

## Introduction

This report provides a comprehensive explanation of a simplistic version of the FAT12 file system designed and implemented as a binary file. The FAT12 file system, a legacy from the MS-DOS era, utilizes a directory structure, a File Allocation Table (FAT), and a superblock for storing crucial metadata. Our simplified version mimics these key components to illustrate the primary elements of a file system.

## File System Design

### Directory Table and Directory Entries

In our design, we employ a struct to encapsulate each directory entry. Each struct consists of several fields such as:

- Filename: An array of 8 bytes, adhering to FAT12 standards.
- Extension: An array of 3 bytes, also as per FAT12 standards.
- Attribute Byte: A single byte set to 0x20, indicating the entry as a file.
- Last modification date and time: Both represented as 2-byte fields.
- Starting Block: A 2-byte field denoting the starting block of the file.
- Size: A 4-byte field showing the file's size in bytes.

The Directory Table is an array of these struct entries stored in one or multiple contiguous blocks, contingent upon the maximum number of entries we wish to permit in a directory.

### Free Blocks Management

To manage free blocks, our file system uses the File Allocation Table (FAT). The FAT operates as a mapping table for the disk space, with each entry corresponding to a block on the disk.

The value of a FAT entry signifies the status of the block it represents:

- If the entry value is 0x000, the corresponding block is free.
- If the value is 0xFF7, the block is marked as bad.
- Other values will point to the next block of a file, and a value of 0xFFFF signifies the end of a file.

### Superblock Design

The superblock is an essential component storing significant file system metadata:

- Block size: 2-byte field indicating the size of a block in bytes.
- Number of blocks: 4-byte field showing the total number of blocks in the filesystem.
- Root directory position: 2-byte field representing the root directory's position in the FAT.
- FAT position: 2-byte field indicating the FAT's position.
- FAT length: 2-byte field showing the FAT's length.

## File System Creation Program

The design is implemented in a C++ program that creates an empty file system as a binary file, with a maximum size of 16 MB. This file holds all the information about the file system, including the super block, data blocks, free blocks, directories, and data.

Running the program is straightforward. The user provides the desired block size (in KB) and the name of the file system file. For instance, `makeFileSystem 4 mySystem.dat`, creates a file system with a block size of 4KB named `mySystem.dat`.

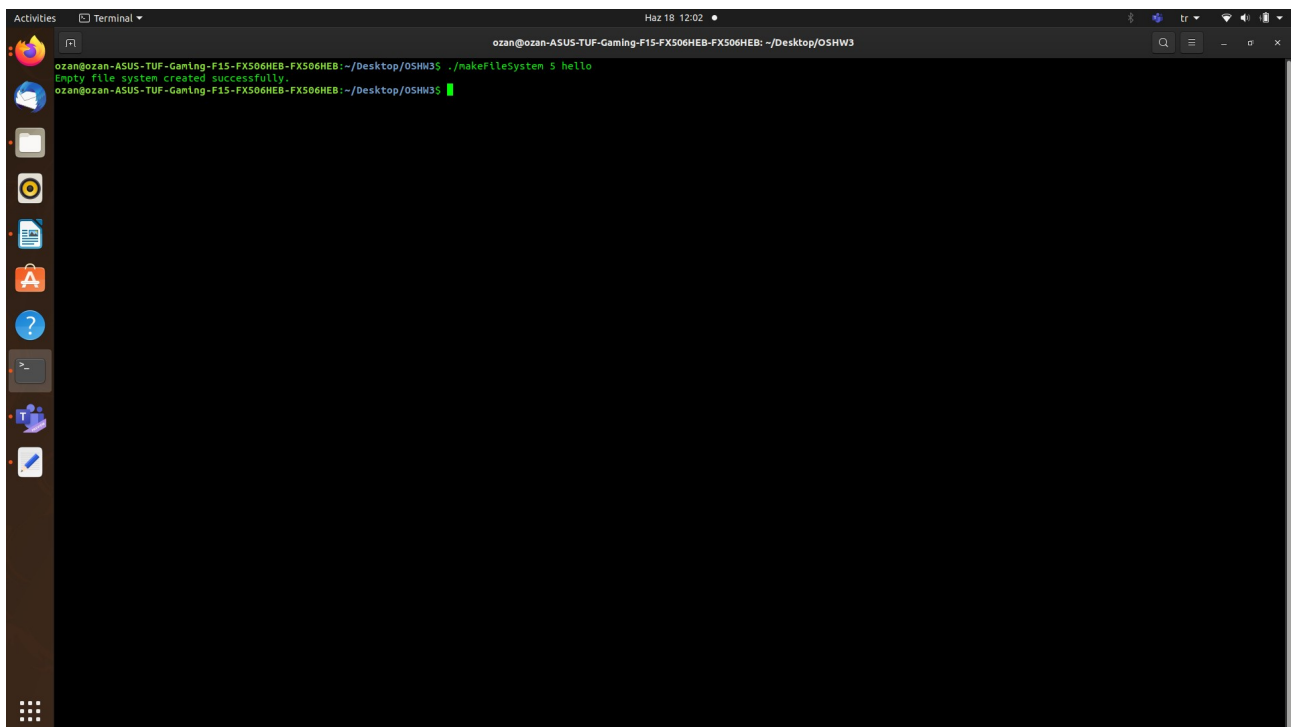
The program initializes an empty file system with a single root directory. All blocks are marked as free in the FAT. The created file is a binary file that follows the structure defined in our design. As a binary file, it cannot be opened with a regular text editor, as it is not human-readable. A hex editor or a purpose-built tool would be required to interpret the file contents accurately.

Please note that this implementation is simplistic and omits many real-world considerations, such as error handling, file system integrity checks, or block and directory caching for performance.

## Conclusion

This report presents a simplified design of a FAT12-like file system. The design includes a superblock, directory entries, a directory table, and a File Allocation Table (FAT) to manage free blocks. The corresponding C++ program illustrates the creation of such a file system as a binary file, with all blocks initially set to free. This exercise offers a basic understanding of how file systems are structured and operate. Further improvements and additions could be introduced to make the file system more robust and functional, reflecting real-world applications.

### TEST:

A screenshot of a Linux terminal window. The window title is "Terminal". The user is "ozan" and the host is "ozan-ASUS-TUF-Gaming-F15-FX506HEB-FX506HEB". The current directory is "~/Desktop/OSHW3". The terminal shows the command `./makeFileSystem 5 hello` being executed, followed by the output "Empty file system created successfully." and a new prompt. The terminal has a dark background and a light-colored cursor. The window is part of a desktop environment with a sidebar on the left containing various application icons.