succès\n",ppid,pid);
84      printf("Processus Père [%d]: attente de 8 secondes\n",ppid);
85      sleep(8);
86      /*##### COMPLETER #####*/
87      /*puis envoie à son fils le signal SIGUSR1*/
88      statut= kill(pid, SIGUSR1);
89
90      if(statut == 0)
91      {
92          /*##### COMPLETER #####*/
93          /*entre ensuite à son tour dans une boucle sans fin*/
94          printf("Processus Père [%d]: SIGUSR1 envoyé au processus Fils [%d]\n",ppid,pid);
95          while(1);
96      }

```

```

95      while(1);
96  }
97  else
98  {
99      /*##### COMPLETER #####*/
100     printf("Processus Père [%d]: Erreur Envoi de SIGUSR1 au processus fils [%d]\n",ppid,pid);
101     }
102 }
103 }

```

## Résultat d'exécution :

```

uir_student@ubuntu:~$ gcc -o ex51 ex51.c
uir_student@ubuntu:~$ ./ex51
Processus Père [3865] lancé.
Processus Père [3865]: Création du processus fils [3866] est terminée avec succès
Processus Père [3865]: attente de 8 secondes
Processus Fils [3866] : lancé avec succès
Processus Fils [3866] : est entré en boucle infini
Processus Père [3865]: SIGUSR1 envoyé au processus Fils [3866]
Fct Comportement : Processus [3866] a reçu SIGUSR1
Fct Comportement : processus [3866] envoi SIGUSR2 au [3865]
Fct Comportement : Processus [3866] se suicide par SIGINT
Fct Comportement : Processus [3865] a reçu SIGUSR2
Fct Comportement : Processus [3865] se suicide par SIGINT

```

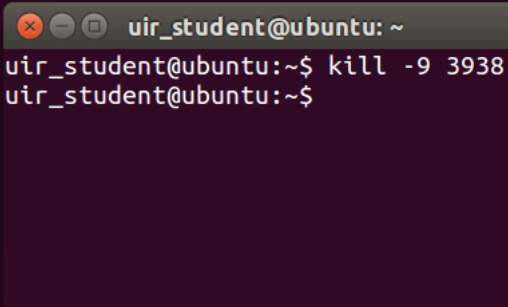
## Exercice2 :

### 2.1

```
1  #include <stdio.h>
2  #include <signal.h>
3  #include <unistd.h>
4
5  /*
6   TP5 - Signaux
7   Intervenants : Profs M.Bakhouya, A.Kharbouch, H.El Khoukhi
8   */
9
10 /*Le signal SIGINT correspond à l'interruption
11 du terminal (CTRL-C). Son action par défaut est la terminaison.
12
13 Pour tester, executer ce programme sur un terminal et appuyer sur Ctrl+c
14 Pour tuer ce programme, utiliser kill -9 pid dans un autre terminal. (pid est le pid du programme)
15 */
16
17 void handINT(int signo){
18     signal(SIGINT,handINT); // set the handler for SIGINT, the process will never stop
19     printf("Reception d'un signal SIGINT\n");
20 }
21
22 void main(){
23     signal(SIGINT,handINT); // set the handler for SIGINT
24     printf("processus : %d \n",getpid());
25     while(1);
26 }
27
```

### Résultat du l'exécution :

```
uir_student@ubuntu:~$ gedit ex52.c
uir_student@ubuntu:~$ gcc -o ex52 ex52.c
uir_student@ubuntu:~$ ./ex52
processus : 3938
^CReception d'un signal SIGINT
^CReception d'un signal SIGINT
^CReception d'un signal SIGINT
^CReception d'un signal SIGINT
^CReception d'un signal SIGINT
^CReception d'un signal SIGINT
^CReception d'un signal SIGINT
Killed
uir_student@ubuntu:~$
```



## 2.2

```
1  #include <stdio.h>
2  #include <unistd.h>
3  #include <signal.h>
4
5  void handUSR1 ( int signo) {
6
7  signal(SIGUSR1, SIG_IGN);
8  printf ("BONJOUR\n") ;
9
10 }
11
12 void handUSR2 ( int signo) {
13
14     //signal(SIGUSR2, handUSR2);
15     printf ("BONSOIR\n") ;
16 }
17
18 void main () {
19     S
20     signal (SIGUSR1, handUSR1) ;
21     signal (SIGUSR2, handUSR2) ;
22     printf("processus : %d\n", getpid ( ));
23     while (1) ;
24
25 }
26
```

### Résultat d'exécution :

```
uir_student@ubuntu:~$ gedit ex522.c
uir_student@ubuntu:~$ gcc -o ex522 ex522.c
uir_student@ubuntu:~$ ./ex522
processus : 3447
BONJOUR
BONSOIR
^C
uir_student@ubuntu:~$ gedit ex522.c
```

uir\_student@ubuntu: ~

bash: /usr/include: Is a directory

uir\_student@ubuntu:~\$ kill -10 3447

uir\_student@ubuntu:~\$ kill -10 3447

uir\_student@ubuntu:~\$ kill -10 3447

uir\_student@ubuntu:~\$ kill -12 3447

uir\_student@ubuntu:~\$

