

LABORATORY 5

—

OPERATING SYSTEMS

RUFINO GARCIA SANCHEZ

EXERCISE 1

Write a program which has installed handlers for SIGUSR1, SIGTERM, SIGINT, SIGTSTP signals.

These handlers should display the messages with names of these signals.

Prepare **two versions of this program**:

- using signal function
- using sigaction function.

Signal versión:

```
}else if(atoi(argv[1]) == 2){  
    printf("\n-----WE ARE USING THE signal FUNCTION-----\n\n");  
    /* Display a message indicating we are registering the signal handler */  
    printf("Registering the signal handler\n");  
    /* Register the signal handler */  
    signal(SIGUSR1, signal_handler);  
    /* Display a message indicating we are raising a signal */  
    printf("Raising a SIGUSR1 signal\n");  
    /* Raise the SIGUSR1 signal */  
    raise(SIGUSR1);  
    /* Display a message indicating we are leaving main */  
    printf("Finished the signal SIGUSR1\n");  
    /* Display a message indicating we are registering the signal handler */  
    printf("\nRegistering the signal handler\n");  
    /* Register the signal handler */  
    signal(SIGTERM, signal_handler);  
    /* Display a message indicating we are raising a signal */  
    printf("Raising a SIGTERM signal\n");  
    /* Raise the SIGTERM signal */  
    raise(SIGTERM);  
    /* Display a message indicating we are leaving main */  
    printf("Finished the signal SIGTERM\n");  
    /* Display a message indicating we are registering the signal handler */  
    printf("\nRegistering the signal handler\n");  
    /* Register the signal handler */  
    signal(SIGINT, signal_handler);  
    /* Display a message indicating we are raising a signal */  
    printf("Raising a SIGINT signal\n");  
    /* Raise the SIGINT signal */  
    raise(SIGINT);  
    /* Display a message indicating we are leaving main */  
    printf("Finished the signal SIGINT\n");  
    /* Display a message indicating we are registering the signal handler */  
    printf("\nRegistering the signal handler\n");  
    /* Register the signal handler */  
    signal(SIGTSTP, signal_handler);  
    /* Display a message indicating we are raising a signal */  
    printf("Raising a SIGTSTP signal\n");  
    /* Raise the SIGTSTP signal */  
    raise(SIGTSTP);  
    /* Display a message indicating we are leaving main */  
    printf("Finished the signal SIGTSTP\n");
```

Signaction version:

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

void signal_handler(int signal);

int main(int argc, const char * argv[]){
    if (atoi(argv[1]) == 1){
        struct sigaction sa;
        sa.sa_handler = signal_handler;

        printf("\n-----WE ARE USING THE Sigaction FUNCTION-----\n\n");

        /* Display a message indicating we are registering the signal handler */
        printf("Registering the signal handler\n");

        /* Register the signal handler */
        sigaction(SIGUSR1, &sa, NULL);

        /* Display a message indicating we are raising a signal */
        printf("Raising a SIGUSR1 signal\n");

        /* Raise the SIGUSR1 signal */
        raise(SIGUSR1);

        /* Display a message indicating we are leaving main */
        printf("Finished the signal SIGUSR1\n");

        /* Display a message indicating we are registering the signal handler */
        printf("\nRegistering the signal handler\n");

        /* Register the signal handler */
        sigaction(SIGTERM, &sa, NULL);

        /* Display a message indicating we are raising a signal */
        printf("Raising a SIGTERM signal\n");

        /* Raise the SIGTERM signal */
        raise(SIGTERM);

        /* Display a message indicating we are leaving main */
        printf("Finished the signal SIGTERM\n");

        /* Display a message indicating we are registering the signal handler */
        printf("\nRegistering the signal handler\n");

        /* Register the signal handler */
        sigaction(SIGINT, &sa, NULL);

        /* Display a message indicating we are raising a signal */
        printf("Raising a SIGINT signal\n");

        /* Raise the SIGINT signal */
        raise(SIGINT);

        /* Display a message indicating we are leaving main */
        printf("Finished the signal SIGINT\n");

        /* Display a message indicating we are registering the signal handler */
        printf("\nRegistering the signal handler\n");

        /* Register the signal handler */
        sigaction(SIGTSTP, &sa, NULL);
```

Here we can observe in the terminal how the code is executed:

```
rufino@rufino:~/Documents/OS/Lab 5/Task1$ ./main 0
ERROR. Execute the program like this:

For signaction function -----> ./main 1

For signal function -----> ./main 2
```

```
rufino@rufino:~/Documents/OS/Lab 5/Task1$ ./main 1
-----WE ARE USING THE Sigaction FUNCTION-----

Registering the signal handler
Raising a SIGUSR1 signal
Received a SIGUSR1 signal
Finished the signal SIGUSR1

Registering the signal handler
Raising a SIGTERM signal
Received a SIGTERM signal
Finished the signal SIGTERM

Registering the signal handler
Raising a SIGINT signal
Received a SIGINT signal
Finished the signal SIGINT

Registering the signal handler
Raising a SIGTSTP signal
Received a SIGTSP signal
Finished the signal SIGTSP

Finished the raising of all the signals
```

```
rufino@rufino:~/Documents/OS/Lab 5/Task1$ ./main 2
-----WE ARE USING THE signal FUNCTION-----

Registering the signal handler
Raising a SIGUSR1 signal
Received a SIGUSR1 signal
Finished the signal SIGUSR1

Registering the signal handler
Raising a SIGTERM signal
Received a SIGTERM signal
Finished the signal SIGTERM

Registering the signal handler
Raising a SIGINT signal
Received a SIGINT signal
Finished the signal SIGINT

Registering the signal handler
Raising a SIGTSTP signal
Received a SIGTSP signal
Finished the signal SIGTSP

Finished the raising of all the signals
```

EXERCISE 2:

Write two programs: *Sender* and *Receiver*. Sender should send a given number, specified by one of the program arguments, of SIGUSER1 signals to Receiver and a SIGUSER2 signal at the end of transmission. Receiver should display the number of SIGUSER1 signals it has received.

Preparte three versions of this program.

Version 1.

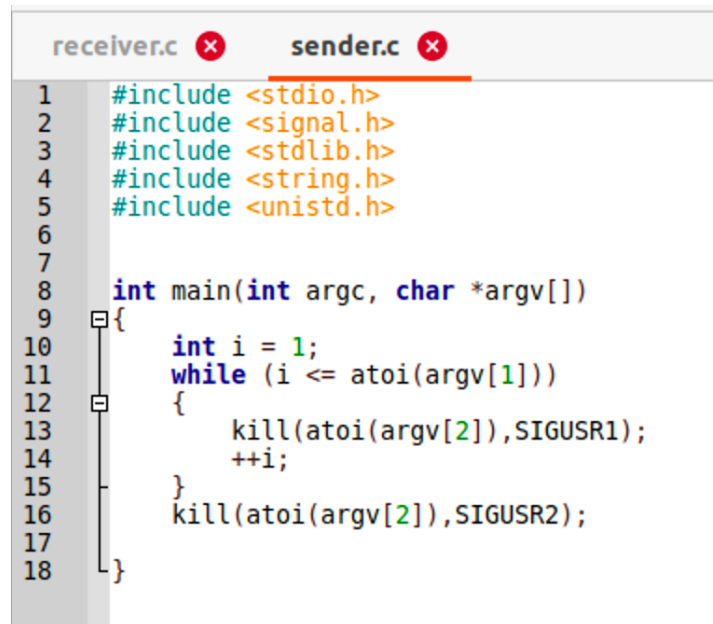
-Sender sends SIGUSR1 signals and one SIGUSR2 to Receiver.

-After receiving the SIGUSR2 signal, Receiver displays the number of received SIGUSR1 signals.

This is the code of the receiver program :

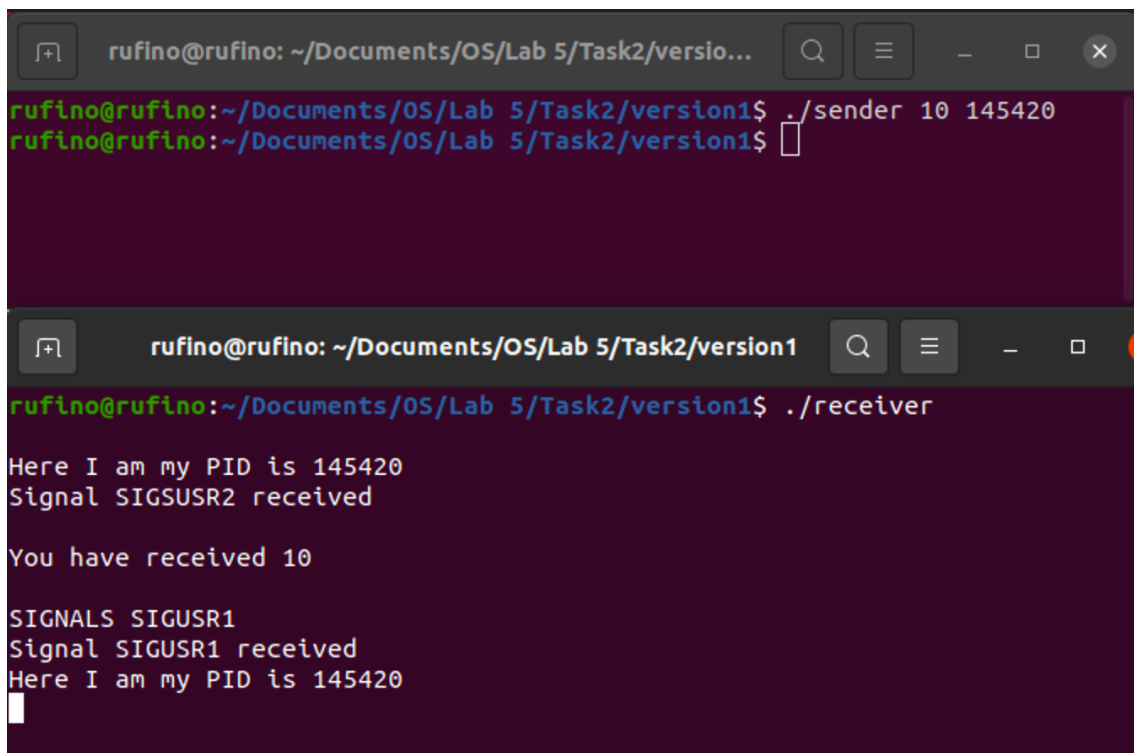
```
receiver.c x sender.c x
1  #include <stdio.h>
2  #include <signal.h>
3  #include <stdlib.h>
4  #include <string.h>
5  #include <unistd.h>
6
7  int x=0;
8  void fun handler(int signo){
9      printf("Signal Sigusr2 received\n");
10     printf("You received %d\n",x);
11     printf("SIGNALS SIGUSR1");
12 }
13
14 void sighand(int signo)
15 {
16     printf("Signal sigusr1 received");
17     x=x+1;
18 }
19
20
21 int main(int argc, char *argv[])
22 {
23     struct sigaction info, newhandler;
24     newhandler.sa handler=&sighand;
25     sigemptyset(&(newhandler.sa mask));
26     newhandler.sa flags=0;
27     sigaction(SIGUSR1,&newhandler,&info);
28
29     struct sigaction info2, newhandler2;
30     newhandler2.sa handler=&fun handler;
31     sigemptyset(&(newhandler2.sa mask));
32     newhandler2.sa flags=0;
33     sigaction(SIGUSR2,&newhandler2,&info2);
34
35
36     while(1){
37         printf("Here I am my PID is %d\n",getpid());
38         sleep(1);
39     }
40 }
41
```

This is the code of the sender program:



```
receiver.c x sender.c x
1  #include <stdio.h>
2  #include <signal.h>
3  #include <stdlib.h>
4  #include <string.h>
5  #include <unistd.h>
6
7
8  int main(int argc, char *argv[])
9  {
10     int i = 1;
11     while (i <= atoi(argv[1]))
12     {
13         kill(atoi(argv[2]), SIGUSR1);
14         ++i;
15     }
16     kill(atoi(argv[2]), SIGUSR2);
17
18 }
```

This is the code execute:



```
rufino@rufino: ~/Documents/OS/Lab 5/Task2/version1$ ./sender 10 145420
rufino@rufino:~/Documents/OS/Lab 5/Task2/version1$

rufino@rufino:~/Documents/OS/Lab 5/Task2/version1$ ./receiver

Here I am my PID is 145420
Signal SIGUSR2 received

You have received 10

SIGNALS SIGUSR1
Signal SIGUSR1 received
Here I am my PID is 145420
```

Version 2.

- Receiver confirms the receipt of each of SIGUSR1 signal by sending it back to the Sender. Sender waits for the SIGUSR1 signal before sending the next signal to the Receiver, to guarantee the correct number of the signal received by the Receiver. The Receiver should display the number of the received SIGUSR signal, such as it was done in **Version 1**.

This is the code of the sender:

```
sender.c x
1  #include <stdio.h>
2  #include <signal.h>
3  #include <stdlib.h>
4  #include <string.h>
5  #include <unistd.h>
6
7
8  void fun handler(int signo){
9
10 }
11
12
13 int main(int argc, char *argv[])
14 {
15     signal(SIGUSR1, fun handler);
16     signal(SIGUSR2, fun handler);
17     for(int i=1; i<=atoi(argv[1]); i++){
18         kill(atoi(argv[2]), SIGUSR1);
19         pause();
20     }
21     kill(atoi(argv[2]), SIGUSR2);
22     return 0;
23 }
24
25
```


This is the code of the receiver:

```
sender.c × receiver.c ×
1  #include <stdio.h>
2  #include <signal.h>
3  #include <stdlib.h>
4  #include <string.h>
5  #include <unistd.h>
6
7  int x;
8  void handler(int signum, siginfo_t *info, void *context){
9      static int counter=0;
10     pid_t pid=info->si_pid;
11     switch (signum){
12         case SIGUSR1:
13             counter++;
14             kill(pid, SIGUSR1);
15             break;
16         case SIGUSR2:
17             printf("I've received %d Signals\n", counter);
18             counter=0;
19             break;
20     }
21 }
22
23
24 int main(int argc, char *argv[])
25 {
26     struct sigaction action;
27     action.sa_flags=SA_SIGINFO;
28     action.sa_sigaction=handler;
29
30     sigaction(SIGUSR1, &action, NULL);
31     sigaction(SIGUSR2, &action, NULL);
32 }
```