Trabajo Práctico N°3
Gestión de Señales y Tuberías

Titular: Ing. Rubén L.M.Castaño
Jefe T.P.: Ing. Roberto A. Miño
Aydante de 1ra.: Lic. Claudio O. Biale

1) Desarrolle un programa que solicite al usuario 10 valores enteros. Por cada valor ingresado se debe obtener la cantidad de números enteros positivos menores al valor ingresado divisibles por 4. Al final el programa debe imprimir cada valor ingresado y la cantidad de números divisibles por 4 existentes. El programa debe aprovechar los tiempos de E/S para calcular lo solicitado, implemente el ejercicio mediante tuberías sin nombre.

2) Realice un programa que reciba un valor entero (N) como parámetro. Se debe crear un proceso hijo que recibirá los valores de la tabla de multiplicar de N por parte del proceso padre. El proceso padre debe esperar por la finalización del proceso hijo. Use tuberías para implementar la comunicación entre el proceso padre y el proceso hijo.

Consideraciones:
- La tabla de multiplicar a realizar va desde el valor cero al valor diez.

3) Implemente el siguiente comando usando tuberías sin nombre:
ps -l | wc -l

4) Implemente el siguiente comando usando tuberías sin nombre:
ls -l | grep ^d | more

5) Implemente el punto 3 usando tuberías con nombre.

6) Escriba un programa en el que dos procesos, padre e hijo, se comunican entre sí a través de dos tuberías:
- el proceso padre lee dos números enteros desde el teclado y los envía al proceso hijo a través de uno de las tuberías;
- el proceso hijo calcula la suma, diferencia, producto y cociente de los dos números y envía los resultados al proceso padre a través de la otra tubería;
- el proceso padre lee los resultados de estas operaciones aritméticas y los muestra en la pantalla

Consideraciones:
- El cociente de dos números enteros puede no ser un entero y la operación de división por cero no es válida.

7) Escriba un programa que cree un proceso hijo. Cada uno de los procesos debe imprimir alternadamente una línea en la salida estándar, el proceso padre debe imprimir en las líneas impares y el proceso hijo en las líneas pares (hasta el valor veinte):

Padre: 1
Hijo: 2
Padre: 3
Hijo: 4

Los procesos deben sincronizarse mediante el envío de la señal SIGUSR1.

8) Desarrolle un programa que imprima *"Hola Mundo, mi id es:* PID*"* cuando se pulse Ctrl-C. El programa debe terminar tras pulsar Ctrl-C 10 veces. Si no se produce ninguna pulsación de Ctrl-C en 15 segundos, el proceso debe terminar.

9) Realice dos programas denominados *"lector"* y *"escritor"*.

- El programa lector crea una tubería con nombre denominada *"nfifo.dat"* y queda esperando la llegada de datos, cuando los datos llegan son impresos en la pantalla.
- El programa escritor recibe como parámetro un programa a ejecutar y guarda la salida de este programa en la tubería con nombre denominada *"nfifo.dat"*. Una vez enviados los datos la tubería es cerrada.

10) Realice un programa que imprima su PID, PPID, PGID, SID y cree dos procesos hijos:

- El primer proceso hijo debe ejecutar la orden *ps –ef* y escribir el resultado de su ejecución en una tubería sin nombre.
- El segundo proceso hijo debe ejecutar la orden *wc -l* tomando como entrada los datos almacenados en la tubería.
- Una vez que crea todos los procesos hijos el proceso padre debe esperar por su finalización indicando el PID del proceso que va terminando. Por último debe enviarse a sí mismo una señal *SIGKILL*.

Consideraciones:

- No se permite la utilización de la llamada al sistema *system()*.

11) Realice un programa que imprima su PID, PPID, PGID y cree tres procesos hijos que colaboran para realizar las siguientes tareas:

- El primer proceso hijo ejecuta la orden *grep palabra fichero1*. Escribiendo el resultado de su ejecución en una tubería.
- El segundo proceso hijo ejecuta la orden *grep palabra fichero2*. Escribiendo el resultado de su ejecución en la misma tubería utilizada por el primer proceso hijo.
- El tercer proceso hijo utilizando la orden *wc -l*, tomará como entrada los datos almacenados en la tubería.
- Una vez que crea todos los procesos hijos, el proceso padre indicará el inicio de la ejecución de sus hijos enviándoles a cada uno una señal *SIGUSR1*.

Ejemplo de ejecución: *./programa palabra fichero1 fichero2*

Consideraciones:

- No se permite la utilización de la llamada al sistema *system()*.
- Validar la cantidad de parámetros enviados al programa, si no coinciden con lo solicitado mostrar un error y terminar.

12) Implemente un programa que cree un proceso hijo. El proceso hijo debe:

- ejecutar un bucle infinito en el que imprimen su *PID* una vez por segundo y
- manejar la señal *SIGUSR1* (El manejador imprime el *PPID* del proceso).

El proceso padre debe realizar las siguientes acciones:

- esperar por tres segundos y suspender al proceso hijo,
- esperar por tres segundos y hacer continuar al proceso hijo,
- esperar cinco segundos y enviar una señal *SIGUSR1* al proceso hijo y
- esperar diez segundos y enviar una señal *SIGKILL* al proceso hijo.