Programación de Servicios y Procesos

gandano@gmail.com

Actividad 1.2

1. Crear una aplicación en la que el proceso padre cree un hijo, y sea el padre quién le pida al hijo que termine (finalice) mediante una señal. El hijo mostará mensajes de su existencia con un período de un segundo. El padre esperará 10 segundos para pedirle su finalización, e irá mostrando el tiempo transcurrido.

2. ¿Es posible implementar la función ***raise()*** mediante el uso de ***kill()***?. En caso afirmativo, escribir un código que lo demuestre. Tomar como base el código realizado en el ejemplo anterior. En este caso el hijo contará 10 segundos y terminará (suicidio). El padre esperará la finalización del hijo e informará de ello cuando ocurra.

3. Escribir una aplicación que gestione la señal ***SIGINT*** que se genera al pulsar ***CTRL+C***. El programa se quedara esperando a esta señal un tiempo indefinido, y al recibir la señal, terminará, mostrando información adecuada.

4. Modificar el ejercicio anterior para que el proceso finalice al pulsar 6 veces la tecla ***CTRL+C***. En cada una de las pulsaciones se irá informando al usuario.

5. Establecer un tratamiento para la señal ***SIGINT*** en la que se evite que pueda llegar otra señal (del mismo tipo) mientras se está atendiendo a una señal ya ocurrida.

6. Escribir un proceso que muera (finalice) al recibir la señal ***SIGTERM*** mostrando su *pid* de proceso, y muestre un número aleatorio al recibir la señal ***SIGUSR1***. Realizar dos versiones de este ejercicio, una enviando la señal desde la consola, y la segunda, desde otro proceso.

7. Realizar un sistema de colaboración entre dos procesos utilizando señales (Para sincronizar sus acciones) y ficheros. Un proceso debe escribir en un fichero, y otro debe escribir en él, de forma que, para que el proceso lector pueda realizar su función, el proceso escritor debe haber colocado algo en el fichero. Para indicarle al receptor que se ha escrito algo, se le envía una señal, y el receptor le enviará otra señal al emisor cuando haya leido la información, indicándole así que está listo para recibir más datos. En programa debe crear un proceso padre y un proceso hijo que realizarán las funciones de escritor y lector respectivamente. Realizar una versión modificada con dos procesos independientes.

8. Escribir un programa para mostrar el funcionamiento de las funciones ***ctime()*** y ***stime()***. ¿Qué parámetros toma? ¿Qué valores retorna?

9. Utilizando las funciones de gestión del tiempo estudiadas, escribir un programa para calcular el tiempo de ejecución de una serie de operaciones matemáticas, en concreto, 1.000.000 de raices cuadradas de números positivos aleatorios de gran tamaño.

10. Desarrollar una aplicación similar al comando time de *UNIX*, llamado ***mytime***, que muestra información del tiempo empleado por un proceso en su ejecución. La forma de invocar a mytime, es *mytime comando argumentos*, donde, argumentos puede ser cualquier programa ejecutable o comando del sistema.

```
mytime ps -ef
 - real 0m 3.28s
 - user 0m 0.08s
 - sys 0m 0.54s
```

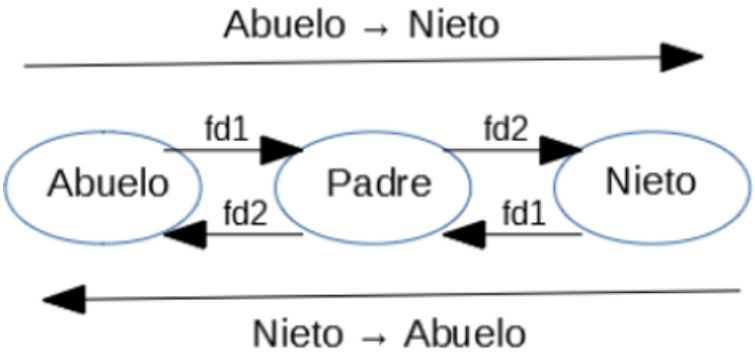
11. Desarrollar una aplicación para lanzar un comando a una hora especificada, llamado ***runat***, con la siguiente sintáxis: ***runat hh:mm:ss comando***.

```
runat 15:00:00 echo CORRE QUE SALE EL BUS
```

    hace que aparerzca en pantalla el mensaje a las 15 : 00 : 00 horas. El programa deberá crear un proceso hijo, que será el encargado de ejecutarse en segundo plano mientras que el padre devolverá el control al sistema. El hijo activará un temporizador que llegará a cero y realizará la acción indicada.

12. Utilizando pipes sin nombre, realizar la comunicación bidireccional entre dos procesos (padre/hijo), de forma que, el proceso padre enviará 10 números aleatorios entre 1 y 100, y el proceso hijo realizará la comprobación de paridad sobre estos, informando al proceso padre de esta paridad con los mensajes de texto "sipar", "nopar", que será el encargado de mostrar estos mensajes por consola de salida. La aplicación finalizará cuando el proceso padre envíe un 0 al proceso hijo. Esto se hará después del envío de los 10 números a comprobar. El número se enviará con una latencia de 1 segundo. ¿Es posible resolver este problema usando tuberías sin nombre? En caso negativo, explicar la razón/es, y proponer alternativas para su solución (todas las que se puedan aplicar para su resolución).

13. Realizar una aplicación con el objetivo de comunicar dos procesos a través de otro que actuará como intermediario. El primer proceso (abuelo) debe comunicarse en ambos sentidos con el proceso (nieto), mediante el proceso intermedio (padre). El proceso abuelo enviará una cadena de texto pedida al usuario, el proceso intermedio, que la invertirá y la pasará al proceso nieto. Este proceso nieto se encargará de imprimirla en la consola. A continuación se realizará el proceso inverso, es decir, el proceso nieto devolverá la cadena recibida al proceso padre, que volverá a invertirla y la devolverá al proceso abuelo que la imprimirá en consola de forma correcta.


Abuelo → Nieto
fd1
fd2
Abuelo
Padre
Nieto
fd2
fd1
Nieto → Abuelo


Happy
coding