File System - Report

Brief description of implemented code

File Create:
Takes 2 input parameters:
1st the name of the file to be created, 2nd the mode of the file which in our case is 11.

The Create method is used to create our file in our implementation. But it however takes up the globally initialize file table and then checks for the next available slot in the memory. If all the blocks are filled, it returns error else it assigns the next available slot to file descriptor and starts initializing the directory structure and inodes.
The inode numbers starts with 0.
It then updates the directory structure with the number of entries and direct structure array.

File Open:
Takes 2 input parameters:
1st the name of the file to be created, 2nd the flags of the file which tell us to open in read mode, write mode or read/write mode.

The read method searches for the file descriptor from the file name passed using the file table array structure, if found then updates the state and flag else returns error.
In the end, the file (with the name that has been passed as an argument) is opened.

File Write
Takes 3 input parameters:
1st the file descriptor, 2nd the buffer which will store the contents of what will be written and 3rd the size parameter.

This method 1st validates the flag to checks the modes, if the file was opened in read only mode then it cannot write to the file so hence it returns error.
Else it takes the fd which is file descriptor and accesses the respective inode and finds out how many blocks of data the inode contains and even the next free blocks using the function getmaskbit().
It then sets the mask bits to update the data blocks used and returns the number of bytes which the method has written.

File Close:
Takes only 1 input parameters which the file descriptor fd.

It changes the state of the file table by accessing the global filetable array through the file descriptor and this is how the file gets changed.

File Read:

Takes 3 input parameters:
1st the file descriptor, 2nd the buffer in which the data will be copied and 3rd is the number of bytes written.

The File descriptor fd is used to retrieve the inode structure of the file and is also used to calculate the number of blocks used by the file. It then uses the inode's structure to get the data from the memory block and copy in the buffer which is the 2nd argument. This way it reads through all the data blocks for the file and finally copies in to the buffer and is then displayed using the same buffer.

File Seek:

Takes 2 input parameters:
1st the file descriptor, 2nd is the number of bytes written rval.
Once the whole file is read, the file pointer will be present at the End of the File. So, to read the file again we must move the file pointer to the starting of the file to read it correctly. This is done by adding the file pointer value and negative of the offset value which is passed in the argument as rval. This way the file pointer will again point to the start of the file.

Lessons learnt

This assignment was a good one to learn having required to understand lots of functionality which we haven't implemented but had to understand the flow to understand the working of the file system.
We learnt the steps required from starting of file creation, file opening, file write and closing a file. Each function requires an implementation of certain structure to store all the metadata of the files and using the same to modify the files based on its parameters like if the file type is read or write or read/write etc. The understanding of each method is already written above.

Contributions

Abhishek Naik wrote – Create, Open and Read functions
Sagar Vora wrote – Seek, Write and Close functions.
However, we both sat together to understand all the functions and discussed how we will implement them.