Akash Sheth - AOS Assignment 6 – Report

Description of implementation of functions

1. fs_create()

→ In this function, before creating a function, it checks whether the file already exists or not. If it exists, error message 'File already exists' is given.
If the file does not exist, it will get memory for the inode and assign file attributes like inode Id, type of the file (file or directory), device id, etc.
And then the inode is put back in the memory. It means that an empty file has been created as an inode represents a file.
After the inode is created, it is linked with root directory of the file system fsd and the number of inodes used is incremented to keep a track.

2. fs_open()

→ While opening a file, the function first checks if the file exists or not. If it does not exist, error message 'File not found' is given.
If the file exists, it's inode is retrieved from the inode block and is put in the filetable with state set to FSTATE_OPEN. Other attributes like fileptr, mode of the file is also set and the inode_number is then returned.

3. fs_write()

→ This function first checks whether the file is open and has proper permission to write in it. If it is closed or does not have privileges, appropriate message is displayed.
If it is open and has permission to write on it, it will start checking for the free blocks from the bitmask. Once it gets a free block, it will call bs_bwrite method to write the buffer passed as parameter to the write function. As the maximum size of a data block is 512bytes, data is written block by block in multiples of 512bytes or less. Once all the bytes are written, it returns the number of bytes written. fileptr keeps track of the offset from where to start writing to a block. Blocks in which the data is written will be maintained in 'blocks' array in the file table.

4. fs_seek()

→ Once the buffer is written to the data blocks, fileptr points to the end of the buffer. A file is always read from the starting. So, the fileptr should point to the start of the file.
fs_seek function brings pointer to the start of the file.

5. fs_read()

→ This function first checks whether the file is open and has proper permission to read from it. If it is closed or does not have privileges, appropriate message is displayed.
If the file is open and has permission to read from it, it will fetch the block_no from 'blocks' array of the inode_number from the file table and read from the block by calling bs_bread.
As the data is written block by block of 512bytes, it will read in a similar fashion and update the buffer shared via pointer. fileptr keeps track of the offset while reading the file.

6. fs_close()

→ This function will check whether the file is closed or not. If it is closed already, it will give error message as 'File already closed'. If the file is open, it will set state of the file to FSTATE_CLOSED and the inode is put back in the inode block.

2. Lessons learned

→ How files are created, how files are represented in an operating system, how files are linked with one another, how read/write access over a file is managed, how files are written in the memory, what inodes are. Overall, a simple file management.