Ejercicios Módulo 3.Señales

Ampliación de Sistemas Operativos. Curso 2017-2018

Señales

handler.c Escribe un programa que capture las señales SIGINT y SIGTSTP, para que se incremente un contador independiente ante cada recepción de dichas señales. El programa entrará en un bucle hasta que se hayan recibido un total de 10 señales, en cuyo caso mostrará por pantalla el número de señales de cada tipo recibidas, y finalizará. Desde otro terminal, envía señales usando el comando `kill`. Usa también las combinaciones de teclado adecuadas para enviar dichas señales desde el terminal en el que se ejecuta el proceso.

suspender.c Escribe un programa que se suspenda hasta la recepción de una señal de tipo SIGINT. En ese caso, el manejador imprimirá un mensaje indicando la recepción, y el programa finalizará.

killer.c Implementa un programa que, a modo de terminal, reciba por entrada estándar un identificador de proceso. Una vez recibido, el programa enviará una señal SIGKILL a dicho proceso, controlando los posibles errores que puedan surgir e informando al usuario de los mismos. Utilízalo para finalizar el proceso creado en el primer apartado (*handler*). Posteriormente, prueba a crear el proceso *handler* con el usuario *osuser* y vuelve a usar *killer* para enviar. señales (con usuario *usuarioso*). ¿Qué ocurre?

alarma.c Completa el programa de modo que se envíen señales de alarma al proceso hasta que finalice su ejecución.

minishell.c Escribe un programa que extienda la shell básica implementada en `minishell.c`, de modo que cuando un proceso termine, informe a su padre de tal circunstancia a través de la señal correspondiente.

bloqueo.c A partir del esqueleto de programa propuesto en el fichero `bloqueo.c` escribe un programa que se comporte de la siguiente forma:

1. El programa recibirá un sólo argumento, que indica los segundos a dormir.
2. Bloqueará las señales SIGINT y SIGTSTP, y ejecutará el comando sleep con tantos segundos como ha indicado el usuario.

3. Durante esta fase, las señales correspondientes permanecerán bloqueadas.
4. Al finalizar la ejecución de sleep, se consultarán las señales pendientes.
5. Si se ha recibido una señal SIGINT, el programa mostrará un mensaje por pantalla indicando esta situación.
6. Si se ha recibido una señal SIGTSTP, el programa desbloqueará dicha señal y continuará. ¿Qué pasa a continuación? ¿Cómo salir del nuevo bloqueo?

sincro(2).c A partir del esqueleto de programa propuesto en el fichero `sincro.c` escribe un programa que se comporte de la siguiente forma:

1. El programa principal (productor) creará un proceso hijo (consumidor).
2. El proceso consumidor leerá un número (float) desde un fichero de intercambio, lo multiplicará por dos, y lo almacenará de nuevo en el fichero. En ese momento, notificará al productor a través de la señal `SIGUSR1`.
3. El proceso productor leerá un número (float) desde un fichero de intercambio, lo mostrará por pantalla. A continuación, solicitará un nuevo número al usuario, y lo escribirá en el fichero, notificando al proceso consumidor de tal circunstancia a través del envío de una señal `SIGUSR1`.
4. El acceso al fichero debe ser mútuamente exclusivo.
5. En la primera implementación `sincro.c` se implementará una espera activa, consultando el valor de dos variables, y variando su valor en las rutinas de tratamiento de la señal `SIGUSR1`.
6. En la segunda implementación `sincro2.c` se implementará un mecanismo de bloqueo (suspensión) a la espera de la recepción de la señal `SIGUSR1`.

Entrega

La entrega consistirá en un solo fichero comprimido (`.tgz`) que contendrá:

- Todos los ficheros fuentes completos (NO debe incluirse ningún fichero objeto o ELF).
- El fichero *makefile*.
- Un fichero de texto o PDF con las respuestas a las preguntas planteadas en cada ejercicio, así como una copia de las fuentes de cada apartado completado.