FILE SYSTEM

Elias Cherkaoui

There is no organization to the FAT folder structure, and files are given the first available location on the volume. The starting cluster number is the address of the first cluster used by the file. Each cluster contains a pointer to the next cluster in the file, or an indication (0xFF) that this cluster is the end of the file.

The layout of a FAT32 filesystem is simple. The first sector is the ID, which is followed by some unused space which are called reserved sectors. After the reserved sectors are two copies of the FAT. The rest of the filesystem is data arranged in clusters. The root directory is stored in the clusters following the ID's.

Disks follow a layout containing:
The Boot Block, FAT, Root Dir, clusters.

Boot Block holds some special instructions which are used to load the operating system into memory. And it also contains data which help describe how the rest of file system looks.

FAT works in this way, there is one entry in the FAT for every cluster (data block) on the disk. Every entry relates to the same cluster. The first byte of the first entry is the media descriptor byte and the second byte is 0xff. Both bytes in the second entry should be 0xff.

For simplified explanations it you can basically say that FAT is a list of blocks, where each block can contain some set amount of bytes. FAT keeps track of each of these blocks and if any of blocks contain data it will relate the block with it's corresponding file. (Basically maps the file info to block locations and reads those).

If we want to read a file using FAT we have to know a couple of things, The root directory, starting cluster of file, etc.
Let's say we have the root directory starting at 0x40 and the starting cluster of file is 0x1. We can add 0x40 to 0x1 and get 0x5. If we take 0x5 and check the next 4 bytes we will now get the file size. Now that we have the file size we also imagine how many blocks are required to store the data.

FAT Blocks: These block can have a flag assigned to them, this is so we can tell if the block is used, empty and etc.

Root Directory Region: This region is a directory table that contains the information about the directories and files.

When a new file is created the FAT has to be updated. The FAT should be updated in the root directory of the file, where the filename length and adress that translates to the block of the file. We also should updated the block flag to tell it that it's now being

used. The file block should tell us that it's a file and how big the file is. The FAT also has to be updated on which blocks the file holds.

We can keep the structures in memory and on disk synchronized by updating the FAT when needed.

File System Calls

The system calls are few and include open, write, format, and dump.

Open:
As the name implies, this system call opens or creates a file depending on if it exists or not. It does this by creating a file descriptor, which then can be used by e.g mmap or read() to read from the file. When attempting to use system call open() it will try to locate the file and return flags. These flags can contain information such as read/write permission or if the file exists.

Write:
Say you have used open() on a file and now want to write something to this file. This is where the write() system call comes in. Write() is essentially a buffer of bytes. It takes a few inputs: the file descriptor, buffer, number of bytes. The write() will fill the file with the bytes. However, it's important to note that on this file, the block sizes are limited, passing this size requires another block.

Format:
The purpose of format() call is to clear the file system. This essentially means that the appropriate disk structure has to be written. All files have to be cleared out on the empty file-system. It's kind of like formatting your USB. The formatted file-system should contain appropriate structure and, in this case, being the FAT system, which means FAT blocks. Doing this should result in a directory with 0 entries.

Dump:
This system call is to loop through the directory structure and output information describing the directories and files stored. In cases after using format() syscall we should get a "clean" dump. Which means we should get a line with no directorie(s) or files.