# Design Document

### CMPS 111, Winter 2019

## Goal

The goal of this assignment is to use the FUSE file system framework to implement an All-in-One Folder File System (AOFS).

## Design (FS\_FILE)

The file system, FS\_FILE consists of 4KB blocks as specified. We have a superblock (with magic number 0xfa19283e) which contains high-level metadata about the file system. Inside the superblock, we have a bitmap of free blocks where each bit corresponds to a 4KB block. Each 4KB block consists 123 bytes of metadata followed by file data. The data for each block is distributed as follows:

* 0-1 first block number (set to 0 if block is off)
* 2-3 next block number (set to 0 if this block is the last block of the file)
* 4-103 filename
* 104-111 the creation time of the file
* 112-119 the last time the file was modified
* 120-121 size of the file
* 122 bit that specifies if a file has been opened with fopen()
* 123-4094 actual data

## Implementation (hello.c)

In the FUSE interface, there is a list of operations (fuse\_operations). We aimed to build the following operations:

* hello\_getattr(const char \*path, struct stat \*stbuf)
  + Gets attributes, such as:
    - If what the user is requesting exists
    - If the user is requesting a file or a directory
    - The permissions of requested file/directory
  + Sets metadata in superblock
* hello\_readdir(const char \*path, void \*buf, fuse\_fill\_dir\_t filler, off\_t offset, struct fuse\_file\_info \*fi)
  + Checks if path is just “/”
    - If it is, then do the following:
      * Check if offset is 0
      * If yes
        + output “.”, “..”
        + Then loops through blocks, outputting each filename until buf is full.
      * If not
        + skip entries until reach offset, including “.” and “..”
        + Keep looping through blocks, outputting each filename until buf is full
    - If not
      * Check if file exists
        + If yes, output filename
        + If not, output empty buffer
* hello\_open(const char \*path, struct fuse\_file\_info \*fi )
  + Loops through every block
    - If filename is found, the file marked as opened for read/write/other access functions
    - If filename is not found nothing happens
* hello\_read(const char \*path, char \*buf, size\_t size, off\_t offset, struct fuse\_file\_info \*fi)
  + Loops through every block
    - If filename is found
      * Check if marked as open by hello\_open
      * If not open, return 0
      * If open
        + Skip bytes until reach the offset
        + Put bytes into buf until buf is full or until there are no more bytes in the file
        + Return number of bytes put in buf
    - If not found, return 0
* hello\_unlink(const char \*path)
  + Loops through every block
    - If filename is found
      * Go through the blocks the file takes up and recursively empty out blocks from the last block back
      * Disable the corresponding bits in the bitmap
    - If filename is not found nothing is done
* hello\_create(const char \*path, mode\_t mode, struct fuse\_file\_info \*fi)
  + Loops through every block
    - If filename already exists, mark the file as opened for read/write/other access functions
    - If filename does not exists next loop starts
  + Loops through bitmap
    - If free block is found
      * Create and initialize the metadata for the new block
      * Turn the corresponding bit in the bitmap on
      * Mark the file as opened for read/write/other access functions
    - If no free blocks are available error message is displayed
* hello\_write(const char \*path, const char \*buf, size\_t size, off\_t offset, struct fuse\_file\_info \*fi)
  + Loops through every block
    - If filename is found
      * Check if marked as open by hello\_open()
      * If not open, return 0
      * If open
        + Skip bytes until reach the offset
        + Put bytes into file until buf is empty
        + Return number of bytes put into file
    - If not found, return 0
* hello\_release (const char \*path, struct fuse\_file\_info \*fi)
  + Loops through every block
    - If filename is found, the file marked as closed for read/write/other access functions
    - If filename is not found nothing happens
* hello\_truncate (const char \*path, off\_t offset, struct fuse\_file\_info \*fi)
  + Loops through every block
    - If filename is found
      * Compare current size to offset number
      * If size is bigger and more blocks must be allocated
        + Go through bitmap and find open blocks if needed

Switch those bits on in the bitmap

Initialize the metadata on those blocks

* + - * If size is smaller and less blocks must be allocated
        + Delete blocks starting from the end of the metadata until the needed size is reached if necessary

Switch the bits corresponding those blocks off

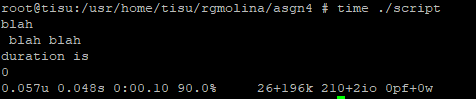
* + - * + If needed, clear bytes from the block at the end of the file until size is reached
      * Set the size of that block to the new size in the metadata
    - If not found, report an error

## Benchmark

The benchmark we created tests the filesystem by doing a series of tasks we expect the filesystem to be able to handle. The tasks are as follows:

* Create a single file
* Input text into one file
* Appends text to a file
* Read text from one file
* Delete a file
* Create 100 files

By timing how long it does these tasks, we can compare the speed of our new filesystem with the base filesystem of freebsd. Unfortunately, because we were unable to finish implementing the filesystem, we could not test it and get any results from our filesystem.

Below is the result of running the benchmark script on the original filesystem freebsd uses.

## Bugs/Unfinished Work

We were unable to implement most of the planned functions. Specifically, we were unable to pin down how to solve a segmentation fault caused by storing blocks from the file in arrays during hello\_create, during the part where we attempt to create a file that does not exist. This issue stopped us from being able to test other aspects of our code, as without creating a file in FILE\_FS, we could perform any of the other actions in FILE\_FS. Because we were planning on reusing much of the code from the create function for other functions that required access to multiple blocks, this issue stopped us from being able to implement those functions.