SESSIÓN TP6

Sistemas Operativos 1
Abril 2023

1 INTRODUCCIÓN

En esta sesión de problemas se presentan ejercicios relacionados con la tercera práctica (Comunicación entre procesos). Esta sesión de problemas está centrada en utilizar tuberías para comunicar datos entre procesos, así como las señales que sirven para que procesos se envíen "interrupciones" entre sí. Sin embargo, estas interrupciones son un mecanismo gestionado a nivel de software y no de hardware.

Esta sesión tiene un ejercicio que debe entregarse. Los problemas que se plantean, así como el ejercicio a entregar están relacionados con la práctica a realizar.

2 LAS TUBERÍAS

Las tuberías son un mecanismo de comunicación entre procesos que permite enviar información de un proceso a otro. Típicamente se utiliza en la línea de comandos, para enviar, por ejemplo, la salida estándar de un proceso a la entrada estándar de otro proceso.

Las tuberías se utilizarán aquí (y, en particular, en la práctica 3) para que un proceso pueda enviar datos a otro proceso. Se realizarán varios experimentos para que se puedan entender bien los conceptos asociados.

1. Empezaremos analizando el tamaño de la tubería. Para ello utilizaremos el archivo `pipe_size.c`, que hace que el proceso hijo escriba en la tubería. Pero el padre no lee ninguna cada de la tubería. Esto hará que la tubería se llene. Observa que llega un momento en que el programa es "cuelga". ¿Por qué se cuelga? ¿Qué información nos está dando el programa sobre el tamaño de la tubería?
2. En Linux existe una forma de aumentar el tamaño de la tubería. Tiene un ejemplo en `pipe_large.c`. Las llamadas a sistema que se utilizan para aumentar no forman parte del estándar POSIX y, por tanto, no tienen por qué funcionar en otros sistemas Unix (como el Mac).

¡Pruebe el código! Puede parecer útil tener una tubería grande, pero no es eficiente, dado que una tubería grande hace que se utilicen más recursos del sistema operativo. Además, existen otros sistemas más eficientes de comunicación interproceso en caso de que se quiera transferir gran cantidad de información (p.ej. comunicación vía red, archivos, archivos mapeados a memoria). Tenga en cuenta que la tubería es la forma más "antigua" de comunicación interproceso.

3. A continuación, haremos que el buffer de la tubería se llene por el hijo. Luego el padre leerá los datos escritos por el hijo. Tiene un ejemplo en el código `pipe_write_read.c`. Analice el código, ejecute y observe qué hace. ¡Observe que el hijo acaba antes de que el padre empieza a leer! Además, verá que el código se "cuelga" al final. ¿Por qué?
4. Por último, haremos una prueba con un código que genera un "flujo continuo" de datos: el hijo escribe datos y el padre los lee. Tiene el código en el archivo `pipe_write_read_continuous.c`. Este último ejemplo es la forma en la que se comunican dos procesos cuando generamos una tubería.

Se proponen a continuación un par de ejercicios, relacionados con la práctica, que utilizan las tuberías.

5. En el archivo `pipe_size.c` hemos visto el tamaño, en bytes, del buffer asociado a la tubería. Modifique el código para escribir valores enteros. Vaya incrementando el valor que escriba en el buffer. ¿Cuántos valores enteros puede escribir?
6. Tome el archivo `pipe_write_read.c` para que el hijo escriba enteros en formato binario (p. ej. utilizando las funciones `read` y `write`). Vaya incrementando el valor que escriba en el buffer. Una vez el buffer esté lleno, el padre empezará a leer los valores. Imprima los valores leídos por pantalla para asegurar que los valores que lea son los correctos.

3 LAS SEÑALES

Las señales son un mecanismo de comunicación muy sencillo entre procesos. Son interrupciones de software que dos procesos pueden enviar entre sí para señalizar que ha ocurrido un evento. Veamos un par de ejemplos

1. El archivo `sigterm_sigint.c` contiene un ejemplo en el que la aplicación espera a recibir una señal. Observe el código e indique cuál es el comportamiento del mismo: ¿cuándo imprimirá el mensaje "*Exiting main*"?
2. El archivo `sigprocesses.c` contiene un segundo ejemplo en el que dos procesos, padre e hijo, se envían señales entre sí. Éste es un mecanismo de sincronización sencillo entre dos procesos (en la actualidad existen mecanismos de sincronización más avanzados que se verán en Sistemas Operativos 2 como, por ejemplo, los semáforos). ¿Cómo sabe el padre cuál es su hijo? ¿Cómo sabe el hijo cuál es su padre?

4 EJERCICIO A ENTREGAR

Se propone a continuación un ejercicio que forma parte de las actividades presenciales (AP) de la asignatura. Los detalles de la evaluación se indican al final.

En concreto, el código debe hacer las siguientes tareas

1. El proceso padre crea un proceso hijo. El proceso padre y el proceso hijo se comunicarán entre sí mediante una tubería.
2. El proceso padre genera N valores enteros aleatorios y los escribe en la tubería a medida que los va generando. Para ello, escribe primero el valor de N y después los valores enteros aleatorios (puede utilizar la función C "rand" para generar números aleatorios).
3. El proceso hijo, mientras, se espera a recibir una señal del padre.
4. Una vez el proceso padre ha escrito los N valores en la tubería envía la señal `SIGUSR2` al proceso hijo.
5. Al recibir el proceso hijo la señal, ésta lee primero el valor de N (que le indica el número de valores enteros escritos) así como los valores enteros que hay a continuación. El hijo realizará la suma de los valores enteros que hay en la tubería.
6. Mientras el proceso hijo realiza la suma el proceso padre se espera.
7. Una vez que el proceso hijo ha realizado la suma escribe el resultado en la tubería y envía la señal `SIGUSR1` al padre.
8. El padre, al recibir la señal, lee de la tubería el valor de la suma y la imprime por pantalla.
9. ¡Los dos procesos finalizan contentos y felices!

Para realizar el ejercicio se proporciona una plantilla que será de ayuda para hacerlo. En la plantilla se utiliza un valor de N=1000. Puede utilizar otro valor de N. Se recomienda utilizar un valor de N lo suficientemente grande para que se tarde un "cierto tiempo" para realizar la inserción y lectura de los datos. De la misma forma, se recomienda que sea lo suficientemente pequeño para que la aplicación no se quede colgada al insertar los datos en la tubería (recuerde que el buffer de la tubería tiene un tamaño relativamente pequeño; ver los ejercicios de la tubería).

5 PROCESO DE EVALUACIÓN

Se pide entregar mediante la tarea del campus virtual el ejercicio propuesto en la sección 4. Se realizará una única entrega por cada grupo de laboratorio. Cada pareja tiene un número asignado, que puede consultar en el campus virtual. El nombre de archivo deberá indicar sólo la pareja que entrega: así, por ejemplo, **el grupo 31** entregará un único archivo llamado **grupo31.sh**.

La fecha límite de entrega es mañana jueves hasta las 15h. A las 18h se harán disponibles todas las entregas en el campus virtual. Es muy importante que haga las entregas con el nombre de archivo correcto para evitar retrasos para hacer disponibles las entregas.

Cada grupo evaluará dos ejercicios entregados por los compañeros y que tengan un número de grupo inmediatamente inferior al que tengan asignados ell@s. El grupo 31 tendrá que evaluar pues los grupos 30 y 29. En caso de que, por ejemplo, el 30 no haya entregado, el grupo 31 evaluará el 29 y 28. El grupo 02 evaluará el grupo 01 y el que tenga el número más grande (es decir, la numeración es circular).

La evaluación se utilizará utilizando la **rúbrica** disponible en el campus. Utilice los listados de archivos proporcionados para asegurar el buen funcionamiento del ejercicio entregado. Evaluar cada ejercicio no debería llevar más de 15 minutos. Cada grupo entregará sus dos rúbricas vía campus virtual. Para facilitar la revisión de las rúbricas se pide que cada rubrica tenga el nombre **rubrica_grupoXX_grupoYY.ods** donde grupoXX es el grupo evaluado y grupoYY es el grupo que realiza la evaluación.

La fecha plazo para entregar las dos rúbricas asignadas es el viernes (23:59). La evaluación asignada por usted nos ayudará a decidir qué soluciones se muestran en la próxima sesión de problemas.

La evaluación de este ejercicio tendrá valores entre 0 y 10. El ejercicio debe intentar resolver el problema propuesto utilizando las instrucciones presentadas en el primer y segundo tutorial. A la hora de evaluar no se penalizará entregar un ejercicio que no funcione. En cambio, en caso de que el código de un grupo no funcione y la rúbrica correspondiente de quien lo ha evaluado no lo refleje, este hecho quedará reflejado en la nota. Por último, **en caso de que no se entregue el ejercicio o la rúbrica, o la solución entregada utilice instrucciones no permitidas o la solución no tenga nada que ver con el ejercicio propuesto la evaluación será de cero**.