

Tratamiento de señales recibidas por un proceso

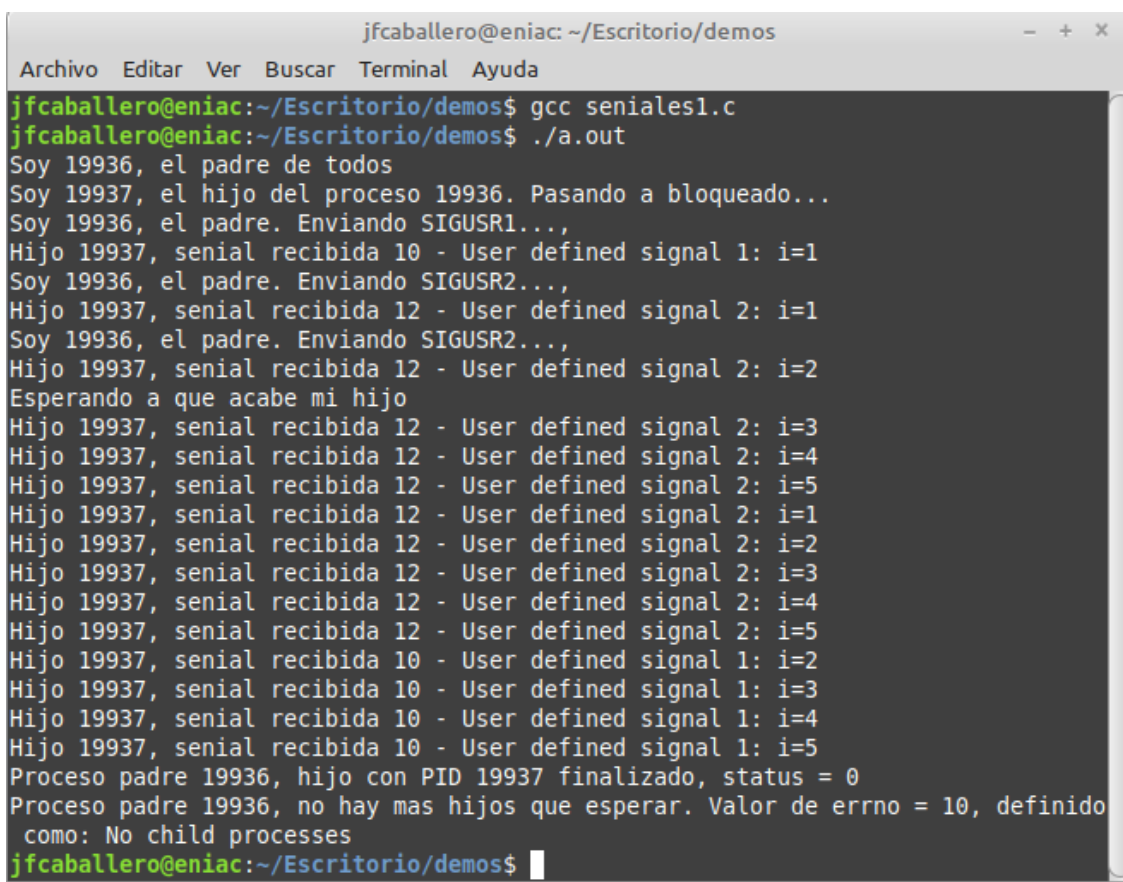
Programa seniales1.c

El programa “seniales1.c” crea un proceso hijo que realiza un tratamiento personalizado de los tipos de señales SIGUSR1 y SIGUSR2. El tratamiento personalizado consiste en imprimir por pantalla 5 veces el tipo de señal que ha recibido, durmiendo 1 segundo entre cada impresión.

Por otro lado, el proceso padre envía la señal SIGUSR1 al hijo, duerme 1 segundo, envía posteriormente la señal SIGUSR2, duerme otro segundo y vuelve a enviar la señal SIGUSR2.

Después de esto el padre espera a que termine el proceso hijo.

¿Qué conclusiones puede sacar de la ejecución del programa?



```
jfcaballero@eniac: ~/Escritorio/demos
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
jfcaballero@eniac:~/Escritorio/demos$ gcc seniales1.c
jfcaballero@eniac:~/Escritorio/demos$ ./a.out
Soy 19936, el padre de todos
Soy 19937, el hijo del proceso 19936. Pasando a bloqueado...
Soy 19936, el padre. Enviando SIGUSR1...,
Hijo 19937, senial recibida 10 - User defined signal 1: i=1
Soy 19936, el padre. Enviando SIGUSR2...,
Hijo 19937, senial recibida 12 - User defined signal 2: i=1
Soy 19936, el padre. Enviando SIGUSR2...,
Hijo 19937, senial recibida 12 - User defined signal 2: i=2
Esperando a que acabe mi hijo
Hijo 19937, senial recibida 12 - User defined signal 2: i=3
Hijo 19937, senial recibida 12 - User defined signal 2: i=4
Hijo 19937, senial recibida 12 - User defined signal 2: i=5
Hijo 19937, senial recibida 12 - User defined signal 2: i=1
Hijo 19937, senial recibida 12 - User defined signal 2: i=2
Hijo 19937, senial recibida 12 - User defined signal 2: i=3
Hijo 19937, senial recibida 12 - User defined signal 2: i=4
Hijo 19937, senial recibida 12 - User defined signal 2: i=5
Hijo 19937, senial recibida 10 - User defined signal 1: i=2
Hijo 19937, senial recibida 10 - User defined signal 1: i=3
Hijo 19937, senial recibida 10 - User defined signal 1: i=4
Hijo 19937, senial recibida 10 - User defined signal 1: i=5
Proceso padre 19936, hijo con PID 19937 finalizado, status = 0
Proceso padre 19936, no hay mas hijos que esperar. Valor de errno = 10, definido
como: No child processes
jfcaballero@eniac:~/Escritorio/demos$
```

Respuesta: Parece que las señales enviadas se van encolando y que establecen como un orden de prioridad ¿Será así?

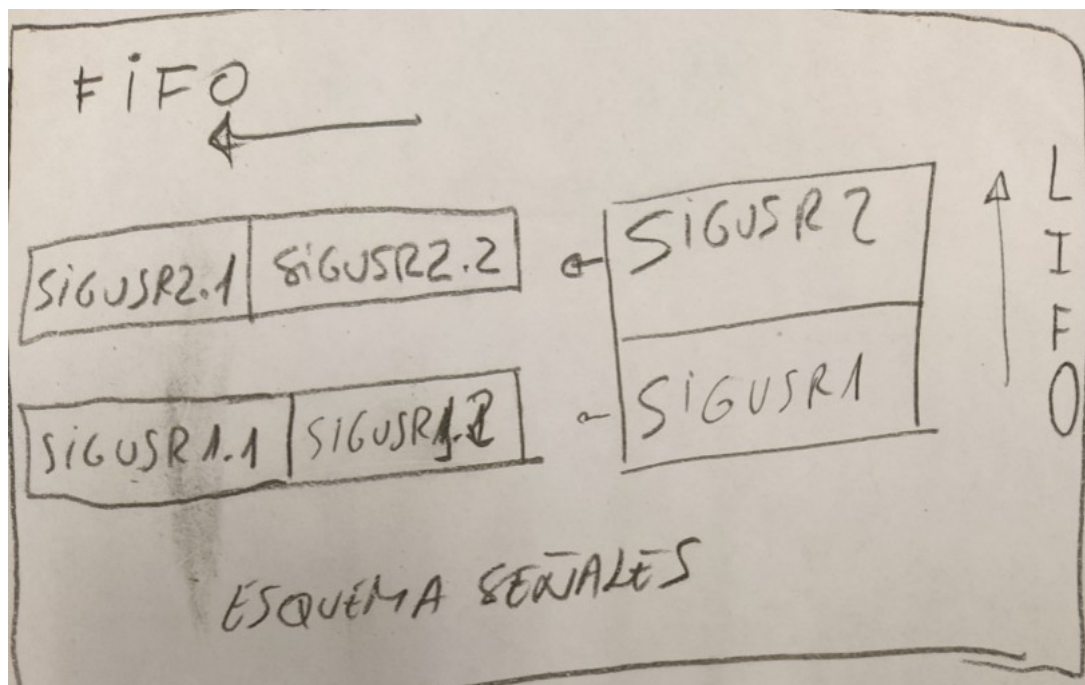
El proceso empieza a ejecutar el *callback* asociado a SIGUSR1, pero de repente llega una señal SIGUSR2, por lo que pasa a ejecutar el *callback* de esta señal, y mientras lo esta ejecutando se encola otra SIGUSR2. Cuando se termina con el primer SIGUSR2 se atiende al segundo y finalmente se atiende a lo que queda de SIGUSR1.

Por tanto, la impresión que da está salida es que el sistema trata la señales mediante una lógica de tipo **LIFO** (último en entrar primero en salir), y en esa cola se almacena el tipo de señal enviada, es decir, dentro de esa cola LIFO se irán tratando en primer lugar los últimos **tipos de señales** que se hayan enviado al proceso. Una vez se esté tratando el tipo de señal, se irán tratando ahora mediante una lógica de tipo **FIFO** (primero en entrar primero en salir) todas las señales pertenecientes **al mismo tipo de señal** enviada.

En nuestro ejemplo se lanza SIGUSR1 -> SIGUSR2.1 -> SIGUSR2.2.

Teniendo en cuenta lo anterior se tendrían que tratar en primer lugar señales de tipo SIGUSR2, una vez se está tratando ese tipo de señal se trata de forma secuencial las señales pertenecientes al dicho tipo, por lo que quedaría lo siguiente en relación al esquema de colas:

SIGUSR2.1 -> SIGUSR2.2 -> SIGUSR1.



Programa seniales2.c

El programa “seniales2.c” es una modificación del programa “seniales1.c”. El proceso padre envía las señales SIGUSR1, SIGUSR2, SIGUSR2 Y SIGUSR1 respectivamente, con intervalos de 1 segundo entre ellas.

Ejecutando dicho programa, **¿podemos decir que da igual el orden en que se invoque a la sentencia `signal()` de asignación de `callback`?**

```
jfcaballero@eniac: ~/Escritorio/demos
Archivo Editar Ver Buscar Terminal Ayuda
jfcaballero@eniac:~/Escritorio/demos$ gcc seniales2.c
jfcaballero@eniac:~/Escritorio/demos$ ./a.out
Soy 21556, el padre de todos
Soy 21557, el hijo del proceso 21556. Pasando a bloqueado...
Soy 21556, el padre. Enviando SIGUSR1...,
Hijo 21557, senial recibida 10 - User defined signal 1: i=1
Soy 21556, el padre. Enviando SIGUSR2...,
Hijo 21557, senial recibida 12 - User defined signal 2: i=1
Soy 21556, el padre. Enviando SIGUSR2...,
Hijo 21557, senial recibida 12 - User defined signal 2: i=2
Soy 21556, el padre. Enviando SIGUSR1...,
Esperando a que acabe mi hijo
Hijo 21557, senial recibida 12 - User defined signal 2: i=3
Hijo 21557, senial recibida 12 - User defined signal 2: i=4
Hijo 21557, senial recibida 12 - User defined signal 2: i=5
Hijo 21557, senial recibida 12 - User defined signal 2: i=1
Hijo 21557, senial recibida 12 - User defined signal 2: i=2
Hijo 21557, senial recibida 12 - User defined signal 2: i=3
Hijo 21557, senial recibida 12 - User defined signal 2: i=4
Hijo 21557, senial recibida 12 - User defined signal 2: i=5
Hijo 21557, senial recibida 10 - User defined signal 1: i=2
Hijo 21557, senial recibida 10 - User defined signal 1: i=3
Hijo 21557, senial recibida 10 - User defined signal 1: i=4
Hijo 21557, senial recibida 10 - User defined signal 1: i=5
Hijo 21557, senial recibida 10 - User defined signal 1: i=1
Hijo 21557, senial recibida 10 - User defined signal 1: i=2
Hijo 21557, senial recibida 10 - User defined signal 1: i=3
Hijo 21557, senial recibida 10 - User defined signal 1: i=4
Hijo 21557, senial recibida 10 - User defined signal 1: i=5
Proceso padre 21556, hijo con PID 21557 finalizado, status = 0
Proceso padre 21556, no hay mas hijos que esperar. Valor de errno = 10, definido
como: No child processes
jfcaballero@eniac:~/Escritorio/demos$
```

Respuesta: Da igual el orden en el que se invoque la sentencia *signal()*, dado que el proceso lo que almacena son los tipos de señales que ha recibido, y se va haciendo, o el tratamiento estándar, o el definido por el programador con la función *callback*.

El proceso tratará el tipo de la última señal que haya recibido en LIFO (evidentemente, antes de que se hayan encolado todos los tipos de señales se ejecutarán las que se van recibiendo) . Una vez esté tratando el tipo de señal particular recibida tratará todas aquellas señales pertenecientes a dicho tipo que se vayan encolando. Por tanto, el esquema sería una cola de tipo **LIFO para los diferentes tipos de señales** que va recibiendo el proceso, y otra estructura de tipo **FIFO donde se van almacenando todas las señales del mismo tipo**.

En nuestro ejemplo se lanza SIGUSR1.1 -> SIGUSR2.1 -> SIGUSR2.2 -> SIGUSR1.2

Teniendo en cuenta lo anterior se tendrían que tratar en primer lugar señales de tipo SIGUSR2, una vez se está tratando ese tipo de señal se trata de forma secuencial las señales pertenecientes al dicho tipo. Después se tratarán secuencialmente las señales SIGUSR1, por lo que quedaría lo siguiente en relación al esquema de colas:

SIGUSR2.1 -> SIGUSR2.2 -> SIGUSR1.1->SIGUSR1.2