Práctica 2 - Opciones

Se entregará en el campus virtual la última versión **estable** de la práctica, comprimida en `.zip` o `.tar.gz`. Si no hubiese versión estable se entregará **la última versión de la opción 1**. El directorio raíz de desarrollo se llamará L<LL>P<PP>P2, donde <LL> es el laboratorio y <PP> es el puesto asignado (p.e. L03P18P2). Luego, se creará una subcarpeta para cada opción desarrollada (L03P18P21 para desarrollar la opción 1, L03P18P22 para desarrollar la opción 2, etc.). Recuerda entregar únicamente las versiones **estables**, aquellas de las que tenga constancia el profesor de laboratorio.

Opción 1 (Obligatoria, 5 puntos) - Script de comprobación

Completar el script de la parte obligatoria, para que realice las tareas que se indican en el guión de la práctica. Demostrar al profesor que la implementación funciona correctamente ejecutando el script.

Opción 2 (2.5 puntos) - Opción -m

Implementar la opción `-m` en el proyecto, de forma que se permita montar un sistema de ficheros existente. Esto implica implementar la función:

```
int myMount(MyFileSystem* myFileSystem, char* backupFileName);
```

, que llama a su vez a las siguientes funciones:

```
int readBitmap(MyFileSystem* myFileSystem);
int readDirectory(MyFileSystem* myFileSystem);
int readSuperblock(MyFileSystem* myFileSystem);
int readInodes(MyFileSystem* myFileSystem);
```

Demostrar al profesor que la implementación funciona correctamente con una prueba simple.

Opción 3 (2.5 puntos) - Añadir campo tipo y soporte parcial para directorios

Modificar la estructura del nodo-i para que aparezca también el tipo, que declararemos como un entero

- `tipo = 0` → Archivo regular
- `tipo = 1` → Directorio
- `tipo = 2` → Enlace simbólico

Modificar la funciones de `fuseLib.c` necesarias para que cuando se cree un archivo este aparezca por defecto como tipo archivo. Modificar `my_getattr` para tener en cuenta que nuestro sistema puede tener archivos y directorios.

Modificar las funciones directamente relacionadas con archivos para que sólo puedan ejecutarse si el nodo-i es de tipo archivo regular y den error en otro caso (EISDIR (The named file is a directory)). Añadir la función `my_mkdir`, que cree un directorio (No se va a acceder al directorio, cuando hagamos un `ls` debe aparecer como un directorio).

Demostrar al profesor que la implementación funciona correctamente con una prueba simple.

Opción 4 (5 puntos) - Soporte para enlaces rígidos

Dar soporte en el sistema de ficheros a enlaces rígidos. Básicamente consiste en realizar las siguientes modificaciones (puede que haya que realizar modificaciones adicionales, dependiendo de cada proyecto):

- Añadir un campo `nlinks` al nodo-i que lleva la cuenta del número de enlaces.
- Modificar la función `copyNode` de `myFS.c` para copiar también este campo.
- Modificar la función `my_mknod`, para que inicialice el número de enlaces a 1 al crear el nodo-i.
- Modificar la función `my_getattr`, para que lea el número de enlaces del nodo-i.
- Modificar la función `my_unlink`, para que decremente el contador y borre sólo el contenido si el contador de enlaces se hace 0.
- Implementar la función link de fuse:
  ```
  int my_link(const char *path, const char *lpath);
  ```
  que crea un enlace lpath a un fichero path. Registrar dicha función en la estructura de operaciones de fuse.

Demostrar al profesor que la implementación funciona correctamente con una prueba simple.

Opción 5 (5 puntos) - Soporte para enlaces simbólicos

De la misma forma que en la opción 4, dar soporte en el sistema de ficheros a enlaces simbólicos.

Para poder realizar este apartado se puede, por ejemplo, modificar la estructura del nodo-i para que contenga un campo entero llamado `tipo` que identifique el tipo de nodo-i, que podrá ser bien un fichero regular (tipo=0) o bien un enlace simbólico (tipo=1). Para crear un enlace simbólico se debe implementar la función:

```
int my_symlink(const char* path1, const char* path2);
```

Para leer un enlace simbólico se debe completar adecuadamente la función `getattr`, actualizando la estructura stat según el campo tipo mencionado anteriormente

```
stbuf->st_mode = S_IFLNK | 0644;
```

Además, se debe implementar la función:

```
int fuse_operations::readlink(const char*, char*, size_t);
```

Demostrar al profesor que la implementación funciona correctamente con una prueba simple.