# Project 3: Document

Qianlan Yang, Maolong Yang

December 13, 2020

## 1 Main data structures

Our project can be divided into two levels, the first level will translate the fuse functions into some small functions using lfs, and the second level will implement lfs, which will interact with disk. The first level simply need to implement the functions according to their meaning.
In our lfs, we design the structure as following. We have `100` segments and one buffer, each one has `1024` blocks with `1KB` space. The buffer and each segments' imap is stored in main memory, and if `fls_fflush()` is preformed the buffer will become a new segment in disk. Our inode only stores the data blocks of its file inside the segment, i.e., if one file is separated into several segments, the inode in each segment can only find data in such segment, this will help implement garbage collection.

## 2 Code organization

Our source code and test cases are at `https://github.com/YMDragon/OperatingSystem/tree/main/fuse-3.10.0/lfs`. Out source code is using **SCons** to compile. **Please install and use it.**
"lfs.c", "lfs.h" and "lfs_*.h" contains data structures of LFS and basic interfaces.
"directory.c" contains some universal directory operation. "permission.c" contains permission checking.
"index.c" contains main function of whole code. And other files contains one functionality corresponding to its filename.

## 3 Details

**Requirement 7:** We use standard Linux file system form to restore these data. When we want access a file or directory, we should check read permission in all the path.

**Requirement 8:** We use pthread_mutex_t to make sure only one process is running at same time.

**Requirement 9:** We simply use `lfs_fflush()` to flush all data to disk.

**Requirement 10:** We cannot survive in every crash, in most case, we will quickly store all in-memory data to disk, then we can restore it.

## 4 Explanation of test cases

We do not have many unit tests, we almost choose integration test. Codes under `lfs/test` are two pressure test, we can copy them into the mounted directory, compile and run them. Also, we can use `vim` to modify data in the file system, or watch videos inside it or any operations you can do under standard linux file system.

## 5 User manual

### 5.1 How to compile and run our LFS

1 Download code at `https://github.com/YMDragon/OperatingSystem/tree/main/fuse-3.10.0/lfs`.

2 Install SCons. Can use "apt install scons" to install.

3 Run "scons" and you will get an executable file

## 6 Known bugs and limitations

If user write more than `100MB` data into our file system, the system cannot correctly ignore the operations and tell user, we will fix it if we have enough time.