Microsoft's Original DOS File System, FAT12, and its Evolution to Become FAT32, Windows 95's Most Current File System

Introduction

In 1977, a small, budding company, named Microsoft, invented a file system called the File Allocation Table, or FAT. This system was developed for Microsoft's Stand-alone Disk Basic interpreter. It was later adapted to Microsoft's Disk Operating System, or DOS. The FAT file system was intended for floppy disks, so the maximum amount of memory the file system could allocate was a whopping 8 megabytes of information. Surely, the 8 megabyte ceiling was all but unattainable; imagine, a personal computer that would require 8 million bytes of information.

During DOS 2.0, Microsoft saw the need for allocation of more memory, so in DOS 3.0, Microsoft introduced FAT16. This new FAT system could allocation up to 32 megabytes. This seemed large enough, but in 1987, the size envelope was pushed to its limit by the hard disks of the time. Microsoft responded with DOS 4.0. This extended the allocable memory size to 128 megabytes. Minor extensions then raised this ceiling to 2 gigabytes.

For about ten years, the 2 gigabyte ceiling was more than enough. Now, with hard drives commonly available of more than 2 gigabytes in size, a new FAT file system has been developed called FAT32. The FAT32 file system is capable of allocating up to 2 terabytes of information.

FAT32 is the latest version of the FAT file system offered by Microsoft in its extremely successful DOS/Windows 95 operation system series. It is the product of a constantly evolving FAT file system. Aside from the amount of allocable memory, FAT32 is far superior than the original FAT file system in many ways such as the sizes of a valid filename. This paper will discuss and explain the original FAT file system. It will then discuss the three major upgrades to this file system: FAT16, VFAT, and FAT32.

Basic Structure of Disks

Floppy disk drives and hard disk drives are slightly different in structure, but both consist of magnetic platters, or disks, that hold magnetic charges in certain areas. A hard drive can contain many disks stacked on top of each other. Floppy disk contains only one disk. Areas with a magnetic charge are designated as digital ones, while areas without a magnetic charge are interpreted as digital zeros. The mechanical read/write disk head is responsible for reading and writing the magnetic fields on the disk. The disk is then spun at very high velocities. By doing this, the heads only need to move from the outside of the disk to the center. Figure 1 illustrates this.

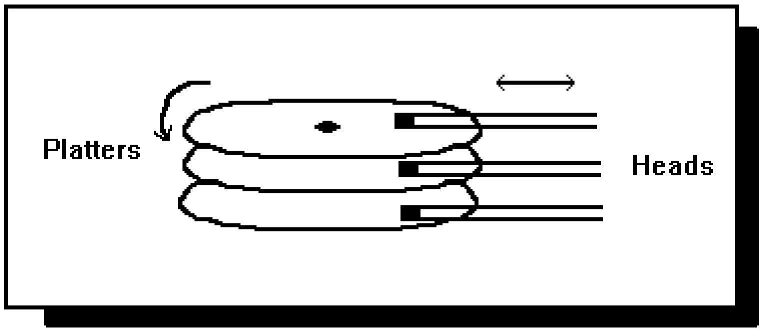


Figure 1: The Disk and Head Layout

The disks are divide into concurrent circles called tracks. Each track contains a line of changing magnetic charges in the shape of a circle. The disk is also divided into small pie shaped slices called sectors. Figure 2 further illustrates the sectors and tracks on a disk.

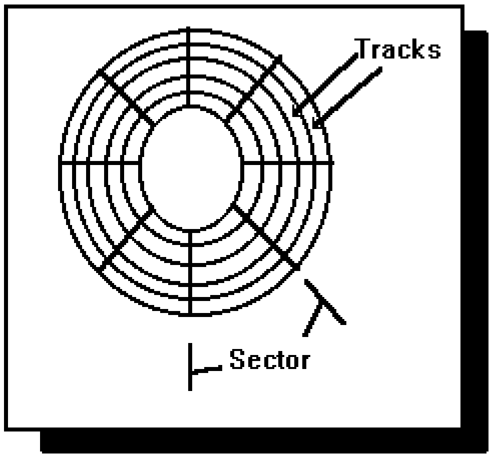


Figure 2: The Sectors and Tracks of a Disk

Each sector contains 512 bytes of information. This is the smallest amount of information that a program can access. A program accesses this by a function call to the disk driver. This function call is not normally called by an individual program. A normal program calls a function that belongs to the operating system. This function retrieves or writes a whole cluster of information. A cluster is defined as a group of sectors. The exact size of a cluster can range from one sectors per cluster (512 bytes), up to 64 sectors per cluster (32,768 bytes). The size of a cluster is determined by the size of the disk drive and the version of the FAT that is used. The advantages and disadvantages to the cluster size is discussed in detail later in the paper.

FAT12

FAT12 is the unofficial name of the original FAT file system. This paper will address the original FAT file system as FAT12 to differentiate the original system from the later versions. The volume is the basic unit for the DOS FAT file system. From a user standpoint, the volume corresponds to the drive letter at the command prompt. One volume is contained on a floppy disk, while a hard disk drive can be partitioned into many volumes. For the FAT12 file system, which is found in DOS 1.0 and 2.0, the maximum size of a volume is 8 Megabytes.

The volume is divided into four major sections: the boot sector, file allocation tables, root directory, and data area.

The Boot Sector

The boot sector is the heart of the FAT file system. It contains all the information needed by DOS to call the disk driver functions to access individual sectors. The boot sector is always located in the first sector of the disk. Table 1 shows the fields of information that are found in the boot sector.

Table 1: The Boot Sector Field Locations

Address	Field Description	Length in Bytes
00h	Jump to boot routine	3
03h	Manufacturer's information	8
0Bh	Bytes per sector	2
0Dh	Sectors per cluster	1
0Eh	Number of reserved sectors	2
10h	Number of FATs	1
11h	Number of entries in the root directory	2
13h	Number of sectors in volume	2
15h	Media descriptor	1
16h	Number of sectors per FAT	2
18h	Sectors per track	2
1Ah	Number of read/write heads	2
1Ch	Number of hidden sectors	2
1Eh	Boot routine	Variable

.

The Boot Routine: The boot sector is named boot sector because it contains the boot routine. The very first three bytes on the disk is an actual jump instruction to the boot routine. In floppy disks that are not boot disks, the jump to the boot routine is not necessary, but with hard disks, it is very necessary. When a PC is first turned on, the ROM BIOS executes and performs many low level tasks. These tasks do not include loading up DOS. One of the last commands in the ROM BIOS is a jump to the first byte in the first sector on the default drive. These bytes then tell the CPU to jump to the boot routine. This is the routine that actually loads up DOS.

Lastly, the "boot routine" starts at address 1Eh, and can extend to the end of the sector, or even to the next couple of sector. The "number of reserved sectors" field contains the total number of sectors needed by the boot sector.

The Manufacturer: The next eight bytes of information, starting at location 03H, contain information on the manufacturer, where the disk was formatted, and the version of DOS used.

The BIOS Parameter Block: The next eleven fields (starting at location 0Bh) are called the BIOS Parameter Block (BPB). These fields contain the information needed by the BIOS when a low level BIOS function is called (namely the 13H BIOS functions). The "number of reserved sectors" field contains the number of sectors the boot sector fills up. Remember, the boot routine is found in the boot sector section; this can get fairly large.

As explained in this paper, the FAT is very important. Because of this, multiple copies of the FAT can be stored; the second FAT is stored right after the first. If an error is detected in the first FAT, a new FAT can be loaded up. Usually, only two copies are stored.

The "media descriptor" field contains a code for the type of disk that the boot sector is on. Table 2 contains the codes and corresponding description.

Table 2: The Codes for the Media Descriptor Field

Code	Device	Description
F0h	3.5 " disk drive	2 sides, 80 tracks, 18 sectors per track
F8h	Hard drive	Varies
F9h	5.25" disk drive 3.5" disk drive	2 sides, 80 tracks, 15 sectors per track 2 sides, 80 tracks, 9 sectors per track
FAh	5.25" disk drive 3.5" disk drive	1 side, 80 tracks, 8 sectors per track 1 side, 80 tracks, 8 sectors per track
FBh	5.25" disk drive 3.5" disk drive	2 sides, 80 tracks, 8 sectors per track 2 sides, 80 tracks, 8 sectors per track
FCh	5.25" disk drive	1 side, 40 tracks, 9 sectors per track
FDh	5.25" disk drive	2 sides, 40 tracks, 9 sectors per track
FEh	5.25" disk drive	1 side, 40 tracks, 8 sectors per track
FFh	5.25" disk drive	2 sides, 40 tracks, 8 sectors per track

The Root Directory

The file allocation tables are found after the boot sector, and the root directory is found after the file allocation tables (the root directory is being discussed out of order because knowledge of the root directory is needed to fully explain the file allocation table). The DOS file system is a hierarchically based system like the UNIX system. The root directory is the father of all the files and subdirectories. Subdirectories can be placed in the root directory, and can contain other subdirectories and files. Look at figure 3 for an illustration of this.

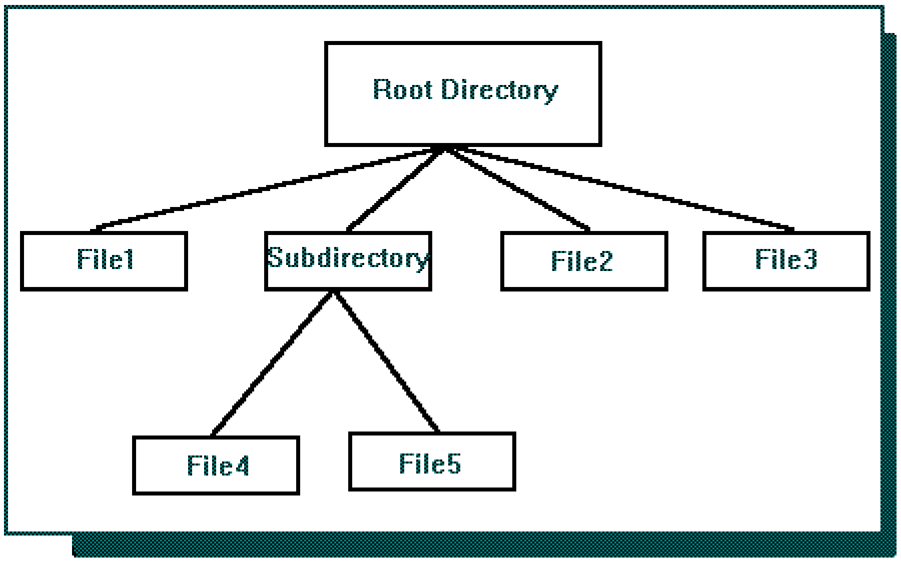


Figure 3: The Hierarchical Directory Arangement

Each file and subdirectory contained in the root directory has a 32 byte entry that describes the file or subdirectory and location. A description of the directory entry structure is found in Table 3.

Table 3: The 32 Byte Directory Entry

Address	Field Description	Length in Bytes

00h	Filename	8
08h	File extension	3
0Bh	File Attribute	1
0Ch	Reserved	10
16h	Time of last change	2
18h	Date of last change	2
1Ah	First cluster of file	2
1Ch	File size	4

The Filename: DOS sees a filename as eight characters, then a period, then a three character extension (for example, FILENUM1.TXT); this is referred to as the 8.3 filename. DOS stores each character as an ASCII character, so only one byte per character is needed. The first field in the directory entry is the first part of the file name. The second field is the extension. The period is not stored because it is common to all filenames. If the filename is less than eight character with a three character extension, spaces (ASCII 32) are placed in the unused fields.

The first character in the directory entry, which is the first character of the filename field, can contain special codes. Table 5 contains these special codes and meaning.

Table 5: The Special Codes Found in the First Character of the Directory Entry

Code	Meaning
00h	Last directory entry
05h	First character of filename has ASCII code E5h
E5h	File Deleted

File Attributes: The next field, the “file attribute” field, is a byte long of bitwise flags that describe the attributes of the file or directory. Table 6 describes the bit meanings. The “write protected”, “hidden file”, and “system file” bits are self-explanatory. The volume name bit is set to a one if the file that the entry describes is the volume name.

There is only one volume name file in a particular volume. This directory entry describes a file that contains a description of the volume. If the command VOL is typed at the command prompt of a computer running DOS, the contents of the file is displayed.

Table 6: The File Attribute Field

Bit	Description
0	1 = Read only
1	1 = Hidden file
2	1 = System file
3	1 = Volume Name
4	1 = Subdirectory
5	Archive bit
6	Reserved
7	Reserved

The "archive" bit is used by backup programs. This bit is set to a one if the file that the entry describes is changed. When a backup program runs, it looks at this attribute; if it is a one, the program backs up the file. The backup program will then place a zero in the "archive" bit. With the "archive" bit, a backup program does not have to backup a whole hard drive, just the files that have changed since the program was last backed up. The "subdirectory" bit is changed to a one if the file that the directory entry describes is actually a subdirectory. This is the only difference between a file and directory entry (shortly, the way FAT12 handles subdirectories is discussed).

<u>The Time and Date Stamp:</u> The "time stamp" and "date stamp" fields contain the time and date when the file that the directory entry describes was last revised. Figure 4 and figure 5 describe the bits found in the fields.

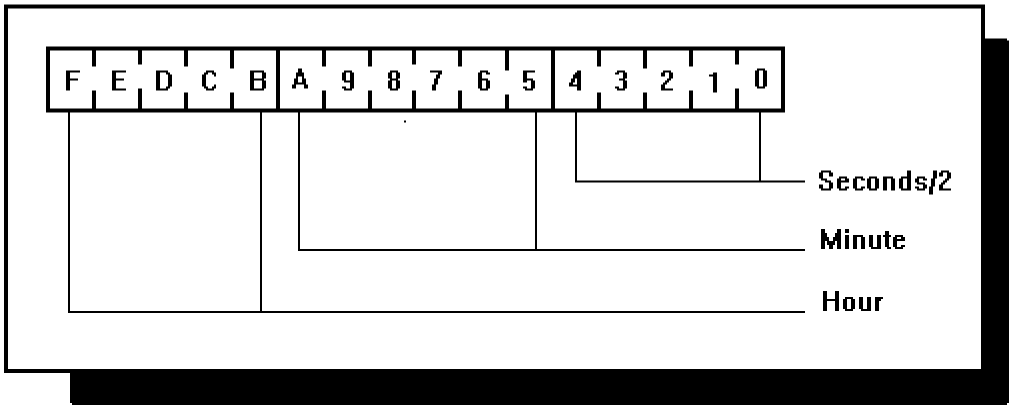


Figure 4: The Time Stamp Field

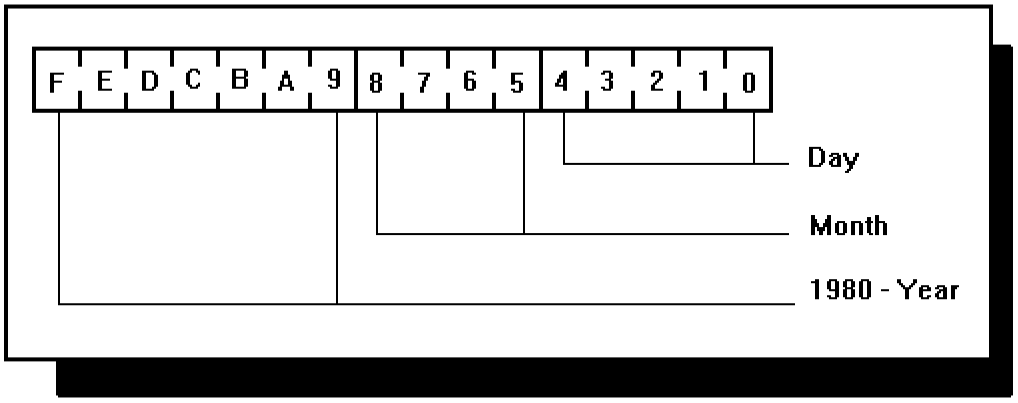


Figure 5: The Date stamp Field

The First Cluster of a File: As mentioned earlier, DOS breaks up memory into clusters, which are a number of sectors. Each cluster is described by a unique number. The number that describes the cluster that the file resides in is found in the next field of the 32 byte directory entry. When a file is too large to be stored in one cluster, the starting cluster is placed in this field. The file allocation table is then used to locate the rest of the clusters that contain the file (more on this later).

File Size: The file size is placed in the last field of the directory entry. It is stored in two two byte packages, or words. The least significant word is stored first, while the most significant word is stored second.

Subdirectories

As noted earlier, a subdirectory entry is flagged by the "subdirectory" bit in the "attribute" byte. The cluster that the "first cluster " field describes is the memory locations that store the subdirectory information. The subdirectory information is laid out almost exactly like the root directory using the 32 byte descriptions. When subdirectories are created though, two entries are created that can never be deleted: the "." and ".." entries. The "." entry points to the current subdirectory; the "first cluster" field points to the cluster of the subdirectory. The ".." entry points to the parent directory. If the parent directory is the root directory, the entry for the "first cluster" field is a zero.

The File Allocation Table (FAT)

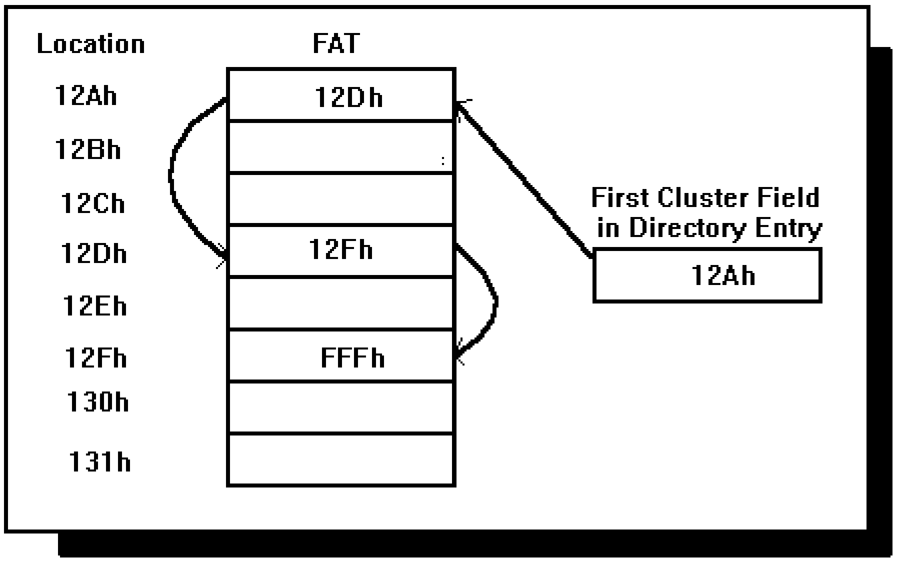
When DOS is asked to retrieve the contents of a file, it must know which clusters the file is located in. If a file is 20 kilobytes long, and the cluster size is 4 kilobytes, five clusters are needed to contain the file. These clusters may not be in sequence because the number of clusters needed to store a file may be more than the largest string of sequential clusters. The file allocation table is designed to overcome this obstacle.

The file allocation table is basically a linked list of clusters. The table consists of twelve bit entries, one for each cluster of memory. Incidentally, the length of the entries in bits is the suffix of the FAT12 designation. When DOS is asked to retrieve a file, DOS first looks at the directory entry for the file. The "first cluster number" field points to the first cluster that the file is stored in. This data is then retrieved. DOS then looks at the entry in the FAT table for the corresponding cluster. If the number in this entry is not a special number (special numbers are found in Table 7), the number stored is the number corresponding to the next cluster that the file is stored in. Once DOS finds a FF8h or greater in a FAT12 field, DOS knows that the cluster is the last cluster in the linked list. Figure 6 illustrates the linked list concept. Lastly, when DOS searches for a cluster to place a file in, it looks for the zero entries in the FAT.

Table 7: The Special Entries in a FAT

Code	Meaning
000h	Cluster is available
FF0h-FF6h	Reserved cluster
FF7h	Bad cluster
FF8h-FFFh	Last cluster in file

Figure 6: The FAT Linked List



The first two entries in the FAT are reserved for special assignments, not cluster assignments. The “media descriptor” field found in the boot sector is also found in the first byte of the FAT. The rest of the reserved bytes contain 255.

Limitations of FAT12

The FAT12 file system is fairly efficient, and works very well for its age, but it still has many limitations.

8.3 Filenames

8.3 filenames are filenames with a maximum of eight characters, a period, and an extension of three character. This limitation is very cumbersome because the filenames do not describe a file’s contents very well. For example, the first quarter’s sales report of 1997 for a company must be describe by something like Q1SALE97.EXC. This may make sense now, but two years from now, and in a directory with fifty other files, this name does a poor job in describing the file.

8 Megabyte Limit

The 8 Megabyte limit is caused by a number of factors. First, the twelve bit entry to the FAT limits the size. Twelve bits have a total of 4096 unique combinations of ones and zeros. Since each entry must have a number corresponding to it that is totally unique, there can be only 4096 clusters (actually the special numbers, such as FFFh used to denote the end of the FAT linked list, drop this number slightly). Also, only four sectors are allocated per cluster. So:

4096 clusters * 4 sectors/cluster * 512 bytes/sector = 8 Megabytes

FAT12 set a maximum of 4 sectors per cluster. When a file is allocated to a cluster of memory, it gets a cluster of memory. Even if the file was 1 byte long, a whole cluster of memory is allocated. As a result, if the cluster size was 2048 bytes, 2047 bytes will go unused. If the size of a cluster is decrease to 2 sectors per cluster, or 1024 bytes, less memory would be wasted, but the maximum amount of memory that could have a unique cluster number goes down to 4 Megabytes. Plus, DOS must traverse the FAT linked list to find the locations of the file fragments, then accesses that memory. If the cluster size was cut in half, the number of clusters that would have to be accessed is doubled. This leads to a slower performance in DOS.

Fragmentation of Files

The fragmentation of a file is when the file is stored in clusters that are not sequential. Figure 6, found earlier in this paper, illustrates file fragmentation. If the clusters that need to be accessed are sequential, the read/write heads of the disk drive only needs to move minute amounts. Fragmented clusters, on the other hand, cause the heads of the disk drive to move all over the disk to retrieve each cluster of information. Therefore, more time is required to move the head to non-sequential clusters than to sequential clusters. DOS provides a utility program to fix the fragmentation, but this cannot be done automatically when a file is fragmented. Plus, this utility takes a while to execute.

FAT16

DOS 3.0

Microsoft realized that the 8 Megabyte barrier must be overcome to handle the hard disk drives that were starting to be installed in computers. As a result, Microsoft released DOS 3.0. This version of DOS had FAT16, as well as FAT12. FAT16 was set up exactly like FAT12, except the file allocation table had entries that were sixteen bits in length. By adding four more bits, the number of unique clusters that were available were 65,536 clusters. If less than 4096 clusters were needed, FAT12 was used. If there were more than 4096 clusters, FAT16 was used.

Even thought the number of clusters possible were increased by a factor of sixteen, 4K to 64K, the maximum number of bytes in a volume was raised to only 32 megabytes. The problem existed in a low level system call to retrieve a sector of information. This function had a sixteen bit number passed to it to retrieve a sector. Since there are only 65,536 distinct 16-bit values, only 65,536 x 512, or 32 megabytes, of information could have a unique sector number.

DOS 4.0

Once the 32 megabyte envelope was being threatened by newer hard drives, Microsoft released DOS 4.0. In DOS 4.0, the low level system function to retrieve a sector was changed to accept a 32-bit value. Now, DOS could handle up to 2 gigabytes in a single volume. The reason for the 2 gigabyte limit in size of a volume is not a fault in DOS, it is a fault in existing programs. Most programs assume that the cluster size can be fit into a 16-bit number because the FAT was 16 bits long. Since FAT16 can handle up to

65,536 unique clusters, the cluster size is just the quotient of the disk size and 65,536 rounded up to a power of two. Table 8 shows disk sizes with the minimum cluster size. Notice, a disk drive capacity of more than 2 gigabytes has to have exactly 65,536 bytes in a cluster; 65,535 is the largest number that can be represented by 16 bits.

Table 8: Disk Sizes and Cluster Sizes

	Sectors per Cluster		**Cluster Size**	
Disk Size	Decimal	Binary	Decimal	Binary
Less than 128 MB	4	0000 0100	2K	0000 1000 0000 0000
Less than 256 MB	8	0000 1000	4K	0001 0000 0000 0000
Less than 512 MB	16	0001 0000	8K	0010 0000 0000 0000
Less than 1GB	32	0010 0000	16K	0100 0000 0000 0000
Less than 2 GB	64	0100 0000	32K	1000 0000 0000 0000
Less than 4 GB	128	1000 0000	64K	Error with Programs
More than 8 GB	256	Error in DOS		

If the companies that wrote the programs that assume 16-bit cluster sizes were to rewrite their code, DOS itself would have a maximum volume size of 4 gigabytes. This is caused by the “sectors per cluster” field found in the BPB. This field is 8-bits long; 255 is the largest number that can be represented. A disk drive of more 4 gigabytes or more would require 256 sectors per cluster.

VFAT

VFAT (virtual file allocation table) is the file allocation system used by Widows 95. This file system allows for long file names, accepting virtually any character. VFAT is also backwards compatible with DOS and most DOS based programs. VFAT does this by cleverly utilizing the attribute field in the 32-byte directory entry.

Through testing, Microsoft found that if a directory entry was marked with attributes of read-only, hidden, system, and volume name (an attribute value of 0Fh), DOS, and most programs, simply ignore the entry. DOS ignores the entry because it is impossible for a file to have this combination of attributes.

When a file is saved, VFAT first truncates the filename to the first six letters, a tilde, and a number. The number is usually a one, but if a file already exits with the same name, the number is incremented to two, then three, etc.. This name is then stored as a 32-bit directory entry. This version of the directory entry is the entry that DOS sees. The full filename is then placed in other 32-bit directory entries located directly in front of the short filename entry. The format for these directory entries are different than normal entries. The format is found in Table 9.

Table 9: The Long Filename Directory Entry

Byte Number	Meaning
00h	Sequence byte Bits 0-4: Sequence number Bit 5: 0 Bit 6: 1 = final Component Bit 7: 0
01h-0Ah	First 5 characters
0Bh	File attribute
0Ch	Type indicator
0Dh	Checksum
0D-18h	Next 6 characters
19h-1Ch	Starting cluster number (always 0)
1c-1Fh	Next 2 characters

Notice that only thirteen characters in the filename can be stored in one 32-bit directory entry. As a result, up two ten long filename directory entries are used to store the filename (to get a 128 character filename). VFAT just uses the minimum number of entries to save memory. If the filename is less than six characters long, a long filename entry is still placed before the short entry. It does this because the long filename accepts characters that the short directory entry does not.

The "Sequence" field in the long filename directory entry contains the number of the segment of the filename that the long directory entry contains. This number is in the first 5 bits. A one is placed in the sixth bit if the directory entry is the last entry. The next ten bytes contain five characters from the filename. Ten bytes are required because the characters are stored using Unicode; two bytes are required for each character. The "attribute" field follows. This field must be in the exact same place as the "attribute" field in the short filename directory entry because the 0Fh must be found there. When DOS or an MS- DOS program encounters a 0Fh in the "attribute" field, it ignores the entry. The next field is the "type indicator". This is always zero.

When a file is created in Windows 95, and then is deleted in DOS, DOS does not know the long filename directory entries exist for the file being deleted. As a result, these directory entries called orphans, are just left there. These orphans can build up to a sizable amount. This is where the "checksum" field in the long filename directory entry is important. The "checksum" field in the long filename directory entry contains a number that is the sum of a few bits location from the next directory entry modulated with 256. With this field, a utility program can get rid of these orphans by checking the "checksum" field and the next directory entry.

The next twelve bytes contain the next six characters in the filename. After that, the "starting cluster" field is found. Since the "starting cluster" field in the short filename directory entry is already filled with the starting cluster number of the file the entry points to, this field is just set to zero. Lastly, the last four bytes contain the next two characters of the filename.

Drawbacks and Limitations

Maximum Number of Files on the Root Directory: As is the case with FAT12 and FAT16, the root directory is fixed. This limitation is even more detrimental for the VFAT file system because up to eleven directory entries could be used for a single file.

Orphan Long Filename Directory Entries: As previously mentioned, orphan directory entries are caused by DOS when it deletes a file created or modified by Windows 95. These orphans can take up some disk space, but with a 1.5 Gigabyte hard drive, the 32 bytes per entry is not very big. Where these orphans can make the most mess is in the root directory. The root directory is fixed in size. If the orphans build up too much, a file may not be able to be saved in the root directory.

FAT32

FAT32 is the latest version of the FAT file system. Microsoft quietly released this version in their OEM Service Pack 2. As a result, only new computers will contain this file system in their version of Windows 95. FAT32 finally raises the 2 gigabyte barrier to 2 terabytes. It accomplishes this by having 32 bit FAT entries, and by modifying the BIOS parameter block (BPB) found in the boot sector.

32 Bit FAT Entries

By expanding the 16 bit FAT entry to 32 bits, a whopping 4,294,967,296 clusters can have unique names. By increasing the number of clusters that can be unique, smaller cluster sizes can be made for the very large hard drives. Table 9 shows the cluster sizes that VFAT calls for with relation to the hard drive size. With 32 bits, a hard drive of 2 terabytes can have a cluster size of one sector, but FAT32 calls for a larger cluster size. It calls for larger clusters keep the speed in retrieving a file low; the smaller the cluster size, the more FAT entries that have to be retrieved.

Table 9: The Disk Size and the Cluster Size in a FAT32 File System

Disk Size	Number of Sectors per Cluster	Cluster size in Bytes
less than 8 GB	8	4K
Less than 16 GB	16	8K
Less than 32 GB	32	16K
More than 32 GB	64	32K

Because the FAT entries are changed from 16 to 32 bits, the “starting cluster” field in the directory entries must be expanded to 32 bits. Microsoft has been gracious enough to give up two of its reserved bytes in the directory entries to accommodate the two extra bytes needed.

Root Directory Modification

In FAT32, Microsoft finally made the root directory dynamic in size like the subdirectories. FAT32 simply stores the root directory in a cluster somewhere. The exact cluster number is stored in the BPB.

BPB Modification

The BPB only needed to be modified to accommodate the larger numbers of sectors and cluster. Microsoft decided, though, that while they were modifying the BPB anyway, they could add a few more fields. Table 10 shows the new structure.

Table 10: The BPB in the FAT32 File System

Address Offset	Description	Length in Bytes
00h	Bytes per sector	2
02h	Sectors per cluster	1
03h	Reserved sector	2
05h	Number of FATs	1
06h	Number of root directory entries	2
08h	Number of sectors in the volume	2
0Ah	Media descriptor	1
0Bh	Sectors per FAT	2
0Dh	Sectors per track	2
0Fh	Read/write heads	2
11h	Number of hidden sectors low	2
13h	Number of hidden sectors high	2
15h	Total sectors low	2
17h	Total sectors high	2
19h	Sectors per FAT low	2
1Bh	Sectors per FAT high	2
1Dh	Mirror flags	2
1Fh	File system version	2
21h	First cluster of root directory low	2
23h	First cluster of root directory high	2
25h	File system information sector	2
27h	Backup boot sector location	2
29h	Reserved	12

The most significant field that was added was the “root directory starting cluster low and high” fields. This field actually stores the cluster number of the root directory. Now, the root directory is not fixed in size; actually, it is exactly like a subdirectory. With this modification, there is no limit to the number of files that can be stored in the root directory.

The first nine fields of the new BPB are exactly the same fields as the original BPB. The only difference is that the "total sectors" and "sectors per FAT" fields contain zeros. This is because these fields are too small to hold 32 bit information. Microsoft has the additional fields in the BPB handle this information. The "big total sectors low and high" fields contain the real information on the total sectors in the volume. The same concept applies to the "big sectors per FAT low and high" fields.

Another feature of the FAT32 file system is that the program can access each copy of the FAT individually. This is called FAT mirroring. This is useful because it allows better error recovery when a bad sector is detected in one of the FATs. The "BPB extension flag" contains the active FAT table in the lower four bits. These bits have meaning only if the eighth bit is set. The other bits in the field are reserved.

The "file system version" field contains the version of the file system used. The high byte is the major version number, and the low byte is the minor version number. Lastly, the "backup boot sector" field contains the location of a backup boot sector, if there is one. It there is no backup boot sector, a 0FFFFh is found there.

Limitations

Compatibility: The biggest limitation of FAT32 is that it is not backwards compatible. This is a result of the larger size of the FAT entries. Many FAT16 low level disk functions fail, causing this incompatibility. Of course, FAT16 can be read by FAT32. If a disk is partitioned into less than 512 Megabyte partitions, FAT32 is not used. This way, a FAT16 partition can be placed on a hard drive to run the non-FAT32 compatible programs.

Upgrade: As mentioned earlier, FAT32 is offered in Microsoft's OEM Service Pack 2. Unfortunately, it is not available in an Windows 95 upgrade. FAT32 will be offered in the upcoming Windows 97 release.

Conclusion

FAT12 is a file system that was developed twenty years ago for Microsoft's Standalone Disk Basic interpreter. This system was more than adequate then, but as computer disk drive sizes increased with leaps and bounds, FAT12 started to lag. After many modifications, and years, FAT32 was produced. With up to a 2 Terabyte volume size, this is the most advanced FAT offered by Microsoft. Surprisingly, with only a few ingenious modifications, the underlying structure of the FAT file system is preserved. This is a result of the need for compatibility, but FAT32 does not sacrifice much efficiency for this compatibility. This illustrates the power of the FAT file system. Of coarse, in just a few years, this FAT32 system will probably be obsolete. FAT evolution continues.

Appendix A: The Bibliography

“BPB (FAT32).” http://www.microsoft.com/msdn/sdk/platforms/doc/sdk/win32 /95guide/fat32/sec/far32_37.htm.

“FAT Mirroring.” http://www.microsoft.com/msdn/sdk/platforms/doc/sdk /win32/95guide/src/far32ovr_3.htm.

Prosise, Jeff. “Can VFAT Handle Large Disks?” PC Magazine 10 October 1995: 345-350.

Prosise, Jeff. “How Windows 95 Stores Long Filenames.” PC Magazine 25 June 1996: 217-218.

Prosise, Jeff. “Is VFAT Ready for Prime Time?” PC Magazine 13 June 1995: 247-253.

Rigney, Steve. “The Long and Short of Windows 95 Filenames.” PC Magazine 10 October 1995: 369-370.

Simon, Barry. “An Enhanced File System for Windows 95.” PC Magazine 8 April 1997: 279-280.

“The FAT32 File Sytem.” http://www.microsoft.com/windows/pr/fat32.htm.

Tischer, Michael. PC Intern: System Programming Grand Rapids: Abacus, 1992

“Windows 95 FAT System Expands to Handle Terabyte Hard Disks.” http://www.microsoft.com/syspro/technet/tnnews/webnews/w950617.htm.