Benemérita Universidad Autónoma de Puebla
Facultad de Ciencias de la Computación

SISTEMAS OPERATIVOS I

PRACTICA 8: SEÑALES

Docente:
Prof. Marcos González Flores

Alumno:	Matricula:
Jesús Huerta Aguilar	202041509
Alex Abdiel Ruano Flores	202075025

NRC: 46152
Sección: 003

QUINTO SEMESTRE

Puebla, Pue. 16/03/2022

Practica No.8

Nombre: Señales

Objetivo: Aprender a enviar, eliminar y programar señales entre los procesosprender a crear tuberías con nombre y sin nombre, así como enviar mensajes a través de una tubería.

Desarrollo:

EJERCICIO 1

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

void atrapa(int);

int main() {
      int i;
      for(i=1;i<=64; i++)
            signal(i, atrapa);
      printf("Identificativo de proceso: %d\n", getpid());
      pause();
      printf("Continuando...\n");
      /* return 0; */
}

void atrapa(int sig) {
      signal(sig, atrapa);
      printf("Recibida la señal: %d\n", sig);
}
```

TRANSCRITO + EJECUCIÓN:

```
  GNU nano 6.2                          signal.c
#include <signal.h>
#include <unistd.h>
#include <stdio.h>

void atrapa(int);

int main(){
        int i;
        for(i=1;i<=64;i++)
                signal(i, atrapa);
        printf("Identificativo de proceso: %d\n", getpid());
        pause();
        printf("COntinuando...\n");
        /* return 0; */
}

void atrapa(int sig){
        signal(sig, atrapa);
        printf("Recibida la señal: %d\n", sig);
}
```

```
alumno@alumno-VirtualBox:~$ nano signal.c
alumno@alumno-VirtualBox:~$ gcc -o signal signal.c
alumno@alumno-VirtualBox:~$ ./signal
Identificativo de proceso: 2395
^CRecibida la señal: 2
COntinuando...
alumno@alumno-VirtualBox:~$
```

```
alumno@alumno-VirtualBox:~$ nano signal.c
alumno@alumno-VirtualBox:~$ gcc -o signal signal.c
alumno@alumno-VirtualBox:~$ ./signal
Identificativo de proceso: 2395
^CRecibida la señal: 2
COntinuando...
alumno@alumno-VirtualBox:~$ nano signal.c
alumno@alumno-VirtualBox:~$ gcc -o signal signal.c
alumno@alumno-VirtualBox:~$ ./signal
Identificativo de proceso: 2498
^\
Recibida la señal: 3
Continuando...
alumno@alumno-VirtualBox:~$ ./signal
Identificativo de proceso: 2502

Recibida la señal: 28
Continuando...
alumno@alumno-VirtualBox:~$ ./signal
Identificativo de proceso: 2536
^X
^C

Recibida la señal: 2
RRecibida la señal: 28

Recibida la señal: 28
Continuando...
alumno@alumno-VirtualBox:~$ ./signal
Identificativo de proceso: 2539
^T^C
Recibida la señal: 2
Continuando...
alumno@alumno-VirtualBox:~$
```

EJERCICIO 2:

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

void atarapa(int);

int main() {
      int i;
      signal(SIGALRM, atrapa);
      printf("Identificativo de proceso: %d\n", getpid());
      alarm(5);
      pause();
      alarm (3);
      pause();
      for(i=1;i<10; i++)
      {
            alarm(1);
            pause();
      }
      return 0;
}

void atrapa(int sig) {
      signal(sig, atrapa);
      printf("RIIIIIIING!\n");
}
```

TRANSCRITO + EJECUCIÓN:

```
  GNU nano 6.2                                          sigs.c *
#include <signal.h>
#include <unistd.h>
#include <stdio.h>

void atarapa(int);

int main(){
        int i;
        signal(SIGALRM, atrapa);
        printf("Identificativo de proceso: %d\n", getpid());
        alarm(5);
        pause();
        alarm(3);
        pause();
        for(i=1;i<10;i++)
        {
                alarm(1);
                pause();
        }
        return 0;
}

void atrapa(int sig){
        signal(sig, atrapa);
        printf("RIIIIIIING!\n");
}
```

```
alumno@alumno-VirtualBox:~$ nano sigs.c
alumno@alumno-VirtualBox:~$ gcc -o sigs sigs.c
alumno@alumno-VirtualBox:~$ ./sigs
Identificativo de proceso: 2614
^C
alumno@alumno-VirtualBox:~$ ./sigs
Identificativo de proceso: 2618
RIIIIIIING!
RIIIIIIING!
RIIIIIIING!
RIIIIIIING!
RIIIIIIING!
RIIIIIIING!
RIIIIIIING!
RIIIIIIING!
RIIIIIIING!
RIIIIIIING!
RIIIIIIING!
alumno@alumno-VirtualBox:~$
```

EJERCICIO 3:

```
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>
#include <signal.h>
#include <stdlib.h>

void atrapa(int sig) {
      signal(sig, atrapa);
      printf("SIGUSR1, magnicidio\n");
}

int main() {
      pid_t padre, hijo;
      padre = getpid();
      signal(SIGUSR1, atrapa);
      if((hijo=fork())==0){
            sleep(1);
            kill(padre, SIGUSR1);
            sleep(1);
            kill(padre, SIGUSR1);
            sleep (1);
            kill(padre, SIGUSR1);
            sleep (1);
            kill(padre, SIGKILL);
            exit(0);
      }
      else
      { /* padre */
            for (;;);
      }
}
```

TRANSCRITO + EJECUCIÓN:

```
  GNU nano 6.2                                  senial.c
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>
#include <signal.h>
#include <stdlib.h>

void atrapa(int sig){
        signal(sig, atrapa);
        printf("SIGUSR1, magnicidio\n");
}

int main(){
        pid_t padre, hijo;
        padre = getpid();
        signal(SIGUSR1, atrapa);
        if((hijo=fork())==0){
                sleep(1);
                kill(padre, SIGUSR1);
                sleep(1);
                kill(padre, SIGUSR1);
                sleep(1);
                kill(padre, SIGUSR1);
                sleep(1);
                kill(padre, SIGKILL);
                exit(0);
        }
        else
        { /* padre */
                for(;;);
        }
}
```

```
alumno@alumno-VirtualBox:~$ nano senial.c
alumno@alumno-VirtualBox:~$ gcc -o senial senial.c
alumno@alumno-VirtualBox:~$ ./senial
SIGUSR1, magnicidio
SIGUSR1, magnicidio
SIGUSR1, magnicidio
Terminado (killed)
alumno@alumno-VirtualBox:~$
```

EJERCICIO 4

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

int main() {
      int pid, i=10;
      if((pid=fork())==0){
            while(i!=0)
            {
            printf("HIJO, PID = %d\n", getpid());
            i--;
            sleep(1);
            }
      }
      else{
            sleep(10);
            printf("PADRE. Terminacion del proceso %d\n", getpid());
            kill (pid, SIGTERM);
      }
      exit(0);
}
```

TRANSCRITO + EJECUCIÓN:

```
  GNU nano 6.2                                  senial2.c
#include <stdio.h>
#include <signal.h>
#include <stdlib.h>

int main(){
        int pid, i=10;
        if((pid=fork())==0){
                while(i!=0)
                {
                printf("HIJO, PID = %d\n", getpid());
                i--;
                sleep(1);
                }
        }
        else{
                sleep(10);
                printf("PADRE. Terminacion del proceso %d\n", getpid());
                kill(pid,SIGTERM);
        }
        exit(0);
}
```

```
alumno@alumno-VirtualBox:~$ nano senial2.c
alumno@alumno-VirtualBox:~$ gcc -o senial2 senial2.c
senial2.c: In function ‘main’:
senial2.c:7:17: warning: implicit declaration of function ‘fork’ [-Wimplicit-function-declar
ation]
    7 |         if((pid=fork())==0){
      |                 ^~~~
senial2.c:10:44: warning: implicit declaration of function ‘getpid’ [-Wimplicit-function-dec
laration]
   10 |                 printf("HIJO, PID = %d\n", getpid());
      |                                            ^~~~~~
senial2.c:12:17: warning: implicit declaration of function ‘sleep’ [-Wimplicit-function-decl
aration]
   12 |                 sleep(1);
      |                 ^~~~~
alumno@alumno-VirtualBox:~$ ./senial2
HIJO, PID = 2774
HIJO, PID = 2774
HIJO, PID = 2774
HIJO, PID = 2774
HIJO, PID = 2774
HIJO, PID = 2774
HIJO, PID = 2774
HIJO, PID = 2774
HIJO, PID = 2774
HIJO, PID = 2774
PADRE. Terminacion del proceso 2773
alumno@alumno-VirtualBox:~$
```

EJERCICIO 5:

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

jmp_buf entorno;
void atrapa(int);

int main() {
      int valor;
      printf("longjmp y setjmp son una forma de simular el
\'goto\'\n\n");
      signal(SIGUSR1, atrapa);
      valor=setjmp(entorno);
      if(valor==0){
            printf("Inicia el punto de interrupcion del proceso.\n");
            sleep (1);
      }
      printf("Regreso al punto de interrupcion del proceso.\n");
}

Void atrapa(int sig) {
      signal(SIGUSR1, atrapa);
      long jmp (entorno, sig);
}
```

TRANSCRITO + EJECUCIÓN:

```
  GNU nano 6.2                                  sixsig.c
#include <signal.h>
#include <stdio.h>
#include <stdlib.h>
#include <setjmp.h>

jmp_buf entorno;
void atrapa(int);

int main(){
        int valor;
        printf("longjmp y setjmp son una forma de simular el \'goto\'\n\n");
        signal(SIGUSR1, atrapa);
        valor=setjmp(entorno);
        if(valor==0){
                printf("Inicia el punto de interrupcion del proceso.\n");
                sleep(1);
        }
        printf("Regreso al punto de interrupcion del proceso.\n");
}

void atrapa(int sig){
        signal(SIGUSR1,atrapa);
        longjmp(entorno,sig);
}
```

```
PADRE: Terminacion del proceso 2903
alumno@alumno-VirtualBox:~$ nano sixsig.c
alumno@alumno-VirtualBox:~$ gcc -o sixsig sixsig.c
sixsig.c: In function ‘main’:
sixsig.c:16:17: warning: implicit declaration of function ‘sleep’ [-Wimplicit-function-decla
ration]
   16 |                 sleep(1);
      |                 ^~~~~
alumno@alumno-VirtualBox:~$ ./sixsig
longjmp y setjmp son una forma de simular el 'goto'

Inicia el punto de interrupcion del proceso.
Regreso al punto de interrupcion del proceso.
alumno@alumno-VirtualBox:~$
```