# Practica: Procesos y Señales

José Luis Quiroz Fabián

## 1 Introducción

Una señal es un evento (interrupción) que puede enviar un proceso a otro proceso. El sistema operativo se encarga de que el proceso que recibe la señal la trate inmediatamente. De hecho, termina la línea de código que está ejecutando y salta a la función que trata a la señal. Cuando termina de ejecutar el código que marca la señal (función), continua en la ejecución de la línea código donde se había quedado. El sistema operativo envía señales a los procesos en determinadas circunstancias. Si un proceso intenta acceder a una memoria no válida (por ejemplo, accediendo al contenido de un puntero a NULL), el sistema operativo envía una señal de terminación inmediata. Las señales van identificadas por un número entero. Entre la principales señales tenemos:

1. SIGINT: Señal enviada al presionar Ctrl+C en la terminal. Puede ser manejada para detener o interrumpir la ejecución del programa.
2. SIGALRM: Señal que se utiliza para programar alarmas.
3. SIGUSR1 y SIGUSR2: Señales personalizadas que pueden ser utilizadas por el programa para comunicarse con otros programas o componentes del sistema.
4. SIGKILL: Señal enviada al programa para forzar su terminación inmediata. No puede ser manejada por el programa y el sistema operativo la utilizará para finalizar el programa en caso de ser necesario.
5. SIGSTOP: Señal que se utiliza para detener temporalmente un proceso en ejecución. Cuando un proceso recibe esta señal, se detiene inmediatamente y se coloca en un estado de ”suspensión”. El proceso permanece suspendido hasta que recibe una señal SIGCONT.
6. SIGCONT: Señal que se utiliza para reanudar un proceso que ha sido detenido con la señal SIGSTOP. Cuando un proceso recibe esta señal, se reanuda su ejecución normal desde el punto en el que se detuvo.

# 2 Ejemplos

## 2.1 Cachando la señal de Ctrl-C

Compila y ejecuta el siguiente programa:

```
1  #include<stdio.h>
2  #include<signal.h>
3  #include<unistd.h>
4
5  void manejador(int signo){
6
7      printf("\n%d :)\n",getpid());
8  }
9
10 int main(){
11   if (signal(SIGINT, manejador) == SIG_ERR)
12           printf("\nNo se puede cachar la senial: SIGINT\n")
                 ;
13
14   while(1)
15     sleep(1);
16
17   return 1;
18 }
```

Trata de terminar el programa con Ctrl-C. ¿Qué ocurre?. Como puedes observar el programa no termina debido a que para el proceso le hemos cambiado el comportamiento cuando recibe la señal Ctrl-C, ahora solo muestra un mensaje. La función *signal* indica que vamos a cambiar el comportamiento de una señal. El primer parámetro indica a cual señal le vamos a cambiar el comportamiento, en particular en este ejemplo la señal SIGINT (Ctrl-C). Cuando se recibe la señal Ctrl-C el proceso ejecuta la función que se especifica en el segundo parámetro, la función *manejador*.

## 2.2 Activando una alarma

En este ejemplo un proceso activa una alarma de 5 segundo (línea 24). Después queda en espera de la alarma (línea 25). Al pasar los 5 segundo, se activa la alarma y el proceso ejecuta el código del manejador (líneas 10-13). Al terminar de ejecutar el manejador el proceso activa nuevamente la alarma (sigue en el ciclo while) y se repite el proceso.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <sys/types.h>
4  #include <unistd.h>
5  #include <signal.h>
6
7  int pid=0;
```

```
8  int i=0;
9
10 void manejador(int sig){
11     pid = getpid();
12     printf("5 segundos mas!!!\n");
13 }
14
15 int main() {
16
17
18     if (signal(SIGALRM,manejador) == SIG_ERR) {
19          printf("\nNo se puede cachar la senial: SIGALRM\n"
                );
20         exit(0);
21     }
22
23     while(1){
24         alarm(5);
25         pause();
26     }
27     return 1;
28 }
```

## 2.3 Enviando una señal desde un proceso

En este ejemplo se utiliza la función kill (línea 19) para eviarle una señal SIGUSR1 a un proceso hijo creado con fork.

```
1  #include <sys/types.h>
2  #include <signal.h>
3  #include <unistd.h>
4  #include <stdio.h>
5
6  void manejador (int senial){
7     printf("Senial del padre\n");
8  }
9
10 int main(){
11    int pid=fork();
12    if(pid == 0){
13       signal(SIGUSR1, manejador);
14          while(1)
15             pause ();
16    }else{
17       while(1){
18          sleep(1);
19          kill(pid, SIGUSR1);
20       }
21    }
22 }
```

# 3 Activando un proceso

En este ejemplo se muestra como es posible desactivar y activar un proceso. Esto llega ser muy útil en el caso de los depuradores para controlar la ejecución de un proceso.

```
1  #include<stdio.h>
2  #include<signal.h>
3  #include<unistd.h>
4
5  void manejador(int sig){
6
7      switch(sig){
8           case SIGKILL:
9              printf("Evento: SIGKILL\n");
10             break;
11          case SIGSTOP:
12             printf("Evento: SIGSTOP\n");
13          case SIGTSTP:
14             printf("Evento: SIGTSTP\n");
15             break;
16
17     }
18 }
19
20 int main(){
21
22     if (signal(SIGKILL, manejador) == SIG_ERR)
23         printf("\nNo se puede cachar la senial: SIGKILL\n")
               ;
24     if (signal(SIGSTOP, manejador) == SIG_ERR)
25          printf("\nNo se puede cachar la senial: SIGSTOP\n"
                );
26     if (signal(SIGTSTP, manejador) == SIG_ERR)
27          printf("\nNo se puede cachar la senial: SIGSTOP\n"
                );
28
29
30     while(1) {
31         printf("Proceso %d Hola!\n",getpid());
32         sleep(1);
33     }
34     return 1;
35 }
```

Para este programa, ejecute en otra terminal el comando:

kill -SIGSTOP PID ¿qué ocurre? ahora ejecute:

kill -SIGCONT PID ¿qué observa?

Ahora ejecute:

CTRL-C

¿Qué pasa?

Ejecute

kill -SIGINT PID

## 3.1 Enviando señales a si mismo

En este ejemplo se muestra el uso de raise (línea 8) para que un proceso se puden enviar una señal a si mismo.

```
1  #include<stdio.h>
2  #include<signal.h>
3  #include<unistd.h>
4
5  void manejador(int sig){
6
7        printf("Proceso %d Error!!!!!\n",getpid());
8        raise (SIGKILL);
9  }
10
11
12 int main(){
13     int contador=0;
14     int r,a=0;
15     if (signal(SIGFPE, manejador) == SIG_ERR)
16          printf("\nNo se puede cachar la senial: SIGFPE,\n"
                );
17
18
19     while(contador<5) {
20         printf("Proceso %d Hola!\n",getpid());
21         sleep(1);
22         contador++;
23     }
24     r=contador/a;
25     return 1;
26
27 }
```

# 4 Ejercicios

1. En uno de los ejemplos, investigue por qué el proceso no se detuvo si no cambiamos el comportamiento de CTRL-C (SIGINT).

2. Modificar el programas de las n-reinas para que un conjunto de procesos encuentren el total de soluciones para un tablero de NxN. Un proceso raíz debe crear N hijos y cada hijo debe iniciar en el renglón 0 pero solo tomar una columna. El valor de la columna el padre se lo pasa por una señal.