



**SISTEMAS OPERATIVOS - LABORATORIO**  
**19 de Diciembre de 2024, Turno 1**

Nombre \_\_\_\_\_ DNI \_\_\_\_\_

Apellidos \_\_\_\_\_ Grupo \_\_\_\_\_

#### INSTRUCCIONES

- Se debe entregar una carpeta individual con el código de cada apartado (1A, 1B, etc.), para así garantizar la evaluación independiente de cada uno.
- Si el código no compila o su ejecución produce un error grave la puntuación de ese apartado será 0.

**Cuestión 1. (5 puntos)** Diseñe un programa que sirva como interfaz para ejecutar el programa student-records implementado en la práctica 2 con distintas opciones. El comportamiento de este nuevo programa dependerá de los argumentos que se pasen por la línea de comandos:

- -b: Se ejecutará student-records con las opciones adecuadas para que genere un fichero binario de la base de datos y después se realizará una comparación con un fichero de referencia que se pasará a este programa con la opción -i. NOTA: contemple que el fichero de referencia puede ser diferente al generado.
- -t: Se ejecutará student-records con las opciones adecuadas para que se muestre por la salida estándar el contenido de la base de datos en formato texto. No se genera ningún fichero ni se realiza comparación.
- -a: Se realizarán concurrentemente las acciones descritas para las opciones -b y -t, realizándose la comparación con el fichero de referencia al final.
- -i <fichero\_bin\_antiguo>: fichero binario a comparar con el nuevo. Parámetro obligatorio con las opciones -b y -a.
- -o <fichero\_bin\_nuevo>: nuevo fichero binario que se generará con la opción -b o -a.
- -h: Muestra por terminal un mensaje de ayuda con las instrucciones de uso.

**NOTA:** Se recomienda que, antes de empezar, compile y compruebe que el programa student-records genera las siguientes salidas cuando se ejecuta con los parámetros del ejemplo.

```
$ ./student-records -i students-db.txt -o students-db.bin
4 student records written successfully to binary file students-db.bin
$ ./student-records -i students-db.txt -p
[Entry #0]
    student_id=27
    NIF=67659034X
    first_name=Chris
    last_name=Rock
[Entry #1]
    student_id=34
    NIF=78675903J
    first_name=Antonio
    last_name=Banderas
[Entry #2]
    student_id=3
    NIF=58943056J
    first_name=Santiago
    last_name=Segura
[Entry #3]
    student_id=4
    NIF=6345239G
    first_name=Penelope
    last_name=Cruz
```

La evaluación de esta pregunta será la siguiente:

- **(2 puntos)** Añada a la función main, que se da en la plantilla de la solución, el código de *parsing* de las opciones descritas anteriormente utilizando la función getopt.
- **(3 puntos)** Complete las funciones run\_bin\_option, run\_text\_option y run\_all\_option suministradas en la plantilla de soluciones utilizando fork, execvp y waitpid de forma que:

- run\_bin\_option debe crear un proceso que ejecutará el programa student-records con las opciones adecuadas para crear un fichero binario con el contenido de la base de datos. Cuando el proceso termine se realizará una comparación del fichero binario nuevo con el antiguo utilizando la función run\_sh\_script suministrada con la plantilla de soluciones.
- run\_text\_option: debe crear un proceso que ejecute student-records con las opciones adecuadas para mostrar el contenido de la base de datos por la salida estándar en formato texto.
- run\_all\_option debe crear dos procesos, que ejecutarán las acciones anteriores concurrentemente, excepto la comparación con el fichero de referencia, que se hará cuando estos dos procesos hayan finalizado.
- En el Campus Virtual se proporciona un archivo de base de datos que puede utilizarse para comprobar si el programa funciona correctamente. Una posible salida del programa sería:

```
$ ./question1 -i students-db.bin -o students-db_new.bin -b
4 student records written successfully to binary file students-db_new.bin
Binary files contents are identical

$ ./question1 -t
[Entry #0]
    student_id=27
    NIF=67659034X
    first_name=Chris
    last_name=Rock
[Entry #1]
    student_id=34
    NIF=78675903J
    first_name=Antonio
    last_name=Banderas
[Entry #2]
    student_id=3
    NIF=58943056J
    first_name=Santiago
    last_name=Segura
[Entry #3]
    student_id=4
    NIF=6345239G
    first_name=Penelope
    last_name=Cruz

$ ./question1 -i students-db.bin -o students-db_new.bin -a
[Entry #0]
    student_id=27
    NIF=67659034X
    first_name=Chris
    last_name=Rock
[Entry #1]
    student_id=34
    NIF=78675903J
    first_name=Antonio
    last_name=Banderas
[Entry #2]
    student_id=3
    NIF=58943056J
    first_name=Santiago
    last_name=Segura
[Entry #3]
    student_id=4
    NIF=6345239G
    first_name=Penelope
    last_name=Cruz
4 student records written successfully to binary file students-db_new.bin
Binary files contents are identical

$ ./question1 -h
Usage: ./question1 [-b] [-t] [-a] -i <old_bin_file> -o <new_bin_file>
```

**Cuestión 2. (5 puntos)** Modifique el programa de la práctica 5 (disco.c) para que simule el comportamiento de una discoteca con dos pistas de baile. A cada cliente se le asignará una pista de baile en la que bailar en el momento de la creación del hilo, además de su condición de vip o no vip. Para entrar en la discoteca se siguen las mismas condiciones que en la práctica original:

- Los clientes vip y no vip deben entrar respetando el orden de llegada respecto de otros clientes de su tipo.
- Los clientes no vip deben esperar si hay clientes vip esperando para entrar
- Los clientes no pueden entrar si el aforo de la discoteca está completo.

Una vez dentro, independientemente de si son vip o no y de su orden de llegada, deberán esperar para entrar en la pista de baile que les fué asignada si hay dos o más clientes bailando en dicha pista. Una vez que el cliente consiga acceder a la pista, bailará un tiempo aleatorio hasta que decida salir. Cuando termine de bailar saldrá de la discoteca liberando así espacio en la pista de baile y la discoteca.

La evaluación de esta pregunta será la siguiente:

- **(1,5 puntos)** Modifique el programa principal de modo que se pasen tres argumentos a los hilos: id, isvip y dancefloor (pista de baile). Como en la práctica original, el programa leerá los parámetros de un fichero clients.txt con un formato similar al de la práctica original: la primera línea indica el número de clientes a crear y las restantes (una por cliente) indican si el cliente correspondiente es vip o no y la pista de baile que le corresponde. En el Campus Virtual se proporciona un fichero de ejemplo.
- **(2 puntos)** Modifique el código del cliente para que antes de bailar se acceda a la pista de baile que se le pasa como parámetro, cumpliendo las condiciones indicadas arriba, añadiendo las variables globales y/o recursos de sincronización que sean necesarios.
- **(1,5 puntos)** Modifique la función de salida de la discoteca para que se cumplan las condiciones de sincronización indicadas arriba.

Una posible salida del programa sería:

```
$ ./disco clients.txt
client 0 is not vip and assigned to Dancefloor 1
client 1 is    vip   and assigned to Dancefloor 2
Starting client 0
Client 0 (not vip) entering the disco
Client 0 (not vip) dancing on Dancefloor 1
client 2 is not vip and assigned to Dancefloor 1
Starting client 1
Client 1 (  vip  ) entering the disco
Client 1 (  vip  ) dancing on Dancefloor 2
client 3 is not vip and assigned to Dancefloor 2
Starting client 2
Client 2 (not vip) entering the disco
Client 2 (not vip) dancing on Dancefloor 1
Starting client 3
Client 3 (not vip) entering the disco
Client 3 (not vip) dancing on Dancefloor 2
client 4 is    vip   and assigned to Dancefloor 1
Starting client 4
Client 4 (  vip  ) entering the disco
Client 4 (  vip  ) waiting for Dancefloor 1
Client 2 (not vip) exits the disco
Client 4 (  vip  ) dancing on Dancefloor 1
Client 1 (  vip  ) exits the disco
Client 0 (not vip) exits the disco
Client 3 (not vip) exits the disco
Client 4 (  vip  ) exits the disco
```