Práctica 2: Señales y semáforos.

Autores: Marcos Aarón Bernuy, Kevin de la Coba Malam. Pareja 04 del grupo 2922.

**Ejercicio 1: Comando kill de Linux.**

1. *Buscar en el manual la forma de acceder a la lista de se˜nales usando el comando kill. Copiar en la memoria el comando utilizado.*

Kill -l

1. *¿Qué número tiene la señal SIGKILL? ¿Y la señal SIGSTOP?*

SIGKILL tiene un valor igual a 9.

SIGSTOP tiene un valor igual a 19.

**Ejercicio 2: Envío de Señales.**

1. *Completar el programa en C anterior para reproducir de forma limitada la funcionalidad del comando de shell kill con un formato similar:*

*$ ./ sig\_kill -< signal >*

*El programa debe recibir dos parámetros: el primero, representa el identificador numérico de la señal a enviar; el segundo, el PID del proceso al que se enviara la señal.*

kill(pid, sig);

1. *Probar el programa enviando la señal SIGSTOP de una terminal a otra (cuyo PID se puede averiguar fácilmente con el comando ps). ¿Qué sucede si se intenta escribir en la terminal a la que se ha enviado la señal? ¿Y después de enviarle la señal SIGCONT?*

La terminal no reacciona hasta que se envía SIGCONT, al enviar SIGCONT aparecen los caracteres escritos.

**Ejercicio 3: Captura de Señales.**

1. *¿La llamada a sigaction supone que se ejecute la función manejador?*

No lo supone ya que el programa puede recibir la señal o no.

1. *¿Se bloquea alguna señal durante la ejecución de la función manejador?*

Sí, se bloquea la señal que se captura, a parte de esta ninguna.

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

Cuando el proceso recibe la señal SIGINT.

1. *Modificar el programa anterior para que no capture SIGINT. ¿Qué sucede cuando se pulsa Ctrl + C ? En general, ¿qué ocurre por defecto cuando un programa recibe una señal y no tiene instalado un manejador?*

El programa se cierra. Por defecto el proceso detiene su ejecución y bifurca a la rutina de tratamiento de la señal (SIG\_DFL), este tratamiento consiste en terminar el proceso y en algunos casos, generar un fichero core.

1. *A partir del código anterior, escribir un programa que capture todas las señales (desde la 1 hasta la 31) usando el mismo manejador. ¿Se pueden capturar todas las señales? ¿Por qué?*

No se pueden capturar todas, SIGKILL y SIGSTOP son especiales ya que son las encargadas de detener y matar procesos. El sistema operativo puede necesitar matar un proceso. Es una cuestión de seguridad.

**Ejercicio 4: Captura de SIGINT Mejorada.**

1. *En esta versión mejorada del programa del Ejercicio 3, ¿en qué líneas se realiza realmente la gestión de la señal?*

Aunque exista un manejador el print de que se ha recibido la señal se ejecuta en las líneas 28-31 (fuera del manejador).

1. *¿Por qué, en este caso, se permite el uso de variables globales?*

Porque si no, el proceso desde el manejador no sería capaz de decir que ha recibido la señal, la única forma que tiene de hacerlo es modificando esa variable para que cuando salga pueda hacer el print.

**Ejercicio 5: Bloqueo de Señales.**

1. ¿Qué sucede cuando el programa anterior recibe SIGUSR1 o SIGUSR2? ¿Y cuando recibe SIGINT?

El programa no reacciona, no hace nada cuando recibe cualquiera de las señales SIGUSR1 o SIGUSR2.

Al recibir SIGINT el programa se cierra.

1. *Modificar el programa anterior para que, en lugar de hacer una llamada a pause, haga una llamada a sleep para suspenderse durante 10 segundos, tras la que debe restaurar la máscara original. Ejecutar el programa, y durante los 10 segundos de espera, enviarle SIGUSR1. ¿Qué sucede cuando finaliza la espera? ¿Se imprime el mensaje de despedida? ¿Por qué?*

Cuando finaliza la espera el programa imprime "Fin del programa". Esto sucede porque al hacer sleep el proceso está dormido y las señales se quedan en espera, a parte se quedan en espera porque tenemos la máscara de señales. Al despertar las señales siguen bloqueadas por la máscara, al poner la máscara antigua, las señales llegan al proceso y vemos en pantalla el mensaje "Señal definida por el usuario 1".