CSE 312/504 OPERATING SYSTEMS

HOMEWORK 4 REPORT

PART 1

In designed my superblock like that:

```
|   blockSize   |   firstValidAddresss   |   rootDirectoryEntry   |
```

And I designed my directory entry like that:

```
|   name   |   attribute   |   date   |   time   |   firstBlockAddress   |   size   |
```

Size holds entry count for directories.

And the content of my file is like this:

```
---------------------
superBlock
---------------------
fatAddressingBlocks
---------------------
blocks for the usage of
the files/directories
---------------------
```

And the design of file allocation table (FAT) is like that (I denoted the free blocks with -2):

freeBlock:	0	-2
firstBlockOfA:	1	6
freeBlock:	2	5
firstBlockOfB:	3	2
	4	-2
tailOfB:	5	-1
	6	7
tailOfA:	7	-1

In fact, first blocks are fat addressing system are reserved by the superblock and fat addressing blocks, so that these blocks cannot be taken by any file/directory, and we can handle this situation by firstValidAddress flag which is in the superblock.

PART 2

What I all did was just to implement what I explained in the first part.

PART 3

dir

In this command, I went down in the file till I fo und the folder to be listed.

mkdir

Here I also went down till I found the folder to create a folder inside, then I increased the entry size of the parent folder, and put the new folder by taking a block from the fat allocation table. When I couldn’t find the path to create the directory, I informed the user and exited the program.

rmdir

Here I did almost the same with mkdir, but here I decreased the entry size of the parent, released the fat block and I moved the last entry to the removed index, to keep them sequential.

write

Here firstly, If I couldn’t open the file to write, I informed the user and exited the program. Unless, I went down to the required level with the path input entered. Then I created the file In the file system, allocated adequate number of blocks (size of the file/block size) to the file and linked it to the parent directory.

read

Here I did the inverse of write; I read the file from file system, traversed all the occupied blocks one by one and wrote them to the denoted file block by block.

del

Here what I did was just delete the file from the file system and decrease the size of the parent directory.

dumpe2fs

Here I traversed all the file system with depth first order and wrote the address of all occupied blocks to the console for eack file/directory.