UNIVERSITY OF THESSALY

POLYTECHNICAL SCHOOL

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

«Implementation of a simple file system utilizing the FUSE infrastructure »

4th Project

Operating Systems

Gianni Ioanna (gioanna@uth.gr)

Kapetanos Georgios (kapgeorgios@uth.gr)

Zafeiri Stamatia Varvara (zstamat@uth.gr)

Supervisor: Christos Antonopoulos

2021-22

1. REQUIREMENTS

The goal of this project is to understand how a simple file system in Linux works and implement it. More specifically, we will implement a file system which tries to “compress” its data by recognizing block sharing opportunities in the same or in different files and reusing any blocks with similar content. Those blocks do not have to be saved multiple times to disk. In order not to compare the whole block, as this would be slow, we will maintain a hash for each existing block and after calculating the corresponding hash for the new block which is requested to be written, we will compare it with the hashes of existing blocks. If two hashes are equal, we assume that the contents of the corresponding blocks are equal. With this way are written to disk exclusively unique (new) blocks.

To avoid kernel changes the file system will be implemented as a user level file system utilizing the FUSE infrastructure.

2. Assumption not followed:

We can unmount and mount again and the files will still exist

3. DESIGN DECISIONS

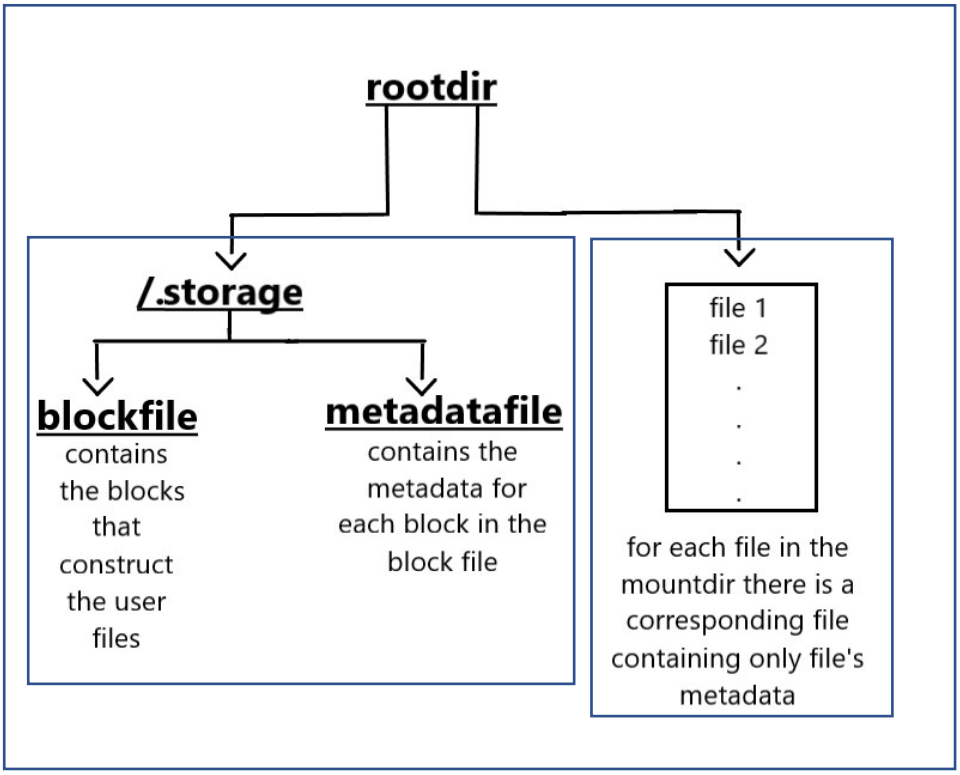
In storage, we are going to have two files, one to save the blocks and one with the metadata of the blocks, which will be:

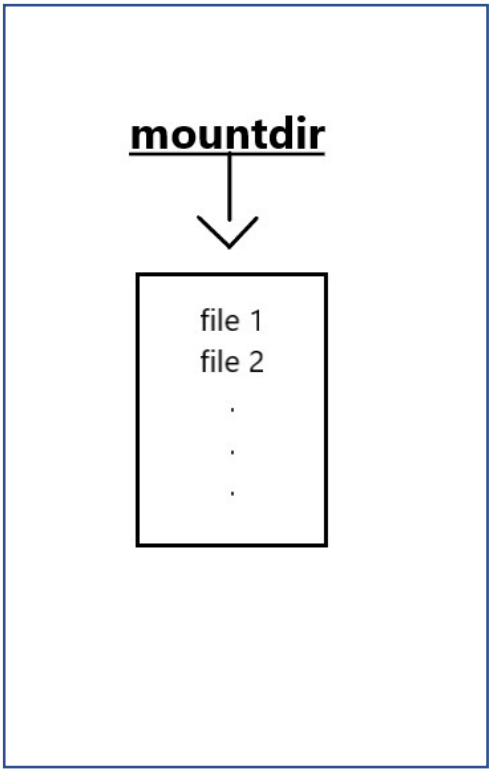
- the **hash** string
- **references**
    - the number of times the block is used
- the **block offset** in block_file

Also, it will be a file for every real file with the following information:

- the **number of blocks** of the file
- an **array with the offsets of blocks** in the block_file
    - pointers to the block to construct the initial file
- an **array with the offsets of the metadata** of the blocks in the metadata file
    - for easier comparing of the hashes

Block size is 4096 bytes.





Functions:

bb_init:

In init we create the Storage folder and the two files that exist in it:

block_file and metadata_file and give the permissions needed

Those two paths are stored in the BB_DATA struct for future use.

bb_getattr:

In getattr we make sure to get the right size of the file by multiplying the block size with the number of blocks that the file has

bb_unlink:

Each time we delete a file, unlink is called.

Via this function, we enter the metatdata_file and traverse it. For each block that was contained in the deleted file, we enter its metadata and decrease the reference number by one.

bb_read:

In read we open the file in storage which contains all the blocks.

We also open the corresponding file in the rootdir to get the file_info data so we can get the block_offset array to use later

Based on the block_offsets of the file, we pass each of these blocks to the buffer to retrieve the information we want to be read in the end.

bb_write:

In the write function, we retrieve the file_info data of the file which bb_write was called on, or generate new data in the case that the file is written for the first time.

We split the buffer in 4096 byte chunks. For every chunk, which corresponds to a block about to be written, we calculate its hash, search the metadata file for either a block with the same hash or an empty space (whose previous data have been deleted). In the first case we increment the “references” value of the block and use this block’s offset. In the second case we update the hash and “references” values and write the new data of the block. If none of the above is found, we generate a new block in the end of the block file, along with a corresponding metadata in the metadata file.

We update the number of blocks value, the block and metadata offsets and save the file info to the appropriate file.

1. EXPERIMENTAL ANALYSIS

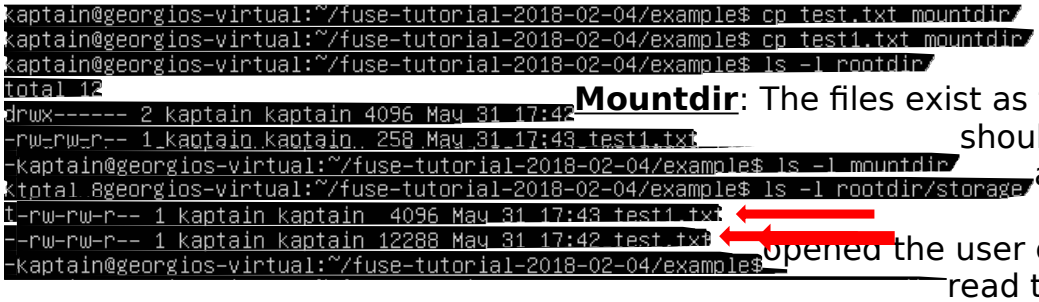
For the experimental analysis, we constructed 2 files. [test.txt & test1.txt] and copied one after another to the mountdir directory. Both files have similar contents. Specifically, test.txt is 12.288 bytes and consists of 2 identical and 1 different block.

Test1.txt consists of 1 block that is similar to a block that exists in test.txt

Taking into consideration the above, we expect for our storage to contain 2 blocks in total, something that is confirmed by the screenshots below.

As we can see below the size of the block_file is 8.192 = 2*BLOCK_SIZE bytes as we expected.

Rootdir: The filesystem recognizes the similar blocks and only one copy of the block is stored in the block_file.



Mountdir: The files exist as they should and if opened the user can read the content with no problem

Next, we test deletion of files, using automated test1.sh, which produces the following output.

Running rm command, calls bb_unlink function, which removes the file specified by the user, and updates the “references” value of the used block’s metadata. Even if the “references” value becomes zero, and no files are using this block, neither the data of the block from the block_file nor the metadata are removed, as a result the block and metadata files are filled with blocks not corresponding to any files and their size is not reduced. Those blocks will be used again on subsequent file creations as described in bb_write.

Upon calling bb_unlink function, the total number of blocks used from files is printed on bbfs.log.

```
bb_unlink: 1 blocks used out of 2
    unlink returned 0
```

In this example, file test.txt is removed, causing a block saved on block_file and its metadata to be unused, until a new “bb_write” call creates a new block and overwrites these unused data.

```
Fuse library version 2.9
about to call fuse_main
Makefile  auto.sh  bbfs.log  mountdir  rootdir  test.txt  test1.sh  test1.txt
ABOUT TO COPY FILES IN MOUNTDIR
total 12
drwx------ 2 kaptain kaptain 4096 Jun  1 14:55 storage
-rw-rw-r-- 1 kaptain kaptain  258 Jun  1 14:55 test.txt
-rw-rw-r-- 1 kaptain kaptain  258 Jun  1 14:55 test1.txt
total 12
-rw-rw-r-- 1 kaptain kaptain 8192 Jun  1 14:55 block_file
-rw-rw-r-- 1 kaptain kaptain   64 Jun  1 14:55 metadata_file
total 8
-rw-rw-r-- 1 kaptain kaptain 12288 Jun  1 14:55 test.txt
-rw-rw-r-- 1 kaptain kaptain  4096 Jun  1 14:55 test1.txt
ABOUT TO REMOVE TEST.TXT
REMOVED AND NOW LS
total 8
drwx------ 2 kaptain kaptain 4096 Jun  1 14:55 storage
-rw-rw-r-- 1 kaptain kaptain  258 Jun  1 14:55 test1.txt
total 12
-rw-rw-r-- 1 kaptain kaptain 8192 Jun  1 14:55 block_file
-rw-rw-r-- 1 kaptain kaptain   64 Jun  1 14:55 metadata_file
total 4
-rw-rw-r-- 1 kaptain kaptain 4096 Jun  1 14:55 test1.txt
```