PREGUNTA 1 . Construya un programa en **C** que realice la labor que se describe a continuación **utilizando únicamente las llamadas al sistema tratadas en los guiones de prácticas.**

Se trata de construir una versión reducida de la orden `tee`. Llamaremos al ejecutable de este programa `mitee.c`. Se ejecutará pasándole como argumento la ruta de un archivo.

El programa `mitee.c` deberá leer todos los caracteres de la entrada estándar y escribirlos en dos destinos: la salida estándar y el archivo.

Estructura obligatoria que debe tener el programa: Debe conseguirse la funcionalidad anterior utilizando dos procesos y cauces como mecanismos de comunicación entre los procesos (con nombre o sin nombre, decídalo usted).

Uno de esos procesos se encarga de que los datos pasen de la entrada estándar al cauce.

Otro proceso se encarga de que los datos pasen del cauce a la salida estándar y al archivo.

Compruebe la corrección de su programa en órdenes como las siguientes en que debe tener el mismo comportamiento que `tee:`

```
$    ls | mitee listado | wc –l

$    echo se valorara la sencillez | mitee consejo
```

PREGUNTA 2 . Nos situamos en el mismo entorno que el enunciado anterior. Ahora queremos ahorrarnos operaciones read y write, por lo que queremos sustituir éstas por la ejecución de un programa que lea de la entrada estándar y escriba en la salida estándar, por ejemplo

```
execlp("cat","cat",NULL);
```

O sea que en la medida de lo posible, queremos que los procesos que componen nuestro programa ejecuten este `execlp("cat","cat",NULL)` habiendo redirigido previamente la entrada y salida estándar. Tal vez esto involucre reestructurar qué procesos se crean y/o qué cauces utilizan. Llame a este programa `miteeExec.c` y en un folio aparte represente la estructura de procesos que ha diseñado junto con los cauces empleados.

PREGUNTA 3. Construya un programa en **C** que realice la labor que se describe a continuación **utilizando únicamente las llamadas al sistema tratadas en los guiones de prácticas.** El programa se llamará `ver.c` y tendrá como argumento la ruta de un directorio.

Queremos mostrar en pantalla los nombres y números de inodo de aquellos archivos regulares que cuelguen directamente de dicho directorio que tengan establecido actualmente algún bloqueo por algún proceso.

Programe la actuación adecuada ante todas las circunstancias de error que puedan ocurrir.