# Ejercicios E/S Multiplexada y Pipes ASO

Usar las funciones disponibles en el API del sistema en los siguientes casos. En cada ejercicio añadir los ficheros de cabecera (#include) necesarios para que no se produzcan avisos en la compilación.

### Ejercicio 1. Comunicación con 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…). A diferencia de los ficheros regulares, las tubería debe abrirse en los dos extremos (lectura/escritura), en caso contrario se bloquean las operaciones.

**1.1.** Usar la orden **`mkfifo`** para crear una tubería (ej. ~/mituberia). 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).

**1.2.** Escribir un programa que abra la tubería con nombre anterior (~/tuberia) en modo solo escritura, y escriba en ella el primer argumento del programa. En otro terminal, leer de la tubería usando un comando adecuado.

**1.3.** Es habitual que un proceso lea o escriba de diferentes flujos. Las llamadas select(), poll() y epoll() permiten multiplexar diferentes las operaciones de E/S sobre múltiples flujos. Crear otra tubería con nombre (ej. ~/tuberia2). Escribir un programa que espere con **`select()`** hasta que haya datos listos para leer en alguna de las tuberías. El programa debe leer entonces de la tubería correspondiente y mostrar los datos leidos en la salida estándar, indicando la tubería desde la que se leyó. Usar lecturas no bloqueantes usando el flag `O_NONBLOCK.`

**1.4** Completar el program del apartado 1.3 añadiendo la posibilidad de que el programa termine cuando el usuario presione en el terminal **CTR+D**

**Nota:** El primer conjunto de descriptores de fichero que se le pasa a select es aquel sobre el que queremos esperar hasta que haya algún dato que leer. Hay una pequeña excepción. Cuando se alcanza el final de un fichero (o se corta la comunicación en una tubería o un socket), select se desbloquea si el objeto está entre los especificados en su primer conjunto de descriptores de fichero. En estos casos al hacer un read sobre uno de ellos, la llamada al sistema devolverá cero inmediatamente, señal que nos sirve para detectar el fin del fichero o el corte de la comunicación.

### Ejercicio 2 Tuberías sin nombre

Escribir un programa que emule el comportamiento de la shell en la ejecución de una sentencia en la forma: **`comando1 argumento 1 | comando2 argumento2`** . El programa abrirá una tubería sin nombre (**`int pipe(int pipefd[2])`**) y creará un hijo:

- El programa padre ejecutará "**`comando1 argumento1`**" y redireccionará la salida estándar al extremo de escritura.
- El hijo, ejecutará "**`comando2 argumento2`**", en este caso la entrada estándar deberá duplicarse con el extremo de lectura de la tubería.

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