磁盘上的数据按照其不同的特点和作用大致可分为 5 部分：MBR 区、DBR 区、FAT 区、DIR 区和 DATA 区。我们来分别介绍一下：

（1）MBR 区（主引导扇区）

MBR（Main Boot Record）,按其字面上的理解即为主引导记录区,位于整个硬盘的 0 磁道 0 柱面 1 扇区。不过,在总共 512 字节的主引导扇区中,MBR 只占用了其中的 446 个字节（偏移 0--偏移 1BDH）,另外的 64 个字节（偏移 1BEH--偏移 1FDH）交给了 DPT(Disk Partition Table 硬盘分区表)（见下表）,最后两个字节"55,AA"（偏移 1FEH- 偏移 1FFH）是分区的结束标志。这个整体构成了硬盘的主引导扇区。

主引导记录中包含了硬盘的一系列参数和一段引导程序。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引导具有激活标志的分区上的操作系统,并将控制权交给启动程序。MBR 是由分区程序（如 Fdisk.com）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,从而实现多系统共存。

偏移 长度 所表达的意义

0 字节 分区状态：如 0-->非活动分区

80--> 活动分区

1 字节 该分区起始头（HEAD）

2 字 该分区起始扇区和起始柱面

4 字节 该分区类型：如 82--> Linux Native 分区

83--> Linux Swap 分区

5 字节 该分区终止头（HEAD）

6 字 该分区终止扇区和终止柱面

8 双字 该分区起始绝对分区

C 双字 该分区扇区数

下面,我们以一个实例让大家更直观地来了解主引导记录：

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

在这里我们可以看到,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。

（2）DBR 区

DBR（Dos Boot Record）是操作系统引导记录区的意思。它通常位于硬盘的 0 磁道 1 柱面 1 扇区,是操作系统可以直接访问的第一个扇区,它包括一个引导程序和一个被称为 BPB（Bios Parameter Block）的本分区参数记录表。引导程序的主要任务是当 MBR 将系统控制权交给它时,判断本分区跟目录前两个文件是不是操作系统的引导文件（以 DOS 为例,即是 Io.sys 和 Msdos.sys）。如果确定存在,就把其读入内存,并把控制权 交给该文件。BPB 参数块记录着本分区的起始扇区、结束扇区、文件存储格式、硬盘介质描述符、根目录大小、FAT 个数,分配单元的大小等重要参数。

（3）FAT 区

在 DBR 之后的是我们比较熟悉的 FAT（File Allocation Table 文件分配表）区。在解释文件分配表的概念之前,我们先来谈谈簇（cluster）的概念。文件占用磁盘空间时,基本单位不是字节而是簇。簇的大小与磁盘的规格有关,一般情况下,软盘每簇是 1 个扇区,硬盘每簇的扇区数与硬盘的总容量大小有关,可能是 4、8、16、32、64……

通过上文我们已经知道,同一个文件的数据并不一定完整地存放在磁盘的一个连续的区域内,而往往会分成若干段,像一条链子一样存放。这种存储方式称为文件的链式存储。硬盘上的文件常常要进行创建、删除、增长、缩短等操作。这样操作做的越多,盘上的文件就可能被分得越零碎（每段至少是 1 簇）。但是,由于硬盘上保存着段与段之间的连接信息（即 FAT）,操作系统在读取文件时,总是能够准确地找到各段的位置并正确读出。不过,这种以簇为单位的存储法也是有其缺陷的。这主要表现在对空间的利用上。每个文件的最后一簇都有可能有未被完全利用的空间（称为尾簇空间）。一般来说,当文件个数比较多时,平均每个文件要浪费半个簇的空间。

好了,我们言归正传,为了实现文件的链式存储,硬盘上必须准确地记录哪些簇已经被文件占用,还必须为每个已经占用的簇指明存储后继内容的下一个簇的簇号,对一个文件的最后一簇,则要指明本簇无后继簇。这些都是由 FAT 表来保存的,表中有很多表项,每项记录一个簇的信息。

由于 FAT 对于文件管理的重要性,所以 FAT 有一个备份,即在原 FAT 的后面再建一个同样的 FAT。初形成的 FAT 中所有项都标明为"未占用",但如果磁盘有局部损坏,那么格式化程序会检测出损坏的簇,在相应的项中标为"坏簇",以后存文件时就不会再使用这个簇了。FAT 的项数与硬盘上的总簇数相当,每一项占用的字节数也要与总簇数相适应,因为其中需要存放簇号。FAT 的格式有多种,最为常见和为读者所熟悉的是 FAT16 和 FAT32,其中 FAT16 是指文件分配表使用 16 位数字,由于 16 位分配表最多能管理 65536（即 2 的 16 次方）个簇,也就是所规定的一个硬盘分区。

由于每个簇的存储空间最大只有 32KB,所以在使用 FAT16 管理硬盘时,每个分区的最大存储容量只有（65536×32 KB）即 2048MB,也就是我们常说的 2G。现在的硬盘容量是越来越大,由于 FAT16 对硬盘分区的容量限制,所以当硬盘容量超过 2G 之后,用户只能将硬盘划分成多个 2G 的分区后才能正常使用,为此微软公司从 Windows 95 OSR2 版本开始使用 FAT32 标准,即使用 32 位的文件分配表来管理硬盘文件,这样系统就能为文件分配多达 4294967296（即 2 的 32 次方）个簇,所以在簇同样为 32KB 时每个分区容量最大可达 65G 以上。此外使用 FAT32 管理硬盘时,每个逻辑盘中的簇长度也比使用 FAT16 标准管理的同等容量逻辑盘小很多。由于文件存储在硬盘上占用的磁盘空间以簇为最小单位,所以某一文件即使只有几十个字节也必须占用整个簇,因此逻辑盘的簇单位容量越小越能合理利用存储空间。所以 FAT32 更适于大硬盘。

（4）DIR 区（根目录区）

DIR（Directory）是根目录区,紧接着第二 FAT 表（即备份的 FAT 表）之后,记录着根目录下每个文件（目录）的起始单元,文件的属性等。定位文件位置时,操作系统根据 DIR 中的起始单元,结合 FAT 表就可以知道文件在硬盘中的具体位置和大小了。

（5）数据（DATA）区

数据区是真正意义上的数据存储的地方,位于 DIR 区之后,占据硬盘上的大部分数据空间。

FAT16 文件系统

The boot sector

A boot sector can be found in the first sector of every logical disk. On a floppy disk, the logical disk takes up all of the physical disk and the boot sector lives in the first physical sector. On a hard disk, a boot sector lives at the start of each partition.

An example boot sec

```
0000 EB 3E 90 4D 53 57 49 4E-34 2E 30 00 02 20 01 00 .>.MSWIN4.0.. ..
0010 02 40 03 00 00 F8 FF 00-3F 00 40 00 3F 00 00 00 .@......?.@.?...
0020 41 DE 1F 00 80 00 29 37-4B 84 32 46 55 4A 49 54 A…..)7K.2FUJIT
0030 53 55 31 32 32 34 46 41-54 31 36 20 20 20 F1 7D SU1224FAT16 .}
0040 FA 33 C9 8E D1 BC FC 7B-16 07 BD 78 00 C5 76 00 .3…..{…x..v.
0050 1E 56 16 55 BF 22 05 89-7E 00 89 4E 02 B1 0B FC .V.U."..~..N….
0060 F3 A4 06 1F BD 00 7C C6-45 FE 0F 8B 46 18 88 45 ……|.E…F..E
0070 F9 FB 38 66 24 7C 04 CD-13 72 3C 8A 46 10 98 F7 ..8f$|…r<.F…
0080 66 16 03 46 1C 13 56 1E-03 46 0E 13 D1 50 52 89 f..F..V..F…PR.
0090 46 FC 89 56 FE B8 20 00-8B 76 11 F7 E6 8B 5E 0B F..V.. ..v….^.
00A0 03 C3 48 F7 F3 01 46 FC-11 4E FE 5A 58 BB 00 07 ..H…F..N.ZX…
00B0 8B FB B1 01 E8 94 00 72-47 38 2D 74 19 B1 0B 56 …….rG8-t…V
00C0 8B 76 3E F3 A6 5E 74 4A-4E 74 0B 03 F9 83 C7 15 .v>..^tJNt……
00D0 3B FB 72 E5 EB D7 2B C9-B8 D8 7D 87 46 3E 3C D8 ;.r…+…}.F><.
00E0 75 99 BE 80 7D AC 98 03-F0 AC 84 C0 74 17 3C FF u…}…….t.<.
00F0 74 09 B4 0E BB 07 00 CD-10 EB EE BE 83 7D EB E5 t……}..
0100 BE 81 7D EB E0 33 C0 CD-16 5E 1F 8F 04 8F 44 02 ..}..3…^….D.
0110 CD 19 BE 82 7D 8B 7D 0F-83 FF 02 72 C8 8B C7 48 ….}.}….r…H
0120 48 8A 4E 0D F7 E1 03 46-FC 13 56 FE BB 00 07 53 H.N….F..V….S
0130 B1 04 E8 16 00 5B 72 C8-81 3F 4D 5A 75 A7 81 BF …..r..?Mzu…
0140 00 02 42 4A 75 9F EA 00-02 70 00 50 52 51 91 92 ..Bju….p.PRQ..
0150 33 D2 F7 76 18 91 F7 76-18 42 87 CA F7 76 1A 8A 3..v…v.B…v..
0160 F2 8A 56 24 8A E8 D0 CC-D0 CC 0A CC B8 01 02 CD ..V$.....
0170 13 59 5A 58 72 09 40 75-01 42 03 5E 0B E2 CC C3 .YZXr.@u.B.^….
0180 03 18 01 27 0D 0A 49 6E-76 61 6C 69 64 20 73 79 …"..Invalid sy
0190 73 74 65 6D 20 64 69 73-6B FF 0D 0A 44 69 73 6B stem disk…Disk
01A0 20 49 2F 4F 20 65 72 72-6F 72 FF 0D 0A 52 65 70 I/O error…Rep
01B0 6C 61 63 65 20 74 68 65-20 64 69 73 6B 2C 20 61 lace the disk, a
01C0 6E 64 20 74 68 65 6E 20-70 72 65 73 73 20 61 6E nd then press an
01D0 79 20 6B 65 79 0D 0A 00-49 4F 20 20 20 20 20 20 y key…IO
01E0 53 59 53 4D 53 44 4F 53-20 20 20 53 59 53 80 01 SYSMSDOS SYS..
01F0 00 57 49 4E 42 4F 4F 54-20 53 59 53 00 00 55 AA .WINBOOT SYS..U.
```

Structure of the boot sector

offset	size	meaning
00	3	jumpinstruction
03	8	system name
0b	33	BIOS parameter block
3e	c0	bootstrap code

fe 2 sector signature

Structure of the BIOS paramter block (BPB)

offset	length	meaning
00	0x2	logical bytes per sector
02	0x1	sectors per cluster
03	0x2	number of reserved sectors starting from 0
05	0x1	number of FATs
06	0x2	number of root dir entries
08	0x2	total sectors. 0 if ofs 15h used
0a	0x1	media descriptor
0b	0x2	number of sectors per FAT
0d	0x2	number of logical sectors per track
0f	0x2	number of logical heads
11	0x4	number of hidden sectors
15	0x4	total sectors
19	0x1	physical drive number
1a	0x1	reserved
1b	0x1	signature byte for extended boot record
1c	0x4	serial number
20	0xb	label
2b	0x8	FAT type

Values for the example boot sector

offset	length	meaning
00	0x2	logical bytes per sector
02	0x1	sectors per cluster
03	0x2	number of reserved sectors starting from 0
05	0x1	number of FATs
06	0x2	number of root dir entries
08	0x2	total sectors. 0 if ofs 15h used
0a	0x1	media descriptor
0b	0x2	number of sectors per FAT
0d	0x2	number of logical sectors per track
0f	0x2	number of logical heads
11	0x4	number of hidden sectors
15	0x4	total sectors
19	0x1	physical drive number
1a	0x1	reserved
1b	0x1	signature byte for extended boot record
1c	0x4	serial number
20	0xb	label
2b	0x8	FAT type

Check the C source code to define the FAT boot sector.

Structure of a FAT disk

disk address	length	contents
0	1	MBR
1	PhysSectPerTrck - 1	Undefined
PhysSectPerTrck	1	Boot sector
PhysSectPerTrck+1	NumFATs*NumSectPerFAT	FATs
BookKeep-SectRootDir	SectRootDir	Root directory
BookKeep	LogcDiskSize-BookKeep	Data space

SectRootDir = RootDirSize / 16

BookKeep = PhysSectPerTrck + 1 + NumFATs * NumSectPerFAT + SectRootDir

FAT

The FAT is an array of up to 65,536 16-bit unsigned integers. The first 2 and last 16 entries in the FAT are reserved. All other elements are of type: index into the FAT array. Each of the non-reserved FAT entries correspond to a cluster on the disk.

FAT index	Meaning as index	Meaning as value
0x0000	reserved	cluster is available
0x0001	reserved	undefined
0x0002-0xffef	data cluster	next cluster in chain is at this index
0xfff0-0xfff6	reserved	undefined
0xfff7	reserved	bad cluster
0xfff8-0xffff	reserved	end of the current cluster chain

FAT entries 0x0000 and 0x0001 contain other information, usually 0xfff8 and 0xffff respectively.

The fragment below is from the start of a FAT

FFF8 FFFF 0000 0004 FFFF 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000

The fragment above shows a file using clusters 0x0003 & 0x0004.

Folders

A folder is a data structure stored on the disk. The root folder is stored before the data area. Other folders are stored in the data area in the same way as files; These folders may grow in size, shrink in size and be stored non-contiguously in the same way as files. The structure of all folders is as an array of folder entries. The end of the folder is marked by a folder entry starting with a 00h byte.

Folder entry structure

offset	length	contents
0x00	0x8	name
0x08	0x3	extension
0x0b	0x1	flags
0x0c	0xa	reserved
0x16	0x2	time
0x18	0x2	date
0x1a	0x2	first cluter
0x1c	0x4	size

The date field is in DOS date format, and the time field is in DOS time.