OS HW3

Zafer Altay

In this project an i-node based file system was created. I-nodes are structures that hold file attributes and data blocks.

An entry is 16 bytes hold 14 bytes of file or directory name and 2 bytes hold the i-node id. When a file is added to the file system, an entry is created and added to a block. This block is filled with entries indicating many files of the same level. The starting point is the root directory. I keep 2 directories in each entry. One for inode numbers and one for directory names.If it is added under an directory, I add it's name and inode number to the arrays inside the parent first. Then I create that new entry and add it to the datablock section.

For example, we added a folder called abc under the root folder.First, I create the entry for abc and add it to the data block.Then I add abc's name and abc's number to the arrays I keep in the root entry.Thus, it is easy to reach the folders's name and inode numbers under the directory.

A free space management partition showing empty and full spaces is required. I used two bitmaps this partition in the structure I designed. These two bitmaps hold all inodes and blocks. It indicates whether it is used or free with 1 and 0.

The disk also has a super block containing the necessary information. Super block is the first block on disk.

Inodes are in the structure we know. It has a unique number and keeps the blocks of the relevant entry.

In the project I set it to 8MB file size and 100 inodes. If a change is required, it can be easily corrected via .h files.

Structure:

```
#define FILE_SIZE 8388608
#define TOTAL_BLOCKS 13
#define NUMOFINODE 100


struct SuperBlock
{
    int inodeSize;
    int inodeStartPos;

    int blockSize;
    int numberOfBlock;
    int blockStartPos;

    int startDatasPos;
    int startFreeSpaceManagPos;

    int inodeFreeListStartPos;
    int blockFreeListStartPos;

    int sizeDirectory;

};



struct INode
{
    int inodeNum;
    int fileSize;
    time_t creationTime;
    time_t lastModificationTime;
    char fileName[14];
    int fileType;       // 0 for regular files, 1 for directory files
    int blockNumbers[TOTAL_BLOCKS];

};


struct DirectoryEntry
{
    int inner;
    int inodeNum;
    char directoryName[14];
    int inodeNumbers[10];
    char directoryNames[10][14];
};
```

PART3:

void myMkdir(FILE * fp, char * filesystem, char * op, char * path)

void myDir(FILE * fp, char * filesystem, char * op, char * path)

void myDumpe2fs(FILE * fp, char * filesystem, char * op)