**FileSystem Module Description Document**

**1.Introduction**

The file system is the organizational structure of files on a disk. The main function of this module is to enable normal operations on files stored on the disk. The main application scenario for this file system is music playback, and functions have been added and key parameters optimized for this scenario. All interfaces related to file numbers are designed for music playback, so file numbering only applies to music files. Non-music files are not numbered and cannot be opened using the open function by file number.

**2.Main Features**

* Supports basic operations on FAT16, FAT32, and exFAT file systems;
* File number operations for files and folders specifically designed for screenless music playback;
* Maximum supported directory depth: 8 levels
* Maximum drive capacity of 256TB, with a maximum file size of 4GB per file
* File number operations include accelerated file search
* Accelerated allocation of free clusters
* Supports sorting of up to 2048 files by file name or file creation time

**3.Function Description**

**3.1. Initialization Process**

When the file system is initialized, it reads the MBR (Master Boot Record) sector of the disk, analyzes the DPT (Disk Partition Table) within it to locate and load the first partition. If the disk does not have an MBR, this step is skipped. The DBR (DOS Boot Record) sector is located in the partition table to obtain basic information about the file system. Subsequently, all files and folders within the current partition are traversed to count the total number of song files, folders, etc. Traversing files during file system initialization prolongs the initialization time, but the benefit is that it enables the retrieval of total file and folder counts, allowing the creation of an accelerated node list to speed up access to files located further back in the storage hierarchy.

**3.2. File Search Order**

The numbers assigned to files and folders are arranged according to their respective search order. Understanding the search order facilitates the use and understanding of the entire file system. Folder searches are performed using a depth-first traversal algorithm. File searches are simply folder searches with the addition of traversing the files in the current folder. This order is designed to suit the application scenario of music playback. Figure 1 assumes a file tree structure, where the numbers assigned to files and folders also indicate the traversal order.

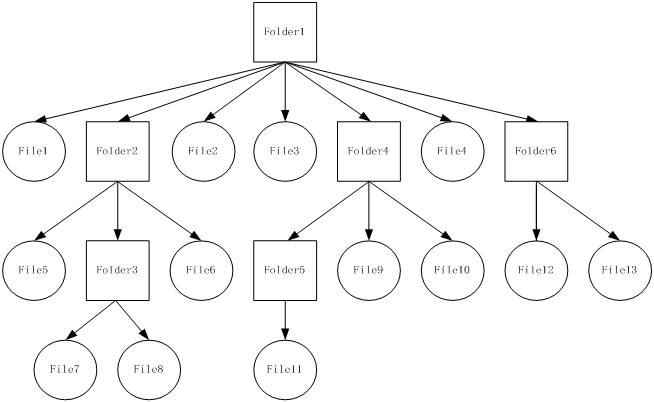


Figure 1 File and folder traversal order

**3.3. Support for exFAT**

Support for exFAT is a feature added in subsequent versions. To unify the underlying processing and API interfaces for different file systems, the approach of converting exFAT to the FAT32 format was adopted. Specifically, whenever an exFAT sector is read into memory, its data organization format is converted to FAT32 format, and subsequent processing uses the same functions and API interfaces as FAT32. Support for exFAT consumes an additional 768 bytes of memory space. If this feature is not required, it can be disabled in the configuration file.

Due to the conversion from exFAT to FAT32, the precision of file creation, modification, and access times in exFAT is reduced to the precision of FAT32 format. Operations related to long file names are not currently supported.

**3.4. Brief Description of Opening File Acceleration Methods**

File opening acceleration applies only to functions that open files by file number; functions that open files by file name are not included in this scope. The acceleration utilizes the acceleration node table established during file system initialization. Through these nodes, the target file's location can be roughly pinpointed, thereby reducing the search range to achieve acceleration. Acceleration nodes are approximately evenly distributed across all files, thereby dividing all files into several regions of equal size. Each node records its own file number and physical sector address information. When searching for a file by file number, the existing node information can be used to roughly lock the target file’s region, allowing the system to skip over previous regions and start searching directly from the node’s position.

It can be seen that the larger the file number, the more time can be saved, which basically balances the time required to open files stored at the beginning and end of the storage location. Otherwise, the time required to open a file would be directly proportional to the size of the file number.