DM510 OPERATING SYSTEMS ASSIGNMENT 3

ESRA DURAN

I have completed this assignment with Tugra Demirel.

tudem23@student.sdu.dk

MAY 20, 2023

SDU

1.Introduction

The objective of this project is to develop a file system in Linux using FUSE (File System in Userspace), which allows users to create their own file systems without modifying the kernel. The implementation includes support for directories and files, and provide functions to create, delete, and list them. Additionally, the file system should support opening, closing, reading, and writing files. It should adjust the writing by looking at the free spaces and organize them. Also, the file system should be able to display the size, access time, and modification of all the files in a folder.

The steps of that project can be listed as follows:

1. Design Decisions and Implementation

- Header File
- lfs.c
- Makefile

2.Discussions

3. Creating some tests for our file system

4.Conclusion

Before starting all the steps, we need to be sure that we have upladed a FUSE system into our Linux.

DESIGN DECISIONS and IMPLEMENTATION

The file system is initially designed as a tree that has inodes inside. A sturuct type “inode” has those attributes inside:

-access time

-modification time

-id of the parent folder

-the type of the inode (editable or not)

-an array that keeps the id of the blocks inside this inode

-size of the inode (bits)

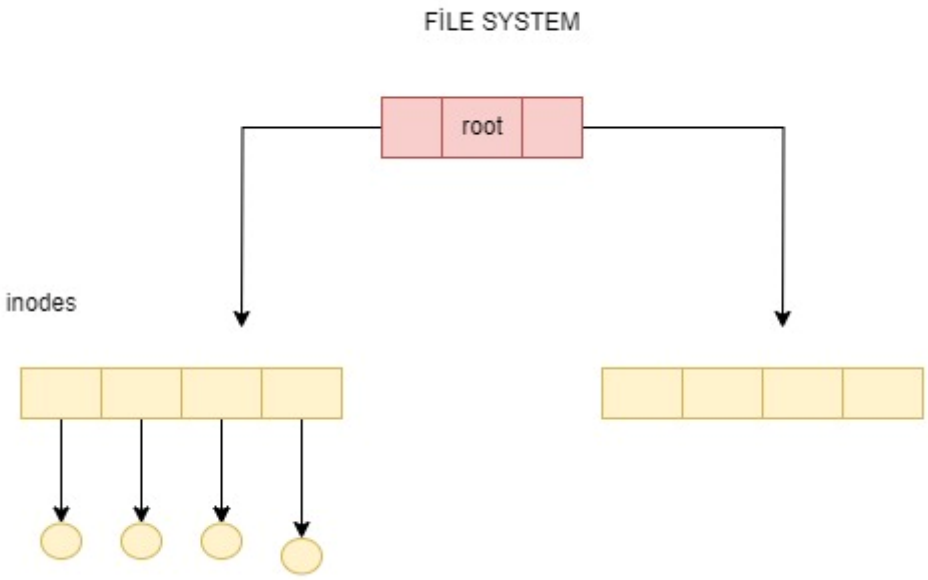
- number of blocks allocated for the inode

-name length of the inode

Similarly we also have an union type of blocks. They have 2 attributes:

-inode that they are residing

-an array that will keep the data for the block



We have a tree structure to keep folders as parent and leaves. If the folder is not root, it should have only one parent node.

When deleting a parent, we make to sure that it should be empty. So, the file system does not allow it to be deleted. Instead, we need to use “rm -r” command to delete the node, even it has leaves in it.

Before setting up the file system, we allocated first 6 blocks for the initial map. And the next one is for the root inode. This map will help us to find some free bits in the block to use.

Also , we have an unlinked function that takes the removed block, and delete the link between this parent and its block. It simply happens by making free the data from the struct inode parent.

- ***Header File (lfs.h)***

In this file, we initially have the names of the given functions and a struct named fuse_operations. Additionally, we have new-defined struct types “inode_type” and “ type block” in order to keep data and make corresponding dependencies between blocks and inodes.
After that, we created many helper functions for our main functions. These helps us fo getting the block, writing and reading from the blocks, setting up the system, freeing the used blocks after completed, getting their names and id’s, finding the free slots, finding the block from its path, setting and updating the data in blocks etc.

- ***lfs.c***

  In this main “.c” file, we implemented fundamental functions for our FUSE. These are :
  Lfs_write
  Lfs_getattr
  Lfs_mkdir
  Lfs_rmdir
  Lfs_readdir
  Lfs_release
  Lfs_unlink
  Lfs_open
  Lfs_read
  Even tough we are not asked to implement all, we tried to understand and implement all functions.
  In order to find the specified address from the blocks, we first need to divide the address to the 4096 bit (4 mb). This is the storage that we allocated for each block. Then, the remainder will be the offset of that address and it shows the location that we are looking for. This method can be categorized under contagious allocation. Because, we cannot keep all the empty spots together. In order to find, we need to traverse all the blocks and find suitable one.

DISCUSSIONS

Our file system has limited amount of inodes and blocks and also sizes. Since it is custom, it may not be powerful for semi-huge files. Not all of the error cases are handled, so it would be better to control existence and validities at some points. Since this is a basic file system, it is not guaranteed to rename a file in our FUSE system. For the further improvements, we might extend this file system to allow multiple writers at the same time by using some lockers. So this way, we can create an algorithm to order them and make use of it in a shared computer system.

TESTS

As the test cases, I firstly tried to make a directory by using the command “mkdir”. And then I tried to write something in it as a txt file. As the test cases, short and long messages are written and observed whether they change or not. We also could read the data from those files; we could open it and see all the details of the files including their last modification and access times.

Another test case was deleting of the files. As the purpose of deleting, the system should not delete a folder if it has leaves in it. So, I created children folder inside the parent folder and tried to delete it. However, it did not allow it to be removed. So, it worked when we add “rm -r” command.

CONCLUSION

Overall, we created a basic File system by using FUSE, which allows users to open, read and write to files. We decided to make it as a tree structure, but it is also possible to create lists and other data algorithms. We could also list the files and their attributes. Most of the fundamental requirements are met successfully. However, we had some problems while unmounting. We could not figure this out as it can be seen in the video test.