作为一份入门的参考，还是可以借鉴的。

**FAT-文件分配表文件系统**

**第一节FAT的概要**

FAT

**文件配置表**（英文：File Allocation Table，首字母缩略字：FAT）又称文档分配表，是一种由微软发明并拥有部分专利的文件系统，供MS-DOS使用，也是所有非NT内核的微软窗口使用的文件系统。FAT文档系统考虑当时计算机性能有限，所以未被复杂化，因此几乎所有个人计算机的操作系统都支持。这特性使它成为理想的软盘和记忆卡文件系统，也适合用作不同操作系统中的数据交流。

File Allocation Table

File Allocation Table (FAT) is a computer file system architecture now widely used on many computer systems and most memory cards, such as those used with digital cameras. FAT file systems are commonly found on floppy disks, flash memory cards, digital cameras, and many other portable devices because of their relative simplicity. For floppy disks, the FAT has been standardized as ECMA-107 and ISO/IEC 9293. Those standards include only FAT12 and FAT16 without long filename support; long filenames with FAT is partially patented.

The FAT file system is relatively straightforward technically and is supported by virtually all existing operating systems for personal computers. This makes it a useful format for solid-state memory cards and a convenient way to share data between operating systems.

**但FAT有一个严重的缺点：当文件被删除并且在同一位置被写入新数据，他们的片段通常是分散的，减慢了读写速度。**磁盘碎片重整是一种解决方法，但必须经常重组来保持FAT文件系统的效率。

FAT的缺点还有以下几点：

1.大磁盘浪费空间。

2.磁盘利用效率抵。

3.文件存储受限制。

4.不支持长文件名，只能支持8个字符。

5.安全性较差。

历史

　　FAT文件系统遵行已用了多年的软件方法来进行规范。它在1977年由比尔·盖茨（Bill Gates）和马斯·麦当劳（）为了管理磁盘而发明，并在1980年被添·彼得逊（Tim Paterson）的86-DOS操作系统采用。这也是86-DOS操作系统与CP/M操作系统最大的不同点，若非此项差异，86-DOS操作系统与CP/M操作系统几乎可说完全相同。

History

The name originates from the usage of a table which centralizes the information about which areas belong to files, are free or possibly unusable, and where each file is stored on the disk. To limit the size of the table, disk space is allocated to files in contiguous groups of hardware sectors called clusters. As disk drives have evolved, the maximum number of clusters has dramatically increased, and so the number of bits used to identify each cluster has grown. The successive major versions of the FAT format are named after the number of table element bits: 12, 16, and 32. The FAT standard has also been expanded in other ways while preserving backward compatibility with existing software.

FAT12

　　早期版本的FAT就是现在俗称的FAT12。主要用于软盘，它对**簇（文件分配最小的存储空间，大于等于扇区,1簇=n扇区，n大于等于1）**的编址采用12bit宽度的数，所以称为FAT12。12bit的地址可以寻址4096(=2^12)个簇，事实上在FAT12中只能寻址4078个簇（在Linux下可寻址4084个簇），有一些簇号是不能用的，在后面会给出具体的说明。磁盘的扇区标识是用16bit的数进行计算的，所以磁盘的容量就被局限在32M（512B\*(2^16)）空间之内。作为**软盘的文件系统**，它有几项限制：不支持分层目录结构，簇寻址只有12位（这使得控制FAT有些棘手）而且只支持最多32M（2^16个）的分区。

当时入门级的磁盘是5.25"、单面、40磁道、每个磁道8个扇区、容量略少于160KB。（1\*40\*8\*512B/1024=160KB）上面的限制超过了这个容量一个或几个数量级，同时允许**将所有的控制结构放在第一个磁道**，这样在读写操作时移动磁头。这些限制在随后的几年时间里被逐步增大。

由于唯一的根目录也必须放在第一个磁道，能够存放的文件个数就限制在了几十个。

FAT12

The initial version of FAT is now referred to as FAT12. Designed as a file system for floppy disks, it limited cluster addresses to 12-bit values, which not only limited the cluster count to 4078, but made FAT manipulation tricky with the PC's 8-bit and 16-bit registers. (Under Linux, FAT12 is limited to 4084 clusters.) The disk's size is stored as a 16-bit count of sectors, which limited the size to 32 MB. FAT12 was used by several manufacturers with different physical formats, but a typical floppy disk at the time was 5.25-inch, single-sided, 40 tracks, with 8 sectors per track, resulting in a capacity of 160 KB for both the system areas and files. The FAT12 limitations exceeded this capacity by a factor of ten or more.

By convention, all the control structures were organized to fit inside the first track, thus avoiding head movement during read and write operations, although this varied depending on the manufacturer and physical format of the disk. At the time FAT12 was introduced, DOS did not support hierarchical directories, and the maximum number of files was typically limited to a few dozen. Hierarchical directories were introduced in MS-DOS version 2.0.

A limitation which was not addressed until much later was that any bad sector in the control structures area, track 0, could prevent the disk from being usable. The DOS formatting tool rejected such disks completely. Bad sectors were allowed only in the file area, where they made the entire holding cluster unusable as well. FAT12 remains in use on all common floppy disks, including 1.44MB ones.

目录

　　为了很好地支持以带有10MB硬盘为特色的IBM PC XT新计算机，在1983年年初MS-DOS 2.0几乎与该计算机在同时发布，它引进了**层次目录结构**。除了允许更好地组织文件外，目录允许在硬盘上存储更多的文件，这是因为最大文件个数不再受制于（仍然是固定的）根目录大小。这个数目现在能够等同于簇的数目（甚至更大，这是考虑到长度为0的文件并不占据任何FAT簇）。

FAT本身的格式并没有改变。PC XT的10MB的硬盘有4KB大小的簇。如果后来安装了一个20MB的硬盘，并且使用MS-DOS 2.0格式化，最后的簇大小将变为8KB，硬盘容量将变为15.9MB。（簇的总数=2^12=4096，固定使用(FAT等)数：19，簇大小=4KB，15.9MB=16308KB=16699392B=4077\*4096B）

最初的FAT16

　　在1984年，IBM发布PC AT，这是一个20 MB的硬盘。微软公司也同步发布了MS-DOS 3.0。簇的地址增加至16位，允许更大数量的簇（最大 65,517），所以有更大的文件系统大小。但是，最大数量磁区及最大分割区(相当于磁盘)的大小仍是32 MB。所以，尽管技术上已经是“FAT16”，这种格式并不是我们今天常见到的这个名字所代表的格式。在MS-DOS 3.0格式化一个20 MB的硬盘，这硬盘将不能被MS-DOS 2.0或之前的版本所存取。当然，MS-DOS 3.0 仍然可存取MS-DOS 2.0的格式(8 KB簇的分割区)。

MS-DOS 3.0也开始支持**高密度**1.2MB 5.25"磁盘，最著名的**是每个磁道有15个扇区**，这样就允许FAT有更大的空间。这或许促进了一个对于簇大小的不确定的优化，**簇大小从2个扇区减到1个**。这样做的最后结果是**高密度磁盘**比旧的**双密度磁盘**的速度大幅度降低。

Initial FAT16

In 1984, IBM released the PC AT, which featured a 20 MB hard disk. Microsoft introduced MS-DOS 3.0 in parallel. (The earlier PC XT was the first PC with a hard drive from IBM, and MS-DOS 2.0 supported that hard drive with FAT12.) Cluster addresses were increased to 16-bit, allowing for up to 65,517 clusters per volume, and consequently much greater file system sizes, at least in theory. However, the maximum possible number of sectors and the maximum (partition, rather than disk) size of 32 MB did not change. Therefore, although technically already "FAT16", this format was not what today is commonly understood as FAT16. With the initial implementation of FAT16 not actually providing for larger partition sizes than FAT12, the early benefit of FAT16 was to enable the use of smaller clusters, making disk usage more efficient, particularly for files several hundred bytes in size, which were far more common at the time. Also, the introduction of FAT16 actually did bring an increase in the maximum partition size under MS-DOS, since the implementation of FAT12 for hard disks in MS-DOS 2.0 was limited to 15 MB. (That is, the initial FAT16 did not support larger drives than FAT12, but MS-DOS 3.0 using FAT16 did support larger drives than MS-DOS 2.0 using FAT12, by a factor of two.)

A 20 MB hard disk formatted under MS-DOS 3.0 was not accessible by the older MS-DOS 2.0. (This was because MS-DOS 2.0 did not support version 3.0's FAT-16 and because it did not support hard disk partitions over 15 MB in size.) Of course, MS-DOS 3.0 could still access MS-DOS 2.0 style 8 KB-cluster partitions.

MS-DOS 3.0 also introduced support for high-density 1.2 MB 5.25" diskettes, which notably had 15 sectors per track, hence more space for the FATs. This probably prompted a dubious optimization of the cluster size, which went down from 2 sectors to just 1. The net effect was that high density diskettes were significantly slower than older double density ones.

扩展分区和逻辑驱动器

　　除了改进FAT文件系统本身的结构之外，一个并行的提高FAT存储空间的开发路线是支持**多个磁盘分区**。最初，由于主引导分区中文件分配表的固定结构的原因一个硬盘只能使用多达4个分区。然而，由于DOS设计要求只能有一个分区标识为“活动的”，它也是主引导代码启动所用的分区。使用DOS工具不可能创建几个“主”DOS分区，并且第三方的工具也至少会警告这样一个机制将与DOS不兼容。

为了用一种兼容的方式使用更多的分区，一种**新的分区类型**被开发出来（1986年1月的MS-DOS 3.2），**扩展分区**实际上是另外称为逻辑分区的一个容器。最初它里面只允许有一个逻辑分区、支持最大64MB的硬盘。在MS-DOS 3.3（1987年8月）这个限制更改到24个分区；它可能来自于强制性的C:-Z:的磁盘命名规则。逻辑分区表使用盘上的数据结构来描述，可能是为了简化编码它与主引导记录非常相似，并且它们组织成类似于**俄罗斯套娃**那样的结构。系统中只能有一个扩展分区。

俄罗斯套娃（матрёшка或матрешка）是俄罗斯 特产木制玩具，一般由多个一样图案的空心 木娃娃一个套一个组成，最多可达十多个，通常为圆柱形，底部平坦可以直立。最普通的 图案是一个穿着俄罗斯民族服装的姑娘，叫做“玛特罗什卡”，这也成为这种娃娃的通称。



在扩展分区引入之前，一些硬盘控制器（当时是独立的板子，因为还没有IDE标准）能够将大硬盘显示为两个独立的硬盘。作为一种选择，如Ontrack的磁盘管理员这样的一些特殊软件可以用于同样目的。

Partitioning and logical drives

Apart from improving the structure of the FAT file system itself, a parallel development allowing an increase in the maximum possible FAT size was the introduction of multiple FAT partitions on a hard disk. Extra DOS partitions could not be used as boot partitions. Simply allowing several identical-looking DOS partitions could lead to naming problems: behaviour if more than one partition was marked active was undocumented (although well defined), as was the behaviour if there was more than one hard disk in the computer (which was machine dependent), or if the system was booted from a diskette. The use of third party formatting tools under the DOS complicated the problem even more.

To allow the use of more FAT partitions in a compatible way, a new partition type was introduced (in MS-DOS 3.2, January 1986), the extended partition, which is a container for additional partitions called logical drives. A useful side-effect of the extended partition scheme was to significantly increase the maximum number of partitions possible on a PC hard disk beyond the four which could be described by the MBR alone.

最终的FAT16

　　1987年11月我们今天称为FAT的格式最终到来，它在康柏 DOS 3.31中去掉了磁盘扇区的16位计数器。这个结果曾经一度被称为DOS 3.31大文件系统。尽管看起来磁盘上的变动很小，这个DOS的磁盘代码都必须检查并转换到32位的扇区数，由于它全部是16位的汇编语言这样一个现实，这项工作就变得非常复杂。

1988年这项改进通过MS-DOS 4.0得到广泛应用。现在分区大小受限于每个簇的8位有符号扇区计数，它最大能达到2的64次方，对于一个常用的有32KB个簇每扇区512字节的硬盘来说，将FAT16分区大小的“明显”限制扩充到2GB。在磁光盘媒体上，它能使用1或者2KB的扇区，这样大小限制也就成比例地增大。

后来，WindowsNT通过将每个簇的扇区数当作无符号数将最大的簇大小增加到64KB。然而这个格式与当时其它的任何FAT实现都不兼容并且产生了大量内部碎片。Windows98也支持这种格式的读写操作但是它的磁盘管理工具不支持这种格式。

在FAT16中，采用了16bit宽的簇地址，32bit宽扇区地址。虽然32bit的扇区地址可以寻址2^32\*512，约2个TB的容量，但于由规定每簇最大的容量不超过1024\*32，所以FAT16文件系统的容量也就限制到了2^16\*1024\*32，大约2.1GB的空量，并且实际还达不到这个值。

Final FAT16

Finally in November 1987, Compaq DOS 3.31 (an OEM version of MS-DOS 3.3 released by Compaq with their machines) introduced what is today called the FAT16 format, with the expansion of the 16-bit disk sector count to 32 bits. The result was initially called the DOS 3.31 Large File System. Although the on-disk changes were minor, the entire DOS disk driver had to be converted to use 32-bit sector numbers, a task complicated by the fact that it was written in 16-bit assembly language.

In 1988 this improvement became more generally available through MS-DOS 4.0 and OS/2 1.1. The limit on partition size was dictated by the 8-bit signed count of sectors per cluster, which had a maximum power-of-two value of 64. With the standard hard disk sector size of 512 bytes, this gives a maximum of 32 KB clusters, thereby fixing the "definitive" limit for the FAT16 partition size at 2 gigabytes. On magneto-optical media, which can have 1 or 2 KB sectors instead of 1/2 KB, this size limit is proportionally larger.

Much later, Windows NT increased the maximum cluster size to 64 KB by considering the sectors-per-cluster count as unsigned. However, the resulting format was not compatible with any other FAT implementation of the time, and it generated greater internal fragmentation. Windows 98 also supported reading and writing this variant, but its disk utilities did not work with it.

The number of root directory entries available is determined when the volume is formatted, and is stored in a 16-bit signed field, defining an absolute limit of 32767 entries (32736, a multiple of 32, in practice). For historical reasons, FAT12 and FAT16 media generally use 512 root directory entries on non-floppy media. Other sizes may be incompatible with some software or devices (entries being file and/or folder names in the original 8.3 format).[9] Some third party tools like mkdosfs allow the user to set this parameter.

长文件名（VFAT, LFN）

　　Windows 95设计人员的一个用户体验目标就是除了传统的8.3文件名以外在新操作系统中使用长文件名（LFN）。长文件名通过在目录条目排列时使用一个工作区来实现（参见下面）。按照Windows 95VxD设备驱动程序的命名规则这个新扩充的文件系统通常称为VFAT。 　　有意思的是，VFAT驱动在早于Windows 95的Windows for Groups 3.11中就已经出现，但它仅仅用于实现32位文件访问，一个绕过DOS的视窗自带高性能保护模式文件管理系统，它能够直接使用BIOS或者更好的32位磁盘访问，如视窗自带的保护模式磁盘驱动程序。它是一个后门；微软为Windows for Groups 3.11所作的广告说32位文件访问基于“芝加哥项目的32位文件系统”。

在Windows NT中，FAT文件系统对于长文件名的支持从3.5版就已经开始了。

Long file names

One of the user experience goals for the designers of Windows 95 was the ability to use long filenames (LFNs—up to 255 UTF-16 code points long), in addition to classic 8.3 filenames. LFNs were implemented using a workaround in the way directory entries are laid out (see below).

The version of the file system with this extension is usually known as VFAT after the Windows 95 virtual device driver, also known as "Virtual FAT" in Microsoft's documentation. Interestingly, the VFAT driver actually appeared before Windows 95, in Windows for Workgroups 3.11, but was only used for implementing 32-bit file access and did not support long file names.

In Windows NT, support for long filenames on FAT started from version 3.5. OS/2 added long filename support to FAT using extended attributes (EA) before the introduction of VFAT; thus, VFAT long filenames are invisible to OS/2, and EA long filenames are invisible to Windows.

FAT32

　　为了解决FAT16对于卷大小的限制同时让DOS的真实模式在非必要情况下不减少可用常规内存状况下处理这种格式，微软公司决定实施新一代的FAT，它被称为FAT32，带有32位的簇数，目前用了其中的28位。

理论上，这将支持总数达268,435,438（<2^28）的簇，允许磁盘容量达到2字节。然而，由于微软公司scandisk工具的限制，FAT不能大于4,177,920（<2^22）个簇，这将卷的容量限制在了124.55GB，除非不再使用“scandisk”。

FAT32文件系统使用了32bit宽的簇地址，所以称为FAT32。但在微软件的文件系统中**只使用了低28位，最大容量为2^28\*1024\*32,约8.7TB的空量。**有的人认为32bit全用，最大容量为2^32\*1024\*32，这种说法是不正确的。

虽然FAT32具有容纳近乎8.7TB的容量，但实际应用中通常不使用超过32GB的FAT32分区。WIN2000及之上的OS已经不直接支持对超过32GB的分区格式化成FAT32，但WIN98依然可以格式化大到127GB的FAT32分区，但不推荐这样做。

FAT32随着Windows95 OSR2(Windows97)发布，尽管需要重新格式化才能使用这种格式并且DriverSpace 3（Windows95 OSR2和Windows98所带版本）从来都不支持这种格式。Windows98提供了一个工具用来在不丢失数据的情况下将现有的硬盘从FAT16转到FAT32格式。在NT产品线上对于它的支持从Windows2000开始。

Windows2000和WindowsXP能够读写任何大小的FAT32文件系统，但是这些平台上的格式化程序只能创建最大32GB的FAT32文件系统。Thompson and Thompson (2003)写道“奇怪的是微软公司说这种现象是故意设计的”微软公司知识库文章 184006的确是这么说的，但是没有提出任何关于这个限制的合理解释。Peter Norton的观点是“微软公司在有意地削弱FAT32文件系统”。

FAT32

In order to overcome size limit of FAT16, while at the same time allowing DOS real mode code to handle the format, and without reducing available conventional memory unnecessarily, Microsoft implemented a next generation, known as FAT32. Cluster values are represented by 32-bit numbers, of which 28 bits are used to hold the cluster number, for a maximum of approximately 268 million (228) clusters. This allows for drive sizes of up to 8 TiB with 32 KiB clusters, but the boot sector uses a 32-bit field for the sector count, limiting volume size to 2 TiB on a hard disk with 512 byte sectors.

On Windows 95/98, due to the version of Microsoft's SCANDISK utility included with these operating systems being a 16-bit application, the FAT structure is not allowed to grow beyond 4,177,920 (< 222) clusters, placing the volume limit at 127.5 GiB (≈137 GB). A limitation in original versions of Windows 98/98SE's Fdisk utility causes it to incorrectly report disk sizes over 64 GiB. A corrected version is available from Microsoft, but it cannot partition drives larger than 512 GiB (≈550 GB). The Windows 2000/XP installation program and filesystem creation tool imposes a limitation of 32 GiB. However, both systems can read and write to FAT32 file systems of any size. This limitation is by design and according to Microsoft was imposed because many tasks on a very large FAT32 file system become slow and inefficient. This limitation can be bypassed by using third-party formatting utilities. Windows Me supports the FAT32 file system without any limits. However, similarly to Windows 95/98/98SE there is no native support for 48-bit LBA in Windows ME, meaning that the maximum disk size for (parallel) ATA disks is 128 GiB (≈137 GB), the maximum size of an ATA disk using the previous long-standard 28-bit LBA.

FAT32 was introduced with Windows 95 OSR2, although reformatting was needed to use it, and DriveSpace 3 (the version that came with Windows 95 OSR2 and Windows 98) never supported it. Windows 98 introduced a utility to convert existing hard disks from FAT16 to FAT32 without loss of data. In the NT line, native support for FAT32 arrived in Windows 2000. A free FAT32 driver for Windows NT 4.0 was available from Winternals, a company later acquired by Microsoft. Since the acquisition the driver is no longer officially available.

The maximum possible size for a file on a FAT32 volume is 4 GiB minus 1 byte (232−1 bytes). Video applications, large databases, and some other software easily exceed this limit. Larger files require another formatting type such as NTFS.

FAT16和FAT32文件系统最大分区容量

　　在FAT16中，采用了16bit宽的簇地址，32bit宽扇区地址。虽然32bit的扇区地址可以寻址2^32\*512，约2个TB的容量，但于由规定每簇最大的容量不超过1024\*32，所以FAT16文件系统的容量也就限制到了2^16\*1024\*32，大约2.1GB的空量，并且实际还达不到这个值。

FAT32文件系统使用了32bit宽的簇地址，所以称为FAT32。但在微软件的文件系统中只使用了低28位，最大容量为2^28\*1024\*32,约8.7TB的空量。有的人认为32bit全用，最大容量为2^32\*1024\*32，这种说法是不正确的。

虽然FAT32具有容纳近乎8.7TB的容量，但实际应用中通常不使用超过32GB的FAT32分区。WIN2000及之上的OS已经不直接支持对超过32GB的分区格式化成FAT32，但WIN98依然可以格式化大到127GB的FAT32分区，但不推荐这样做。

第三方支持

　　其它IBM PC的可选操作系统—如Linux、FreeBSD和BeOS—都支持FAT格式，并且大部分都在相应的视窗版本发布以后很快就支持VFAT和FAT32格式。早期的Linux发布版本还包括称为UMSDOS的格式，它是保存在一个独立的称为--linux-.---的带有Unix文件属性（如长文件名和访问许可）的FAT。UMSDOS在VFAT发布以后就不再使用从Linux内核2.5.7开始就禁止了这项功能。Mac OS X操作系统在除启动盘之外的其它卷上也支持FAT文件系统。

FAT和其它数据流

　　FAT文件系统本身不是为支持ADS而设计的，但是一些高度依赖它们的操作系统创造出不同的方法在FAT驱动器上处理它们。这些方法或者在额外的文件或路径中存储附加的信息（Mac OS），或者给那些磁盘数据结构中以前没有使用的变量赋予新的含义（OS/2和视窗NT）。第二种设计，尽管想象起来会更有效率，但是它们不能被不认识这种格式的工具复制或者备份；使用不能识别这种格式的磁盘工具（如碎片整理或CHKFSK）控制这些磁盘可能会破坏这些信息。

Mac OS使用PC Exchange存储不同的数据，文件属性和文件名存在一个名为FINDER.DAT的隐藏文件中，资源分支（ADS）存在名为RESSOURCE.FRK的子目录中，这些数据都存在使用它们的每个目录中。从PC Exchange 2.1开始，它们将Mac OS的长文件名保存为标准的FAT长文件名，并且将超过31个字符的FAT长文件名转换为唯一的31字符能够被Macintosh应用程序识别的文件名。

Mac OS X将元数据（资源分支、不同的ADS、文件属性）保存在与所有人相同并以“.\_”开始的名字的隐藏文件中，并且Finder将一些文件夹和文件元数据存在名为“.DS\_Store”的隐藏文件中。

OS/2高度依赖于扩展属性（EA）并且将它们存在位于FAT12或FAT16的根目录下名为“EA DATA. SF”的隐藏文件中。这个文件使用以前文件（或者目录）的目录清单中的两个保留字节索引。在FAT32格式中，这些字节中存有文件或者目录开始簇号的高16位，这样就使它难于在FAT32上保存EA。扩展属性可以通过Workplace Shell桌面、REXX脚本、许多系统图形用户接口和命令行工具（如4DOS|4OS2）来访问。

Windows NT支持HPFS、NTFS和FAT中所有扩展属性的处理（所用处理机制完全类似于OS/2），但是不能处理其它一些存于NTFS驱动器的ADS数据。试图从复制带有与NTFS驱动器属性不同扩展属性的ADS到FAT驱动器将报告一个警告信息提示ADS将会丢失。 　　Windows 2000以后产品的处理类似于视窗NT但复制到FAT32时它们没有显示任何警告信息直接丢弃扩展属性（但报告其它象“Macintosh Finder Info”和“Macintosh Resource Fork”这些ADS引起的警告）。

前景

　　微软公司最近获得了VFAT和FAT32的专利（但没有得到最初的FAT的专利），这引起人们对于微软将会寻求从Linux OS发布和初始化他们产品的媒体厂商收取专利费的担忧（参见下面的FAT授权协议）。尽管最初的裁定不利于微软公司，但是微软仍然取得了胜利并且得到了专利授权。

由于微软公司已经宣布不再开发基于MS-DOS的消费用操作系统视窗Me的后续版本，所以不再有可能会有新版的FAT。对于大多数用途来说，为视窗NT系列开发的NTFS文件系统从效率、性能和可靠性来说都优于FAT；它的主要缺点是小容量文件所占的额外空间以及除了基于NT的视窗之外的很少有其他操作系统支持。由于确切的规范是微软公司的商业秘密，这就使得使用一个DOS软盘用于恢复目的很困难。微软公司提供了一个恢复界面来解决这个问题，由于安全的原因它严重限制了缺省情况下它能解决的问题。

FAT仍然是活动媒体的常用文件系统（CD和DVD是例外），软盘使用的是FAT12，其它多数活动媒体用的是FAT16（如用于数位相机的快闪记忆卡和USB盘）。许多活动媒体还没有达到能够享用FAT32。出于兼容性和存储空间利用率的考虑FAT仍然用在这些驱动器上，同时也是由于这些活动媒体上的文件的许可更容易遇到麻烦而不是更重要这样一个事实。

视窗2000和XP支持的FAT32格式化的限制是32GB，这非常有效地强迫使用现代硬盘的用户或者使用NTFS或者在视窗之外格式化驱动器。一个解决的办法是使用从Linux移植到视窗平台的一个工具version of mkdosfs。

这里也有一个自由开源的工具。

**FAT12 1.44MB-16MB**

**FAT16 17MB-512MB**

**FAT32 513MB-MAX**

**主引导记录**

主引导记录（Master Boot Record，缩写：MBR），又叫做主引导扇区，是计算机开机后访问硬盘时所必须要读取的首个扇区，它在硬盘上的三维地址为（柱面，磁头，扇区）＝（0，0，1）。

主引导扇区记录着硬盘本身的相关信息以及硬盘各个分区的大小及位置信息，是数据信息的重要入口。如果它受到破坏，硬盘上的基本数据结构信息将会丢失，需要用繁琐的方式试探性的重建数据结构信息后才可能重新访问原先的数据。主引导扇区内的信息是通过FDISK写入的，它是低级格式化的产物，和操作系统没有任何关系（操作系统是创建在高级格式化的硬盘分区之上，是和一定的文件系统相联系的）。

对于硬盘而言，一个扇区可能的字节数为128×2n (n=0,1,2,3)。大多情况下，取n=2，即一个扇区（sector）的大小为512字节。

**低级格式化**

低级格式化（Low-Level Formatting）又称低层格式化或物理格式化（Physical Format），对于部分硬盘制造厂商，它也被称为初始化（initialization）。最早，伴随着应用CHS编址方法、频率调制（FM）、改进频率调制（MFM）等编码方案的磁盘的出现，低级格式化被用于指代对磁盘进行划分柱面、磁道、扇区的操作。现今，随着软盘的逐渐退出日常应用，应用新的编址方法和接口的磁盘的出现，这个词已经失去了原本的含义，大多数的硬盘制造商将低级格式化（Low-Level Formatting）定义为创建硬盘扇区（sector）使硬盘具备存储能力的操作。现在，人们对低级格式化存在一定的误解，多数情况下，提及低级格式化，往往是指硬盘的填零操作。

对于一张标准的1.44 MB软盘，其低阶格式化将在软碟上创建160个磁道（track）（每面80个），每磁道18个磁区（sector），每磁区512位位组（byte）；共计1,474,560字节。需要注意的是：软盘的低级格式化通常是系统所内置支持的。通常情况下，对软盘的格式化操作即包含了低级格式化操作和高级格式化操作两个部分。

主引导记录的组成

启动代码(0000-01B7,440)

主引导记录最开头是第一阶段引导代码。其中的硬盘引导程序的主要作用是检查分区表是否正确并且在系统硬件完成自检以后将控制权交给硬盘上的引导程序（如GNU GRUB）。MBR是由分区程序（如Fdisk）所产生的，它不依赖任何操作系统，而且硬盘引导程序也是可以改变的，从而能够实现多系统引导。

选用磁盘标志(01B8-01BB,4)

(01BC-01BD,2)0x0000

**硬盘分区表**

硬盘分区表(01BE-01FD,64)（Disk Patition Table,DPT）

硬盘分区表占据主引导扇区的64个字节(偏移01BEH--偏移01FDH)，可以对四个分区的信息进行描述，其中每个分区的信息占据16个字节。具体每个字节的定义可以参见硬盘分区结构信息。下面是一个例子：

如果某一分区在硬盘分区表的信息如下

80 01 01 00 0B FE BF FC 3F 00 00 00 7E 86 BB 00

则我们可以看到，最前面的"80"是一个分区的激活标志，表示系统可引导；"01 01 00"表示分区开始的磁头号为01，开始的扇区号为01，开始的柱面号为00；"0B"表示分区的系统类型是FAT32，其他比较常用的有04（FAT16）、07（NTFS）；"FE BF FC"表示分区结束的磁头号为254，分区结束的扇区号为63、分区结束的柱面号为764；"3F 00 00 00"表示首扇区的相对扇区号为63；"7E 86 BB 00"表示总扇区数为12289622。

80 **01 01 00** 0E **0F 3F 40** 3F 00 00 00 B1 FF 00 00

结束标志字(01FE-01FF,2)

结束标志字55，AA（偏移1FEH－偏移1FFH）是主引导扇区的最后两个字节，是检验主引导记录是否有效的标志。

主引导扇区的读取流程

系统开机或者重启。

1.BIOS 加电自检 ( Power On Self Test -- POST )。BIOS执行内存地址为 FFFF:0000H 处的跳转指令，跳转到固化在ROM中的自检程序处，对系统硬件(包括内存)进行检查。

2.读取主引导记录(MBR)。当BIOS检查到硬件正常并与 CMOS 中的设置相符后，按照 CMOS 中对启动设备的设置顺序检测可用的启动设备。BIOS将相应启动设备的第一个扇区(也就是MBR扇区)读入内存地址为0000:7C00H 处。

3.检查0000:7DFEH-0000:7DFFH(MBR的结束标志位)是否等于55AAH，若不等于则转去尝试其他启动设备，如果没有启动设备满足要求则显示"NO ROM BASIC"然后死机。

4.当检测到有启动设备满足要求后，BIOS将控制权交给相应启动设备。启动设备的MBR将自己复制到0000:0600H处, 然后继续执行。

5.根据MBR中的引导代码启动引导程序。

**硬盘分区表**

一、分区表的位置及识别标志

　　分区表一般位于硬盘某柱面的0磁头1扇区。而第1个分区表(也即主分区表)总是位于(0柱面，0磁头，1扇区)，剩余的分区表位置可以由主分区表依次推导出来。分区表有64个字节，占据其所在扇区的[447-510]字节。要判定是不是分区表，就看其后紧邻的两个字节(也即[511-512])是不是 "55AA"，若是，则为分区表。

二、分区表的结构

　　分区表由4项组成，每项16个字节.共4×16 = 64个字节。每项描述一个分区的基本信息。每个字节的含义如下：

分区表项含义

字节 含义

0 Activeflag.活动标志.若为0x80H,则表示该分区为活动分区.若为0x00H,则表示该分区为非活动分区.

1,2,3 该分区的起始磁头号,扇区号,柱面号磁头号 -- 1字节, 扇区号 -- 2字节低6位,柱面号 -- 2字节高2位 + 3字节

4 分区文件系统标志：

　　分区未用: 0x00H.

　　扩展分区: 0x05H, 0x0FH.

　　FAT16分区: 0x06H.

　　FAT32分区: 0x0BH, 0x1BH, 0x0CH, 0x1CH.

　　NTFS分区: 0x07H.

5,6,7 该分区的结束磁头号,扇区号,柱面号，含义同上.

8,9,10,11 逻辑起始扇区号。表示分区起点之前已用了的扇区数.

12,13,14,15 该分区所占用的扇区数.

　　分区表项有几个字节比较重要，下面分别阐述之：

　　1、（1，2，3）字节

　　磁头号由（1）字节8位表示，其范围为（0 -- 2^8 - 1），也即（0 磁头-- 255磁头）。 　　扇区号由（2）字节低6位表示，其范围为（0 -- 2^6 - 1），由于扇区号从1开始，所以其范围是（1扇区-- 63扇区）。

　　柱面号由（2）字节高2位 + （3）字节，共10位表示，其范围为（0 --2^10 - 1），也即（0 柱面-- 1023柱面）。

　　当柱面号超过1023时，这10位依然表示成1023，需要注意。

　　（5，6，7）字节含义同上。

2、（8, 9, 10, 11）字节

　　如果是主分区表，则这4 个字节表示该分区起始逻辑扇区号与逻辑0扇区（0柱面，0磁头，1扇区）之差。如果非主分区表，则这4 个字节要么表示该分区起始逻辑扇区号与扩展分区起始逻辑扇区号之差，要么为63。详细情况在后面有所阐述。

注意：

　　1、扇区上的字节是按左边低位，右边高位的顺序排列的。所以在取值时，需要把字节再反一下，让高位字节在左边，低位字节在右边，这一点在读取逻辑起始扇区号和分区大小时需要注意。举个例子:第一项的逻辑起始扇区为(3F 00 00 00)，转换为十进制前要先反一下字节顺序，为（00 00 00 3F）然后在转换为十进制，即63 .同理分区大小为（3F 04 7D 00），先反为（00 7D 04 3F）再转换为十进制，即8193087。

　　2、逻辑扇区号与（柱面，磁头，扇区）的相互转换：

　　令L = 逻辑扇区号，C = 柱面号，H = 磁头号，S = 扇区号。

　　每道扇区数 = 63

　　每柱面磁头数 = 255

　　每柱面扇区数 = 每道扇区数\* 每柱面磁头数

　　= 63 × 255

　　= 16065

　　柱面号下标从0开始。磁头号[0 -- 254]，扇区号[1 -- 63]。

　　逻辑扇区号下标也从0开始。

　　(柱面，磁头，扇区)转换成逻辑扇区号的公式为：

　　L = C×16065 + H ×63 + S - 1 ;

　　比如（1柱面，1磁头，1扇区），其逻辑扇区号为：

　　L = 1×16065 + 1×63 + 1 – 1

　　= 16128

　　逻辑扇区号转换成(柱面，磁头，扇区) 公式为：

　　C = L / 16065

　　H = (L % 16065) / 63

　　S = (L % 16065) % 63 + 1

　　比如逻辑扇区号 16127：

　　C = 16127 / 16065 = 1

　　H = (16127 % 16065) / 63 = 0

　　S = (16127 % 16065) % 63 + 1 = 63

　　即（1柱面，0磁头，63扇区）

　　3、分区表上有四项，每一项表示一个分区，所以一个分区表最多只能表示4个分区。主分区表上的4项用来表示主分区和扩展分区的信息。因为扩展分区最多只能有一个，所以硬盘最多可以有四个主分区或者三个主分区，一个扩展分区。余下的分区表是表示逻辑分区的。这里有必要阐述一点：逻辑区都是位于扩展分区里面的，并且逻辑分区的个数没有限制。

4、分区表所在扇区通常在（0磁头，1扇区），而该分区的开始扇区通常位于（1磁头，1扇区），中间隔了63 个隐藏扇区。

三、分区表链的查找

　　分区表链实际上相当于一个单向链表结构。第一个分区表，也即主分区表，可以有一项 描述扩展分区。而这一项就相当于指针，指向扩展分区。然后我们根据该指针来到扩展分 区起始柱面的0头1扇区，找到第二个分区表。对于该分区表，通常情况下：第一项描述了 扩展分区中第一个分区的信息，第二项描述下一个分区,而这第二项就相当于指向第二个分 区的指针，第三项，第四项一般均为0。我们可以根据该指针来到扩展分区中第二个分区起 始柱面的0头1扇区，找到第三个分区表。以此类推，直到最后一个分区表。而最后一个分 区表只有第一项有信息,余下三项均为0.相当于其指针为空.所以只要找到了一个分区表就可 以推导找出其后面所有分区表。不过该分区表前面的分区表就不好推导出来了。但令人高兴 的是这个链表的头节点，也即主分区表的位置是固定的位于（0柱面, 0磁头, 1扇区）处， 我们可以很轻易的找到它，然后把剩下的所有分区表一一找到。

　　以笔者的硬盘为例:一个主分区（C盘）, 一个扩展分区.扩展分区中有两个逻辑分区（D盘,E盘）其分区表链示意图如下:

　　C盘的主分区表

　　图一 分区表链示意图

　　(一).读取（0柱面，0磁头，1扇区）处

　　[80 01 01 00 0B FE 7F FD 3F 00 00 00 3F 04 7D 00 ]

　　[00 00 41 FE 0F FE FF FF 7E 04 7D 00 1F 2C B4 00 ]

　　[00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ]

　　[00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ]

第一项：

　　（80）（01 01 00）（0B）（FE 7F FD ）（3F 00 00 00 ）（3F 04 7D 00）

　　描述的是C盘的情况。

　　1.（80）：表示C盘为活动分区。即系统会从C盘启动。

　　2.（01 01 00）：表示C盘的起始扇区为（0柱面，1磁头，1扇区）。

　　3.（0B）：表示C盘的文件系统为FAT32。

　　4．（FE 7F FD）：

　　(FE) 16 = (254) 10； (7F) 16 = (0111 1111) 2； (FD) 16 = (1111 1101) 2。

　　磁头号：(254) 10；

　　扇区号：(11 1111) 2 = (63) 10.

　　柱面号：(01 1111 1101) 2 = (509) 10

　　故C盘结束扇区为（509柱面，254磁头，63扇区）。

5.（3F 00 00 00）：

　　反向，(00 00 00 3F) 16 = (63) 10,为C盘起始逻辑扇区号与逻辑0扇区号之差。表示C盘前面已有63个扇区，这63个扇区为系统隐藏扇区。

　　6. (3F 04 7D 00)：

　　反向，(00 7D 04 3F) 16 = (8193087) 10。表明C盘有8193087个扇区。即（0柱面，1磁头，1扇区）至（509柱面，254磁头，63扇区）

　　共有8193087个扇区。

第二项：

　　(00）（00 41 FE）（0F）（FE FF FF）（7E 04 7D 00）（1F 2C B4 00）

　　描述的是扩展分区的情况。

　　1. (00): 表示该分区不是活动分区。

　　2. (00 41 FE):

　　(00) 16 = (0) 10　(41) 16 = (0100 0001) 2 (FE) 16 = (1111 1110) 2

　　磁头号：(0) 10；

　　扇区号：(00 0001) 2 = (1) 10.

　　柱面号：(01 1111 1110) 2 = (510) 10

　　所以扩展分区的起始扇区为（510柱面，0磁头，1扇区）。

　　3. (0F): 表示该分区为扩展分区。

　　4. (FE FF FF):

　　(FE) 16 = (254) 10 (FF) 16 = (1111 1111) 2 (FF) 16 = (1111 1111) 2;

　　磁头号：(254) 10；

　　扇区号：(11 1111) 2 = (63) 10.

　　柱面号：(11 1111 1111) 2 = (1023) 10

　　但这是不准确的，因为当柱面号的真实值超过1023时，表示柱面号的10位也依然是1023。

　　5. (7E 04 7D 00):

　　反向，(00 7D 04 7E) 16 = (8193150) 10. 表示扩展分区的起始扇区号为8193150，即(510 柱面，0磁头，1扇区)。这是真实准确的，我一般都用这一项来定位分区起点。

　　6. (1F 2C B4 00):

　　反向, (00 B4 2C 1F) 16 = (11807775) 10. 表示扩展分区共有11807775个扇区。通过上面得到的起点和分区的大小，可以推导出扩展分区的结束位置：8193150 + 11807775 = 20000925号扇区，即（1244 柱面，254磁头，63扇区）。

四、查找D盘分区表

　　根据上面的信息，第二个分区表，也即D盘分区表在（510柱面，0磁头，1扇区）处。读取该扇区，得到分区表如下：

　　[00 01 41 FE 0B FE FF 7B 3F 00 00 00 BF A3 5D 00 ]

　　[00 00 C1 7C 05 FE FF FF FE A3 5D 00 21 88 56 00 ]

　　[00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ]

　　[00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ]

第一项：

　　（00）（01 41 FE）（0B）（FE FF 7B）（3F 00 00 00 ）（BF A3 5D 00）

　　描述的是D盘的情况。

　　1.（00）：表示D盘不是活动分区。

　　2.（01 41 FE）：

　　(01) 16 = (1) 10 (41) 16 = (0100 0001) 2 (FE) 16 = (1111 1110) 2.

　　磁头号：(1) 10；

　　扇区号：(00 0001) 2 = (1) 10.

　　柱面号：(01 1111 1110) 2 = (510) 10;

　　故D盘开始扇区为（510柱面，1磁头，1扇区）。

　　3.（0B）：表示D盘的文件系统为FAT32。

　　4．（FE FF 7B）：

　　(FE) 16 = (254) 10 。 (FF) 16 = (1111 1111) 2 (7B) 16 = (0111 1011) 2.

　　磁头号：(254) 10；

　　扇区号：(11 1111) 2 = (63) 10.

　　柱面号：(11 0111 1011) 2 = (891) 10

　　故D盘结束扇区为（891柱面，254磁头，63扇区）。

　　5.（3F 00 00 00）：

　　反向，(00 00 00 3F)16 = (63)10,为D盘起始逻辑扇区号（510柱面，1磁头，1扇区）与扩展分区起始逻辑扇区号

　　（510柱面，0磁头，1扇区）之差。表示D盘前面已有63个扇区，这63个扇区为系统隐藏扇区。

　　6. (BF A3 5D 00)：

　　反向，(00 5D A3 BF) 16 = (6136767) 10。表明D盘有6136767个扇区。

　　通过上面得到的起点和分区的大小，可以推导出D盘的结束位置：8193150 + 63 + 6136767 = 14329980号扇区。即（891柱面，254磁头，63扇区）。与上面的正好吻合。

第二项：

　　(00)（00 C1 7C）（05）（FE FF FF）（FE A3 5D 00）（21 88 56 00）

　　描述的是E盘的情况。

　　1. (00)： 表示E盘不是活动分区。

　　2. (00 C1 7C)：

　　(00) 16 = (0) 10 (C1) 16 = (1100 0001) 2 (7C) 16 = (0111 1100) 2

　　磁头号：(0) 10

　　扇区号：(00 0001) 2 = (1) 10

　　柱面号：(11 0111 1100) 2 = (892) 10

　　所以E盘的起始扇区为（892柱面，0磁头，1扇区）。

　　3. (05): 表示E盘的在扩展分区里面。

　　4．(FE FF FF)：

　　(FE) 16 = (254) 10 (FF) 16 = (1111 1111) 2 (FF) 16 = (1111 1111) 2

　　磁头号：(254) 10

　　扇区号：(11 1111) 2 = (63) 10

　　柱面号：(11 1111 1111) 2 = (1023) 10

　　但这是不准确的，原因同上。

　　5．(FE A3 5D 00):

　　反向，(00 5D A3 FE) 16 = (6136830) 10. 这一项非常重要，它定位了E盘分区表所在扇区。其值为E盘分区表所在扇区号与扩展分区起始扇区号之差。所以，E盘分区表所在扇区号为：

　　8193150 + 6136830 = 14329980。即（892柱面，0磁头，1扇区）。

　　6．(21 88 56 00):

　　反向，(00 56 88 21) 16 = (5670945) 10.表示E盘共有11807775个扇区。通过上面得到的起点和分区的大小，可以推导出E盘的结束位置：14329980 + 5670945 = 20000925号扇区，即（1244 柱面，254磁头，63扇区）。

五、查找E盘分区表

　　根据上面的信息，第三个分区表，也即E盘分区表在（892柱面，0磁头，1扇区）处。读取该扇区，得到分区表如下：

　　[00 01 C1 7C 0B FE FF FF 3F 00 00 00 E2 87 56 00]

　　[ 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ]

　　[00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ]

　　[00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ]

　　该分区表第二项全为0，说明没有下一个分区表了。该分区表就是分区表链的最后一个节点。

　　第一项：

　　（00）（01 C1 7C）（0B）（FE FF FF）（3F 00 00 00 ）（E2 87 56 00）

　　该项与D盘分区表的第二项描述的都是E盘的情况，但它们在某些细节上又有所区别。

　　1.（00）：表示E盘不是活动分区。

　　2.（01 C1 7C）：

　　(01) 16 = (1) 10 (C1) 16 = (1100 0001) 2 (7C) 16 = (0111 1100) 2.

　　磁头号：(1) 10

　　扇区号：(00 0001) 2 = (1) 10

　　柱面号：(11 0111 1100) 2 = (892) 10

　　故E盘起始扇区为（892柱面，1磁头，1扇区）。

　　3.（0B）：表示E盘的文件系统为FAT32。

　　4．（FE FF FF）：

　　(FE) 16 = (254) 10 。 (FF) 16 = (1111 1111) 2 (FF) 16 = (1111 1111) 2.

　　磁头号：(254) 10；

　　扇区号：(11 1111) 2 = (63) 10.

　　柱面号：(11 1111 1111) 2 = (1023) 10

　　但这是不准确的，原因同上。

　　5.（3F 00 00 00）：

　　反向，(00 00 00 3F) 16 = (63) 10.这一项与D盘分区表相应项有所不同.为E盘起始逻辑 　　扇区号（892柱面，1磁头，1扇区）与（892柱面，0磁头，1扇区）之差。表示E盘前面已有63个扇区, 这63个扇区为系统隐藏扇区。操作系统无法对这些扇区进行读写,所以可以把自己的秘密信息写在这里.

　　6. (E2 87 56 00):

　　反向，(00 56 87 E2) 16 = (5670882) 10。表明E盘有5670882个扇区。而D盘分区表相应项为5670945. 5670945 - 5670882 = 63.

　　正好等于63个隐藏扇区.这是因为D盘分区表描述的是(892柱面,0磁头，1扇区) 到 (1244 柱面，254磁头，63扇区) 之间的扇区数。而E盘分区表描述的是(892柱面, 1磁头,1扇区) 到 (1244 柱面，254磁头，63扇区) 之间的扇区数。

**第二节FAT的详细存储**

设计

一个FAT分区的构成概况

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 保留扇区 | | FAT区域 | | 数据区域 | |
| 引导扇区  (BPB+  引导代码) | | 其他  保护扇区 | FAT表1 | FAT表2 | 根目录 | 数据区 | |
| 可选 | 12Bit/16/32 | 12Bit/16/32 | 仅12/16 | 目录和数据 | |
| 1个扇区  0000-01FF | | N扇区(N>=0) | 9扇区  0200-13FF | 同FAT表1  1400-25FF | 224(E0)  14扇区  2600-41FF | F6 | |

引导扇区的结构：

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Jmp3 | | OEMNm8 | | | | BPS2 | | CPS1 | 保留扇区数2 |
| F1 | 目2 | TS | M | Fat扇2 | SPT2 | NH2 | Hid Sec4 | | |
| SectorsBig4 | | D1 | R1 | B1 | VolID4 | VolLab11 | | | |
| VolLab11 | | | | FileSysType8 | | | | | 引导代码448 |
| 引导代码448 | | | | | | | | | |
| 引导代码448 | | | | | | | | | EndFlag2 |

根目录的数据项结构：

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 文件名称8 | | | | 扩展3 | | 属性1 | 保留1 | 创建时间1 | 创建时间HMS2 |
| 创建日期YMD2 | 最后访问日期2 | ea Index2 | 最终修改时间2 | 最终修改 日期2 | 起始簇号2 | | 文件大小4 | | |

主磁盘结构

　　一个FAT文件系统包括四个不同的部分。

**保留扇区**，位于最开始的位置。第一个保留扇区是**引导扇区**（分区启动记录）。它包括一个称为**基本输入输出参数块(BPB)**的区域（包括一些基本的文件系统信息尤其是它的类型和其它指向其它扇区的指针），通常包括操作系统的启动调用代码。**保留扇区的总数记录在引导扇区中的一个参数中。**引导扇区中的重要信息可以被DOS和OS/2中称为驱动器参数块的操作系统结构访问。

**FAT区域**。它包含有两份文件分配表，这是出于系统冗余考虑，尽管它很少使用，即使是磁盘修复工具也很少使用它。它是分区信息的映射表，指示簇是如何存储的。

**根目录区域**。它是在根目录中存储文件和目录信息的目录表。在FAT32下它可以存在分区中的任何位置，但是在早期的版本中它永远紧随FAT区域之后。

**数据区域**。这是实际的文件和目录数据存储的区域，它占据了分区的绝大部分。通过简单地在FAT中添加文件链接的个数可以任意增加文件大小和子目录个数（只要有空簇存在）。然而需要注意的是每个簇只能被一个文件占有，这样的话如果在32KB大小的簇中有一个1KB大小的文件，那么31KB的空间就浪费掉了。

例外情况

Apricot PC的MS-DOS所用FAT的实现有一个不同的启动扇区组织以使用计算机与IBM不兼容的基本输入输出系统。跳转指令和OEM名被省略并且MS-DOS文件系统参数位于0x50（在标准扇区中偏移为0x0B - 0x17）。后来的Apricot MS-DOS版本除了Apricot特有的引导区之外也具有了读写标准启动分区的能力。

BBC Master 512 上的DOS Plus根本就不使用传统的引导区。数据磁盘省略了引导区并且以一个单份的FAT开始（FAT的第一个字节用来确定磁盘容量），启动磁盘使用一个包含启动调用程序的小型ADFS文件系统，后面跟随一个单份的FAT。

**文件分配表**

　　一个分区分成同等大小的簇，也就是连续空间的小块。簇的大小随着FAT文件系统的类型以及分区大小而不同，典型的簇大小介于2KB到32KB之间。每个文件根据它的大小可能占有一个或者多个簇；这样，一个文件就由这些这些（称为单链表）簇链所表示。然而，这些链并不一定一个接着一个在磁盘上存储，它们经常是在整个数据区域零散的储存。

文件分配表（FAT）是映射到分区每个簇的条目列表。

每个条目记录下面五种信息中的一种。

1.链中下一个簇的地址

2.一个特殊的文件结束符（EOF）符号指示链的结束

3.一个特殊的符号标示坏簇

4.一个特殊的符号标示保留簇

5.0来表示空闲簇

每个版本的FAT文件系统使用不同大小的FAT条目。这个大小已经由名字表示出来，例如FAT16文件系统的每个条目使用16位表示，32位文件系统使用32位表示。这个不同意味着FAT32系统的文件分配表能比FAT16映射更多的簇，它也允许FAT32有更大的分区大小。这也使得FAT32比FAT16更能有效地利用磁盘空间，因为每个驱动器能够寻址更小的簇，这也就意味着更少的空间浪费。

目录表

　　目录表是一个表示目录的特殊类型文件（现今通常称为文件夹）。它里面保存的每个文件或目录使用表中的32位条目表示。每个条目记录名字、扩展名、属性（档案、目录、隐藏、只读、系统和卷）、创建的日期和时间、文件/目录数据第一个簇的地址，最后是文件/目录的大小。

除了FAT12和FAT16文件系统中的根目录表占据特殊的根目录区域位置之外，所有其它的目录表都存在数据区域。

合法的DOS文件名包括下面一些字符：

大写字母A-Z

数字0-9

空格（尽管结尾的空格被作为填充而不是文件名的一部分）

! # $ % & ( ) - @ ^ \_ ` { } ~ '

数值 128-255

DOS文件名位于OEM字符集。

长文件名（LFN）使用一个技巧存储在FAT文件系统上——在目录表中添加假的条目。这些条目使用一个普通文件无法使用的卷标属性标识，普通文件无法使用是由于它们被大多数旧的MS-DOS程序忽略。很显然，一个只包含卷标的目录被当作空卷，这样就允许删除；使用长文件名创建的文件在从普通的DOS删除就会发生这样的情形。

校验和也允许检验长文件名是否与8.3文件名匹配；当一个文件删除之后使用DOS在同一个目录位置重新创建之后就会出现不匹配现象。校验和使用下面的算法计算。（注意pFcbName是指向如正常目录条目中所显示的文件名的指针，例如前八个字符是文件名，最后三个是扩展名。点是隐含的。文件名中没有使用的空间将使用空格（ASCII 0x20）补齐。例如，“Readme.txt”将记录为"README TXT"。

unsigned char lfn\_checksum(const unsigned char \*pFcbName) {

int i;

unsigned char sum=0;

for (i=11; i; i--)

sum = ((sum & 1) ? 0x80 : 0) + (sum >> 1) + \*pFcbName++;

return sum;

}

旧版的PC-DOS错误地将根目录中的长文件名当作卷标，这样它们就会显示错误的卷标。 　　每个假条目包含13UTF-16个字符（26字节），通过使用包含文件大小或者时间记录的区域获得除了旧的8+3之外的另外15个字节（但是出于安全和磁盘检查工具的考虑开始簇的区域没有使用保留值为0）。参见8.3中另外的解释。

|  |  |  |
| --- | --- | --- |
| **字节偏移** | **长度** | **描述** |
| 0x00 | 1 | 串行号 |
| 0x01 | 10 | 名称字符（5个UTF-16字符） |
| 0x0b | 1 | 属性（永远是0x0F） |
| 0x0c | 1 | 保留（永远是0x00） |
| 0x0d | 1 | DOS文件名校验和 |
| 0x0e | 12 | 名称字符（6个UTF-16字符） |
| 0x1a | 2 | 第一个簇（永远是0x0000） |
| 0x1c | 4 | 名称字符（两个UTF-16字符） |

如果一个文件名只包含小写字母、或者是一个小写字母的名加上大写扩展名的混合或者与此相反，没有特殊的字符并且满足8.3的限制，在视窗NT上就不创建VFAT的条目。相反，在目录条目的偏移0x0c处的没有说明的位用来指示文件名全部或者部分是小写字母。特别明确的是，位4意味着小写字母的扩展名，位3意味着名是小写字母，这样就允许如“example.TXT”和“HELLO.txt”这样的组合，但是不允许“Mixed.txt”这样的组合。很少有操作系统支持这种功能。非NT视窗版本当这个扩展使用时将把文件名当作大写字母。缺省情况下，Linux的最近版本将认识这个扩展但是在写时并不使用它。

需要说明的是：

1．引导扇区和其他保留扇区一起称为保留扇区，而其他保留扇区是可选的，当没有时候，引导扇区后紧跟的就是FAT表1

2．根目录区是仅FAT12/16才有，FAT32的目录项位于数据区。由于FAT12/16的根目录区是一个固定的区域，所以它的根目录的项数是有限制的，意即不能在根录建立超过这个定数的目录项数。

（一）引导扇区与BPB

BPB（BIOS Parametre Block）是FAT文件系统中第一个重要的数据结构，它位于该FAT分区的第一个扇区，同时也属于FAT文件系统基本区域的保留区，在下面的描述中。凡名称以BPB\_开头的都是BPB的一部分，凡名称与BS\_开头的项都是启动扇区的一部分，而不是属于BPB的内容，以下是启动扇区的结构

|  |  |  |  |
| --- | --- | --- | --- |
|  | offset（byte） | 长度（byte） | 描述 |
| BS\_jmpBoot | 0x00 | 3 | 跳转指令，指向启动代码 |
| BS\_OEMName | 0x03 | 8 | 建议值为“MSWIN4.1”。有些厂商的FAT驱动可能会检测此项，所以设为“MSWIN4.1”可以尽量避免兼容性的问题 |
| BPB\_BytsPerSec | 0x0b | 2 | 每扇区的字节数，取值只能是以下几种：512，1024，2048或是4096。设为512会取得最好的兼容性，目前有很多FAT代码都是硬性规定每扇区的字节数为512，而不是实际的检测此值。但微软的操作系统能够很好支持1024，2048或是4096 |
| BPB\_SecPerClus | 0x0d | 1 | 每簇的扇区数，其值必须中2的整数次方（该整数必须>=0），同时还要保证每簇的字节数不能超过32K，也就是1024\*32字节 |
| BPB\_RsvdSecCnt | 0x0e | 2 | 保留扇区的数目，此域不能为0，FAT12/FAT16必须为1，FAT32的典型值取为32，，微软的系统支持任何非0值 |
| BPB\_NumFATs | 0x10 | 1 | 分区中FAT表的份数，任何FAT格式都建议为2 |
| BPB\_RootEntCnt | 0x11 | 2 | 对于FAT12和FAT16此域包含根目录中目录的个数（每项长度为32bytes），对于FAT32，此项必须为0。对于FAT12和FAT16，此数乘以32必为BPB\_BytesPerSec的偶数倍，为了达到更好的兼容性，FAT12和FAT16都应该取值为512 |
| BPB\_ToSec16 | 0x13 | 2 | 早期版本中16bit的总扇区，这里总扇区数包括FAT卷上四个基本分区的全部扇区，此域可以为0，若此域为0，那么BPB\_ToSec32必须为0，对于FAT32，此域必为0。对于FAT12/FAT16，此域填写总扇区数，如果该值小于0x10000的话，BPB\_ToSec32必须为0 |
| BPB\_Media | 0x15 | 1 | 对于“固定”（不可移动）存储介质而言，0xF8是标准值，对于可移动存储介质，经常使用的数值是0xF0，此域合法的取值可以取0xF0,0xF8,0xF9,0xFA,0xFC,0xFD,0xFE,0xFF。另外要提醒的是，无沦此域写入什么数值，同时也必须在FAT[0]的低字节写入相同的值，这是因为早期的MSDOS 1.x使用该字节来判定是何种存储介质 |
| BPB\_FATz16 | 0x16 | 2 | FAT12/FAT16一个FAT表所占的扇区数，对于FAT32来说此域必须为0，在BPB\_FATZ32中有指定FAT表的大小 |
| BPB\_SecPerTrk | 0x18 | 2 | 每磁道的扇区数，用于BIOS中断0x13，此域只对于有“特殊形状”（由磁头和柱面每分割为若干磁道）的存储介质有效，同时必须可以调用BIOS的0x13中断得到此数值 |
| BPB\_NumHeads | 0x1A | 2 | 磁头数，用于BIOS的0x13中断，类似于上面的BPB\_ SecPerTrk，只对特殊的介质才有效，此域包含一个至少为1的数值，比如1,4M的软盘此域为2 |
| BPB\_HidSec | 0x1C | 4 | 在此FAT分区之前所隐藏的扇区数，必须使得调用BIOS的0x13中断可以得到此数值，对于那些没有分区的存储介质，此域必须为0，具体使用什么值由操作系统决定 |
| BPB\_ToSec32 | 0x20 | 4 | 该卷总扇区数（32bit），这里的扇区总数包括FAT卷四个个基本分的全部扇区，此域可以为0，若此域为0，BPB\_ToSec16必须为非0，对FAT32，此域必须是非0。对于FAT12/FAT16如果总扇区数大于或等于0x10000的话，此域就是扇区总数，同时BPB\_ToSec16的值为0。 |

FAT32的BPB的内容和FAT12/16的内容在地址36以前是完全一样的，从偏移量36开始，他们的内容有所区别，具体的内容要看FAT类型为FAT12/16还是FAT32，这点保证了在启动扇区中包含一个完整的FAT12/16或FAT32的BPB的内容，这么做是为了达到最好的兼容性，同时也为了保证所有的FAT文件系统驱动程序能正确的识别和驱动不同的FAT格式，并让他们良好地工作，因为他们包含了现有的全部内容

从offset 36开始FAT12/FAT16的内容开始区别于FAT32，下面分两个表格列出，下表为FAT12/FAT16的内容

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | offset（byte） | 长度（byte） | 描述 |
| BS\_drvNum | 0x24 | 1 | 用于BIOS中断0x13得到磁盘驱动器参数，（0x00为软盘，0x80为硬盘）。此域实际上由操作系统决定 |
| BS\_Reseved1 | 0x25 | 1 | 保留（供NT使用），格式化FAT卷时必须设为0 |
| BS\_VolID | 0x26 | 1 | 扩展引导标记（0x29）用于指明此后的3个域可用 |
| BS\_BootSig | 0x27 | 4 | 卷标序列号，此域以BS\_VolLab一起可以用来检测磁盘是否正确，FAT文件系统可以用此判断连接的可移动磁盘是否正确，引域往往是由时间和日期组成的一个32位的值 |
| BS\_VolLab | 0x2B | 11 | 磁盘卷标，此域必须与根目录中11字节长的卷标一致。 |
| FAT文件系统必须保证在根目录的卷标文件列改或是创建的同时，此域的内容能得到时的更新，当FAT卷没有卷标时，此域的内容为“NO NAME” |
| BS\_FilSysType | 0x36 | 8 | 以下的几种之一：“FAT12”，“FAT16”，“FAT32” |
| 不少人错误的认为FAT文件系统的类型由此域来确认，他细点你就能发现此域并不是BPB的一部分，只是一个字符串而已，微软的操作系统并不使用此此域来确定FAT文件的类型，；因为它常常被写错或是根本就不存在。 |

下表为FAT32的内容

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | offset（byte） | 长度（byte） | 描述 |
| BPB\_FATSz32 | 0x24 | 4 | 一个FAT表所占的扇区数，此域为FAT32特有，同时BPB\_FATSz16必须为0 |
| BPB\_Flags | 0x28 | 2 | 此域FAT32特有。 |
| Bits0-3:不小于0的FAT（active FAT）数目，只有在镜像（mirrorig）禁止时才有效。 |
| Bits 4-6: 保留 |
| Bits7： 0表示FAT实时镜像到所有的FAT表中 |
| 1 表示只有一个活动的FAT表。这个表就是Bits0-3所指定的那个 |
| Bits8-15：保留 |
| BPB\_FSVer | 0x2A | 2 | 此域为FAT32特有， |
| 高位为FAT32的主版本号，低位为次版本号，这个版本号是为了以后更高级的FAT版本考虑，假设当前的操作系统只能支持的FAT32版本号为0.0。那么该操作系统检测到此域不为0时，它便会忽略FAT卷，因为它的版本号比系统能支持的版式本要高 |
| BPB\_RootClus | 0x2C | 4 | 根目录所在第一个簇的簇号，通常该数值为2，但不是必须为2 |
| 磁盘工具在改变根目录位置时，必须想办法让磁盘上第一个非坏簇作为根目录的第一个簇（比如第2簇，除非它已经被标记为坏簇），这样的话，如果此域正好为0的话磁盘检测工具也能轻松的找到根目录所在簇的位置 |
| BPB\_FSIfo | 0x30 | 2 | 保留区中FAT32卷FSINFO结构所占的扇区数，通常为1 |
| 在Backup Boot 中会有一个FSINFO的备份，但该备份只是更新其中的指针，也就是说无论是主引导记录还是备份引导记录都是指向同一个FSINFO结构 |
| BPB\_\_BkBootSec | 0x32 | 2 | 如果不为0，表示在保留区中引导记录的备数据所占的扇区数，通常为6。同时不建议使用6以外的其他数值 |
| BPB\_Reserved | 0x34 | 12 | 用于以后FAT扩展使用，对FAT32。此域用0填充 |
| BS\_DrvNum | 0x40 | 1 | 与FAT12/16的定义相同，只不过两者位于启动扇区不同的位置而已 |
| BS\_Reserved1 | 0x41 | 1 | 与FAT12/16的定义相同，只不过两者位于启动扇区不同的位置而已 |
| BS\_BootSig | 0x42 | 1 | 与FAT12/16的定义相同，只不过两者位于启动扇区不同的位置而已 |
| BS\_VolID | 0x43 | 4 | 与FAT12/16的定义相同，只不过两者位于启动扇区不同的位置而已 |
| BS\_FilSysType | 0x47 | 11 | 与FAT12/16的定义相同，只不过两者位于启动扇区不同的位置而已 |
| BS\_FilSysType | 0x52 | 8 | 通常设置为“FAT32”，请参照FAT12/16此部分的陈述。 |

关于FAT启动扇区还有一点重要的说明，我们假设里面的内容是按字节排序的，那么扇区[510]的内容一定0x55，扇区[511]的内容一定是0xAA。

很多FAT资数文档会把0xAA55说成是“启动扇区最后两字节的内容”，这样的陈述是正确的—仅仅是如果—BPB\_BytsPerSec的值为512的话。若BPB\_BytsSec的值大于512，该标记的位置并没有改变（虽然在启动扇区的最后两个字节写0xAA55并没有问题）。

关于BPB\_ToSec16/32这里再作一点补充：假设一现在我们有一块磁盘或一个分区，它的扇区数为DskSz，如果BPB\_aToSec(BPB\_ToSec16或是BPB\_ToSec32基中不为0的那个)的值小于或等于DskSz并不会使该FAT卷在使用中出现什么错误，实际上BPB\_ToSec16/32的值不要比DskSz小得离谱就不会有什么错误。

这样做将造成磁盘空间的浪费，程序本身并不会认为该FAT卷存在什么错误，但是，如果BPB\_ToSec16/32的值比DskSz大将会使FAT卷遭到严重的损坏，因为它超出了存储介质或是磁盘分区的边界。当BPB\_ToSec16/32的值比DskSz大时，一些数据将不幸地被丢失.

第三方扩展

　　在微软公司添加长文件名和创建/访问时间戳之前，其它的操作系统使用目录表字节0x0C-0x15存储其它的元数据。

FAT授权

专利

　　在二十世纪九十年代中期微软公司申请并获得了一系列的FAT文件系统内核部分的专利。由于具有广泛的兼容性和易于理解，FAT经常被选为用于数码相机和个人数码助理中闪存的数据交换格式。

2003年12月3日微软公司宣布使用FAT规范和“相关智慧产权”将需要得到授权，被个销售单元支付0.25美元的版权费，每个授权协议最多250,000美元的版权费。

为了这个目的，微软公司提及了四个关于FAT文件系统的专利作为它的知识产权主张的基础。所有这些与长文件名扩展有关的FAT首先出现在Windows 95中：

美国专利 5,745,902 - 访问使用不同文件名格式的文件名的方法和系统。1992年7月6日备案。这包括生成、联系一个8.3兼容格式的短文件名和长文件名的方法（如“Microsoft.txt”和“MICROS~1.TXT”），以及列举相互冲突的短文件名的方法（如“MICROS~2.TXT”和“MICROS~3.TXT”）。现在还不清楚这个专利是否覆盖不具有显式长文件名能力的FAT实现。Unix文件系统中的硬连接看起来不是先行者：从长文件名删除一个FAT文件也将删除它的短文件名。将一个文件重命名为一个“短”文件名也将一致地更改长文件名；同样，将一个文件重命名为“长”文件名也将重新生成一个“短”文件名。在NTFS中，硬连结和两个名字是不同的概念，并且每个硬连结都有两个名字。最后，在API的层面上，当在系统中进行目录搜索时两个文件名都会出现；它们看起来不是两个独立的文件并且它们也没有有必要去“映射”确定同一个文件。

美国专利 5,579,517 - 长、短文件名公用的命名空间。1995年4月24日备案。这包括将多个连续8.3目录条目链接在一起支持长文件名的方法，其中一些条目特殊进行标记阻止可能引起混淆地早期的不支持长文件名的FAT实现。

公共专利基金会成功地对这项专利发起了挑战；这个专利申请由于所申请的技术在专利美国专利 5,307,494 和 美国专利 5,367,671中的先期发现在2004年9月14日被驳回]。这个决定后来在2006年1月10日被美国专利局所推翻。

美国专利 5,758,352 - 长、短文件名公用的命名空间。1996年9月5日备案。它非常类似于5,579,517。

公共专利基金会成功地对这项专利发起了挑战；美国专利商标局基于"the six assignees names were incorrect"在2005年10月5日驳回了这项专利。这个决定也在后来的2006年1月10日被美国专利局推翻。

美国专利 6,286,013 - 在操作系统中为长、短文件名提供一个公用的命名空间的方法和系统。1997年1月28日备案。它所申请的内容包括Windows 95、Windows 98和Windows Me的长文件名提供给它们MS-DOS兼容层所用的方法。它看起来不影响非微软的FAT实现。 　　许多技术评论断言这些专利仅仅涵盖了支持长文件名的FAT实现，那些只使用短名字的移动固态媒体和消费设备将不受影响。

另外，在微软2000年12月6日出版的"Microsoft Extensible Firmware Initiative FAT 32 File System Specification, FAT: General Overview of On-Disk Format"，微软公司明确地给出了一些授权，许多读者将它认为是微软允许操作系统厂商实现FAT。

控诉

　　由于人们广泛要求重新审查这些专利，公众专利基金会向美国专利和商标局（USPTO）提出了一些证据争辩这些专利的有效性，其中包括施乐公司和IBM的早期参考资料。美国专利商标局承认这些证据提出了“可专利性的实质性的新问题”并且对于微软公司FAT专利的有效性展开调查。

2004年9月30日，美国专利商标局主要基于公共专利基金会所提供的证据驳回了美国专利 5,579,517的专利主张。这个基金会的执行总裁Dan Ravicher说“现在专利局只不过是确认了我们已经知道了一段时间的事情，微软公司的专利是假的。”

PUBPAT的新闻发布会说，“微软公司仍然有机会回应专利局的驳回。有代表性的是第三方的重新审查要求如PUBPAT提供的资料成功地减小了专利的范围或者有70%的机会完全驳回专利。”

2005年10月5日，专利局宣布随着调查的深入它驳回了专利5.579,517的专利主张，另外它发现专利美国专利 5,758,352有错误的专利受益人而无效。

最后在2006年1月10日，专利局裁定微软公司的FAT系统的实现特点是“新颖和非显然的”，推翻了早期的两个非最终裁决。

注解

专利申请的是在文件系统中支持长文件名的技术，而不是文件系统内核本身。

Duncan, Ray (1989). Design goals and implementation of the new High Performance File System. Microsoft Systems Journal 4 (5).

Thompson, Robert Bruce and Barbara Fritchman Thompson, PC Hardware in a Nutshell, 3rd Edition,, O'Reilly, ISBN 059600513X (p. 506 re Microsoft "bizarrely" saying 32 GB limitation is by design

Limitations of FAT32 File System, Microsoft knowledge base article 184006

Norton, Peter (2002页面文件)

FAT32和NTFS相互转化

一.FAT32转NTFS

　　在开始——运行中输入CMD ，再输入 convert X：/FS:NTFS(X：是所要转化的分区盘符)重新启动之后就开始转化了。不过这样的转化方式有三个弊端：

1.其实也不能算是弱点，主要是转化为NTFS之后，系统默认Everyone用户对所有目录、文件拥有完全权限，其实FAT32本来就默许所有用户访问!为了安全起见，需要重新设置安全特性。对一些重要文件，应该取消everyone用户的完全权限。

2.转化后的NTFS分区，第一次可能很容易产生磁盘碎片。

3.C盘在转换为NTFS分区前，一定要注意是不是多系统的，有没有安装WIN9X、ME等不认识NTFS分区的系统，如果安装了，那么C盘千万不能转换为NTFS分区，一定要保持原来的FAT分区。扩展分区里的逻辑分区则除了安装有WIN9X/ME系统的分区外，其他的分区都可以转换为NTFS分区。

二.NTFS转FAT32

　　目前为止，比较常用的方法就是用PQ Magic进行转化(也可用Acronis公司的Partation Expert，英文版，但完全支持中文)，但是一定要注意，如果有中文目录、文件，一定要集中起来用Winrar打包成英文名称的压缩包，否则这些中文目录、文件都将无法访问!!!

常见问题处理

　　1.中途断电，分区成了PQRP类型，无法访问，怎么办?

可以利用PQ安装目录下的PTEDIT32.exe(\PowerQuest\PartitionMagic 8.0)进行修复，只需将标志为PQRP的分区(类型为3C)，转化为FAT32/NTFS(类型为07/0B)即可。如果是C盘出错，可以借助实模式 DOS加载DOS版的PTEDIT.exe(\PowerQuest\PartitionMagic 8.0\DOS)进行修复。注：其他PQ操作断电导致的问题也可以依法炮制，而且相对于Diskgenius，PTEDIT32.exe可以支持NTFS 分区。

2.转化分区后，中文目录无法访问，怎么办?

可以利用GHOST克隆问题分区，然后用ghost Explorer打开gho备份文件，重命名中文目录即可。

FAT文件系统数据结构C语言表示

FAT文件系统相关数据结构

struct fat\_boot\_sector {

uint8\_t ignored[3]; /\* 0x00 Boot strap short or near jump \*/

int8\_t system\_id[8]; /\* 0x03 Name - can be used to special case

partition manager volumes \*/

uint8\_t sector\_size[2]; /\* 0x0B bytes per logical sector \*/

uint8\_t sectors\_per\_cluster; /\* 0x0D sectors/cluster \*/

uint16\_t reserved; /\* 0x0E reserved sectors \*/

uint8\_t fats; /\* 0x10 number of FATs \*/

uint8\_t dir\_entries[2]; /\* 0x11 root directory entries \*/

uint8\_t sectors[2]; /\* 0x13 number of sectors \*/

uint8\_t media; /\* 0x15 media code (unused) \*/

uint16\_t fat\_length; /\* 0x16 sectors/FAT \*/

uint16\_t secs\_track; /\* 0x18 sectors per track \*/

uint16\_t heads; /\* 0x1A number of heads \*/

uint32\_t hidden; /\* 0x1C hidden sectors (unused) \*/

uint32\_t total\_sect; /\* 0x20 number of sectors (if sectors == 0) \*/

/\* The following fields are only used by FAT32 \*/

uint32\_t fat32\_length; /\* 0x24=36 sectors/FAT \*/

uint16\_t flags; /\* 0x28 bit 8: fat mirroring, low 4: active fat \*/

uint8\_t version[2]; /\* 0x2A major, minor filesystem version \*/

uint32\_t root\_cluster; /\* 0x2C first cluster in root directory \*/

uint16\_t info\_sector; /\* 0x30 filesystem info sector \*/

uint16\_t backup\_boot; /\* 0x32 backup boot sector \*/

uint8\_t BPB\_Reserved[12]; /\* 0x34 Unused \*/

uint8\_t BS\_DrvNum; /\* 0x40 \*/

uint8\_t BS\_Reserved1; /\* 0x41 \*/

uint8\_t BS\_BootSig; /\* 0x42 \*/

uint8\_t BS\_VolID[4]; /\* 0x43 \*/

uint8\_t BS\_VolLab[11]; /\* 0x47 \*/

uint8\_t BS\_FilSysType[8]; /\* 0x52=82\*/

/\* \*/

uint8\_t nothing[420]; /\* 0x5A \*/

uint16\_t marker;

} \_\_attribute\_\_ ((\_\_packed\_\_));

struct msdos\_dir\_entry {

int8\_t name[8],ext[3]; /\* 00 name and extension \*/

uint8\_t attr; /\* 0B attribute bits \*/

uint8\_t lcase; /\* 0C Case for base and extension \*/

uint8\_t ctime\_ms; /\* 0D Creation time, milliseconds \*/

uint16\_t ctime; /\* 0E Creation time \*/

uint16\_t cdate; /\* 10 Creation date \*/

uint16\_t adate; /\* 12 Last access date \*/

uint16\_t starthi; /\* 14 High 16 bits of cluster in FAT32 \*/

uint16\_t time; /\* 16 time, date and first cluster \*/

uint16\_t date; /\* 18 \*/

uint16\_t start; /\* 1A \*/

uint32\_t size; /\* 1C file size (in bytes) \*/

};

/\* Up to 13 characters of the name \*/

struct msdos\_dir\_slot {

uint8\_t id; /\* 00 sequence number for slot \*/

uint8\_t name0\_4[10]; /\* 01 first 5 characters in name \*/

uint8\_t attr; /\* 0B attribute byte \*/

uint8\_t reserved; /\* 0C always 0 \*/

uint8\_t alias\_checksum; /\* 0D checksum for 8.3 alias \*/

uint8\_t name5\_10[12]; /\* 0E 6 more characters in name \*/

uint16\_t start; /\* 1A starting cluster number, 0 in long slots \*/

uint8\_t name11\_12[4]; /\* 1C last 2 characters in name \*/

};

簇

从统计学上讲，平均每个文件浪费0.5簇的空间，簇越大，存储文件时空间浪费越多，利用率越低。因此，簇的大小决定了该盘数据区的利用率。FAT16系统簇号用16位二进制数表示，从0002H到FFEFH个可用簇号(FFF0H到FFFFH另有定义,用来表示坏簇，文件结束簇等)，允许每一逻辑盘的数据区最多不超过FFEDH(65518)个簇。FAT32系统簇号改用32位二进制数表示，大致从00000002H到FFFFFEFFH个可用簇号。FAT表按顺序依次记录了该盘各簇的使用情况，是一种位示图法。

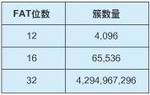
每簇的使用情况用32位二进制填写，未被分配的簇相应位置写零；坏簇相应位置填入特定值；已分配的簇相应位置填入非零值，具体为：如果该簇是文件的最后一簇，填入的值为FFFFFF0FH，如果该簇不是文件的最后一簇，填入的值为该文件占用的下一个簇的簇号，这样，正好将文件占用的各簇构成一个簇链，保存在FAT表中。0000000H、00000001H两簇号不使用，其对应的两个DWORD位置(FAT表开头的8个字节)用来存放该盘介质类型编号。FAT表的大小就由该逻辑盘数据区共有多少簇所决定，取整数个扇区。

**第三节 簇的概念**

簇的概念

磁盘上最小可寻址存储单元称为扇区，通常每个扇区为512个字节(或字符)。由于多数文件比扇区大得多，因此如果对一个**文件分配最小的存储空间**，将使存储器能存储更多数据，这个最小存储空间即称为**簇**。根据存储设备(磁盘、闪卡和硬盘)的容量，簇的大小可以不同以使存储空间得到最有效的应用。在早期的360KB磁盘上，簇大小为2个扇区(1,024字节)；第一批的10MB硬盘的簇大小增加到8个扇区(4,096字节)；现在的小型闪存设备上的典型簇大小是8KB或16KB。2GB以上的硬盘驱动器有32KB的簇。表2列出在给定簇大小和FAT类型的最大分区。

表2



**为什么采用簇呢？**通常，存储设备上的空间分配是随机的。在一个新存储设备上，文件连续存储，并知道开始和结束扇区和长度，在读取时可以根据这些信息重新得到所存储的文件。但是，过一段时间后，有些文件将会被擦掉，同时可能有些文件增大，这时不能保证同一个文件存储在连续的一系列扇区里。因此，需要一种方法来辨别哪个扇区被分配到某些文件，以及还有哪些扇区可用。这时可以采用一种表结构（FAT）来实现这种功能，使每个扇区对应一个表记录。**然而，由于大多数文件存储在多个扇区，一个记录对应一个扇区的作法将造成一些浪费。**因此，采用每个记录代表一个固定数量扇区将更有意义，这个固定扇区被称为簇。

给文件分配扇区

如果给一个**文件分配的最小存储空间(簇)**大小取决于保存该文件的介质类型，我们必须有一种方法来标明某个簇是否可用或者已分配给一个文件。如果已将一个簇分配给一个文件，我们必须知道这个簇分配给文件的哪部分了，这就是通过一个**链接列表**来实现，即文件分配表(FAT)。FAT仅仅是一个包含N个整数的列表，N是存储设备上最大的簇数。表中每个记录的位数称为FAT大小，是12、16或32三个数之一。早期的存储设备使用12位(1.5字节)FAT以减少浪费，12位可以提供4,096个簇。假定一个簇具有两个扇区(1,024字节)，则代表存储设备将近有4.2MB存储容量。更大的簇就能实现更大的存储空间，**但是在存储较小文件时，由于簇没有完全装满而将造成存储空间的浪费。**

操作系统根据表现整个磁盘空间所需要的簇数量来确定使用多大的FAT。如果磁盘要求的簇少于4,096个，则可以使用12位FAT；如果小于65,536但大于4,096个簇，就使用16位FAT；否则，必须采用32位FAT。在存储设备的第一个扇区内以数据列表的形式定义了簇的大小，该数据列表被称为BIOS参数块。在系统引导期间操作系统可以读该扇区，这样就能设定如何从存储设备中读取文件。

FAT是一种链接列表，链接列表中相关记录之间互相指向对方。在存储设备目录中包含了FAT表名称、该文件大小和分配给该文件的第一个簇的编号。存储该文件的第一个簇所对应的表记录中包含了该文件的第二个簇号码。同样，第二个簇对应的记录里包含有存储该文件的第三个簇编号，以此类推直到该文件的最后一个簇。在新存储设备上存储的第一个文件将保存在连续的簇内，因此第一个簇会指向第二个，第二个指向第三个，如此类推。

实际上，第一个簇(簇0)总是保留用于存储操作系统信息、根目录和两份FAT。系统建立两份FAT，其目的是在当修改其中一个时如果系统发生中断(崩溃)，另一份还完整无缺，恢复程序会检查FAT并使用这份FAT来恢复。

FAT16各分区与簇大小的关系如下表：

|  |  |
| --- | --- |
| 分区大小 | FAT16簇大小 |
| 16MB-127MB | 2KB |
| 128MB-255MB | 4KB |
| 256MB-511MB | 8KB |
| 512MB-1023MB | 16KB |
| 1024MB-2047MB | 32KB |

如果你在一个1000MB的分区中存放50KB的文件，由于该分区簇的大小为16KB，因此它要用到4个簇才行。而如果是一个1KB的文件，它也必须使用一个簇来存放。那么每个簇中剩下的空间可否拿来使用呢？答案是不行的，所以在使用磁盘时，无形中都会或多或少损失一些磁盘空间。

由上可知，FAT16文件系统有两个最大的缺点：

（１）磁盘分区最大只能到2GB。当前只要你添购计算机的话，想必其中的硬盘大小必定至少有2GB，而3.2GB、4.3GB以上的硬盘比比皆是，且物美价廉。FAT16文件系统已不能适应当前这种大容量的硬盘，必须被迫分区成几个磁盘空间。而分区磁盘的大小又牵扯出簇的问题来，可谓影响颇大。

（２）使用簇的大小不恰当。试想，如果一个只有1KB大小的文件放置在一个1000MB的磁盘分区中，它所占的空间并不是1KB，而是16KB，足足浪费了15KB！当前流行的HTML文件，其大小几乎多为1KB、2KB，而制作一个网站往往用到数十个HTML文件。如果你的硬盘中有100个这种小文件的话，你浪费的磁盘空间可从700KB(511MB的分区)，到3.1MB(2047MB的分区)。

以上这两个问题常常使得用户在“分多大的分区，才能节省空间，同时又可使硬盘的使用更加方便有效”的抉择中徘徊不定。

**第四节 目录项**

FDT 文件目录表（File Directory Table）：

计算机术语　即根目录区，又称为ROOT区，紧跟在FAT2的下一个扇区，长度为32个扇区（**256个表项**）。如果支持长文件名，则每个表项为64个字节，其中，前32个字节为长文件链接说明；后32个字节为文件属性说明，包括文件长度、起始地址、日期、时间等。如不支持长文件名，则每个表项为32个字节的属性说明。

值得注意的是:

**FAT32没有储存目录的目录区，而 FAT16储存根目录并把子目录放到数据区。表示目录的目录项指出根目录地址同时长度字节为0，表示文件的目录项指出数据地址。**

注意：DOS7前的怪字符为E5H，表示被删除，被删除文件仍旧能够找到开始簇，数据恢复就依靠这一特点。

用FORMAT命令对磁盘（或逻辑盘）进行格式化的时候，就已经为整个硬盘建立了一个根目录FDT。在根目录下，用户可以用DOS命令"MD"再创建不同的各个子目录，以及子目录下的子目录。

根目录以及各级子目录都有自己的FDT。

在具体操作中，系统规定用字母C-Z代表逻辑盘符，所以DOS简单地用“[盘符:]\”表示在根目录下。根目录的作用是分配根目录下的所有文件和子目录的存储空间（逻辑扇区号），并且通过设备驱动程序接口确定有效的最大目录项。

1.根目录中的FDT

根目录的作用是分配根目录下的所有文件和子目录的存储空间（逻辑扇区号），并通过设备驱动程序接口确定有效的最大目录项。

根目录下的所有文件及其子目录在根目录的文件目录表（FDT）中都有一个“目录登记项”或简称为“目录项”。每个目录登记项占用32个字节，分为8个区域，提供有关文件或子目录的信息。其中包括了DOS的系统文件（IO.SYS、MSDOS.SYS、和COMMAND.COM）的目录项。

常用软盘的根目录（FDT）所在逻辑扇区如下：

360KB:5--11扇区，共7个扇区

1.2MB:15--28扇区，共14个扇区

1.44MB:19--32扇区，共14个扇区

在硬盘中，各逻辑盘的根FDT的起始逻辑扇区由分区容量确定。在FAT16中，其大小总是32(20H)个扇区。FAT32中的根FDT大小与其逻辑盘容量有关。

下面是用DEBUG读出1.44MB软盘逻辑13H扇区开始的14个扇区内容的例子。

C>DEBUG

-L 00 0 13 E

-D 00

2.FDT中的目录项

根目录下的所有文件及其子目录在根目录的文件目录表（FDT）中都有一个“目录登记项”或简称为“目录项”。

由于每个文件的目录32个字节，用作目录的一个扇区（512字节）最多只能装入512/32=16个文件。因此，在常用磁盘的根目录下最多可建文件或子目录数如下：

360 KB: 共7个扇区， 7\*16=112个文件或子目录。

1.2 MB: 共14个扇区，14\*16=224个文件或子目录。

1.44MB: 共14个扇区，14\*16=224个文件或子目录。

硬盘逻辑盘：共32个扇区，16\*32=512个文件或子目录。

在FDT中，每个目录登记项占用32个字节，分为8个区域，提供有关文件或子目录的信息，包括DOS的系统文件IO.SYS、MSDOS.SYS的目录项。

在FAT16文件系统中，一个文件目录登记项32个字节中个字节的内容及含义如表6-6所示。

表6-6 **FAT16文件目录项**内容及含义表

|  |  |  |
| --- | --- | --- |
| 字节位置 | 内 容 及 含 义 | |
| 第0--7字节 | 表示文件名 | |
| 第8--10字节 | 表示文件的扩展名 | |
| 第11字节 | 表  示  文  件  属  性 | 00H（读写） |
| 01H（只读） |
| 02H（隐藏） |
| 04H（系统） |
| 08H（卷标） |
| 10H（子目录） |
| 20H（归档——只要完成了写操作并已关闭，则该位置1） |
| 第12--21字节 | 保留未用 | |
| 第22--23字节 | 表示文件的创建时间 | |
| 第24--25字节 | 表示文件的创建日期 | |
| 第26--27字节 | 表示文件的起始簇号（首簇号） | |
| 第28--31字节 | 表示文件的字节数 | |

FAT16文件系统对每个文件来说其数据结构是一个单项簇链表，而文件在文件目录表FDT中占一个文件目录项，每个文件的首簇号就存放在该文件的目录项中，一个文件目录项占32个字节。对FDT中一个文件目录项的几个参数值作如下说明：

(1)文件名

FAT16文件系统中，FDT的文件目录项中第0-7字节为文件名（若有剩余字节则用空白符20H填充）。其中，第1字节又表明了该文件的状态，它有如下3种取值方式：

·00H目录项的空表目。

·E5表示该目录项曾经使用过，但文件已被删除。

·2E表示该项为子目录项。

其它任何字符表示一个文件名（或子目录名）的第一个字符的ASCII码值。

(2)文件属性

第11字节为文件属性字节，按位规定如下：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
| 未用 | 未用 | 归档 | 子目录 | 卷标 | 系统 | 隐藏 | 只读 |

(3)文件创建（修改）时间

·（第22字节0--4位）以2秒为增量的二进制树

·（23字节0--2位）+（22字节5--7位）为分钟

·（第23字节3--7位）为小时

(4)文件创建（修改）日期

·（第24字节0--4位）为日期1--31

·（25字节0位）+（24字节5--7位）为月份1--12

·（第25字节1--7位）为年号0--119（1980--2099）

(5)文件首簇号

第26、27 字节存放该文件的首簇号。系统根据FAT中该文件的单链表即可找到它的全部内容。

要注意区分文件目录表（FDT）和FDT中的目录登记项这两个概念。

目录登记项是一个32字节长的数据块，它记录了一个文件或子目录的有关信息。FDT是由若干目录登记项组成的一个表。根目录区是盘卷上的一个固定区域，根目录及各子目录都有各自的FDT。

DOS**文件名**（附加空格）

第一个字节可以是下面的特殊数值：

0x00 这个条目有用并且后面没有被占用条目

0x05 最初字符确实是 0xE5

0x2E '点'条目；'.' 或者 '..'

0xE5 这个条目曾经被删除不再有用。取消删除文档工具作为取消删除的一步必须使用一个正常的字符取代它。

DOS **文件扩展名**（空格补齐）

**文档属性**

第一个字节可以是下面一些特殊值：

位 掩码 描述

0 0x01 只读 （R）

1 0x02 隐藏 （H）

2 0x04 系统 （S）

3 0x08 卷标

4 0x10 子目录

5 0x20 档案 （A）

6 0x40 设备（内部使用，磁盘上看不到）

7 0x80 没有使用

属性值 **0x0F** 用来表示长文件名条目。

**保留**，NT使用（参见后面）

**创建时间**，最小时间分辨率：10ms 单位，数值从 0 到199。

**创建时间**。小时、分钟和秒根据后面的图示描述进行编码：

位 描述

15-11 小时（0-23）

10-5 分钟（0-59）

4-0 秒／2（0-29）

注意秒只保存了2秒的分辨率。更细分辨率的文档创建时间在偏移 0x0d 处。

**创建日期**。年、月和日根据后面的图示编码：

位 描述

15-9 年（0 = 1980, 127 = 2107）

8-5 月（1 = 1月，12 = 12月）

4-0 日（1 - 31）

**最近访问时间**；参见偏移 0x0e 处的描述。

FAT12和FAT16中的EA-Index（OS/2和NT使用），FAT32中第一个簇的两个高字节

**最后更改时间**；参见偏移 0x0e 处的描述。

**最后更改日期**； 参见偏移 0x10 处的描述。

FAT12和FAT16中的第一个簇。FAT32中第一个簇的两个低字节。

**文件大小**：4字节。

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
| 42 | 42 | 42 | 42 | 42 | 42 | 42 | 42 | 54 | 58 | 54 | 20 | 00 | 1D | D8 | B2 |
| 50 | 3D | 50 | 3D | 00 | 00 | 28 | AA | 50 | 3D | 06 | 00 | 64 | 02 | 00 | 00 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 1 | 2 | 3 | 1 | 1 | 1 | 1 | 2 |
| B | B | B | B | B | B | B | B | T | X | T |  | . | . | . | . |
| P | = | P | = | . | . | ( | . | P | = | . | . | d | . | . | . |

文件名称8：BBBBBBBB，扩展3：TXT

属性1：20 00100000档案属性A，保留1：00

创建时间1：1D=290ms，

创建时间HMS2：B2 D8 （10110，010110，11000）22：22：48

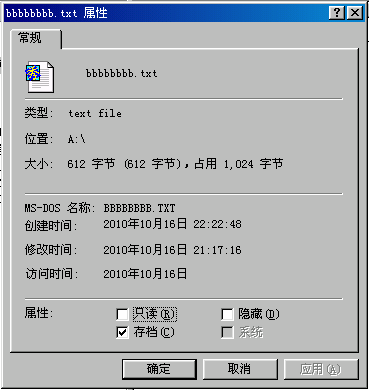
创建日期YMD2：3D 50 （0011110，1010，10000）1980+30年10月16日

最后访问日期2：3D 50 （0011110，1010，10000）1980+30年10月16日

eaIndex2：00 00，最终修改时间2：AA 28 （10101，010001，01000）21：17：16

最终修改日期2：3D 50 （0011110，1010，10000）1980+30年10月16日

起始簇号2：00 06 6，文件大小4：00 00 02 64 612B



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字节偏移 | 字节数 | 定义 | | |
| (16进制) |
| 0x0 | 1 | 属性字节位意义 | 7 | 保留未用 |
| 6 | 1表示长文件最后一个目录项 |
| 5 | 保留未用 |
| 4 | 顺序号数值 |
| 3 |
| 2 |
| 1 |
| 0 |
| 0x1~0xA | 10 | 长文件名unicode码① | | |
| 0xB | 1 | 长文件名目录项标志，取值0FH | | |
| 0xC | 1 | 系统保留 | | |
| 0xD | 1 | 校验值(根据短文件名计算得出) | | |
| 0xE~0x19 | 12 | 长文件名unicode码② | | |
| 0x1A~0x1B | 2 | 文件起始簇号(目前常置0) | | |
| 0x1C~0x1F | 4 | 长文件名unicode码③ | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
| 41 | 61 | 00 | 61 | 00 | 61 | 00 | 61 | 00 | 61 | 00 | 0F | 00 | A2 | 61 | 00 |
| 61 | 00 | 61 | 00 | 00 | 00 | FF | FF | FF | FF | 00 | 00 | FF | FF | FF | FF |
| 41 | 41 | 41 | 41 | 41 | 41 | 41 | 41 | 20 | 20 | 20 | 10 | 00 | C0 | D7 | B2 |
| 50 | 3D | 50 | 3D | 00 | 00 | D8 | B2 | 50 | 3D | 02 | 00 | 00 | 00 | 00 | 00 |
| 41 | 62 | 00 | 62 | 00 | 62 | 00 | 62 | 00 | 62 | 00 | 0F | 00 | 4A | 62 | 00 |
| 62 | 00 | 62 | 00 | 2E | 00 | 74 | 00 | 78 | 00 | 00 | 00 | 74 | 00 | 00 | 00 |
| 42 | 42 | 42 | 42 | 42 | 42 | 42 | 42 | 54 | 58 | 54 | 20 | 00 | 1D | D8 | B2 |
| 50 | 3D | 50 | 3D | 00 | 00 | 28 | AA | 50 | 3D | 06 | 00 | 64 | 02 | 00 | 00 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 1 | 2 | 3 | 1 | 1 | 1 | 1 | 2 |
| A | a | . | a | . | a | . | a | . | a | . | . | . | . | a | . |
| a | . | a | . | . | . | . | . | . | . | . | . | . | . | . | . |
| A | A | A | A | A | A | A | A |  |  |  | . | . | . | . | . |
| P | = | P | = | . | . | . | . | P | = | . | . | . | . | . | . |
| A | b | . | b | . | b | . | b | . | b | . | . | . | . | b | . |
| b | . | b | . | . | . | t | . | x | . | . | . | t | . | . | . |
| B | B | B | B | B | B | B | B | T | X | T |  | . | . | . | . |
| P | = | P | = | . | . | ( | . | P | = | . | . | d | . | . | . |

**第五节 文件数据**

数据区（Data Area）:

紧跟在FDT的下一个扇区，直到逻辑盘的结束地址。它存储着所有的数据，而且即使文件目录被破坏仍旧可能从磁盘里把信息读出，这也就是硬盘数据的理论依据。

到现在为止，硬盘数据结构的理论部分已经讲完。

数据恢复主要是手动找出FAT、目录、数据的对应关系或直接找到数据，现在已经有完善的磁盘编辑器帮助我们做到这一点，使工作大大简化了。有智能化的恢复工具能不依靠FAT而恢复被删除文件，比如RECOVERNT，估计是依靠Win2000的文件使用记录。这种方法在重新启动之前恢复文件的可能性很大。从理论上讲只要数据不被覆盖总能被恢复的。

磁盘大小:1.44MB FAT12

30MB FAT16

FAT的存储结构

00:0000第001扇区(1扇区，1\*512字节)主引导扇区(MBR，Master Boot Record)

**启动代码**

主引导记录最开头是第一阶段引导代码。其中的硬盘引导程序的主要作用是检查分区表是否正确并且在系统硬件完成自检以后将控制权交给硬盘上的引导程序（如GNU GRUB）。MBR是由分区程序（如Fdisk）所产生的，它不依赖任何操作系统，而且硬盘引导程序也是可以改变的，从而能够实现多系统引导。

**硬盘分区表**

硬盘分区表占据主引导扇区的64个字节(偏移01BEH--偏移01FDH)，可以对四个分区的信息进行描述，其中每个分区的信息占据16个字节。具体每个字节的定义可以参见硬盘分区结构信息。

下面是一个例子：

如果某一分区在硬盘分区表的信息如下

80 01 01 00 0B FE BF FC 3F 00 00 00 7E 86 BB 00

则我们可以看到，最前面的"80"是一个分区的激活标志，表示系统可引导；

"01 01 00"表示分区开始的磁头号为01，开始的扇区号为01，开始的柱面号为00；

"0B"表示分区的系统类型是FAT32，其他比较常用的有04（FAT16）、07（NTFS）；

"FE BF FC"表示分区结束的磁头号为254(FE)，分区结束的扇区号为63、分区结束的柱面号为764；

"3F 00 00 00"表示首扇区的相对扇区号为63；(00 00 00 3F)

"7E 86 BB 00"表示总扇区数为~~12289622~~12289662。(00 BB 86 7E)

|  |  |  |
| --- | --- | --- |
| 硬盘分区结构信息 | | |
| 偏移 | 长度(字节) | 意义 |
| 00H | 1 | 分区状态：**00-->非活动分区；80--> 活动分区；** 其它数值没有意义 |
| **01H** | **1** | **分区起始磁头号(HEAD)，用到全部8位** |
| 02H | 2 | 分区起始扇区号(SECTOR)，占据02H的位0－5； 该分区的起始磁柱号(CYLINDER)，占据 02H的位6－7和03H的全部8位 |
| **04H** | **1** | **文件系统标志位** |
| 05H | 1 | 分区结束磁头号(HEAD)，用到全部8位 |
| **06H** | **2** | **分区结束扇区号(SECTOR)，占据06H的位0－5； 该分区的起始磁柱号(CYLINDER)，占据 06H的位6－7和07H的全部8位** |
| 08H | 4 | 分区起始相对扇区号 |
| **0CH** | **4** | **分区总的扇区数** |

80 **01** 01 00 **0E** 0F **3F 40** 3F 00 00 00 **B1 FF 00 00**

00 **00** 00 00 **00** 00 **00** **00** 00 00 00 00 **00 00 00 00**

80 活动分区

**01 HEAD 01**

00 01 SECTOR(0-5) + CYLINDER(6-7 + )

**0E**

0F(15)

**40 3F**

00 00 00 3F(63)★

**00 00 FF B1(65457) (65457\*512)/(1024\*1024)=31.9MB**

**结束标志字**

结束标志字55，AA（偏移1FEH－偏移1FFH）是主引导扇区的最后两个字节，是检验主引导记录是否有效的标志。

00:0200第002扇区(61扇区，61\*512字节)保留扇区

00:7E00第063扇区(1扇区，1\*512字节)分区引导扇区

**引导扇区**

启动扇区是硬盘、软盘或类似的数据存储设备的一个扇区，内含负责启动（booting）“存放在碟片（disk）的其他部份的程序（通常，但不必然是操作系统）”的机器码。

引导扇区有两种：

Volume Boot Record 是软盘未被分区的第一个扇区，或已分区的分区的第一个扇区，包含了加载与唤起操作系统(放在这个分区之内或放在这个软盘上)的码。

Master Boot Record 是软盘已被分区的第一个扇区，它包含定位活动分区区与唤起它的VBR的码。

IBM PC兼容机上，BIOS不在意VBR与MBR的不同，甚至分区区。固件只是加载并运行软盘的第一个扇区(sector)，在MBR里的码，才知道软盘分区消息；且是负责加载启动活动分区区的VBR的地方。

如果你从“一个没有灌操作系统的软盘”启动，屏幕会显示"Please Insert a bootable disc and press a key"；这是开机扇区(boot sector)显示的，而不是机器的固件。

**BPB（BIOS Parameter Block）**

FAT16的BPB（BIOS Parameter Block）表，描述逻辑盘结构组成，包含隐藏扇区数目（从0-1-1开始计算）、FAT扇区数、FAT拷贝数、硬盘磁头总数、根目录表项最大值等。

FAT32系统中，BPB表的偏移与FAT16不同，但表项基本相同。

整个隐藏扇区部分都作为逻辑盘的描述区域。

硬盘BPB主要结构说明：

(Cylinder柱面/磁道-Side/Head磁头-Sector扇区地址以下简称为?-?-?)

主分区 　　名称 地址 长度(扇区)

主引导记录（Main Boot Record) 0-0-1 1

系统扇区（System Secotrs） 0-0-2 0-0-63 62

引导扇区（Boot） 0-1-1 1

扩展分区 　　名称 地址 长度(扇区)

扩展分区（Extend Partition） ?-y-1 1

系统扇区（System Secotrs） ?-y-2 ?-y-63 62

引导扇区（Boot） ?-(y+1)-1 1

其后各项与主分区相同……

隐藏扇区（Hidden Secotrs）:

FAT16 0-1-1 1

FAT32 0-1-1 32

文件分配表(File Allocation Table):

FAT16 0-1-2 根据逻辑盘容量变化

FAT32 0-1-33 根据逻辑盘容量变化

说明：

1. FAT16的每个表项由2字节（16位）组成，通常每个表项指向的簇包含64个扇区，即32K字节。逻辑盘容量最大为2047MB。

2. FAT32的每个表项由4字节（32位）组成，通常每个表项指向的簇包含8个扇区，即4K字节。逻辑盘容量最小为512MB。

3. 对于C分区，在MBR的偏移01c2H处，FAT16为06H，FAT32为0CH。

BIOS parameter block

In computing, the BIOS parameter block, often shortened to BPB, is a data structure in the Volume Boot Record describing the physical layout of a data storage volume. On partitioned devices, such as hard disks, the BPB describes the volume partition, whereas, on unpartitioned devices, such as floppy disks, it describes the entire medium. A basic BPB can appear and be used on any partition, including floppy disks where its presence is often necessary, however, certain filesystems also make use of it in describing basic filesystem structures. Filesystems making use of a BIOS parameter block include FAT12, FAT16, FAT32, HPFS, and NTFS. Due to different types of fields and the amount of data they contain, the length of the BPB is different for FAT16, FAT32, and NTFS boot sectors. ECMA-107 or ISO/IEC 9293 (which describes FAT as for flexible/floppy and optical disk cartridges) also describes this as an FDC Descriptor or an FDC Extended Descriptor.

The standard BPB for FAT16 is as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Field Length | Hex offset | Description |
| BytesPerSector | WORD | 0x000B | Bytes Per Sector |
| SectorsPerCluster | BYTE | 0x000D | Sectors Per Cluster |
| ReservedSectors | WORD | 0x000E | Reserved Sectors |
| FatCopies | BYTE | 0x0010 | Number of FATs |
| RootDirEntries | WORD | 0x0011 | Root Entries |
| NumSectors | WORD | 0x0013 | Small Sectors |
| MediaType | BYTE | 0x0015 | Media Descriptor |
| SectorsPerFAT | WORD | 0x0016 | Sectors Per FAT |
| SectorsPerTrack | WORD | 0x0018 | Sectors Per Track |
| NumberOfHeads | WORD | 0x001A | Number of Heads |
| HiddenSectors | DWORD | 0x001C | Hidden Sectors |
| SectorsBig | DWORD | 0x0020 | Large Sectors |

Extended BPB for FAT16 Volumes:

|  |  |  |
| --- | --- | --- |
| Field Length | Hex offset | Description |
| BYTE | 0x0024 | Physical Drive Number |
| BYTE | 0x0025 | Reserved |
| BYTE | 0x0026 | Extended Boot Signature |
| DWORD | 0x0027 | Volume Serial Number |
| 11 bytes | 0x002B | Volume Label |
| QWORD | 0x0036 | File System Type |

00:8000第064扇区(64扇区，64\*512字节)FAT1扇区

01:0000第128扇区(64扇区，64\*512字节)FAT2扇区

01:8000第192扇区(32扇区，32\*512字节)根目录扇区

01:C000第224扇区(n扇区，n\*512字节)文件存储扇区--512字节单位

FAT16

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 |  | 63 | 64 | | 128 | | 192 | 224 |  |  |
| 0000h |  | 7E00 | 8000 | | 10000 | | 18000 | 1C000 |  |  |
| MBR | … | BS | FAT1 |  | FAT2 |  | RootDir | File1 | File1 | File2 |
| DPT |  | BPB | 0- |  |  |  |  |  |  |  |
| 16\*4 |  | 1 | 64 | | 64 | | 32 |  |  |  |

FAT12

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | | 10 | | 19 | 33 |  |  |
| 0000 | 0200 | | 1400 | | 2600 | 4200 |  |  |
| BS | FAT1 |  | FAT2 |  | RootDir | File1 | File1 | File2 |
| BPB | 0- |  |  |  |  |  |  |  |
| 1 | 9 | | 9 | | 14 |  |  |  |

文件目录表项存储第一个簇0002其余文件的簇号存储于FAT中。

SecNo=（n-2）\*4:BPB\_SecPerClus +224， n=FAT中的簇地址。

n=3,SecNo=(3-2)\*4:BPB\_SecPerClus +224=228.

n=45(2d),SecNo=(45-2)\*4:BPB\_SecPerClus +224=396.