# Internal Design

The file system holds blocks of data. The first block is the superblock which holds data about the rest of the file system. The next 4 data blocks hold inodes which are smaller blocks of data that hold data pertaining to a single file. The 5th block holds the directory which holds data about the organization of all the files. The rest of the blocks hold the actual data of the files. Generally when a process wants to get ahold of data, its file descriptor located in one of its tcbs will have an entry which has data mapping to a file entry in the file table structure. Which is made up of the inodes. The inodes have data that maps to the file.

# SuperBlock Specification

The superblock is a block of information that describes the FileSystem and it’s variables. The superblock is responsible for reading from the disk and making sure that there are no errors read from the disk about the FileSystem. If it detects an error, it will format the block and all the Filesystem.

# Inode Specification

The purpose of the Inode is to hold 11 direct block data pointers and 1 indirect pointer to a block of pointers to data. The Inode also includes the length of the file, the number of filetables that are using this inode, a flag that indicates use. It can store 16 inodes in one block.

# Directory Specification

The directory holds info about all the files and keeps track of which files are being used. Since the directory is a file info about it is stored in inode 0 of disk block 1.

# FileTable Specification

This class holds a list of filetable entries. Each of these entries represent one file descriptor. It is responsible for allocating and removing file entries from a vector when needed.

# FileSystem Specification

This class will preform any disk manipulations. It provides an easy to user list of operations that users can use. The basic functions are format, open, write, read, delete, seek, and close. This is the overarching class that will create the other classes in the filesystem.

# Performance

We ran our File System against the provided test and it passed all the test’s test. There are no actual numbers regarding how fast the performance is because there were too many factors that could have affected the speed. However when comparing it to the provided .class it will run similarly and seems to about the same amount of time.

# Current functionality, and possible extend functionality

Right now files can be create,written to, and read from. Files can be written to and read from any point within the data of the file. Space allocation is set for a certain size of files but can change if the file is large enough.

Possible Extends could be to have a user interface to make using the current functions easier. That is add a gui. We could also add services for other software the O.S might encounter.

# Assumptions/Limitations

One limitation is the size of the file system. There are only 1000 blocks of 512 bytes. So the system can’t handle larger amounts of data. This is also limited by the fact that the inode only has one indirect and 11 direct data block. Another limitation is parent and child processes will share the same files so writing to a file will affect both processes. Additionally, there is no mechanism to duplicate files. We are limited by 4 modes of functionality.

We are assuming that the threads that need to write and read to the files know what point they need to this at in our file system. So we are assuming the threads know about the structure of the file system.