

Sistemas Operativos. Práctica 2.

Pablo Cuesta Sierra y Álvaro Zamanillo Sáez

15 de marzo de 2021

Semana 1

Ejercicio 1

- a) La lista de señales se obtiene con el comando:

```
$ kill -l
```

b) SIGKILL corresponde al número 9, y SIGSTOP, al número 19. En este caso lo podemos ver directamente usando: `kill -l <Nombre_de_la_señal>`

Ejercicio 2

- a) Basta con añadir el siguiente código en el hueco señalado:

```
if(kill(pid, sig)!=0){  
    perror("kill");  
    exit(EXIT_FAILURE);  
}
```

b) Si mandamos la señal SIGSTOP, al intentar escribir en la terminal, el texto que se introduce por el teclado, no aparece. Cuando mandamos la señal SIGCONT, el texto que habíamos intentado insertar antes aparece de golpe y la terminal vuelve a la normalidad. De hecho, si escribimos algún comando y presionamos enter después de haberle mandado la señal SIGSTOP, en cuanto se mande la señal SIGCONT, se ejecuta el mismo de inmediato.

Ejercicio 3

a) La llamada a `sigaction` no supone que se ejecute la función `manejador`, sino que establece que esa es la función que este proceso ejecutará para manejar la interrupción causada por la señal indicada en el primer argumento de `sigaction`.

b) En este caso, la única señal que se bloquea durante la ejecución de `manejador` es la señal que ha provocado su ejecución: SIGINT. Esto se debe a que `act.sa_mask` no contiene ninguna señal.

c) Lo primero que aparece en pantalla es el `printf` del bucle. Luego el proceso se bloquea en la orden `sleep(9999)`, y cuando se manda la señal SIGINT con el teclado, aparece el `printf` de la función `manejador`. Como el manejador no termina el proceso, éste continúa por la orden siguiente a la que ejecutó por última vez, es decir, vuelve a empezar el bucle, imprimiendo el `printf` del `while`.

d) Si modificamos el programa para que no capture SIGINT, cuando pulsemos Ctrl+C, terminará la ejecución. Esto se debe a que si un proceso recibe una señal y no la captura, se ejecutará el manejador por defecto, el cual, en general, termina el proceso.

e) Las señales SIGKILL y SIGSTOP no pueden ser capturadas, ya que sólo las puede manejar el núcleo. Si no fuera así, el sistema operativo no tendría posibilidad de parar un proceso que no responde.

Ejercicio 4

- a) La gestión de la señal se realiza cuando el programa llega al bloque `if(got_signal){...}`
b)

Ejercicio 5

a) Si se envía SIGUSR1 o SIGUSR2, no ocurre nada, ya que las señales están bloqueadas y, por ende, quedan pendientes. En cambio, si se envía la señal SIGINT, el proceso finaliza puesto que esta no está bloqueada.

b) Cuando acaba la espera, finaliza el proceso. No se imprime el mensaje. Si mandamos una de estas señales que están bloqueadas, en cuanto finaliza la espera y se desbloquean las señales, que estaban pendientes, se lleva a cabo la rutina que la maneja, que termina el programa sin que se imprima el mensaje de despedida.

Ejercicio 6

a) Si se le manda al proceso la señal SIGALRM antes de que terminen los 10 segundos, ocurre lo mismo que si se acabara de terminar el tiempo, se ejecuta el manejador establecido por `sigaction` (que finaliza el proceso tras imprimir un mensaje).

b) Si no se ejecuta `sigaction`, al recibir la señal SIGALRM, se atiende con la rutina por defecto, que imprime el mensaje 'Temporizador' y finaliza el proceso.

Ejercicio 7**Ejercicio 8****Ejercicio 9****Ejercicio 10**