FILE SYSTEM WITH FUSE

Team Members

Sharmishtha Singh,01FB16ECS354

Shirisha S Rao,01FB16ECS364

Shivam Ganwani,01FB16ECS365

Shivangi Jadon,01FB16ECS366

Objective

- Create a virtual file system with FUSE which runs on top of OS.
- Implement the required system calls.
- The file system should pass all the test cases.
- The file system should be persistent.

Crux

- Figuring out when to update acess time,modification time and change time for file and directories.
- Making file system persistence.
- How to assign number of blocks.

Phase1

- ### File System Layout
  - We have a structure for inode, file, directory.
  - Inode structure has all the elements required by stat buffer.
  - File structure contains inode number ,name ,path ,parent path ,size and its content.
  - Directory structure contains inode number, name, parent path, path.
  - We also have inode array of size 100 ,having each element as inode structure, 100 indicating that we can have 100 objects in our file system.
  - We have file array of size 80 and directory array of size 20 indicating we can have 80 files and 20 directories. Each array element is file and directory structure respectively.
  - Three variables curr_inode_no, curr_dir_no, curr_file_no are used to represent number of inodes, files and directories.
  - Block size is 512 bytes.
  - We have a shell executable file which runs the command to initialise the file system.
  - For unmounting we can call fusermount or ctrl +c which unmounts the file system.

Phase 2

- ### System calls

  1. #### init
     - Initialises the file system.
     - It calls the persistent() function which copies the metadata from the external files to inodes of our files and updates current inode number. .
     - If the file system is newly initialised it creates a inode for the root directory.

2. getattr

- For each path accessed this function is called and it maps the inode of that path to stat buffer.
- If the path is invalid it returns error.
- Mapping is done by calling helper function which maps the inode to stat. It also calculates the number of blocks required.

3. readdir

- This is called when we call the ls system call.
- It checks if the path is valid else it returns error.
- It updates the access time for the directory accessed.
- It lists all the children nodes of the requested path.
- It parses the inode array and checks if the inode parent path matches the requested path.
- Then it fills the filler buffer with those names.

4. mkdir

- Called while creating the directory.
- Parse the inode array and check if the directory already exists or not.
- If it exists return error or else create a new inode .
- The access ,change and modification time are set to the time the directory was created.
- Increment the link count of the parent directory.
- Update the mtime and ctime of the parent directory.

5. rmdir

- Parse the inode array and check if the children of the requested directory are present.
- If present, return error else remove the directory.
- Remove the inode from the inode array.
- Decrement the links count of the parent directory.
- Update the mtime and ctime of the parent directory.

6. mknod

- Parse the inode array and check if the file exists or not.
- If present return error else create a inode.
- Initialise size to 0 and the file content to “”.
- Initialise the atime, ctime, mtime to current time.
- Increment the link count of the parent directory.
- Update the mtime and ctime of the parent directory.

7. utime

- It updates the access time and modification time of the file requested.
- It updates utimbuf with those values.

8. unlink

- Parse the inode and check if the file exists.
- If it exists remove the inode from the inode array.
- Decrement the link count of the parent directory.
- Update the mtime and ctime of the parent directory.

9. read

- Parse the inode array and get the index of the inode.
- If offset is greater than the size of the file return 0.
- Else copy the content of the file from the offset to the buffer and return the requested size.
- Update the atime of the file requested.

10. write

- Parse the inode array and get the index of the inode.
- Clear the file content and copy the content from the buffer.
- Update the size of the file to the new size.
- Update the atime, mtime and ctime of the requested file.

• Helper functions

1. get_name

- It extracts the name of the current path and returns it .
- The name of the path is used to update the inode name.

2. inode_to stat

- Copies the inode values to stat buffer whenever required.

3. get_parent_path

- It gets the parent path to store in the inode.
- The parent path is required to update the details of the parent whenever its children are modified.

Phase 3

• Persistence

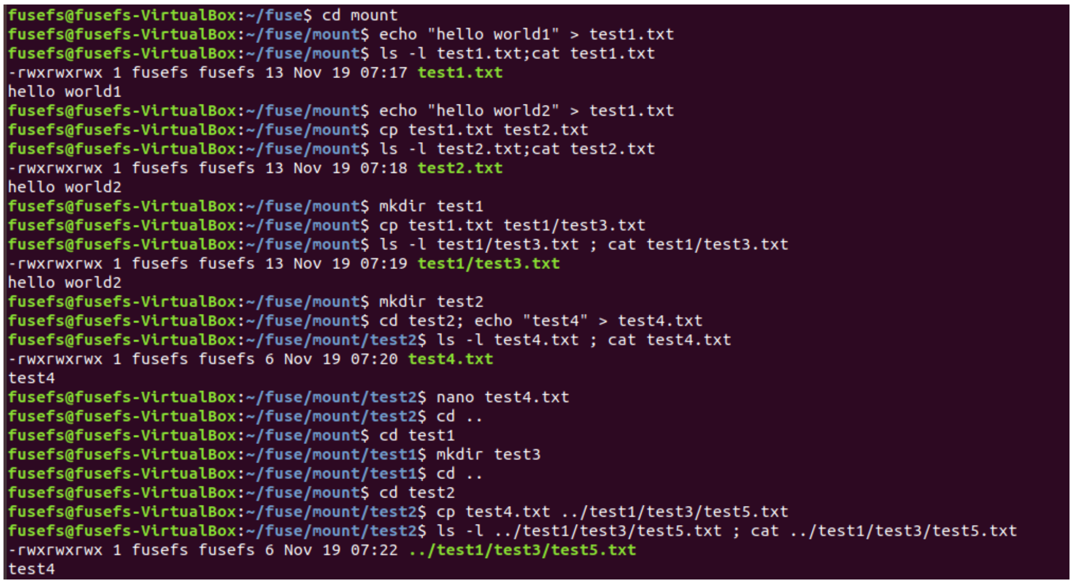
- Persistence is implemented using an external file.
- When you call the ctrl+c command file system is unmounted. destroy() function is called at this time and it copies the whole metadata of the file in the external file.

1. destroy()

- Creates three files inode.txt , file.txt , dir.txt which stores the metadata for inodes, files and directories.
- If the file already exists it appends the newly made changes in these files.

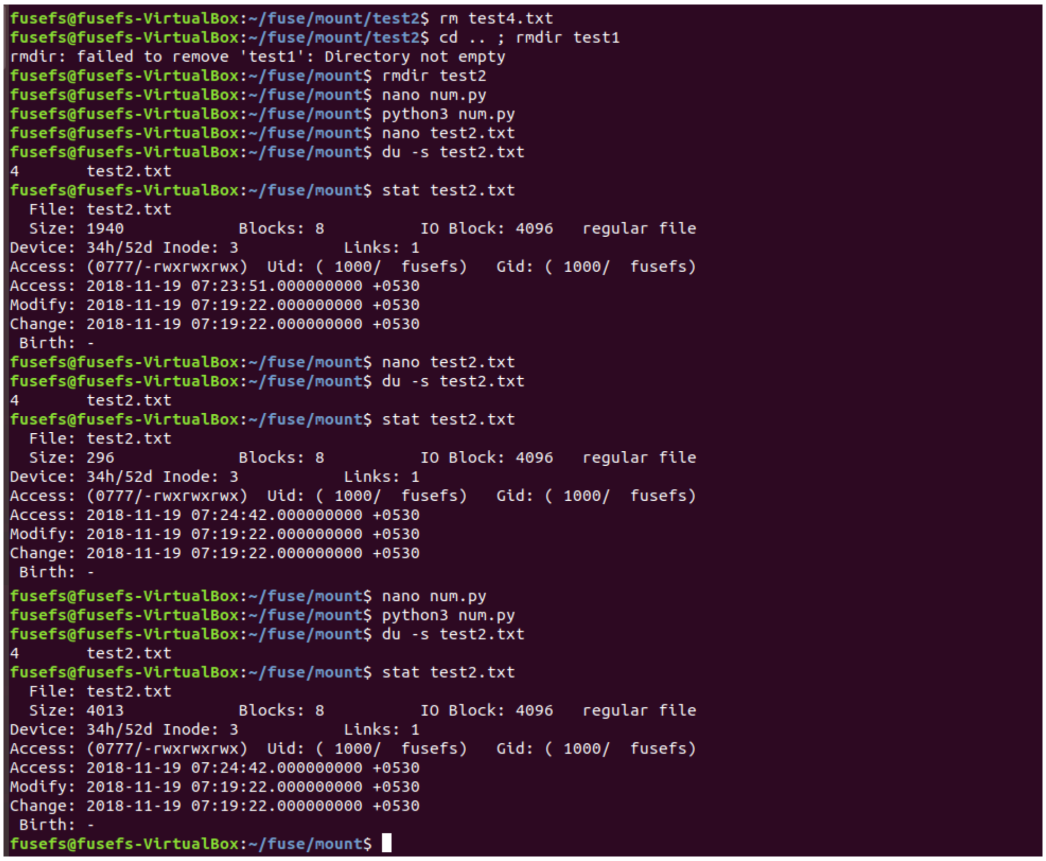
2. persistent()

- This is called while initialising the file system.
- It copies all the metadata from the external metadata files into the inodes.



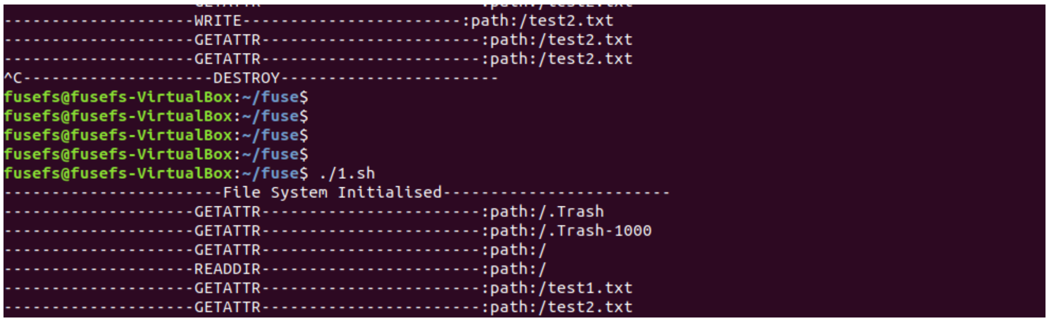
```
fusefs@fusefs-VirtualBox:~/fuse$ cd mount
fusefs@fusefs-VirtualBox:~/fuse/mount$ echo "hello world1" > test1.txt
fusefs@fusefs-VirtualBox:~/fuse/mount$ ls -l test1.txt;cat test1.txt
-rwxrwxrwx 1 fusefs fusefs 13 Nov 19 07:17 test1.txt
hello world1
fusefs@fusefs-VirtualBox:~/fuse/mount$ echo "hello world2" > test1.txt
fusefs@fusefs-VirtualBox:~/fuse/mount$ cp test1.txt test2.txt
fusefs@fusefs-VirtualBox:~/fuse/mount$ ls -l test2.txt;cat test2.txt
-rwxrwxrwx 1 fusefs fusefs 13 Nov 19 07:18 test2.txt
hello world2
fusefs@fusefs-VirtualBox:~/fuse/mount$ mkdir test1
fusefs@fusefs-VirtualBox:~/fuse/mount$ cp test1.txt test1/test3.txt
fusefs@fusefs-VirtualBox:~/fuse/mount$ ls -l test1/test3.txt ; cat test1/test3.txt
-rwxrwxrwx 1 fusefs fusefs 13 Nov 19 07:19 test1/test3.txt
hello world2
fusefs@fusefs-VirtualBox:~/fuse/mount$ mkdir test2
fusefs@fusefs-VirtualBox:~/fuse/mount$ cd test2; echo "test4" > test4.txt
fusefs@fusefs-VirtualBox:~/fuse/mount/test2$ ls -l test4.txt ; cat test4.txt
-rwxrwxrwx 1 fusefs fusefs 6 Nov 19 07:20 test4.txt
test4
fusefs@fusefs-VirtualBox:~/fuse/mount/test2$ nano test4.txt
fusefs@fusefs-VirtualBox:~/fuse/mount/test2$ cd ..
fusefs@fusefs-VirtualBox:~/fuse/mount$ cd test1
fusefs@fusefs-VirtualBox:~/fuse/mount/test1$ mkdir test3
fusefs@fusefs-VirtualBox:~/fuse/mount/test1$ cd ..
fusefs@fusefs-VirtualBox:~/fuse/mount$ cd test2
fusefs@fusefs-VirtualBox:~/fuse/mount/test2$ cp test4.txt ../test1/test3/test5.txt
fusefs@fusefs-VirtualBox:~/fuse/mount/test2$ ls -l ../test1/test3/test5.txt ; cat ../test1/test3/test5.txt
-rwxrwxrwx 1 fusefs fusefs 6 Nov 19 07:22 ../test1/test3/test5.txt
test4
```

Test cases 1-13



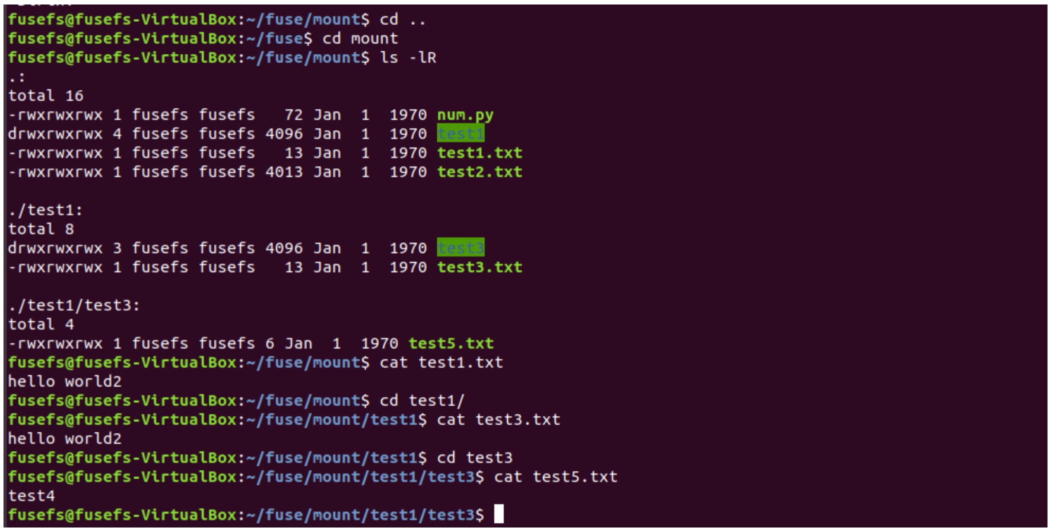
```
fusefs@fusefs-VirtualBox:~/fuse/mount/test2$ rm test4.txt
fusefs@fusefs-VirtualBox:~/fuse/mount/test2$ cd .. ; rmdir test1
rmdir: failed to remove 'test1': Directory not empty
fusefs@fusefs-VirtualBox:~/fuse/mount$ rmdir test2
fusefs@fusefs-VirtualBox:~/fuse/mount$ nano num.py
fusefs@fusefs-VirtualBox:~/fuse/mount$ python3 num.py
fusefs@fusefs-VirtualBox:~/fuse/mount$ nano test2.txt
fusefs@fusefs-VirtualBox:~/fuse/mount$ du -s test2.txt
4       test2.txt
fusefs@fusefs-VirtualBox:~/fuse/mount$ stat test2.txt
  File: test2.txt
  Size: 1940            Blocks: 8          IO Block: 4096   regular file
Device: 34h/52d Inode: 3           Links: 1
Access: (0777/-rwxrwxrwx)  Uid: ( 1000/  fusefs)   Gid: ( 1000/  fusefs)
Access: 2018-11-19 07:23:51.000000000 +0530
Modify: 2018-11-19 07:19:22.000000000 +0530
Change: 2018-11-19 07:19:22.000000000 +0530
 Birth: -
fusefs@fusefs-VirtualBox:~/fuse/mount$ nano test2.txt
fusefs@fusefs-VirtualBox:~/fuse/mount$ du -s test2.txt
4       test2.txt
fusefs@fusefs-VirtualBox:~/fuse/mount$ stat test2.txt
  File: test2.txt
  Size: 296             Blocks: 8          IO Block: 4096   regular file
Device: 34h/52d Inode: 3           Links: 1
Access: (0777/-rwxrwxrwx)  Uid: ( 1000/  fusefs)   Gid: ( 1000/  fusefs)
Access: 2018-11-19 07:24:42.000000000 +0530
Modify: 2018-11-19 07:19:22.000000000 +0530
Change: 2018-11-19 07:19:22.000000000 +0530
 Birth: -
fusefs@fusefs-VirtualBox:~/fuse/mount$ nano num.py
fusefs@fusefs-VirtualBox:~/fuse/mount$ python3 num.py
fusefs@fusefs-VirtualBox:~/fuse/mount$ du -s test2.txt
4       test2.txt
fusefs@fusefs-VirtualBox:~/fuse/mount$ stat test2.txt
  File: test2.txt
  Size: 4013            Blocks: 8          IO Block: 4096   regular file
Device: 34h/52d Inode: 3           Links: 1
Access: (0777/-rwxrwxrwx)  Uid: ( 1000/  fusefs)   Gid: ( 1000/  fusefs)
Access: 2018-11-19 07:24:42.000000000 +0530
Modify: 2018-11-19 07:19:22.000000000 +0530
Change: 2018-11-19 07:19:22.000000000 +0530
 Birth: -
fusefs@fusefs-VirtualBox:~/fuse/mount$ 
```

Test cases 14-21



```
-------------------WRITE----------------------:path:/test2.txt
-------------------GETATTR----------------------:path:/test2.txt
-------------------GETATTR----------------------:path:/test2.txt
^C-------------------DESTROY----------------------
fusefs@fusefs-VirtualBox:~/fuse$
fusefs@fusefs-VirtualBox:~/fuse$
fusefs@fusefs-VirtualBox:~/fuse$
fusefs@fusefs-VirtualBox:~/fuse$
fusefs@fusefs-VirtualBox:~/fuse$ ./1.sh
----------------------File System Initialised-----------------------
-------------------GETATTR----------------------:path:/.Trash
-------------------GETATTR----------------------:path:/.Trash-1000
-------------------GETATTR----------------------:path:/
-------------------READDIR----------------------:path:/
-------------------GETATTR----------------------:path:/test1.txt
-------------------GETATTR----------------------:path:/test2.txt
```

Unmounting the file system



```
fusefs@fusefs-VirtualBox:~/fuse/mount$ cd ..
fusefs@fusefs-VirtualBox:~/fuse$ cd mount
fusefs@fusefs-VirtualBox:~/fuse/mount$ ls -lR
.:
total 16
-rwxrwxrwx 1 fusefs fusefs   72 Jan  1  1970 num.py
drwxrwxrwx 4 fusefs fusefs 4096 Jan  1  1970 test1
-rwxrwxrwx 1 fusefs fusefs   13 Jan  1  1970 test1.txt
-rwxrwxrwx 1 fusefs fusefs 4013 Jan  1  1970 test2.txt

./test1:
total 8
drwxrwxrwx 3 fusefs fusefs 4096 Jan  1  1970 test3
-rwxrwxrwx 1 fusefs fusefs   13 Jan  1  1970 test3.txt

./test1/test3:
total 4
-rwxrwxrwx 1 fusefs fusefs 6 Jan  1  1970 test5.txt
fusefs@fusefs-VirtualBox:~/fuse/mount$ cat test1.txt
hello world2
fusefs@fusefs-VirtualBox:~/fuse/mount$ cd test1/
fusefs@fusefs-VirtualBox:~/fuse/mount/test1$ cat test3.txt
hello world2
fusefs@fusefs-VirtualBox:~/fuse/mount/test1$ cd test3
fusefs@fusefs-VirtualBox:~/fuse/mount/test1/test3$ cat test5.txt
test4
fusefs@fusefs-VirtualBox:~/fuse/mount/test1/test3$ 
```

Persistence after mounting the file system