Rose-Hulman Institute of Technology
Computer Science and Software Engineering
Prof. Archana Chidanandan

An overview of FAT12

The File Allocation Table (FAT) is a table stored on a hard disk or floppy disk that indicates the status and location of all data clusters that are on the disk. The File Allocation Table can be considered to be the "table of contents" of a disk. If the file allocation table is damaged or lost, then a disk is unreadable

In this document, the FAT12 file system is described. The FAT12 is the file system on a floppy disk. The number “12” is derived from the fact that the FAT consists of 12-bit entries.

The storage space on a floppy disk is divided into units called sectors. In larger storage devices, a bunch of sectors form a cluster. However, for the floppy disk, the number of sectors in a cluster is one. Also, the size of a sector (and hence a cluster) is 512 bytes for a floppy disk.

1. Disk organization

A floppy disk layout (FAT-12) consists of four major sections: the boot sector, FAT tables, root directory, and data area:

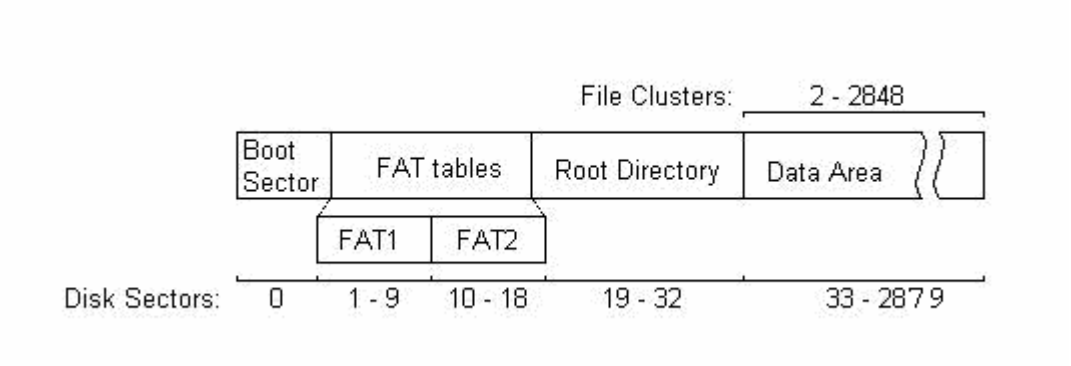

Figure 1 Disk organization of the FAT12 file system[1]

- The boot sector consists of the first sector (sector 0) on the volume or disk. The boot sector contains specific information about the rest of organization of the file system, including how many copies of the FAT tables are present, how big a sector is, how many sectors in a cluster, etc.
- FAT tables contain pointers to every cluster on the disk, and indicate the number of the next cluster in the current cluster chain, the end of the cluster chain, whether a cluster is empty, or has errors. The FAT tables are the only method of finding the location of files and directories on the rest of the disk. There are typically two redundant copies of the FAT table on disk for data security and recovery purposes. On a floppy, since a cluster consists of just one sector, there is a FAT entry pointer to every sector on the disk.
- The root directory is the primary directory of the disk. Unlike other directories located in the data area of the disk, the root directory has a finite size (For FAT12, 14 sectors * 16 directory

[1] The figure has been obtained from the CS324 course web-site from Brigham Young University.

entries per sector = 224 possible entries), restricting the total amount of files or directories that can be created therein.

- Data Area. - The first sector or cluster of the data area corresponds to cluster 2 of the file system (the first cluster is *always* cluster 2). The data area contains file and directory data and spans the remaining sectors on the disk.

A summary of the disk organization is given below:

Logical Sector	**Content**
0	Boot Sector
1	First sector in the (first) FAT
10	First sector in the second FAT
19	First sector in the floppy disk's root directory
XX	Last sector in the root directory (see bytes 17 and 18 in the boot sector)
XX + 1	Beginning of data area for the floppy disk

For FAT12, XX = 32 as 14 sectors are reserved for the root directory.

2. The Boot Sector

The boot sector exists at sector 0 on the disk and contains the basic disk geometry, which is the set of information needed by the operating system to use the disk correctly. Whenever the disk is used, the information from the boot sector is read and any needed information is extracted from it. The boot sector on a DOS formatted floppy is a sequence of bytes that looks as follows:

Starting byte	Length (in bytes)	Stored data
0	11	Ignore
11	2	Bytes per sector
13	1	Sectors per cluster
14	2	Number of reserved sectors
16	1	Number of FATs
17	2	Maximum number of root directory entries
19	2	Total sector count [a]
21	1	Ignore

22	2	Sectors per FAT
24	2	Sectors per track
26	2	Number of heads
28	4	Ignore
32	4	Total sector count for FAT32 (0 for FAT12 and FAT16)
36	2	Ignore
38	1	Boot signature[b]
39	4	Volume id [c]
43	11	Volume label[d]
54	8	File system type (e.g. FAT12, FAT16)[e]
62	-	Rest of boot sector (ignore)

a. Total sector count - This field is the 16-bit total count of sectors on the volume. This count includes the count of all sectors in all four regions of the volume. For FAT12 and FAT16 volumes, this field contains the sector count. For FAT32, see bytes 32-35.

b. Boot signature - Extended boot signature. This is a signature byte that indicates that the following three fields in the boot sector are present. The value should be 0x29 to indicate that.

c. Volume id – Also the Volume serial number. This field, together with Volume label, supports volume tracking on removable media. These values allow FAT file system drivers to detect that the wrong disk is inserted in a removable drive. This ID is usually generated by simply combining the current date and time into a 32-bit value.

d. Volume label - This field matches the 11-byte volume label recorded in the root directory. NOTE: FAT file system drivers should make sure that they update this field when the volume label file in the root directory has its name changed or created. The setting for this field when there is no volume label is the string “NO NAME ”.

e. File System type - One of the strings “FAT12 ”, “FAT16 ”, or “FAT ”. NOTE: Many people think that the string in this field has something to do with the determination of what type of FAT—FAT12, FAT16, or FAT32—that the volume has. This is not true. This string is informational only and is not used by Microsoft file system drivers to determine FAT type because it is frequently not set correctly or is not present. This string should be set based on the FAT type though, because some non-Microsoft FAT file system drivers do look at it.

3. FAT (File Allocation Table)

The FAT, as stated earlier, is a data structure that maps the data sectors of the storage device. It is similar to an array and each entry in the FAT corresponds to a cluster of data on the disk. The values in each entry of the FAT that are of interest are:

- A value signifying that this data cluster is the last cluster of a file
- A value signifying that this data cluster is currently unused
- A value signifying where the NEXT data cluster of the current file is located.

Specifically, the FAT entry values signify the following:

Value	Meaning
0x00	Unused
0xFF0-0xFF6	Reserved cluster
0xFF7	Bad cluster
0xFF8-0xFFF	Last cluster in a file
(anything else)	Number of the next cluster in the file

Translation from physical to logical data sector numbers:

The FAT works off logical data sector values. For the FAT12 system, while determining the logical sector number from the physical sector number, the following two factors need to be taken into account.

- From the organization of the disk, it is seen that the first 33 sectors are predefined. The actual data sector that holds user data does not exist in these first 33 sectors and starts at sector number 33 (remember we start with 0).
- The entries in positions 0 and 1 of the FAT are reserved. Therefore, it is entry 2 of the FAT that actually contains the description for physical sector number 33.

Therefore, **physical sector number = 33 + FAT entry number - 2**

For example, entry 5 of the FAT would actually refer to physical data sector number 36.

4. Directories

Directories (such as the root directory) exist like files on the disk, in that they occupy one or more sectors. Each sector (512 bytes) of a directory contains 16 directory entries (each of which is 32 bytes long). Each directory entry describes and points to some file or subdirectory on the disk. Thus, the collection of directory entries for a directory specify the files and subdirectories of that directory.

Each directory entry contains the following information about the file or subdirectory to which it points.

Offset (in bytes)	Length (in bytes)	Description
0	8	Filename (but see notes below about the first byte in this field)
8	3	Extension
11	1	Attributes (see details below)
12	2	Reserved
14	2	Creation Time
16	2	Creation Date
18	2	Last Access Date
20	2	Ignore in FAT12
22	2	Last Write Time
24	2	Last Write Date
26	2	First Logical Cluster
28	4	File Size (in bytes)

Note: We have already established that in the FAT12 system a cluster holds just one sector. Therefore, the two words are used interchangeably in the rest of this document.

Notes on directory entries:

1. The First Logical Cluster field specifies where the file or subdirectory begins. Thus the directory entry *points to* a file or subdirectory. Note that it gives the value of the FAT index. For example, if the First Logical Cluster value is "2", then it implies that the index to the FAT array should be "2", which is physical cluster "33" in the FAT12 system. If the value of the First Logical Cluster is "0", then it refers to the first cluster of the root directory and that directory entry is therefore describing the root directory. (Keep in mind that the root directory is listed as the ".." entry i.e. the parent directory in all its sub-directories.)
2. If the first byte of the Filename field is 0xE5, then the directory entry is *free* (i.e., currently unused), and hence there is no file or subdirectory associated with the directory entry.
3. If the first byte of the Filename field is 0x00, then this directory entry is free and all the remaining directory entries in this directory are also free.
4. The Attributes field of a directory entry is an 8-bit quantity where each bit refers to an attribute (property) of the file or subdirectory pointed to by this directory entry, as follows:

Bit	Mask	Attribute
0	0x01	Read-only
1	0x02	Hidden
2	0x04	System
3	0x08	Volume label
4	0x10	Subdirectory
5	0x20	Archive
6	0x40	Unused
7	0x80	Unused

   a. If a bit in the Attributes field is set (i.e., is 1), that means that the file or subdirectory to which this directory entry points has the attribute associated with that bit. For example, if the Attributes field is 0001 0010, then the file/subdirectory pointed to by this directory entry is a hidden subdirectory. (Bit 1 is on, indicating that it is *hidden*. Bit 4 is also on, indicating that it is a *subdirectory* and not a file. Remember, bits are numbered right-to-left.)
   b. If the Attributes byte is 0x0F, then this directory entry is part of a long file name and can be ignored for purposes of this assignment. (The updated version of the Microsoft white paper on FAT systems includes details about long file names in FAT12, if you want to deal with them.)

5. The formats for the time and data fields are specified in a Microsoft white paper on FAT systems. (But you won't need to know those formats for this assignment.)
6. The directory entry specifies where the file or subdirectory starts (First Logical Cluster field) and the length of the file or subdirectory (File Size field). However, the file or subdirectory is NOT stored contiguously, in general. For a file that is more than 1 cluster long, you need to use the FAT to find the remaining clusters, per the next section of this document.

FAT-12 file name and extension representation

File names in DOS traditionally have a limit of 8 characters for the name, and 3 characters for the extension. There are a few things to be aware of:

- File/directory names and extensions are *not* null-terminated within the directory entry
- File/directory names always occupy 8 bytes--if the file/directory name is shorter than 8 bytes (characters) pad the remaining bytes with spaces (ASCII 32, or Hex 0x20). This also applies to 3-character extensions.

- File/directory names and extensions are *always* uppercase. Always convert given file/directory names to uppercase.
- Directory names can have extensions too.
- "FILE1" and "FILE1.TXT" are unique (the extension *does* matter).
- Files and directories *cannot* have the same name (even though the attributes are different).

Here are examples of how some file names would translate into the 11 bytes allocated for the file/directory name and extension in the directory entry (white space between quotes should be considered as spaces).

- `filename provided [01234567012]`
- `"foo.bar"      -> "FOO     BAR"`
- `"FOO.BAR"      -> "FOO     BAR"`
- `"Foo.Bar"      -> "FOO     BAR"`
- `"foo"          -> "FOO        "`
- `"foo."         -> "FOO        "`
- `"PICKLE.A"     -> "PICKLE  A  "`
- `"prettybg.big" -> "PRETTYBGBIG"`
- `".big"         -> illegal! file/directory names cannot begin with a "."`

5. Why do we need a FAT?

The directory entry has a field called the First Logical Cluster field which specifies where the file or subdirectory begins. Since files and directories can be larger than a sector, a directory or file may have to be stored across more than one sector. The data sectors belonging to a file or a directory are not always stored in contiguous locations in memory. A FAT therefore is used to keep track of which sectors are allocated to which file.

To retrieve the entire contents of a file, for example, the First Logical Cluster field would point to the sector number that holds the first 512 bytes of data. The data from this sector needs to be read in. To determine if there is more data, one must examine the FAT entry that corresponds to the First Logical Cluster. By examining the FAT entry value, it can be determined if there is another sector allocated to this file. If there is, then the logical sector value is translated to physical sector value and the data from that sector is read in. Next, the FAT entry for the second data sector is examined to see if it is the end of the file. If not, the process is continued.

Therefore, the FAT allows the access of data stored in non-contiguous sectors of the storage device.

In Figure 2, File1.txt is stored in logical sectors 2, 4, 6 and 7. The directory entry field “Start Cluster” i.e. First Logical Cluster field points to sector number 2 which is the first data sector. In the FAT, the value at FAT entry 2 is 4, indicating that the next data sector of File1.txt is stored in logical sector 4. The last sector is sector 7, which is evident as the FAT entry 7 holds the EOC value.

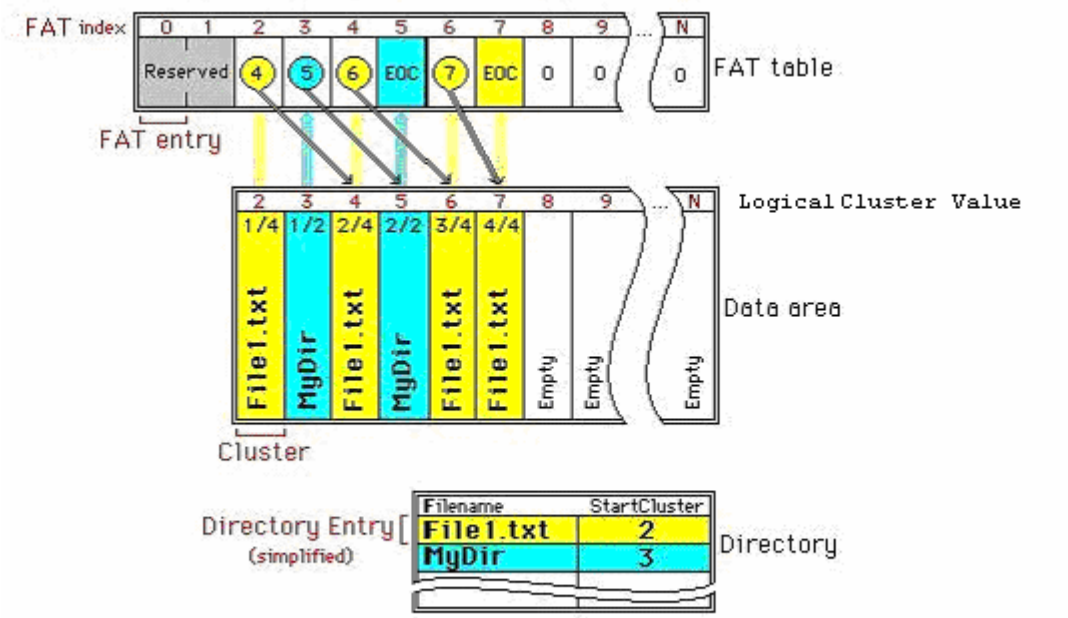

Figure 2 Example showing the use of the FAT[1]

6. Fat Packing

In this section, the choice of the value "12" is explained followed by a description on how a 12-bit value is stored in the FAT.

The space on a floppy disk = 1.44 Mbytes.
The number of bytes in a sector = 512
The number of sectors in 1.44 Mbytes = x ≈ 2812
Therefore, the minimum number of bits required to address "x" sectors = 12 bits ($2^{11} < 2812 < 2^{12}$)

It can be seen from the above computations that 12 bits is the minimum number of bits needed to access the entire 1.44M space of a floppy disk.

The challenge of 12 bits is that computers store everything in multiples of 8 bits (1 byte). So when storing the 12 bit quantity, the option of using 16 bits to store the 12 bits was unsatisfactory as it would leave 4 bits unused for every FAT entry. Since disk space is already at a premium on floppies another solution was designed. This solution involves packing 2 FAT entries (a total of 24 bits) into three 8 bit locations. This is great from an efficiency point of view but it means you have to do a little bit of work to extract a single entry. To further clarify examine the snapshot of the FAT. 8-bit entries are examined:

[1] The figure has been obtained from the CS324 course web-site from Brigham Young University.

Position Byte

0 **76543210**

1 *5432**1098***

2 *32109876*

This space holds 2 FAT entries. The first entry would be 109876543210 where the first 4 bits come from position 1. The second entry is 321098765432 where the last 4 bits come from position 1. Since the FAT was developed for IBM PC machines, the data storage is in little-endian format i.e. the least significant byte is placed in the lowest address.

So how do we work with the FAT? First, we think of the FAT as an array of bytes (8 bit quantities) since that is the only way we will be able to represent it in C. Now, if we want to access the n^{th} FAT entry then we need to convert between 12 bit and 8 bit values.

- If n is even, then the physical location of the entry is the low four bits in location 1+(3*n)/2 and the 8 bits in location (3*n)/2
- If n is odd, then the physical location of the entry is the high four bits in location (3*n)/2 and the 8 bits in location 1+(3*n)/2

You are provided with the functions to read and write values to the FAT.

References

- http://students.cs.byu.edu/~cs345ta/labs/fall03_specs/lab_fat_help.htm#Directory%20structures%20and%20their%20fields
- White paper on FAT file systems.