13. Trabajo Práctico 13 - Pipes y Named Pipes (FIFOs)

13.1. Ejercicio 1

Realizar un programa que genere dos procesos, uno padre y uno hijo. Dichos procesos deben comunicarse a través de dos pipes (debido a que son unidireccionales). El proceso padre debe enviarle al proceso hijo la leyenda "Luke, yo soy tu padre" a lo que el proceso hijo debe responder "Noooooooooo!!!". Colocar los printf() que considere necesario para demostrar el correcto funcionamiento de la comunicación.

Ayuda

Conviene declarar un string genérico para la comunicación para saber de antemano la cantidad de bytes a recibir (por ejemplo, **char mensaje[MSJ_MAX]**).

13.2. Ejercicio 2

Realizar un programa que primeramente deberá crear un pipe y luego generar dos procesos hijo. El primer proceso hijo debe redirigir stdout al extremo de escritura del pipe y luego ejecuta con execlp() el comando "ps -e -o user,pid,%mem,command –sort -%mem". El segundo proceso hijo debe redirigir su entrada desde stdin al extremo de lectura del pipe, y luego ejecuta el comando "head". Después de crear ambos procesos, el proceso padre simplemente debe esperar a que los procesos hijo finalicen su ejecución. En definitiva, el proceso padre hace lo mismo que un shell que ejecuta el comando "ps -e -o user,pid,%mem,command –sort -%mem | head".

Ayuda

Revisar las funciones **dup2()** y **execlp()**.

13.3. Ejercicio 3

Realizar dos programas que se comunicarán a través de un named pipe (FIFO). El primer programa debe enviarle al segundo programa un comando ("/refran", "/cancion" o "/pelicula") a lo que el segundo programa debe responder con un refrán, el nombre de una canción o el nombre de una película. Luego ambos programas deben terminar. Colocar los printf() que considere necesario para demostrar el correcto funcionamiento de la comunicación. **Opcional: Hacer que lo devuelto varíe aleatoriamente dentro de una lista de posibles valores**.

13.4. Ejercicio 4

Realizar un chat simple entre dos programas. Dichos programas se comunicarán a través de un named pipe (FIFO). Tener en cuenta que la comunicación será por turnos (un mensaje de cada programa por vez), y la misma terminará cuando uno de los dos programas envíe el comando "/exit".

13.5. Ejercicio 5

Realizar un programa, llamado "servidor", el cual recibirá el nombre de un archivo de audio como argumento. Luego, enviará dicho archivo de audio a un programa "cliente" a través de un named pipe (FIFO). El programa cliente deberá recibir dicho archivo y reproducirlo. Al finalizar el envío, el programa servidor terminará su ejecución y lo mismo ocurrirá en el programa cliente al terminar la reproducción. **Nota: Se debe utilizar como programa cliente el mismo programa "play" desarrollado en el ejercicio 10.2**.