09/04/2018

Sistemas Operativos

Práctica 2





**Pareja 9**

Lucía Fuentes

Mihai Blidaru

# Ejercicio 2:

Cada uno de los hijos creados solo imprime el mensaje “Soy el proceso hijo <pid>”:

|  |
| --- |
| root@soper-5929753:/home/ubuntu/workspace/practica2# ./ejercicio2  Soy el proceso hijo 3774  Soy el proceso hijo 3775  Soy el proceso hijo 3776  Soy el proceso hijo 3777  root@soper-5929753:/home/ubuntu/workspace/practica2# |

Esto ocurre porque la acción por defecto que se toma cuando se recibe la señal SIGTERM es de terminar el proceso y por tanto nunca llega a imprimir el segundo mensaje.

# Ejercicio 3:

**a. ¿La llamada a signal supone que se ejecute la función captura?**

La llamada a signal no supone la ejecución de captura, solamente cambia el manejador de la señal, de forma que si un proceso recibe la señal SIGINT, ejecuta la acción que se le indica. En este caso, cuando el proceso recibe SIGINT es cuando se ejecuta la función captura.

**b. ¿Cuándo aparece el printf en pantalla?**

Cuando el proceso recibe un SIGINT, una señal que se puede lanzar por teclado (C^), en vez de interrumpir la ejecución del programa se ejecuta la función de captura y aparece el printf.

**c. ¿Qué ocurre por defecto cuando un programa recibe una señal y no la tiene capturada?**

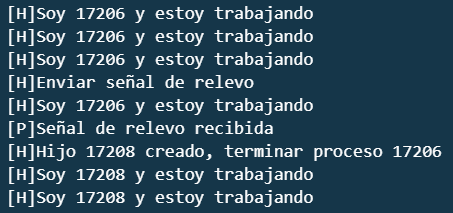
Cuando un programa recibe una señal que no tiene capturada se ejecuta el manejador por defecto de la señal, se ejecuta la acción por defecto que esa señal tiene asignada.

**d. El código de un programa captura la señal SIGKILL y la función manejadora escribe “He conseguido capturar SIGKILL”. ¿Por qué nunca sale por pantalla “He conseguido capturar SIGKILL”?**

SIGKILL es una señal de terminación abrupta, supone la terminación instantánea del programa ya que la llamada a su manejador es asíncrona, y es la única señal que no se puede ignorar.

# Ejercicio 4:

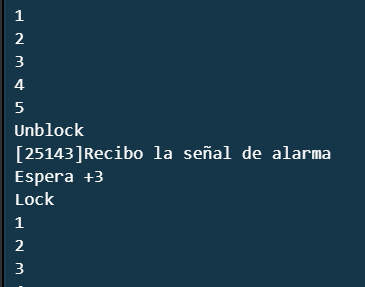
Este ejercicio es un ejemplo del intercambio de señales entre un proceso padre y sus procesos hijos. Es fácil acceder de un proceso hijo a un proceso padre con las llamadas a la función getppid(). En este programa el proceso hijo es el que envía una señal al padre para que el padre termine su ejecución, cuando el padre la recibe, le devuelve un SIGKILL.



# Ejercicio 6a:

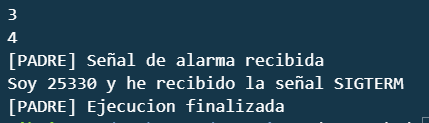
**¿Qué sucede cuando el hijo recibe la señal de alarma?**

Solo se va a ejecutar la función de alarma (imprimir “Recibo la señal de alarma”) cuando coincida que la señal SIGALRM esté desbloqueada. Supongamos que el hijo programa una alarma en el segundo 0 para el segundo 40. La señal se bloquea en el segundo 0 hasta el segundo 5, y luego se desbloquea del segundo 5 al 8, si hacemos cuentas, en el segundo 40 (cuando se recibe la señal de alarma), hijo va a estar “trabajando” y por lo tanto la señal de alarma va a estar bloqueada, lo que significa que no se va a imprimir el mensaje de “Recibo la señal de alarma” hasta que la señal se desbloquee.



# Ejercicio 6b:

Es un ejercicio muy parecido al anterior en el que el padre se pone una alarma mientras el hijo trabaja. Pasados los 40 segundos de alarma el padre envía un SIGTERM al hijo y termina. Cuando el hijo termina, el padre también lo hace. En este caso podríamos no haber cambiado el manejador de SIGTERM, ya que por defecto es una señal que termina un proceso, pero no podría imprimir antes el mensaje “He recibido la señal SIGTERM”.



# Ejercicio 7:

Comprobamos efectivamente que tras la ejecución del programa no hay ningún semáforo, todos han sido liberados.

