East African Scholars Journal of Engineering and Computer Sciences
Abbreviated Key Title: East African Scholars J Eng Comput Sci
ISSN: 2617-4480 (Print) & ISSN: 2663-0346 (Online)
Published By East African Scholars Publisher, Kenya
Volume-3 | Issue-3 | Mar-2020 | DOI: 10.36349/easjecs.2020.v03i03.05

OPEN ACCESS

Review Article

Fat32 File System Organization and Storage Mechanism

Nguyen Tien Duy[1]
[1]Thai Nguyen University of Technology – Thai Nguyen University, Thai Nguyen, Vietnam

Article History
Received: 08.02.2020
Accepted: 18.03.2020
Published: 26.03.2020

Journal homepage:
https://www.easpublisher.com/easjecs

Quick Response Code

Abstract: Contents of the paper make mention of the physical and logical organization on the hard disk with a FAT32 system. It includes information structures such as Master Boot Record (MBR), Partition Table, DOS Boot Record (DBR), FAT, Root Directory, etc. Meaning of important information fields and their parts in the file FAT system. Base on that, they are used by operating system in the file (folder) management, such as create, delete and update, etc. The article is presented with the purpose of conveying to the readers the necessary information when working (building utility programs) for magnetic disks formatted with the FAT file system.

Keywords: File Allocation Table, Master Boot Record, Partition Table, DOS Boot Record.

INTRODUCTION

In recent times, the rapid development of hard disk drive capacity and BIOS access services have given system developers a new-look at the logical organization of high-quality hard drives (Stallings, W. 2003; Shri Vishnu Engineering College; & https://en.wikipedia.org). In particular, organizing the file system in FAT32 format is a focus of those who care about, who build utility programs for hard disk such as checking the drive parameters, formatting, defragment, deleted data recovery, etc. (Hordeski, M. 1995; & Andrew, S. T. 1996).

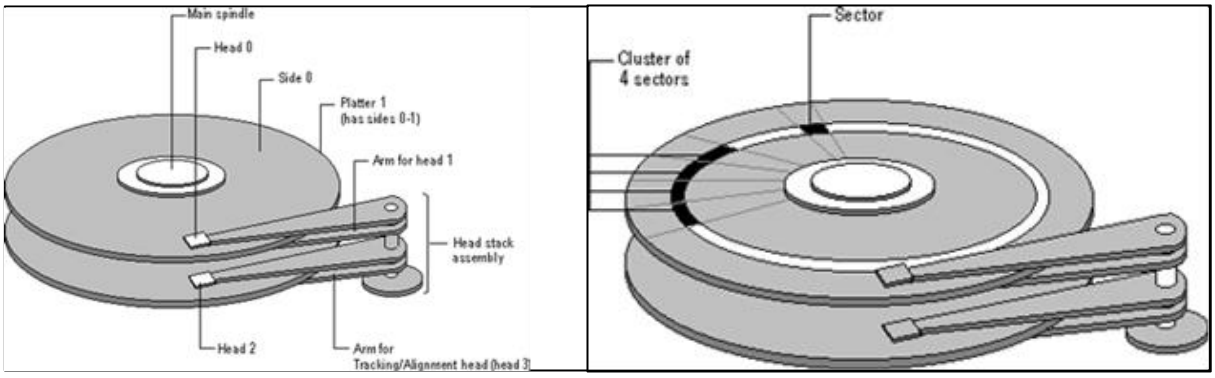

Figure 1. Roller bearing structure

The logical structure of the hard disk in FAT32 format
Master Boot Record (MBR)

The Master Boot Record is created when creating the first partition on the hard disk, this is the most important information area on the disk. It occupies the entire hard disk's first sector, its physical address is always: track (cylinder) 0, side (head) 0, and sector 1.

The Master Boot Record contains a partition table and a small piece of code. On x86-based

***Corresponding Author:** Nguyen Tien Duy

computers, this code identifies the partition as having system properties. MBR finds the location of the boot partition on the disk, loads the code in Boot Sector into memory and transfers control to the code in this Boot Sector (https://en.wikipedia.org/).

Hard drive partition

Information about the main and extended partition is stored in the partition table. The table consists of 4 entries, each of which is 16 bytes in size and located in the MBR. This table has a different structure depending on the operating system. Each entry starts at the offset addresses in the sector as follows:

- Partition 1 0x01BE (446)
- Partition 2 0x01CE (462)
- Partition 3 0x01DE (478)
- Partition 4 0x01EE (494)

The last two bytes of the sector are the operating system signature and are always 0x55AA. The following table describes the content of an entry in the partition table, the example values correspond to the first partition.

Table 1. The content of the entry in the partition table

Byte Offset	Field Length	Sample Value	Meaning
00	BYTE	0x80	Boot Indicator. Indicates whether the partition is the system partition. Legal values are: 00 = Do not use for booting. 80 = System partition.
01	BYTE	0x01	Starting Head.
02	6 bits	0x01	Starting Sector. Only bits 0-5 are used. Bits 6-7 are the upper two bits for the Starting Cylinder field.
03	10 bits	0x00	Starting Cylinder. This field contains the lower 8 bits of the cylinder value. Starting cylinder is thus a 10-bit number, with a maximum value of 1023.
04	BYTE	0x06	System ID. This byte defines the volume type. In Windows NT, it also indicates that a partition is part of a volume that requires the use of the HKEY_LOCAL_MACHINE\SYSTEM\DISK Registry subkey.
05	BYTE	0x0F	Ending Head.
06	6 bits	0x3F	Ending Sector. Only bits 0-5 are used. Bits 6-7 are the upper two bits for the Ending Cylinder field.
07	10 bits	0x196	Ending Cylinder. This field contains the lower 8 bits of the cylinder value. Ending cylinder is thus a 10-bit number, with a maximum value of 1023.
08	DWORD	3F 00 00 00	Relative Sector.
12	DWORD	51 42 06 00	Total Sectors.

Partition Table

Head, Sector, and Cylinder Fields starting and the ending

On x86 computers, the Head, Cylinder, and Sector information fields starting and ending on the boot disk are critical to starting the computer. The part of the program in MBR will use the information in this field to find and load the Boot Sector of the corresponding partition.

The Cylinder field ends in the 10-bit partition table, the start and end number fields are 1 byte long, the Sector field begins and ends 6 bits long.

Hard disk disks are usually low-level formatted from the factory with a sector size of 512 bytes, the maximum disk space described in the partition table can be calculated by the following formula:

MaxCapacity = (sector size)*(sectors per track)*(cylinders)*(heads)

Corresponding is 512 x 63 x 1024 x 256 = 8,455,716,864 bytes or 7.8 GB (≈8 GB)

Sectors and Number of Sectors fields

For a primary partition, the Relative Sectors field represents the deviation from the first sector of the disk to the first sector of the partition in sectors. The Number of Sectors field represents the total number of sectors of the partition. The description of these fields in the extended partition.

Primary Partition, Extended Partition and Logical Drives

Logical Drives and Extended Partitions

When there are more than 4 logical drives on a physical hard disk, the first partition should be used as the primary partition. The second partition can be created as an extended partition and possibly the rest of the disk.

In an extended partition, we can create several logical drives. When we have an extended partition on a hard drive, the entry for this partition in the partition table (the end of the Master Boot Record) points to the first sector of the partition. The first sector of each logical drive in an extended partition also has a partition table, occupying the last 64 bytes of that sector (and 2 bytes marking the sector end).

There are several types of entries in a partition table:

- The first entry for the current logical drive.
- The second entry contains information about the next logical drive in the extended partition.
- Partition 3 and 4 are equal to 0.

This structure is repeated with any logical drive. The last logical drive only lists partition entries. Entries between 2-4 are equal to 0.

The partition table entry only contains information about the first disk, the first cylinder, of each logical drive in the extended partition. The entry for the first partition in each partition table contains the first address of the data area on that logical drive. And the entry for the second partition is the address of the sector that contains the partition table for the next logical drive, etc.

What is FAT?

The FAT file system (File Allocation Table) is a simple file system, originally built for small disks and a simple directory structure. The FAT file system is named after the organization. The file distribution table is located at the beginning of the volume. For safety, FAT is written in 2 identical copies, in case of failure of one table, the other will be used. Each logical drive formatted with the FAT file system will be allocated memory units according to the cluster. The default cluster size depends on disk space.

The figure below shows the logical structure of a volume according to the FAT file system.

Figure 2. Structure of a FAT volume

The table 2 lists a number of different FAT systems

Table 2. Some types of FAT

System	Bytes Per Cluster Within File Allocation Table	Cluster limit
FAT12	1.5	Fewer than 4087 clusters.
FAT16	2	Between 4087 and 65526 clusters, inclusive.
FAT32	4	Between 65526 and 268,435,456 clusters, inclusive.

FAT Partition Boot Sector

The Boot Sector contains information that the file system uses to access the volume. On x86 computers, the MBR uses the Boot Sector on the system partition to load the operating system kernel files. The following table describes the information fields in the Boot Sector with a FAT file system formatted Volume.

Table 3. Information fields in Boot Sector

Byte Offset (in hex)	Field Length	Sample Value	Meaning
00	3 bytes	EB 3C 90	Jump instruction
03	8 bytes	MSDOS5.0	OEM Name in text
0B	25 bytes		BIOS Parameter Block
24	26 bytes		Extended BIOS Parameter Block
3E	448 bytes		Bootstrap code
1FE	2 bytes	0x55AA	End of sector marker

The Table 4 describes the fields in the BIOS parameter block and the expanded BIOS parameter block.

Table 4. BIOS Parameter Block and Extended BIOS Parameter Block Fields

Byte Offset	Field Length	Sample Value	Meaning
0x0B	WORD	0x0002	Bytes per Sector. The size of a hardware sector. For most disks in use in the United States, the value of this field is 512.
0x0D	BYTE	0x08	Sectors Per Cluster. The number of sectors in a cluster. The default cluster size for a volume depends on the volume size and the file system.
0x0E	WORD	0x0100	Reserved Sectors. The number of sectors from the Partition Boot Sector to the start of the first file allocation table, including the Partition Boot Sector. The minimum value is 1. If the value is greater than 1, it means that the bootstrap code is

0x10		BYTE	0x02	Number of file allocation tables (FATs). The number of copies of the file allocation table on the volume. Typically, the value of this field is 2.
0x11		WORD	0x0002	Root Entries. The total number of file name entries that can be stored in the root folder of the volume. One entry is always used as a Volume Label. Files with long filenames use up multiple entries per file. Therefore, the largest number of files in the root folder is typically 511, but you will run out of entries sooner if you use long filenames.
0x13		WORD	0x0000	Small Sectors. The number of sectors on the volume if the number fits in 16 bits (65535). For volumes larger than 65536 sectors, this field has a value of 0 and the Large Sectors field is used instead.
0x15		BYTE	0xF8	Media Type. Provides information about the media being used. A value of 0xF8 indicates a hard disk.
0x16		WORD	0xC900	Sectors per file allocation table (FAT). Number of sectors occupied by each of the file allocation tables on the volume. By using this information, together with the Number of FATs and Reserved Sectors, you can compute where the root folder begins. By using the number of entries in the root folder, you can also compute where the user data area of the volume begins.
0x18		WORD	0x3F00	Sectors per Track. The apparent disk geometry in use when the disk was low-level formatted.
0x1A		WORD	0x1000	Number of Heads. The apparent disk geometry in use when the disk was low-level formatted.
0x1C		DWORD	3F 00 00 00	Hidden Sectors. Same as the Relative Sector field in the Partition Table.
0x20		DWORD	51 42 06 00	Large Sectors. If the Small Sectors field is zero, this field contains the total number of sectors in the volume. If Small Sectors is nonzero, this field contains zero..
0x24		BYTE	0x80	Physical Disk Number. This is related to the BIOS physical disk number. Floppy drives are numbered starting with 0x00 for the A disk. Physical hard disks are numbered starting with 0x80. The value is typically 0x80 for hard disks, regardless of how many physical disk drives exist, because the value is only relevant if the device is the startup disk.
0x25		BYTE	0x00	Current Head. Not used by the FAT file system.
0x26		BYTE	0x29	Signature. Must be either 0x28 or 0x29 in order to be recognized by Windows NT.
0x27		4 bytes	CE 13 46 30	Volume Serial Number. A unique number that is created when you format the volume.
0x2B		11 bytes	NO NAME	Volume Label. This field was used to store the volume label, but the volume label is now stored as special file in the root directory.
0x36	8 bytes	FAT16		System ID. Either FAT12 or FAT16, depending on the format of the disk.

File Allocation System

The file allocation table contains several types of information about each cluster on the volume (see the example below for FAT16).

- Do not use: 0x0000
- Bad cluster: 0xFFF7
- Final cluster of a file: 0xFFF8-0xFFFF
- The cluster is used (of a file) carries any value not equal to the above values.

There is no organization for FAT directory structure and files are given in the first place on the volume. The starting cluster number is the address that is the first cluster used for a file. Each cluster contains a pointer to the next cluster in the file or the value 0xFFFF signals the end of the file.

Example: A cluster chain of some files:

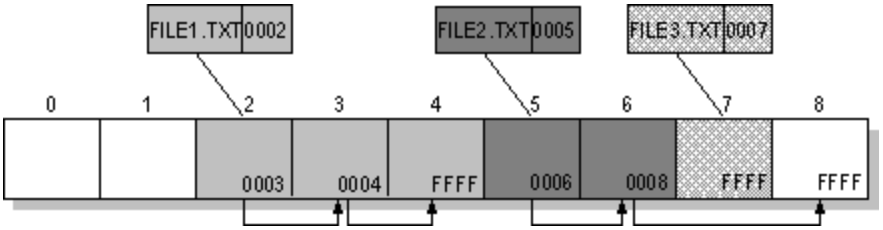

Figure 3. Cluster string of 3 files

As shown in Figure 3, we see there are 3 files. File1.txt file occupies 3 consecutive clusters. The second file: File2.txt is a file and it also takes up 3 clusters. File3.txt file has the smallest size, only occupies 1 cluster. In each case, the directory structure points to the first cluster of the file.

FAT Root Folder

The FAT root directory contains entries that point to each file or directory on the drive. The only difference between the root directory and the other directories are that the root directory is located at a fixed location on the disk and has a fixed size (512 entries for the hard drive, the number of entries on the floppy disk depend on the size of the disk, with a 1.44 MB floppy disk, the number of root directory entries is 224).

FAT Folder Structure

A directory contains a set of entries, each 32 bytes long for each file or a sub-folder. Each entry will contain the following information:

- Name (according to 8.3 standards).
- Attribute byte (each bit carries a piece of attribute information, described later).
- Creation time (24 bits).
- Creation date (16 bits).
- Last access date (16 bits).
- Last modification time (16 bits).
- Last modified date (16 bits).
- The cluster number comes from the FAT table (16 bits).
- File size (32 bits).

The information in the root directory is used for all operating systems that support FAT. Because all entries in the root directory are equal in size, the attribute byte of each entry will indicate the type of the entry. There is a bit that indicates whether the entry is a sub-directory or a file. Usually, only the operating system controls the creation of these bits.

Each FAT file has 4 attribute bits, they can be created/deleted by the user, which are the properties: archive file, system file, hidden file, and read-only file.

Filenames on FAT Volumes

Since Windows NT 3.5, files created or renamed on a FAT volume use attribute bits to support long file names in a way that does not affect MS-DOS and OS/2 access, etc. Although the user creates a file with a long name, Windows creates a "short" file name in accordance with 8.3 for the file. To extend this common entry, Windows creates one or more entries for the file, each containing 13 characters (which are part of) for the Unicode long file name. Windows will set the attributes of the extended entries as part of the long file name. MS-DOS and OS/2 will ignore these entries when accessing because as they are transparent, they only use the information in the first entry (containing the file name in accordance with 8.3).

Example of Folder Entries for the long filename

Figure 4 shows all 4 entries for Thequi ~ 1.fox file, its corresponding long name is The quick brown.fox. The characters in the long name are encoded according to Unicode, so each character in the name occupies 2 bytes in the entry. The attribute field for the long name entry is 0x0F and the attribute field for the short name entry is 0x20.

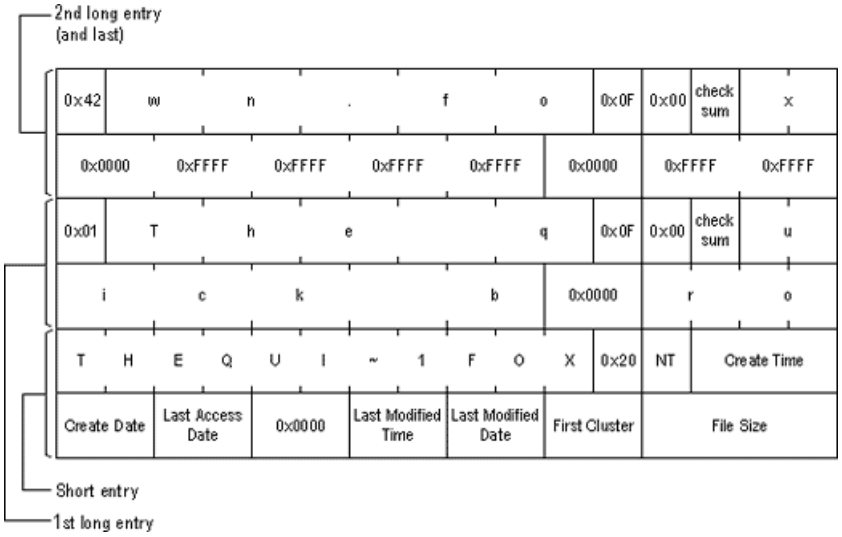

Figure 4. Example of root entry for long file names

Manage files on the hard disk in FAT32 format

File System Specifications

FAT32 is an extension of the FAT file system, it can work on drives over 2 GB. Drives in FAT32 format may have a cluster count of more than 65536, which is much larger than those in FAT16. FAT32 has expanded the management space a lot more than FAT16. With FAT32, a file can be as large as 4GB-2 bytes. FAT32 consists of 4 bytes on an entry in the FAT table, which stores information about the cluster. Of which the highest 4 bits in 32 bits of an element in the FAT32 table are unused and do not belong to the cluster number.

BPB on FAT32

The BPB (BIOS Parameter Block) of FAT32-formatted drives is an extension of BPB to FAT16 / FAT12. It contains the same information as the standard BPB, but it also includes a few extra fields to store specific FAT32 information. Here are some important and commonly used information fields.

Table 5. Fields of BDB

Member Name	Description
A_BF_BPB_BytesPerSector	The number of bytes per sector.
A_BF_BPB_SectorsPerCluster	The number of sectors per cluster.
A_BF_BPB_ReservedSectors	The number of reserved sectors, beginning with sector 0.
A_BF_BPB_NumberOfFATs	The number of File Allocation Tables.
A_BF_BPB_RootEntries	This member is ignored on FAT32 drives.
A_BF_BPB_TotalSectors	The size of the partition, in sectors.
A_BF_BPB_MediaDescriptor	The media descriptor. Values in this member are identical to standard BPB.
A_BF_BPB_SectorsPerFAT	The number of sectors per FAT.
Note: This member will always be zero in a FAT32 BPB. Use the values from **A_BF_BPB_BigSectorsPerFat** and **A_BF_BPB_BigSectorsPerFatHi** for FAT32 media.	
A_BF_BPB_SectorsPerTrack	The number of sectors per track.
A_BF_BPB_Heads	The number of read/write heads on the drive.
A_BF_BPB_HiddenSectors	The number of hidden sectors on the drive.
A_BF_BPB_HiddenSectorsHigh	The high word of the hidden sectors value.
A_BF_BPB_BigTotalSectors	The total number of sectors on the FAT32 drive.
A_BF_BPB_BigTotalSectorsHigh	The high word of the FAT32 total sectors value.
A_BF_BPB_BigSectorsPerFat	The number of sectors per FAT on the FAT32 drive.
A_BF_BPB_BigSectorsPerFatHi	The high word of the FAT32 sectors per FAT value.
A_BF_BPBExtFlags	Flags describing the drive. Bit 8 of this value indicates whether or not information written to the active FAT will be written to all copies of the FAT. The low 4 bits of this value contain the 0-based FAT number of the Active FAT, but are only meaningful if bit 8 is set. This member can contain a combination of the following values.
Value	**Description**
BGBPB_F_ActiveFATMsk (000Fh)	Mask for low four bits.

BGBPB_F_NoFATMirror (0080h)	Mask indicating FAT mirroring state. If set, FAT mirroring is disabled. If clear, FAT mirroring is enabled. Bits 4-6 and 8-15 are reserved.
A_BF_BPB_FS_Version	The file system version number of the FAT32 drive. The high byte represents the major version, and the low byte represents the minor version.
A_BF_BPB_RootDirStrtClus	The cluster number of the first cluster in the FAT32 drive's root directory.
A_BF_BPB_RootDirStrtClusHi	The high word of the FAT32 starting cluster number.
A_BF_BPB_FSInfoSec	The sector number of the file system information sector. The file system info sector contains a BIGFATBOOTFSINFO structure. This member is set to 0FFFFh if there is no FSINFO sector. Otherwise, this value must be non-zero and less than the reserved sector count.
A_BF_BPB_BkUpBootSec	The sector number of the backup boot sector. This member is set to 0FFFFh if there is no backup boot sector. Otherwise, this value must be non-zero and less than the reserved sector count.
A_BF_BPB_Reserved	Reserved member.

CONCLUSION

We can see that the complex management of the FAT file system has brought about efficiency, namely: saving device memory, high access speed, ... However, the complexity of FAT's complexity is a major hindrance to those who need to work deeply with magnetic discs in this format.

Building efficient, concise, highly useful utility programs for hard drives is one of the permanent aims of system programmers, IT students or others interested in this area. And there's no other way we need to find out the nature of storing information on the hard drive - the object we're studying.

For example, a common problem for computer users is data loss (often due to a wrong deletion). When users want to recover their important data in many ways. If you understand the organization of the FAT file system, you can build a utility program "recover data" with a certain degree of efficiency. Thinking, it would be very valuable to recover the deleted data even though the probability of success is not absolute.

Acknowledgments

This research was funded by the science fund of the Thai Nguyen University of Technology.

REFERENCE

1. "Design of the FAT file system", https://en.wikipedia.org/wiki/Design_of_the_FAT_file_system
2. Ahson, S.I. (1984). "Microprocessor with application in Process Control", Tata Mc.Graw Hill.
3. Andrew, S. T. (1996). Modern Operating Systems, Prentice Hall,.
4. Groover, M. P. (2016) Automation, Production System and Computer Integrated Manufacturing, Prentice Hall.
5. Hordeski, M. (1995). Personal Computer Interfaces, Mc. Graw Hill.
6. Olsson, G., & Piani, G. (1990). "Computer Systems for Automation and Control", Prentice Hall.
7. Shri Vishnu Engineering College for women:: Bhimavaram Department of Information Technology, "Computer Organization and Architecture lecture notes".
8. Stallings, W. (2003). Computer Organization and Architecture Designing for Performation; Prentice Hall.