Universidad Carlos III de Madrid

Escuela Politécnica Superior (Leganés)

Grado en Ingeniería Informática (3er curso)

Asignatura: Diseño de sistemas operativos

Práctica 2:
Sistema de Ficheros

Elaborado por:

Estévez Fernández, Andrés Javier
(NIA: 358857 / Grupo 81 /
100358857@alumnos.uc3m.es)

Zhu, Zhenfeng
(NIA: 363798 / Grupo 81 /
100363798@alumnos.uc3m.es)

Leganés, Madrid. 30 de abril de 2019.

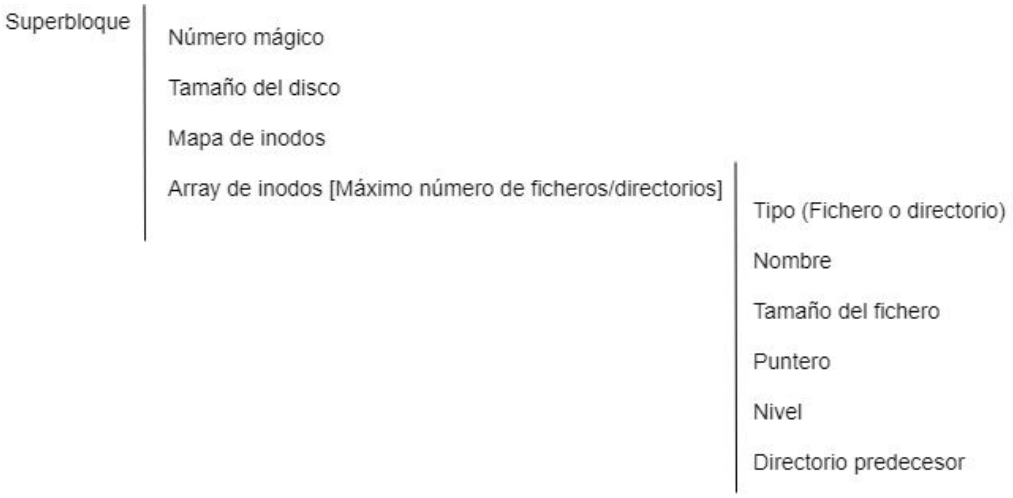
Índice de contenidos

Introducción

Este documento describe detalladamente la solución implementada para el problema propuesto en esta práctica. También incluye una serie de casos de prueba que tienen como finalidad garantizar el buen funcionamiento del programa. Por último el documento termina con una breve conclusión en la que se habla de los problemas encontrados y las opiniones personales.

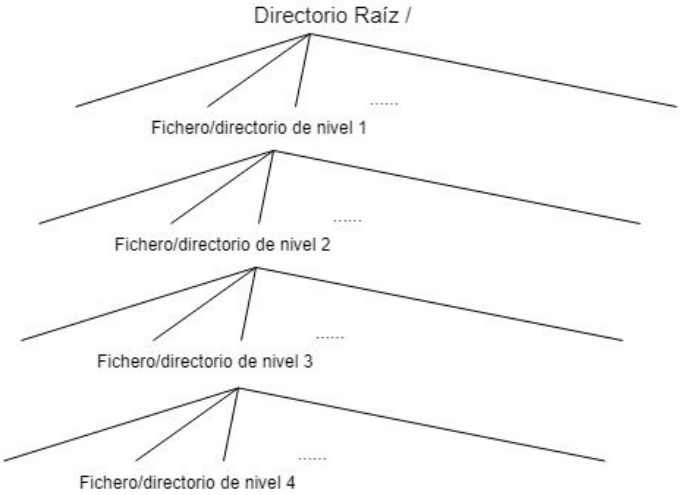
Diseño detallado del sistema de ficheros

Metadatos



Los metadatos que se guardarán en el disco tendrán la siguiente estructura:

- Número mágico: Es un número que inventamos para identificar nuestro sistema de ficheros
- Tamaño del disco: Es el tamaño del disco
- Mapa de inodos: Es un mapa de bits que sirve para asignar sus posiciones i-ésimas a los inodos
- Array de inodos: Es un array que guarda los datos de los inodos: el tipo (si es fichero o directorio), el nombre, el tamaño, el puntero, el nivel (4 niveles en total) y su directorio predecesor.

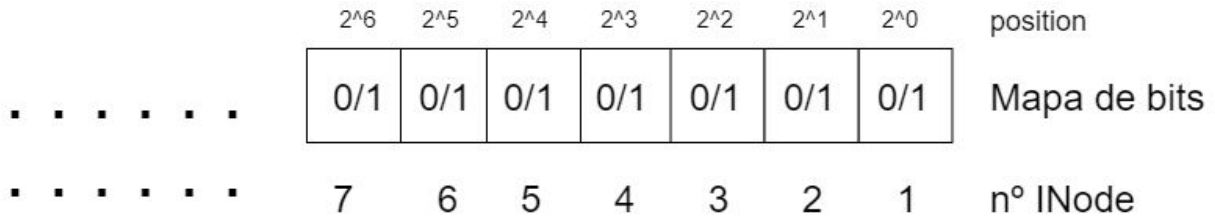


Jerarquía del sistema de ficheros

Asignación de recursos

El mapa de bits tiene inicialmente 64 posiciones, es una variable de tipo uint64_t, por tanto nos servirá de sobra para los 40 ficheros/directorios que tenemos como cantidad máxima establecida en el enunciado.

En cuanto a la asignación de inodos, tal como se puede observar en la siguiente imagen:



El mapa de bits solamente puede tener valor 0 o 1 en cada posición, inicialmente están todas las posiciones inicializadas a 0. Cada posición en el mapa de bits corresponde a un inodo. Cuando asignamos una posición de el mapa de bits a un inodo, esa posición cambia a 1, y el inodo tendrá como su id el nº INode correspondiente.

Nosotros solamente vamos a necesitar las primeras 40 posiciones de el mapa de bits. Por tanto existen en total 40 id para los inodos.

Por otro lado, para averiguar el valor de una posición de el mapa de bits, utilizamos la tabla AND.

INPUT		OUTPUT
A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1

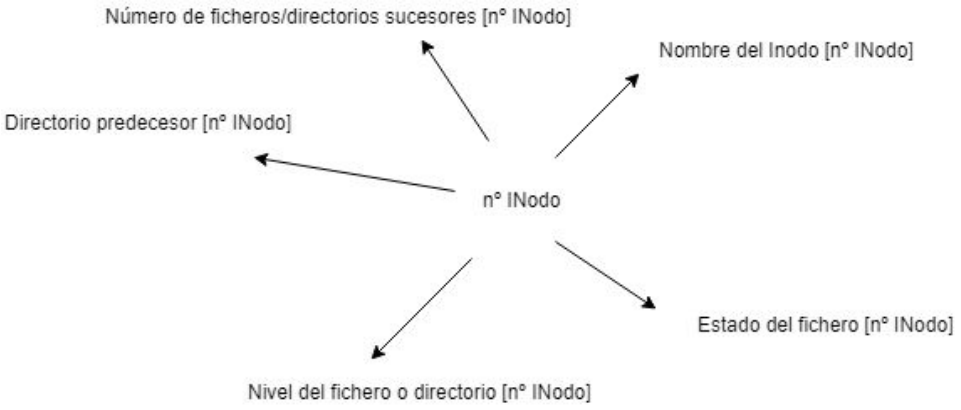
Siendo una posición del mapa de inodos que representa “A” en la tabla AND, creamos una variable de tipo uint64_t que tiene valor 1 en su última posición y que representa como “B” en la tabla. Haciéndolos AND si nos sale OUTPUT 0 entonces esa posición en el mapa de inodos tiene valor 0, mientra si sale 1 entonces tiene como valor 1.

Por último, para averiguar el id de un inodo sabiendo su nombre u otro dato, tenemos que inicializar un contador a 1 en primer lugar, luego hacemos puerta AND con la máscara que tiene valor 1 a cada posición del mapa de bits utilizando desplazamiento “>>”, e incrementamos el contador por cada interacción hasta que el nombre coincida con el nombre contenido en el inodo y así averiguamos el id del inodo que tiene ese nombre. Mientras para encontrar la primera posición libre para asignar un inodo basta con encontrar el primer OUTPUT a 0.

Descripción del código

El número del inodo, o llamado de otra manera, el id del inodo, nos sirve para utilizar también como el descriptor del fichero, ya que cada inodo corresponde únicamente a un fichero, o directorio.

Para facilitar el desarrollo del sistema de ficheros hemos utilizado el id del inodo como índice para los siguientes arrays en la implementación de las distintas funcionalidades pedidas en la práctica.



Estructuras utilizadas para la implementación

En los ficheros filesystem.c y metadata.h contienen toda explicación de implementación en los comentarios.

Plan de pruebas

Pruebas de error

Todas las pruebas de error están contenidas en el fichero test.c con el comentario “Pruebas de error x”, siendo ‘x’ el índice de la prueba de error.

1. En mkFS - error NF9: El tamaño del disco está a menor que 50 KiB
   - Entrada: 49 KiB
   - Salida: -1

```
Error NF9
TEST mkFS FAILED
```

2. En mkFS - error NF9: El tamaño del disco está a mayor que 10 MiB
   - Entrada: 11 MiB
   - Salida: -1

```
Error NF9
TEST mkFS FAILED
```

3. En mountFS y unmountFS da fallos solo cuando queremos escribir más bloques de los que hemos creado.

4. En createFile - error: El nombre del fichero no empiezan por ‘/’ o termina por ‘/’
   - Entrada: ‘/’
   - Salida: -2

```
path incorrect format
error checkFile
TEST createFile FAILED
```

5. En createFile - error NF4: La ruta exceda de 132 caracteres
   - Entrada: 133 caracteres
   - Salida: -2

```
path incorrect format
error checkFile
TEST createFile FAILED
```

6. En createFile - error: No existe la ruta
   - Entrada: "/dir1/file1"
   - Salida: -2

```
Directory does not exist
error checkFile
TEST createFile FAILED
```

7. En createFile - error: Ya existe el fichero
   - Entrada: "/error7"
   - Salida: 0
   - Entrada: "/error7"
   - Salida: -1

```
File or directory with same name already exists
error same name file
TEST createFile FAILED
```

8. En removeFile - error: El nombre del fichero no empiezan por ‘/’ o termina por ‘/’
   - Entrada: ‘/’
   - Salida: -2

```
path incorrect format
TEST removeFile FAILED
```

9. En removeFile - error NF4: La ruta exceda de 132 caracteres
   - Entrada: 133 caracteres
   - Salida: -2

```
path incorrect format
TEST removeFile FAILED
```

10. En removeFile - error: No existe la ruta
    - Entrada: "/dir1/file1"
    - Salida: -2

```
Directory does not exist
TEST removeFile FAILED
```

11. En removeFile - error: No existe el fichero
    - Entrada: "/fileNotExisted"
    - Salida: -1

```
File not exist
TEST removeFile FAILED
```

12. En openFile - error: El nombre del fichero no empiezan por ‘/’ o termina por ‘/’
    - Entrada: ‘/’
    - Salida: -2

```
path incorrect format
TEST openFile FAILED
```

13. En openFile - error NF4: La ruta exceda de 132 caracteres
    - Entrada: 133 caracteres
    - Salida: -2

```
path incorrect format
TEST openFile FAILED
```

14. En openFile - error: No existe la ruta
    - Entrada: "/dir1/file1"
    - Salida: -2

```
Directory does not exist
TEST openFile FAILED
```

15. En openFile - error: No existe el fichero
    - Entrada: "/fileNotExisted"
    - Salida: -1

```
File not exist
TEST openFile FAILED
```

16. En openFile - error: El fichero ya está abierto
    - Entrada: "/fileAlreadyOpened"
    - Salida: 0
    - Entrada: "/fileAlreadyOpened"
    - Salida: -2

```
TEST createFile SUCCESS
TEST openFile SUCCESS
File already opened
TEST openFile FAILED
```

17. En closeFile - error: El descriptor de fichero es menor que 0
    - Entrada: -1
    - Salida: -1

```
Invalid fd
TEST closeFile FAILED
```

18. En closeFile - error: El descriptor de fichero es mayor que el máximo número de ficheros permitidos
    - Entrada: 41
    - Salida: -1

```
Invalid fd
TEST closeFile FAILED
```

19. En closeFile - error: No existe el fichero o ya está cerrado
    - Entrada: 0 (Siento el descriptor 0 un fichero que no existe o cerrado)
    - Salida: -1

```
error: file not exist or already closed
TEST closeFile FAILED
```

20. En readFile - error: El descriptor de fichero es menor que 0
    - Entrada: -1, buffer, sizeof(buffer)
    - Salida: -1

```
Invalid fd
TEST readFile FAILED
```

21. En readFile - error: El descriptor de fichero es mayor que el máximo número de ficheros permitidos
    - Entrada: 41, buffer, sizeof(buffer)
    - Salida: -1

```
Invalid fd
TEST readFile FAILED
```

22. En readFile - error: El número de bytes que se desee leer es menor que 0
    - Entrada: 0, buffer, -1
    - Salida: -1

```
Invalid numBytes
TEST readFile FAILED
```

23. En readFile - error: El número de bytes que se desee leer es mayor que el tamaño del bloque
    - Entrada: 0, buffer, 2049
    - Salida: -1

```
Invalid numBytes
TEST readFile FAILED
```

24. En readFile - error: El fichero no está abierto. (Si el fichero está abierto siempre existe)
    - Entrada: 0, buffer, sizeof(buffer)
      (Siento el descriptor 0 un fichero que no existe o cerrado)
    - Salida: -1

```
TEST createFile SUCCESS
File not opened
TEST readFile FAILED
```

25. En writeFile - error: El descriptor de fichero es menor que 0
    - Entrada: -1, buffer, sizeof(buffer)
    - Salida: -1

```
Invalid fd
TEST writeFile FAILED
```

26. En writeFile - error: El descriptor de fichero es mayor que el máximo número de ficheros permitidos
    - Entrada: 41, buffer, sizeof(buffer)
    - Salida: -1

```
Invalid fd
TEST writeFile FAILED
```

27. En writeFile - error: El número de bytes que se desee escribir es menor que 0
    - Entrada: 0, buffer, -1
    - Salida: -1

```
Invalid numBytes
TEST writeFile FAILED
```

28. En writeFile - error: El número de bytes que se desee escribir es mayor que el tamaño del bloque
    - Entrada: 0, buffer, 2049
    - Salida: -1

```
Invalid numBytes
TEST writeFile FAILED
```

29. En writeFile - error: El fichero no está abierto. (Si el fichero está abierto siempre existe)
    - Entrada: 0, buffer, sizeof(buffer)
      (Siento el descriptor 0 un fichero que no existe o cerrado)
    - Salida: -1

```
TEST createFile SUCCESS
File not opened
TEST writeFile FAILED
```

30. En lseekFile - error: El descriptor de fichero es menor que 0
    - Entrada: -1, buffer, sizeof(buffer)
    - Salida: -1

```
Invalid fd
TEST lseekFile FAILED
```

31. En lseekFile - error: El descriptor de fichero es mayor que el máximo número de ficheros permitidos
    - Entrada: 41, buffer, sizeof(buffer)
    - Salida: -1

```
Invalid fd
TEST lseekFile FAILED
```

32. En lseekFile - error: El fichero no existe
    - Entrada: 0, FS_SEEK_BEGIN, 0
      (Siento el descriptor 0 un fichero que no existe o cerrado)
    - Salida: -1

```
File not exists
TEST lseekFile FAILED
```

33. En lseekFile - error: El puntero fuera de rango
    - Entrada: 0, 40, 0
    - Salida: -1

```
TEST createFile SUCCESS
Pointer out of range
TEST lseekFile FAILED
```

34. En mkDir - error: El nombre del fichero no empiezan por ‘/’ o termina por ‘/’
    - Entrada: “/”
    - Salida: -2

```
path incorrect format
error checkDir
TEST mkDir FAILED
```

35. En mkDir - error NF4: La ruta exceda de 99 caracteres
    - Entrada: 100 caracteres
    - Salida: -2

```
path incorrect format
error checkDir
TEST mkDir FAILED
```

36. En mkDir - error: No existe la ruta
    - Entrada: "/test/file"
    - Salida: -2

```
Directory does not exist
error checkDir
TEST mkDir FAILED
```

37. En mkDir - error: Ya existe el directorio
    - Entrada: "/test"
    - Salida: 0
    - Entrada: "/test"
    - Salida: -1

```
TEST mkDir SUCCESS
There already exist file or directory with the same name
There is a directory with same name
TEST mkDir FAILED
```

38. En rmDir - error: El nombre del fichero no empiezan por ‘/’ o termina por ‘/’
    - Entrada: “/”
    - Salida: -2

```
path incorrect format
error checkPath
TEST rmDir FAILED
```

39. En rmDir - error NF4: La ruta exceda de 99 caracteres
    - Entrada: 100 caracteres
    - Salida: -2

```
path incorrect format
error checkPath
TEST rmDir FAILED
```

40. En rmDir - error: No existe la ruta
    - Entrada: "/test/file"
    - Salida: -2

```
Path does not exist
error checkPath
TEST rmDir FAILED
```

41. En rmDir - error: No existe el directorio
    - Entrada: "/test"
    - Salida: -1

```
Directory no exists
TEST rmDir FAILED
```

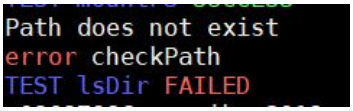
42. En lsDir - error: El nombre del fichero no empiezan por ‘/’ o termina por ‘/’
    - Entrada: “/”
    - Salida: -2

```
path incorrect format
error checkPath
TEST lsDir FAILED
```

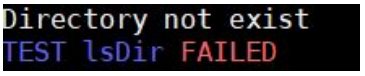
43. En lsDir - error NF4: La ruta exceda de 99 caracteres
    - Entrada: 100 caracteres
    - Salida: -2

```
path incorrect format
error checkPath
TEST lsDir FAILED
```

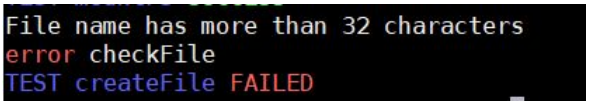
44. En lsDir - error: No existe la ruta
    - Entrada: "/test/file"
    - Salida: -2



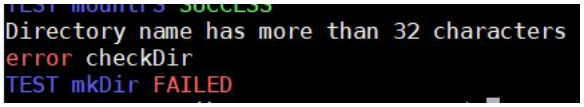
45. En lsDir - error: No existe el directorio
    - Entrada: "/test"
    - Salida: -1



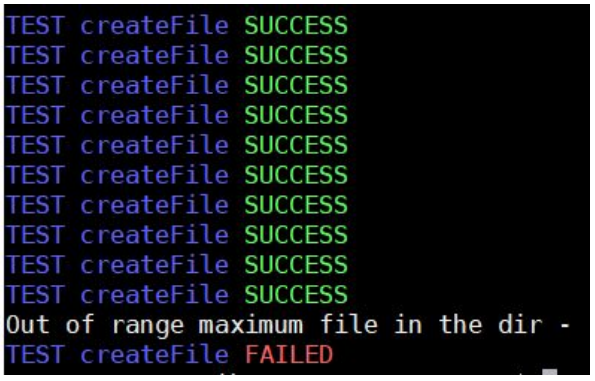
46. En createFile - error NF3: El nombre del fichero tiene más de 32 caracteres.
    - Entrada: "/test/<33 caracteres>"
    - Salida: -1



47. En mkDir - error NF3: El nombre del directorio tiene más de 32 caracteres.
    - Entrada: "/test/<33 caracteres>"
    - Salida: -1



48. En createFile - error NF2: El un directorio existen más de 10 ficheros/directorios
    - Entrada: "/test1"
    - Salida: 0
    - Entrada: "/test2"
    - Salida: 0
    - ……
    - Entrada: "/test10"
    - Salida: 0
    - Entrada: "/test11"
    - Salida: -2



49. En mkDir - error NF2: El un directorio existen más de 10 ficheros/directorios
    - Entrada: "/dir”
    - Salida: 0
    - Entrada: "/dir/1”
    - Salida: 0
    - ……
    - Entrada: “/dir/10”
    - Salida: 0
    - Entrada: "/dir/11”
    - Salida: -2

```
TEST mkDir SUCCESS
TEST mkDir SUCCESS
TEST mkDir SUCCESS
TEST mkDir SUCCESS
TEST mkDir SUCCESS
TEST mkDir SUCCESS
TEST mkDir SUCCESS
TEST mkDir SUCCESS
TEST mkDir SUCCESS
TEST mkDir SUCCESS
Out of range maximum file on the directory dir
TEST mkDir FAILED
```

Pruebas de funcionamiento

Todas las pruebas de funcionamiento están contenidas en el fichero test.c con el comentario “Pruebas de funcionamiento x”, siendo ‘x’ el índice de la prueba de funcionamiento.

1. mkFS - F1.1
    - Entrada: 50 KiB
    - Salida: 0

```
TEST mkFS SUCCESS
```

2. En mountFS y unmountFS mientras que hayamos creado más de un bloque a la hora de crear el disco no da fallos

```
TEST mountFS SUCCESS
```

```
TEST unmountFS SUCCESS
```

3. mkDir - F1.11
    - Entrada:
        - “/dir1"
        - “/dir2"
        - “/dir1/dir11"
        - “/dir1/dir11/dir111"
    - Salida: 0

```
*** Directory named dir1 created with fd 0 in level 0 with predecesor Directoy -
 TEST mkDir SUCCESS
*** Directory named dir2 created with fd 1 in level 0 with predecesor Directoy -
 TEST mkDir SUCCESS
*** Directory named dir11 created with fd 2 in level 1 with predecesor Directoy dir1
 TEST mkDir SUCCESS
*** Directory named dir111 created with fd 3 in level 2 with predecesor Directoy dir11
 TEST mkDir SUCCESS
```

4. rmDir - F1.12
    - Entrada: /dir2"
    - Salida: 0

```
*** Directory named dir2 with fd 1 in level 0 with predecesor Directoy - is removed
 TEST rmDir SUCCESS
```

5. createFile - F1.4
    - Entrada:
        - "/dir1/dir11/dir111/file1111"
        - "/dir1/dir11/dir111/file1112"
        - “/file2”
    - Salida: 0

```
*** File named file1111 created with fd 1 in level 3 with predecesor Directoy dir111
 TEST createFile SUCCESS
*** File named file1112 created with fd 4 in level 3 with predecesor Directoy dir111
 TEST createFile SUCCESS
*** File named file2 created with fd 5 in level 0 with predecesor Directoy -
 TEST createFile SUCCESS
```

6. removedFile - F1.5
    - Entrada: “/file2”
    - Salida: 0

```
 TEST createFile SUCCESS
*** File named file2 with fd 5 in level 0 with predecesor Directoy - is removed
```

7. openFile - F1.6
    - Entrada:
        - "/dir1/dir11/dir111/file1111"
        - "/dir1/dir11/dir111/file1112"
    - Salida: 0

```
*** File named file1111 with fd 1 in level 3 with predecesor Directoy dir111 is opened now
 TEST openFile SUCCESS
*** File named file1112 with fd 4 in level 3 with predecesor Directoy dir111 is opened now
```

8. closeFile - F1.7
    - Entrada: "/dir1/dir11/dir111/file1112"
    - Salida: 0

```
*** File named file1112 with fd 4 in level 3 with predecesor Directoy dir111 is closed now
 TEST closeFile SUCCESS
```

9. writeFile - F1.9
    - Entrada: fd=1, buffer=”dso2019”, sizeof(buffer)
    - Salida: sizeof(buffer)

```
*** File named file1111 writed with fd 1 in level 3 with predecesor Directoy dir111
 TEST writeFile SUCCESS
```

10. readFile - F1.8
    - Entrada: fd=1, buffer, sizeof(buffer)
    - Salida: sizeof(buffer)

```
*** File named file1111 read with fd 1 in level 3 with predecesor Directoy dir111
 TEST readFile SUCCESS
block read is dso2019
```

11. lsDir - F1.13
    - Entrada: "/dir1/dir11/dir111",inodesDir, namesDir
    - Salida: 0

```
TEST lsDir SUCCESS
************************
inodesDir[0] = 1
namesDir[0] = file1111
inodesDir[1] = 4
namesDir[1] = file1112
inodesDir[2] = -1
namesDir[2] =
inodesDir[3] = -1
namesDir[3] =
inodesDir[4] = -1
namesDir[4] =
inodesDir[5] = -1
namesDir[5] =
inodesDir[6] = -1
namesDir[6] =
inodesDir[7] = -1
namesDir[7] =
inodesDir[8] = -1
namesDir[8] =
inodesDir[9] = -1
namesDir[9] =
************************
```

12. lseekFile - 1.10
    - Entrada:
        - fd=1, offset=0, whence=FS_SEEK_END
        - fd=1, offset=0, whence=FS_SEEK_BEGIN
        - fd=1, offset=10, whence=FS_SEEK_CUR
    - Salida: 0

```
*** File with fd 1 named file1111's pointer is now 2047
 TEST lseekFile SUCCESS
*** File with fd 1 named file1111's pointer is now 0
 TEST lseekFile SUCCESS
*** File with fd 1 named file1111's pointer is now 10
 TEST lseekFile SUCCESS
```

Conclusiones y comentarios personales

Durante el desarrollo de la práctica hemos encontrado numerosos problemas a resolver, pero la mayoría de ellos los hemos resuelto gracias a las ayudas proporcionadas por parte de los profesores.

Consideramos que la mayor dificultad de la práctica en diseñar un sistema de ficheros que sea eficiente y bueno, y depurar los errores de implementación.

Por último, en esta práctica hemos aprendido mucho acerca del sistema de ficheros y sus interacciones internas.