Práctica 2.4: Tuberías

Objetivos

Las tuberías ofrecen un mecanismo sencillo y efectivo para la comunicación entre procesos en un mismo sistema. En esta práctica veremos los comandos e interfaz para la gestión de tuberías, y los patrones de comunicación típicos.

Contenidos

Preparación del entorno para la práctica

Esta práctica únicamente requiere las herramientas y entorno de desarrollo de usuario.

Tuberías con nombre

Las tuberías con nombre son un mecanismo de comunicación FIFO, útil para procesos sin relación de parentesco. La gestión de las tuberías con nombre es igual a la de un archivo ordinario (`write`, `read`, `open`…). Revisar la información en `fifo(7)`.

Ejercicio 1. Usar la orden `mkfifo` para crear una tubería (ej. `$HOME/tuberia`). Usar las herramientas del sistema de ficheros (`stat`, `ls`…) para determinar sus propiedades. Comprobar su funcionamiento usando utilidades para escribir y leer de ficheros (ej. `echo`, `cat`, `less`, `tail`).

Ejercicio 2. Escribir un programa que abra la tubería con el nombre anterior (`$HOME/tuberia`) en modo sólo escritura, y escriba en ella el primer argumento del programa. En otro terminal, leer de la tubería usando un comando adecuado.

Multiplexación de canales de entrada/salida

Es habitual que un proceso lea o escriba de diferentes flujos. La función `select()` permite multiplexar las diferentes operaciones de E/S sobre múltiples flujos.

Ejercicio 1. Crear otra tubería con nombre (ej. `tuberia2`). Escribir un programa que espere hasta que haya datos listos para leer en alguna de ellas. El programa debe indicar la tubería desde la que se leyó y mostrar los datos leídos. Consideraciones:

- Para optimizar las operaciones de lectura usar un *buffer* (ej. de 256 bytes).
- Usar `read()` para leer de la tubería y gestionar adecuadamente la longitud de los caracteres leídos.
- Normalmente, la apertura de la tubería para lectura se bloqueará hasta que se abra para escritura (ej. con `echo 1 > tuberia`). Para evitarlo, usar la opción `O_NONBLOCK` en `open()`.
- Cuando un escritor termina y cierra la tubería, `select()` considerará el descriptor siempre listo para lectura (para detectar el fin de fichero) y no se bloqueará. En este caso, hay que cerrar la tubería y volver a abrirla.

Tuberías sin nombre

Las tuberías sin nombre son entidades gestionadas directamente por el núcleo del sistema y son un mecanismo eficiente para procesos relacionados (padre-hijo). La forma de comunicar los identificadores de la tubería es por herencia (en la llamada `fork()`). En este caso no hay una ruta bien conocida en el sistema de ficheros.

Ejercicio 1. Comunicación por tuberías. Escribir un programa que emule el comportamiento de la shell en la ejecución de una sentencia en la forma: `comando1 argumento1 | comando2 argumento2`. El programa creará una tubería sin nombre y creará un hijo:

- El proceso padre redireccionará la salida estándar al extremo de escritura de la tubería y ejecutará `comando1 argumento1`.
- El proceso hijo redireccionará la entrada estándar al extremo de lectura de la tubería y ejecutará `comando2 argumento2`.

Probar el funcionamiento con una sentencia similar a: `./ejercicio1 echo 12345 wc -c`

Nota: Antes de ejecutar el comando correspondiente deben cerrarse todos los descriptores no necesarios.

Ejercicio 2. Para la comunicación bi-direccional es necesario crear dos tuberías, una para cada sentido: `p_h` y `h_p`. Escribir un programa que implemente el mecanismo de sincronización de parada y espera:

- El padre leerá de la entrada estándar (terminal) y enviará el mensaje al proceso hijo escribiendo en la tubería `p_h`. Entonces permanecerá bloqueado esperando la confirmación por parte del hijo en la otra tubería, `h_p`.
- El hijo leerá de la tubería `p_h`, cuando haya leído y procesado el mensaje (escribiéndolo por la salida estándar y esperando 1 segundo) enviará el carácter `'l'` al proceso padre para indicar que está listo escribiendo en la tubería `h_p`. Después de 10 mensajes enviará `'q'` para indicar al padre que finalice.