SISTEMAS OPERATIVOS I

Práctica 3: Uso de señales

Objetivo

Utilización de llamadas al sistema para el envío y procesamiento de señales entre procesos.

Información útil para la práctica

UNIX define un conjunto de señales que pueden enviarse desde un proceso hacia otro el cual, al recibirla, reaccionará de algún modo programado, es decir, las "capturará". Existe un conjunto de señales ya predeterminadas en UNIX; el fichero *signal.h* las enumera, y se pueden ver en el volumen 7 del manual de *signal* (con *man 7 signal*). Por ejemplo, la señal 9 (denominada SIGKILL) se utiliza para finalizar un proceso o la señal 2 (SIGINT) es la asociada a la interrupción desde teclado CtrlC. Otro grupo de señales, SIGUSR1 y SIGUSR2 están pensadas para que las programe el usuario.

POSIX ofrece una función para enviar señales a un proceso denominada *kill* (ver el volumen 2 del manual). Para cada señal existe un modo de actuar por defecto, pero para algunas se puede cambiar usando las llamadas al sistema *signal* o *sigaction*. La llamada al sistema *sigaction* es más versátil y permite más funcionalidades que *signal*.

Enunciado

1. Busca información sobre las funciones comentadas.

```
static void gestion(int numero_de_senhal) { /* Funcion de gestion de señales*/
 switch (numero_de_senhal) {
 case SIGUSR1:                              /*Entra señal SIGUSR1*/
     printf("Señal tipo 1 recibida. Soy %d\n",getpid()); break;
/***Completa para el resto de las señales usadas***/
 return; }
```

4. Programa un proceso con un lazo infinito y en el que se sustituya la acción asociada a CtrlC por la impresión de una frase. Después de recibir 3 veces CtrlC, en las siguientes debe preguntar al usuario si se debe volver a su funcionamiento habitual y hacerlo si la respuesta es afirmativa. Para ello utiliza la opción SIG_DFL. Todas estas funcionalidades deben formar parte de la función que captura la señal. ENTREGABLE 1.

5. Escribe un programa que cree n procesos hijos, denominados H[1], H[2], ..., H[n]. El valor de n debe ser pedido al usuario. El proceso padre P simplemente debe esperar a la finalización del primer hijo H[1] con *waitpid*. Cada hijo H[i] debe finalizar solamente al anterior (H[i-1]) con *kill* cuando el usuario le envíe una señal SIGUSR2 desde la consola. Cuando un proceso muera debe anunciarlo escribiendo una frase en consola. Este comportamiento no es aplicable al primer hijo. Envía esa señal a algún proceso y comprueba su funcionamiento. ENTREGABLE 2.

6. Haz un programa que cree un proceso hijo, espere 4 segundos y le envíe la señal SIGUSR1. El hijo al ser creado ejecuta la llamada al sistema *pause* que lo bloqueará hasta que reciba una señal. Comprueba que el hijo se desbloquea (sale del *pause*) al recibir la señal.

7. Haz un programa que capture las señales SIGUSR1 y SIGUSR2 e ignore la señal SIGINT con las siguientes características. ENTREGABLE 3.

   - Programa un proceso P para que capture las señales SIGUSR1 y SIGUSR2 e ignore la señal SIGINT (señales que posteriormente recibirá de dos procesos hijo) con la llamada al sistema *sigaction*; de tal modo que el manejador de las señales SIGUSR1 y SIGUSR2 simplemente debe mostrar en pantalla un mensaje indicando qué señal se ha recibido. A continuación P crea dos procesos hijo, H1 y H2, y espera por la terminación del primer hijo H1, mostrando en pantalla el valor del código enviado a través del *exit* de ese hijo.
   - El primer proceso hijo H1 envía la señal SIGUSR1 al padre, espera 20 segundos, y a continuación envía la señal SIGUSR2 al padre, espera otros 20 segundos y termina.
   - Además de lo indicado anteriormente, el proceso padre P debe bloquear la señal SIGUSR1 durante un tiempo antes de capturarla. Para poder ver el efecto del bloqueo, P debe bloquear la señal SIGUSR1 antes de que el hijo H1 se la envíe y desbloquearla después de que el hijo haya enviado la señal SIGUSR2. Para ello, una vez que se ha bloqueado la señal SIGUSR1, se debe dormir el proceso padre durante un tiempo suficiente para que el hijo envíe la señal SIGUSR2 antes de que se despierte y desbloquear la señal SIGUSR1 después de despertarse. Antes de desbloquear la señal, el padre debe comprobar que SIGUSR1 está pendiente (bloqueada). Como resultado, el padre debería procesar la señal SIGUSR2 antes que la SIGUSR1 a pesar de haberla recibido más tarde. Utiliza la llamada al sistema *sigprocmask* para bloquear la señal y las llamadas *sigpending* y *sigismember* para comprobar si están bloqueadas.
   - El segundo proceso hijo H2 simplemente, tras esperar 5 segundos, envía la señal SIGINT al proceso padre y termina. Como el padre ignora esta señal, no tendrá como efecto la terminación del proceso padre.
   - Todos los procesos involucrados deben imprimir en consola comentarios exhaustivos sobre todas y cada una de las acciones que realizan, identificándose y mostrando la hora para comprobar de una manera clara que la secuencia de acciones es la esperada.

Nota 1: En la estructura tipo sigaction es conveniente activar el flag SA_RESTART, que fuerza el reiniciado de ciertas llamadas del sistema cuando sean interrumpidas por un gestor de señal; en caso contrario la llamada al sistema se interrumpiría cuando el proceso recibe la señal.

Nota 2: Se recomienda que primero hagas el programa sin bloquear la señal SIGUSR1, y cuando funcione correctamente introduzcas el bloqueo y la comprobación de que está pendiente.

8. Haz un programa que juegue al ping-pong. Para ello un proceso padre P crea dos hijos M (máquina) y J (jugador) que van a jugar. La posición de los jugadores (en su propio campo) y también la de la bola se establecen con un número entero en el intervalo [0-9]. M decide la posición inicial del jugador aleatoriamente en el intervalo [0-9]. La posición inicial de J la decide el ususario por teclado. P lleva cuenta del marcador, mostrándolo a lo largo del partido, y decide quien, M o J, saca en cada punto de manera aleatoria. El partido lo gana el que llegue a 10 puntos. Un jugador puede devolver una bola si está a una distancia de 3 o menos respecto a su posición actual, si no, no puede tocarla y pierde el punto. Tocar la bola supone decidir a que posición en el intervalo [0-9] irá en el campo del contrario, y posteriormente el jugador puede moverse hasta 2 posiciones desde suposición actual. J lo decidirá aleatoriamente y M lo pedirá al usuario. En este programa, los tres procesos deben sincronizar sus acciones exclusivamente con señales. El intercambio de información entre ellos se debe hacer a través de un único archivo. Todos los procesos deben acabar de manera programada (con *exit*). ENTREGABLE 4.

9. OPTATIVA: Haz un programa que calcule la raiz cuadrada de los números primos en orden creciente. Usa el algoritmo que quieras para determinar si un número es primo. Utiliza la función *alarm(1)* para ejecutar una función que muestre el último valor calculado en cada segundo capturando la señal SIGALARM. El programa finalizará cuando el usuario envíe la señal SIGUSR1 al proceso desde el terminal. ENTREGABLE 5.