CSE 312/504 Operating Systems

Abdullah Çelik

171044002

Table of Contents

1. Requirements

- ✓ Creating general structure of file system
- ✓ Preparing a report
- ✓ Creating disk file system image is done (part2)
- ✓ Operating a few commands is done (part3)

2. Design Approach

There are three main classes: **SuperBlock**, **Directory**, and **FAT12**. Each classes is explained related chapter.

FAT12 class is where everything starts to initialize. It handles the initialization of the FAT (File Allocation Table), free block information, super block, and linear disk space.

The FAT12 class includes several important methods.

- **mount()** method reads the disk file and stores the information in the "disk" variable.
- **unmount()** method writes the information from the "disk" variable back to the disk file.
- **readbyte()** method allows us to read a specific block by providing the block number, buffer, and size as arguments. This method reads the specified block from the file and writes it to the given buffer.
- **writeblock()** method enables us to write information to a specific block by providing the block number, buffer, and size as arguments. This method writes the buffer to the specified block as requested by the calling function.

- **command()** method plays a crucial role in directing user commands received through the terminal to the corresponding methods, such as mkdir, dir, and dumpe2fs, and executing them.

3. Disk Structure

In order to handle and control mechanical secondary storage disks, we need to organize the disk's cylinder structure in a linear fashion. Additionally, it is necessary to divide the linear storage into sectors and blocks.

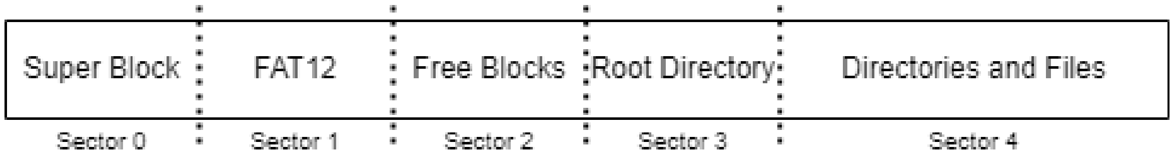
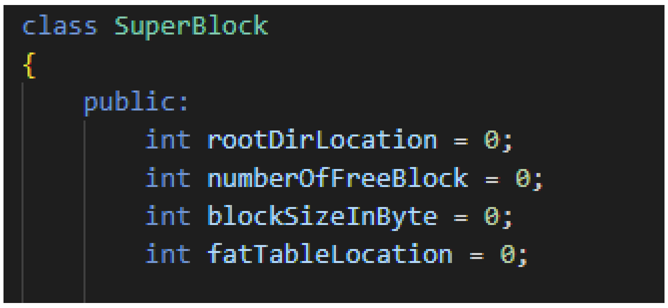


Figure 1: Disk space

a. Super Block

Manages super block properties.



```
class SuperBlock
{
    public:
        int rootDirLocation = 0;
        int numberOfFreeBlock = 0;
        int blockSizeInByte = 0;
        int fatTableLocation = 0;
```

Figure 2: Super block properties

rootDirLocation: Keeps root directory location in disk.

numberOfFreeBlock: Number of free block in the file system.

blockSizeInByte: How many bytes is a block.

fatTableLocation: Keeps FAT12 table location in disk.

b. FAT (File Allocation Table)

Keeps file blocks records. It is form of map. First element of map is wheter busy or not. Second element is pointed block number.

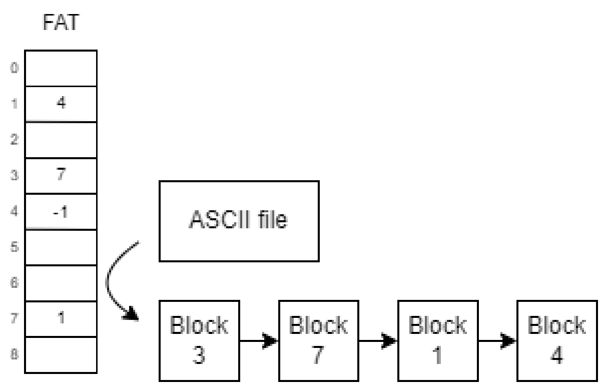


Figure 3: Ascii file in terms of blocks

c. Free Blocks

Keeps free blocks in the file system. Managed bit set data structure.

d. Root Directory

This space is specifically allocated for the root directory. Initially, this area is empty. When you perform file operations, the root directory keeps records of folders. All files and directory records are saved in the remaining space on the disk.

4. Directory Structure

FAT-12 entry structure is used.

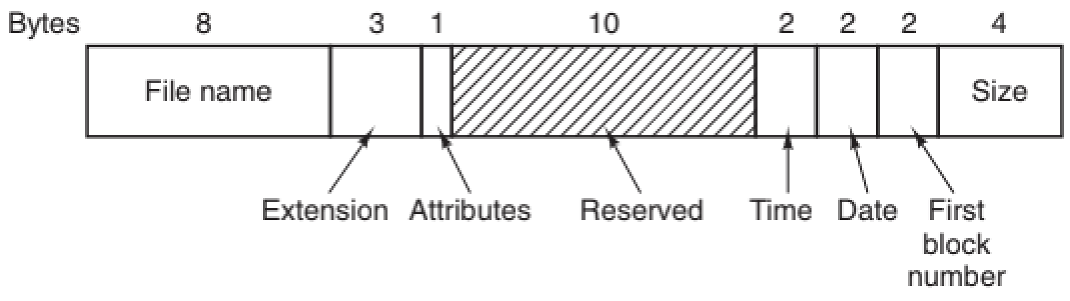
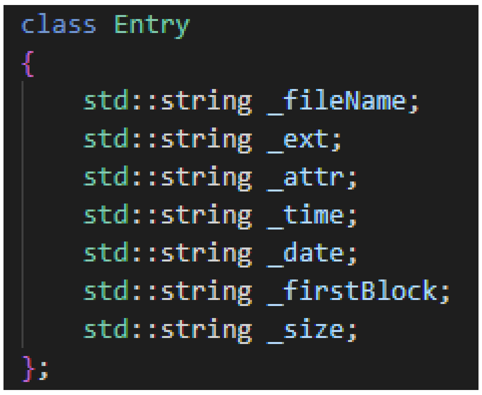


Figure 4: FAT-12 entry structure

Unlike the above structure, I don't use reserved space.



```
class Entry
{
    std::string _fileName;
    std::string _ext;
    std::string _attr;
    std::string _time;
    std::string _date;
    std::string _firstBlock;
    std::string _size;
};
```

Each directory is identified by a directory number and name. Additionally, the Directory class contains a list of file entries (list<Entry*>) and subdirectory lists. Moreover, the

Directory class includes an entry class that manages the file entries, as shown in the figure above.