ECE318 Operating Systems

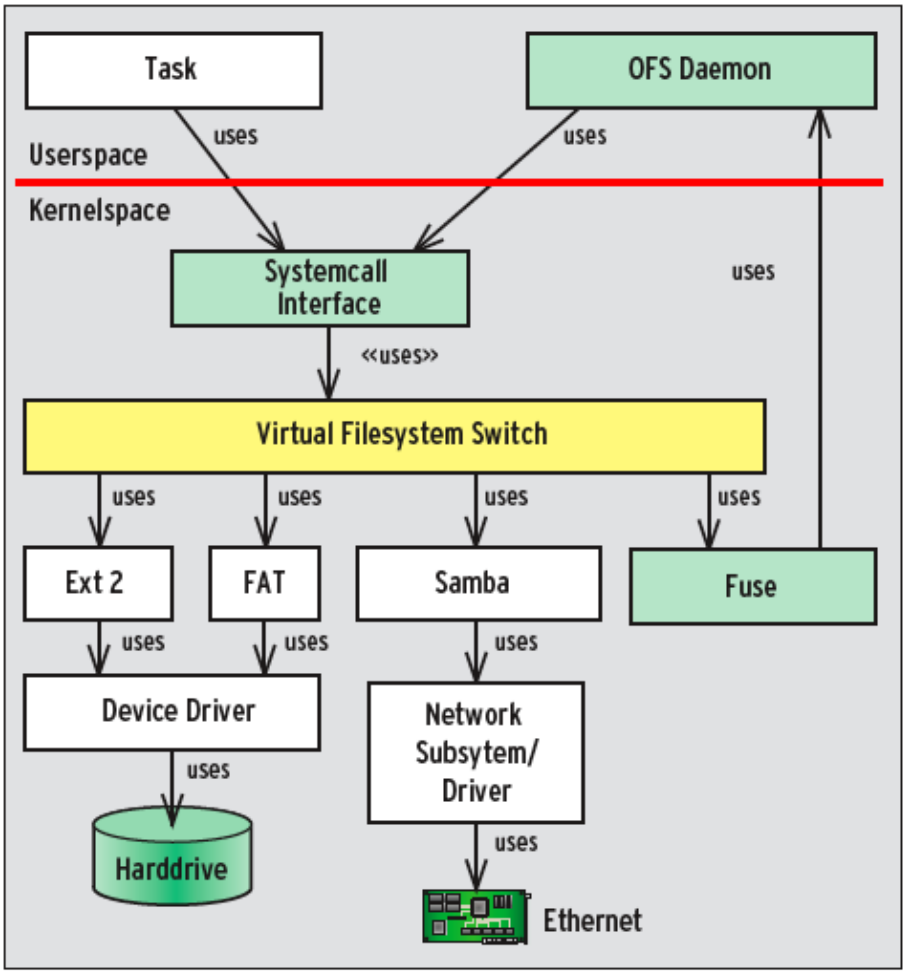
Assignment 4

FUSE File System

Kyritsis Spyridon 2697

Stoltidis Alexandros 2824

Koutsoukis Nikolaos 2907



Project Description

In this project a Big Brother File System (Dummy File System) is modified in order to achieve better data compression and get rid of duplicate blocks on storage memory.

Implementation

The root directory represents the real storage of the file system.

The mount directory is used as a “Mirror Image” of the root directory.

Root contains two unique folders (.storage and .meta) which contain information about the files contained in the real storage.

A file on Mount is split to N number data blocks of 4KiB each.

At the **.storage** directory, data blocks are stored and contain 4KiB of data from the file. Many data blocks might be used to “Reconstruct” a file.

The naming Convention we follow for the Data Blocks is {hash_of_block}_{count}, where the hash is calculated based on the block’s content and the count refers to the number of times the same exact block is used in the file system (many files can use the same data block or even the same file can use the same block multiple times).

At the **.meta** directory, we store meta information for each file. This meta information is a “Path” of data block hashes that need to be read in order to reconstruct the given file.

The naming convention we follow for a meta file that is located in the path: /mount/path/to/file, is {path-to-file}, where ‘-’ are used instead of ‘/’ to name the meta file.

FUSE Initialization

Two directories, **.storage** and **.meta** are created to store block data and meta data respectively.

FUSE Writes

In order to write data, the function bb_write() is called. If the file is N bytes the function bb_write() is Called ceil(N / 4096) times with offset and size set accordingly on each subsequent call by the “Module”.

If a new file is created during the bb_write() call, meaning its meta file on .meta doesn't exist, then the meta file is created as well.

In order to reuse data blocks, a hash is calculated based on the data of the block to be writen. The hash is calculated by the SHA1 hashing algorithm. If the calculated hash already exists in the .storage directory, its Count is incremented by one, else it is created.

Finally, the calculated hash is appended in the meta File and the “Path” of Blocks can be read in order to reconstruct the file.

FUSE Reads

In order to read and “Reconstruct” a file, the function bb_read() is called. The meta file from .meta is parsed and for each retrieved hash a block from .storage is searched and read. The reading process finally depends on the function bb_getattr(), which sets the size of the read buffer.

FUSE Get Attributes

Each entry (Hash) from the Meta File is read and searched on .storage the directory. All info regarding a data blocks is stored on a **struct stat**. This information is used in the reading process in order for the buffer array to be dynamically allocated.

FUSE File Deletions

In order to delete a file, the function bb_unlink() is called. bb_unlink() reads hash entries from the meta file we want to delete. Each hash read from the meta file is searched on the .storage directory. If the storage block’s reference count is one, then the block is deleted. If the storage block’s reference count is more than one, the reference counter is decremented by one. Finally, the file’s meta file is deleted.

Testing our Implementation

1. files_basic.sh

This script tests the write and read operations. The root directory is cleared (blocks from .storage and meta files from .meta are deleted). Three files are created:

i. Two files that consist of 8KiB of 'A' and are copied into mount

ii. The Third File consists of 8KiB of 'B' and is also copied into mount

Finally, the .storage and .meta content is printed.

```
alexstolt@alexstolt:~/fuse-dev/example/tests$ ./files_basic.sh
Enter Absolute Path of Root Directory: /home/alexstolt/fuse-dev/example/rootdir
Root Directory Set: /home/alexstolt/fuse-dev/example/rootdir
Enter Absolute Path of Mount Directory: /home/alexstolt/fuse-dev/example/mountdir
Mount Directory Set: /home/alexstolt/fuse-dev/example/mountdir
    [*] Clearing /home/alexstolt/fuse-dev/example/rootdir/.storage/
    [*] Clearing /home/alexstolt/fuse-dev/example/rootdir/.meta/
    [*] Clearing /home/alexstolt/fuse-dev/example/rootdir
    [*] Creating Two 8KiB Files with the Same Content (Perl Script)...
    [*] Creating One 4KiB File with the Different Content (Perl Script)...
    [*] Copying Files to Mount (BB_WRITE)...
    [*] Differences Between FILE_ONE at Working Directory and Mount (BB_READ)...
    [*] Differences Between FILE_TWO at Working Directory and Mount (BB_READ)...
    [*] Differences Between FILE_THREE at Working Directory and Mount (BB_READ)...
    [*] Storage Content at: /home/alexstolt/fuse-dev/example/rootdir/.storage/: 59682bddd4b2a31b08bc6977169691c10db7a501_1  608cabc0f2fa18c260cafd974516865c772363d5_4
    [*] Meta Content at: /home/alexstolt/fuse-dev/example/rootdir/.meta/: FILE_ONE  FILE_THREE  FILE_TWO
```

At **.storage**

The 'B' block is 4KiB so a unique hash is calculated and the reference count is set to one.

The 'A' block on the other hand is created once, with a reference count of one, and is renamed three times. This happens because the second block of the first file is also 'A's and both blocks from the second file contain only 'A's) so its reference count is set to 4.

At **.meta**

One meta file is created for each file.

2. file_basic_rmv.sh

This script tests the write, read and unlink operations. The root directory is cleared (data blocks from .storage and meta files from .meta are deleted). Three files are also created in this test case:

i. The First two files consist of 8KiB of 'A' and are copied into mount

ii. The Third File consists of 4KiB of 'A' and is also copied into mount

Finally, .storage and .meta content is printed after each deletion.

```
alexstolt@alexstolt:~/fuse-dev/example/tests$ ./file_basic_rmv.sh
Enter Absolute Path of Root Directory: /home/alexstolt/fuse-dev/example/rootdir
Root Directory Set: /home/alexstolt/fuse-dev/example/rootdir
Enter Absolute Path of Mount Directory: /home/alexstolt/fuse-dev/example/mountdir
Mount Directory Set: /home/alexstolt/fuse-dev/example/mountdir
    [*] Clearing /home/alexstolt/fuse-dev/example/rootdir/.storage/
    [*] Clearing /home/alexstolt/fuse-dev/example/rootdir/.meta/
    [*] Clearing /home/alexstolt/fuse-dev/example/rootdir
    [*] Creating Two 8KiB Files with the Same Content (Perl Script)...
    [*] Creating One 4KiB File with the Same Content (Perl Script)...
    [*] Copying Files to Mount (BB_WRITE)...
    [*] Differences Between FILE_ONE at Working Directory and Mount (BB_READ)...
    [*] Differences Between FILE_TWO at Working Directory and Mount (BB_READ)...
    [*] Differences Between FILE_THREE at Working Directory and Mount (BB_READ)...
    [*] Storage Content at: /home/alexstolt/fuse-dev/example/rootdir/.storage/: 608cabc0f2fa18c260cafd974516865c772363d5_5
    [*] Meta Content at: /home/alexstolt/fuse-dev/example/rootdir/.meta/: FILE_ONE  FILE_THREE  FILE_TWO
    [*] Deleting FILE_ONE (BB_UNLINK)...
    [*] Storage Content at: /home/alexstolt/fuse-dev/example/rootdir/.storage/: 608cabc0f2fa18c260cafd974516865c772363d5_3
    [*] Meta Content at: /home/alexstolt/fuse-dev/example/rootdir/.meta/: FILE_THREE  FILE_TWO
    [*] Deleting FILE_THREE (BB_UNLINK)...
    [*] Storage Content at: /home/alexstolt/fuse-dev/example/rootdir/.storage/: 608cabc0f2fa18c260cafd974516865c772363d5_2
    [*] Meta Content at: /home/alexstolt/fuse-dev/example/rootdir/.meta/: FILE_TWO
    [*] Deleting FILE_TWO (BB_UNLINK)...
    [*] Storage Content at: /home/alexstolt/fuse-dev/example/rootdir/.storage/
    [*] Meta Content at: /home/alexstolt/fuse-dev/example/rootdir/.meta/
```

At **.storage**

After the deletion of an 8KiB block the reference count is decremented by 2 (from 5 to 3) since two blocks were deleted.

After the deletion of an 4KiB block the reference count is decremented by 1 (from 3 to 2) since one more block was deleted.

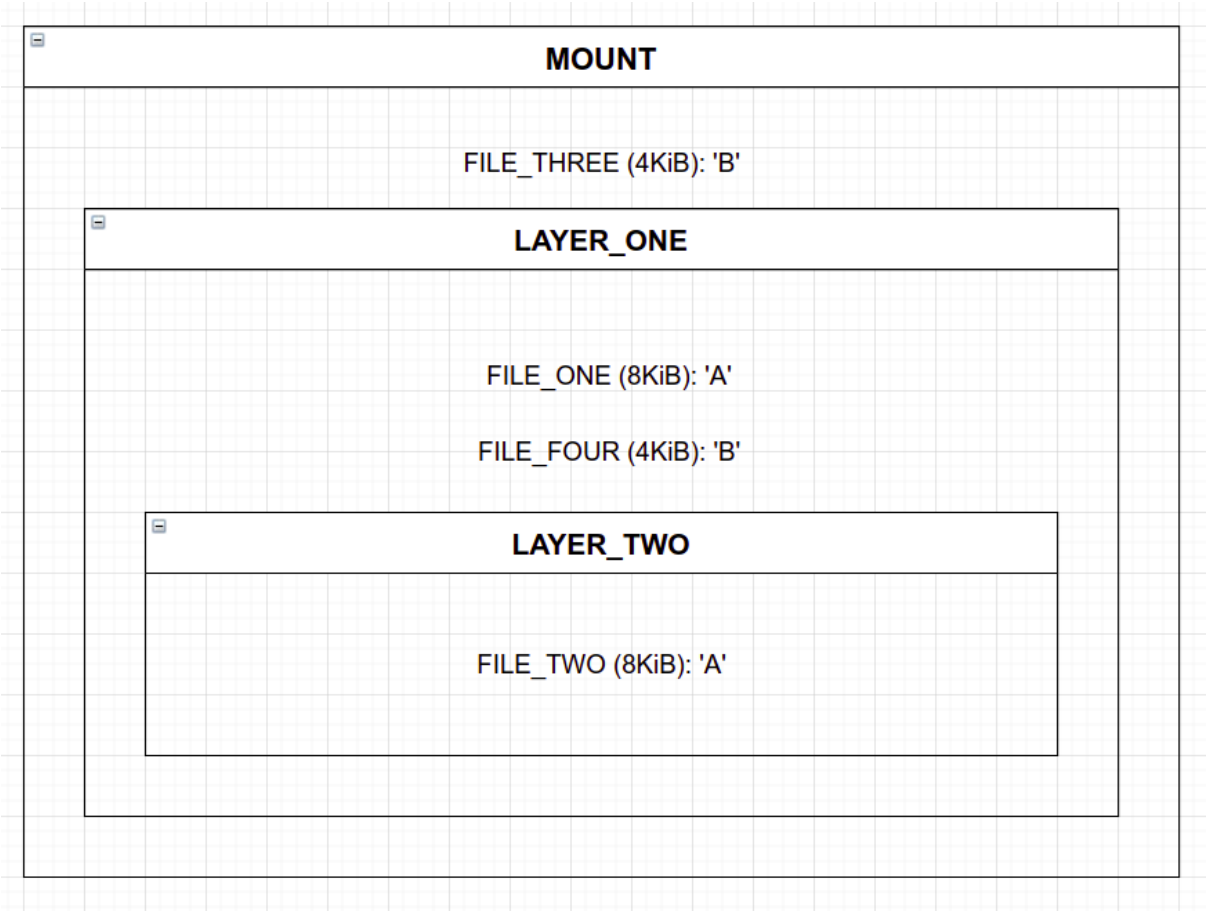
Finally, the deletion of the 8KiB Block Removes the block since the block is not referenced by any file.

At **.meta**

One meta file for each file is created. Upon each file's deletion the corresponding meta file is also deleted.

3. nested_directories.sh

This script tests the write and read operations with nested directories. The root directory is cleared (data blocks from .storage and meta files from .meta are deleted). Four files are created in this test case.



i. The first and second files consist of 8KiB of ‘A’ and are copied into the directories mount/LAYER_ONE and mount/LAYER_ONE/LAYER_TWO respectively

ii. The third and fourth files consist of 4KiB of ‘B’ and are copied into the directories /mount and mount/LAYER_ONE respectively

```
alexstolt@alexstolt:~/fuse-dev/example/tests$ ./nested_directories.sh
Enter Absolute Path of Root Directory: /home/alexstolt/fuse-dev/example/rootdir
Root Directory Set: /home/alexstolt/fuse-dev/example/rootdir
Enter Absolute Path of Mount Directory: /home/alexstolt/fuse-dev/example/mountdir
Mount Directory Set: /home/alexstolt/fuse-dev/example/mountdir
    [*] Clearing /home/alexstolt/fuse-dev/example/rootdir/.storage/
    [*] Clearing /home/alexstolt/fuse-dev/example/rootdir/.meta/
    [*] Clearing /home/alexstolt/fuse-dev/example/rootdir
    [*] Creating Two Nested Directories...
    [*] Creating Two 8KiB Files with the Same Content (Perl Script)...
    [*] Creating Two 4KiB Files with the Same Content (Perl Script)...
    [*] Copying Files to Mount's Layer One and Two (BB_WRITE)...
    [*] Differences Between FILE_ONE at Working Directory and Mount (BB_READ)...
    [*] Differences Between FILE_TWO at Working Directory and Mount (BB_READ)...
    [*] Differences Between FILE_THREE at Working Directory and Mount (BB_READ)...
    [*] Differences Between FILE_FOUR at Working Directory and Mount (BB_READ)...
    [*] Storage Content at: /home/alexstolt/fuse-dev/example/rootdir/.storage/: 59682bddd4b2a31b08bc6977169691c10db7a501_2  608cabc0f2fa18c260cafd974516865c772363d5_4
    [*] Meta Content at: /home/alexstolt/fuse-dev/example/rootdir/.meta/: FILE_THREE  LAYER_ONE-FILE_FOUR  LAYER_ONE-FILE_ONE  LAYER_ONE-LAYER_TWO-FILE_TWO
```

At **.storage**

Just like in the first test .storage contains the hashes with the correct reference count for each block. The fact that the files are located on different directories is not a problem since their hashes are the search key on the .storage directory (where all data blocks are stored).

At **.meta**

One meta File is created for each file. Files that are not on the mount Directory (FILE_ONE, FILE_TWO and FILE_FOUR) are named based on their relative path. A '-' is used instead of a '/' to represent the file of the path, so that we can correctly name their meta files and prevent collisions.

3. nested_directories_rmv.sh

This script tests the write, read and unlink Operations with nested directories. The root directory is cleared (data blocks from .storage and meta files from .meta are deleted). Four Files are also Created in this Test Case.

i. The first and second files consist of 8KiB of 'A' and are copied into the directories mount/LAYER_ONE and mount/LAYER_ONE/LAYER_TWO respectively

ii. The third and fourth files consist of 4KiB of 'B' and are copied into the directories mount/ and mount/LAYER_ONE respectively

Then a recursive deletion is tested (rm -rf *) on the mount directory.

```
alexstolt@alexstolt:~/fuse-dev/example/tests$ ./nested_directories_rmv.sh
Enter Absolute Path of Root Directory: /home/alexstolt/fuse-dev/example/rootdir
Root Directory Set: /home/alexstolt/fuse-dev/example/rootdir
Enter Absolute Path of Mount Directory: /home/alexstolt/fuse-dev/example/mountdir
Mount Directory Set: /home/alexstolt/fuse-dev/example/mountdir
    [*] Clearing /home/alexstolt/fuse-dev/example/rootdir/.storage/
    [*] Clearing /home/alexstolt/fuse-dev/example/rootdir/.meta/
    [*] Clearing /home/alexstolt/fuse-dev/example/rootdir
    [*] Creating Two Nested Directories...
    [*] Creating Two 8KiB Files with the Same Content (Perl Script)...
    [*] Creating Two 4KiB Files with the Same Content (Perl Script)...
    [*] Copying Files to Mount's Layer One and Two (BB_WRITE)...
    [*] Differences Between FILE_ONE at Working Directory and Mount (BB_READ)...
    [*] Differences Between FILE_TWO at Working Directory and Mount (BB_READ)...
    [*] Differences Between FILE_THREE at Working Directory and Mount (BB_READ)...
    [*] Differences Between FILE_FOUR at Working Directory and Mount (BB_READ)...
    [*] Storage Before Removing All Content: /home/alexstolt/fuse-dev/example/rootdir/.storage/: 59682bddd4b2a31b08bc6977169691c10db7a501_2  608cabc0f2fa18c260cafd974516865c772363d5_4
    [*] Meta Before Removing All Content: /home/alexstolt/fuse-dev/example/rootdir/.meta/: FILE_THREE  LAYER_ONE-FILE_FOUR  LAYER_ONE-FILE_ONE  LAYER_ONE-LAYER_TWO-FILE_TWO
    [*] Content Deletion (BB_UNLINK)
    [*] Storage After Removing All Content: /home/alexstolt/fuse-dev/example/rootdir/.storage/
    [*] Meta After Removing All Content: /home/alexstolt/fuse-dev/example/rootdir/.meta/
```

At **.storage**

The directory is now empty since all blocks were successfully deleted.

At **.meta**

The directory is now empty since all blocks were successfully deleted.

Errors: Truncation

We tried to implement truncation but failed due to time limitations.

The truncation concept is that when a block is changed, the function bb_write() is called with its offset parameter set. We open the meta file and go to the (offset / 4096)th entry of the meta file, where the old block is located. We store the old hash in main memory and we overwrite the old hash with the calculated hash of the new data block. Now we search for the old hash in the .storage directory and perform an unlink Operation to the block if the reference count is one, else we decrement the reference count by one. Finally, we write the new hash block in storage. If the data block does not exist, we create it, else we increment its reference counter by one.

Conclusions

The data compression of the blocks is a crucial part of the file system, since many real Files contain headers and footers with the same information that can be compressed to a single block.

However, our implementation has two major flaws

- If the reference count is larger than PATH_MAX a buffer overflow exploit is possible and an INT overflow will also occur
- SHA1 only allows 2^40 Files and many hashes may collide with each other. This will lead to data losses or data corruption in the long run.