FAT 文件系统总结

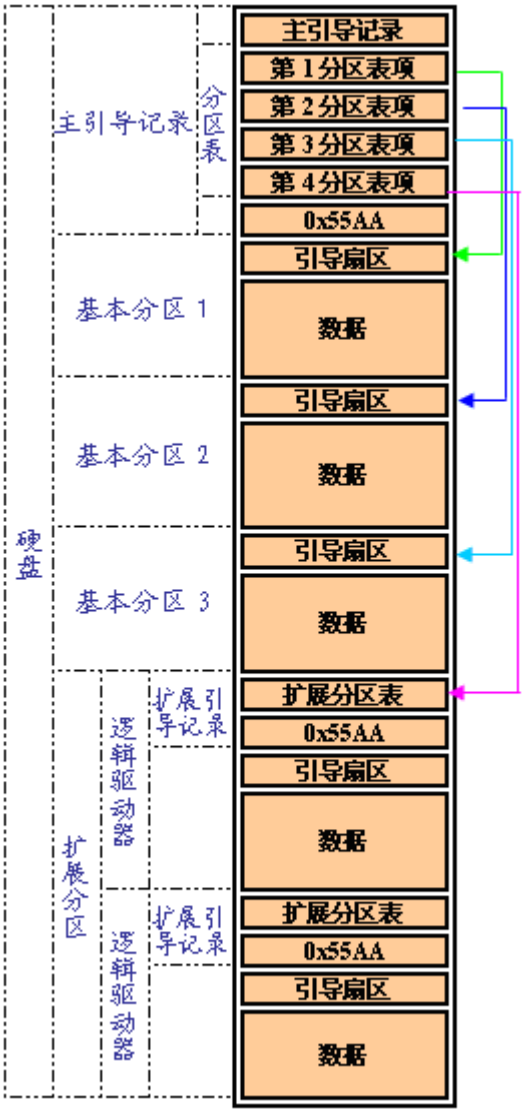
罗流毅
xluoly@msn.com
http://blog.ednchina.com/xluoly

目录

MBR：Master Boot Record (主引导记录)
DBR：DOS Boot Record (DOS 引导记录，位于分区引导扇区)
BPB：BIOS Parameter Block (BIOS 参数块)
FAT：File Allocation Table (文件分配表)
Sector：扇区
Cluster：簇

一、 硬盘组织结构

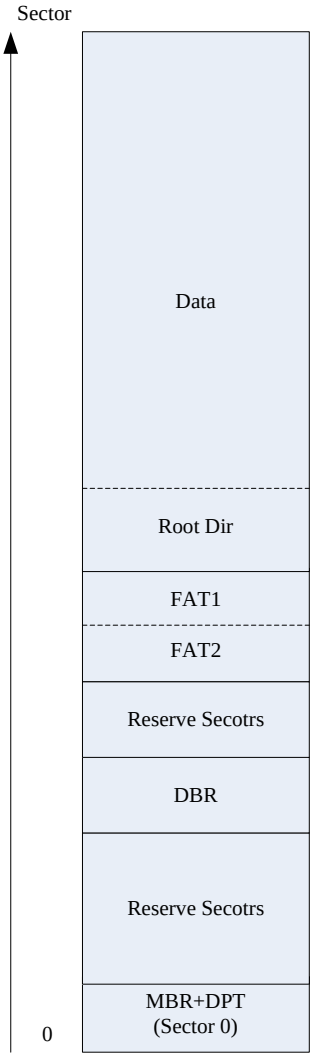
下面是一个包含 4 个分区的硬盘结构示意图，其中分为 3 个基本分区和一个扩展分区。



二、 FAT 文件系统结构

FAT 文件系统是由按照如下顺序排列的几个部分组成的：

- 0 – Reserved Region
- 1 – FAT Region
- 2 – Root Directory Region (FAT32 没有这部分)
- 3 – File and Directory Data Region



FAT 系统的数据存储采用小端（Little Endian）方式，注意到这一点很重要，在使用大端（Big Endian）的系统中，读取多字节数据的时候必须要经过转换，否则，读取到的数据是不正确的。

例如：一个 32-bit 数据 0x12345678 在 FAT 中的保存方式如下图所示：

BYTE[0]	BYTE[1]	BYTE[2]	BYTE[3]
0x12	0x34	0x56	0x78

三、 主引导扇区

硬盘主引导扇区 = 硬盘主引导记录（MBR）+ 硬盘分区表（DPT）

MBR：扇区内偏移地址 0 ~ 0x1BD

DPT：扇区内偏移地址 0x1BE ~ 0x1FD，其中又分为 4 个分区表：

第一个分区表：0x1BE ~ 0x1CD

第二个分区表：0x1CE ~ 0x1DD

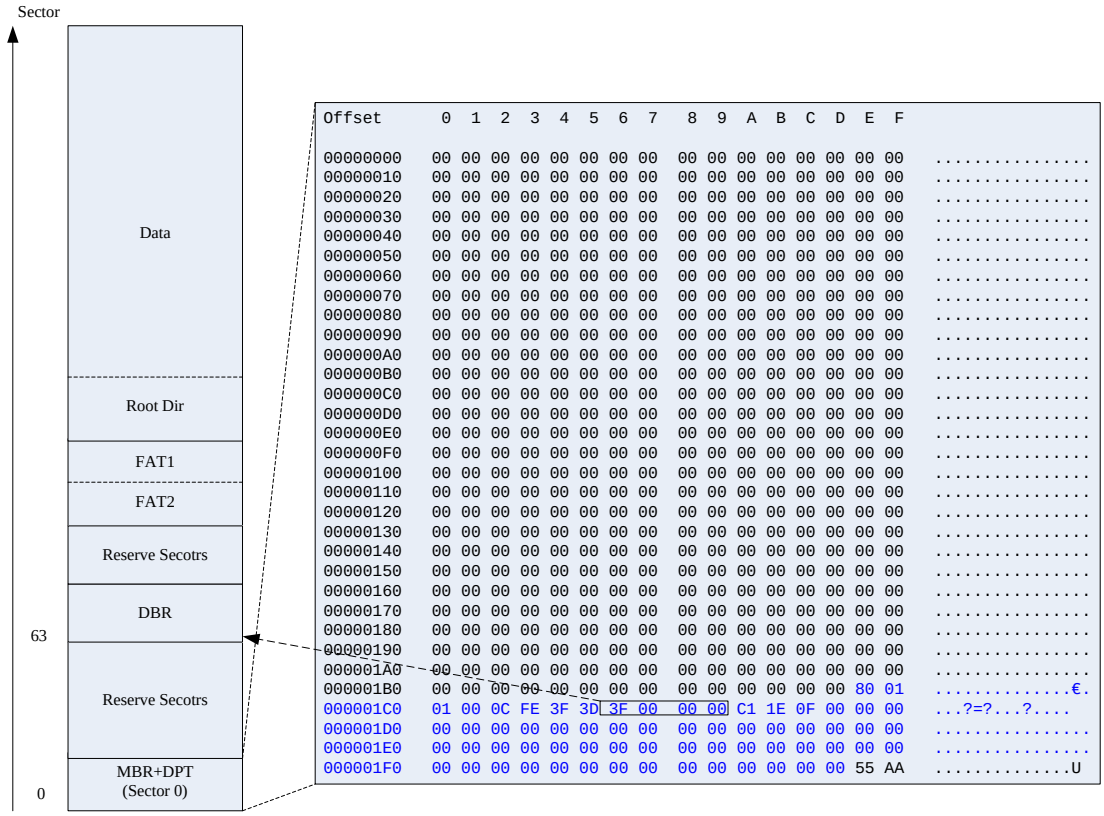
第三个分区表：0x1DE ~ 0x1ED

第四个分区表：0x1EE ~ 0x1FD

每个分区表的信息如下表所示：

分区表信息		
字节位移	字段长度	字段名和定义
0x00	BYTE	引导指示符(Boot Indicator)，指明该分区是否是活动分区，0x80=活动分区，0x00=非活动分区
0x01	BYTE	开始磁头(Starting Head)
0x02	6Bits	开始扇区(Starting Sector)，只用了0~5位。后面的两位(第6位和第7位)被开始柱面字段所使用
0x03	10Bits	开始柱面(Starting Cylinder)，除了开始扇区字段的最后两位外，还使用了1位来组成该柱面值。开始柱面是一个10位数，最大值为1023
0x04	BYTE	系统ID(System ID)，定义了分区的类型，详见下表
0x05	BYTE	结束磁头(Ending Head)
0x06	6Bits	结束扇区(Ending Sector)，只使用了0~5位。最后两位(第6、7位)被结束柱面字段所使用
0x07	10Bits	结束柱面(Ending Cylinder)，除了结束扇区字段最后的两位外，还使用了1位，以组成该柱面值。结束柱面是一个10位的数，最大值为1023
0x08	DWORD	相对扇区数(Relative Sectors)，从该磁盘的开始到该分区的开始的位移量，以扇区来计算
0x0C	DWORD	总扇区数(Total Sectors)，该分区中的扇区总数

分区标志类型值及其含义			
类型值 (HEX)	含义	类型值 (HEX)	含义
0	空。DOS或windows不允许使用，视为非法	**5C**	Priam Edisk
1	FAT12	**61**	Speed Stor
2	XENIX root	**63**	GNU HURD or Sys
3	XENIX usr	**64**	Novell Netware
6	FAT16分区小于32M时为0x04	**65**	Novell Netware
7	HPFS / NTFS	**70**	Disk Secure Mult
8	AIX	**75**	PC/IX
9	AIX bootable	**80**	Old Minix
0A	OS/2 Boot Manage	**81**	Minix/Old Linux
0B	Win95 FAT32	**82**	Linux swap
0C	Win95 FAT32	**83**	Linux
0E	Win95 FAT16	**84**	0s/2 hidden C:
0F	Win95 Extended(大于 8GB)	**85**	Linux extended
10	OPUS	**86**	NTFS volume set
11	Hidden FAT12	**87**	NTFS volume set
12	Compaq diagmost	**93**	Amoeba
14	Hidden FAT16<32MB	**94**	Amoeba BBT
16	HiddenFAT16	**A0**	IBM Thinkpad hidden
17	Hidden HPFS/NTFS	**A5**	BSD/386
18	AST Windows swap	**A6**	Open BSD
1B	Hidden FAT32	**A7**	NextSTEP
1C	Hidden FAT32 partition (using LBA-mode INT 13 extensions)	**B7**	BSDI fs
		B8	BSDI swap
1E	Hidden LBA VFAT partition	**BE**	Solaris boot partition
24	NEC DOS	**C0**	DR-DOS/Novell DOS secured partition
3C	Partition Magic	**C1**	DRDOS/sec
40	Venix 80286	**C4**	DRDOS/sec
41	PPC Perp Boot	**C6**	DRDOS/sec
42	NTFS动态分区	**C7**	Syrinx
4D	QNX4.x	**DB**	CP/M/CTOS
4E	QNX4.x 2nd part	**E1**	DOS access
4F	QNX4.x 3rd part	**E3**	DOS r/0
50	OnTrack DM	**E4**	Speedstor
51	OnTrack DM6 Aux	**EB**	BeoS fs
52	CP/M	**F1**	SpeedStor
53	OnTrack DM6 Aux	**F2**	DOS 3.3+secondary partition
54	OnTrack DM6	**F4**	SpeedStor
55	EZ-Drive	**FE**	LAN step
56	Golden Bow	**FF**	BBT



主引导扇区

上面是从一张 SD 卡读到的主引导扇区信息。可以看出，MBR 区域数据全部为 0，这张 SD 卡只有一个分区，这个分区前的扇区数为 0x0000003F，所以这个分区的开始位置就是扇区 0x0000003F，总扇区数为 0x000F1EC1(990913 个扇区)。

四、 分区引导扇区

也常常称为启动扇区，Microsoft称它为 0 扇区（0th sector），通过前面的介绍我们知道，称它为 0 扇区其实是不正确的，这样容易让人误解它为磁盘的最前面一个扇区，称它为 0 扇区只是表明它是FAT中扇区的参考点而已。

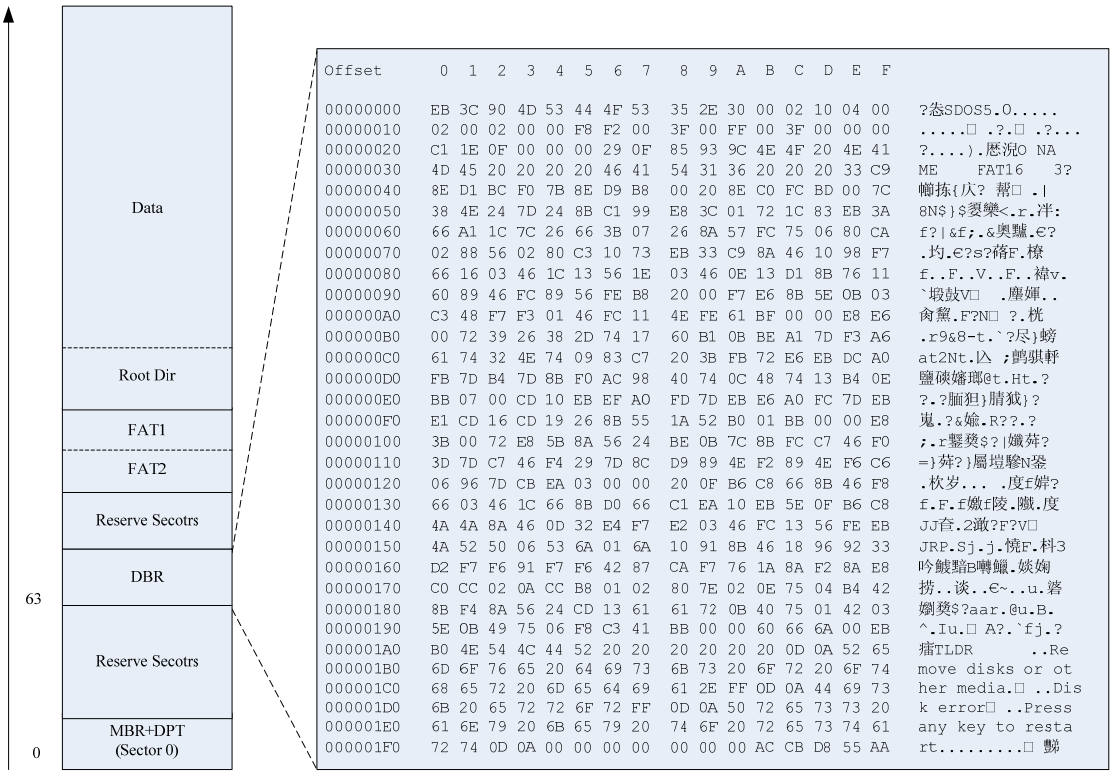


图 2：DBR

该扇区中包含有我们关注的一个重要数据结构 BPB(BIOS Parameter Block)。以下表格内容翻译自 Microsoft 的《Microsoft Extensible Firmware Initiative FAT32 File System Specification–version1.03》，其中包含 BPB 各项的描述。

NOTE：在以下的叙述中，名字以 BPB_开头的属于 BPB 部分，以 BS 开头的属于启动扇区（Boot Sector）部分，实际上并不属于 BPB。

	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_BumFATs	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，无论此域写入什么数值，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_FATSz16	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 0x36 以前是完全一样的，从偏移量 0x36 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 Bits 7： 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”，这样的说法是正确的，但仅仅适用于 BPB_BytsPerSec 值为 512 的情况。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。

五、 FAT 类型识别

FAT 的字类型（FAT12/16/32）只能通过 FAT 卷中的簇（Cluster）数来判定，没有其他的办法。

Cluster 总数的计算：

```
RootDirSectors = (BPB_RootEntCnt*32) /BPB_BytsPerSec
DataSect = TotSec
      – (BPB_RsvdSecCnt +(BPB_NumFATs * FATSz) + RootDirSectors)
CountofClusters = DataSec / BPB_SecPerClus
If(CountofClusters < 4085) {
   /*卷类型是 FAT12 */
} else if(CountofClusters < 65525) {
   /* 卷类型是 FAT16 */
} else {
   /* 卷类型是 FAT32 */
}
```

注意这里的簇数（count of Cluster）是指数据区所占簇的数量（the count of the data cluster），从簇 2 算起，而“最大可用簇数”（Maximun valid cluster number for the volume）是 CountofClusters +1，“包括保留簇的簇数”（count of cluster including the two reserved cluster）则为 CountofClusters +2。

根目录占据的 Sector 数：

```
RootDirSectors = (BPB_RootEntCnt*32) /BPB_BytsPerSec
```

数据区（Cluster 2）的起始 Sector：

```
FirstDataSector = BPB_EsvdSecCnt
                + (BPB_NumFATs * FATSz)
                + RootDirSectors
```

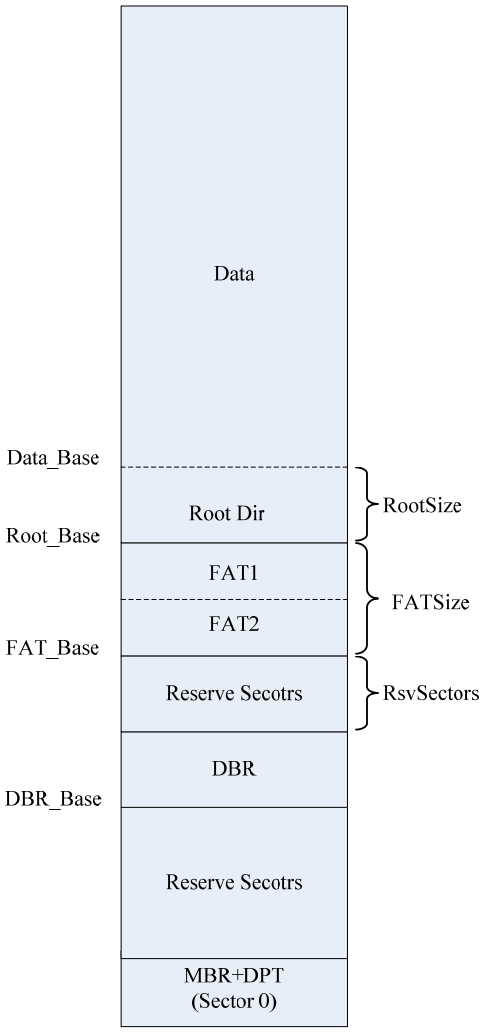
给一个合法的簇号 N，该簇的第一个扇区号由下式计算：

```
FirstSectorofCluster =((N -2) * BPB_SecPerClust)
                     + FirstDataSecot;
```

因为 BPB_SecPerClus 总是 2 的整数次方，这意味着 BPB_SecPerSlus 的乘法运算可以通过移动来进行。

NOTE：这里所说的 Sector 号，指的是针对卷中包 BPB 的第一个扇区（DBR）的偏移量（DBR 设为 Sector 0）。

六、 FAT 各部分位置的计算



DBR_Base：从DPT中偏移8Bytes的地址读取4Bytes数据就可以直接看作是DBR所在的Sector。
RsvSectors = BPB_RsvdSecCnt。
FAT_Base = DBR_Base + RsvSectors
FATSize = BPB_NumFATs * FATSz
其中，FATSz = (FAT32?) BPB_FATSz32 : BPB_FATSz16
Root_Base：如果是FAT12/FAT16，则
    Root_Base = FAT_Base + FATSize
    如果是FAT32，则
    Root_Base = BPB_RootClus
RootSize ：如果是FAT32，则没有限制；否则
    RootSize = (BPB_RootEntCnt*32) / BPB_BytsPerSec
Data_Base：仅对于FAT12/FAT16，
    Data_Base = Root_Base + RootSize

七、 FAT 表结构

FAT 表（File Alloacation Table）是一一对应于数据簇号的列表。

文件系统分配磁盘空间按簇来分配。因此，文件占有磁盘空间时，基本单位不是字节而是簇，即使某个文件只有一个字节，操作系统也会给它分配一个最小单元：即一个簇。为了可以将磁盘空间有序地分配给相应的文件，而读取文件的时候又可以从相应的地址读出文件，我们可以把数据区空间分成 BPB_BytsPerSec*BPB_SecPerClus 字节长的簇来管理，FAT 表项的大小与 FAT 表的类型有关，FAT12 的表项为 12bit ，FAT16 为 16bit， 而 FAT32 则为 32bit。对于大文件，需要分配多个簇。同一个文件的数据并不一定完整地存放在磁盘中一个连续地区域内，而往往会分若干段，像链子一样存放。这种存储方式称为文件的链式存储。为了实现文件的链式存储，文件系统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空的下一个簇的簇号，对于文件的最后一簇，则要指明本簇无后继簇。这些都是由 FAT 表来保存的，FAT 表的对应表项中记录着它所代表的簇的有关信息：诸如是空，是不是坏簇，是否是已经是某个文件的尾簇等。

以 FAT16 为例说明 FAT 的结构如下：

表项	示例代码	描述
0	FFF8	磁盘标识字，必须为 FFF8
1	FFFF	第一簇已经被占用
2	0003	0000h ：可用簇
3	0004	0002h - FFFEF ：已用簇，表项中存放文件下个簇的簇号
……	……	
N	FFFF	FFFF0h - FFFF6 ：保留簇
N+1	0000	FFFF7h ：坏簇
……	……	FFFF8h - FFFFFh ：文件的最后一簇

FAT各系统记录项的取值含义(16进制)			
FAT12记录项的取值	FAT16记录项的取值	FAT32记录项的取值	对应簇的表现情况
0	0	0	未分配的簇
002~FFF	0002~FFEF	00000002~FFFFFFEF	已分配的簇
FF0~FF6	FFF0~FFF6	FFFFFFF0~FFFFFFF6	系统保留
FF7	FFF7	FFFFFFF7	坏簇
FF8~FFF	FFF8~FFFF	FFFFFFF8~FFFFFFFF	文件结束簇

八、 目录结构

目录所在的扇区,都是以32 Bytes划分为一个单位,每个单位称为一个目录项(Directory Entry），即每个目录项的长度都是32 Bytes。

FAT32短文件目录项32个字节的表示定义				
名称	字节偏移	字节数	定义	
DIR_Name	0x0~0xA	11	文件名	
DIR_Attr	0xB	1	属性字节	00000000(读写)
				00000001(只读)
				00000010(隐藏)
				00000100(系统)
				00001000(卷标)
				00010000(子目录)
				00100000(归档)
				00001111(长文件名)
DIR_NTRes	0xC	1	系统保留	
DIR_CrtTimeTenth	0xD	1	创建时间的10毫秒位	
DIR_CrtTime	0xE~0xF	2	文件创建时间	
DIR_CrtDate	0x10~0x11	2	文件创建日期	
DIR_LstAccDate	0x12~0x13	2	文件最后访问日期	
DIR_FstClusHI	0x14~0x15	2	文件起始簇号的高16位	
DIR_WrtTime	0x16~0x17	2	文件的最近修改时间	
DIR_WrtDate	0x18~0x19	2	文件的最近修改日期	
DIR_FstClusLO	0x1A~0x1B	2	文件起始簇号的低16位	
DIR_FileSize	0x1C~0x1F	4	表示文件的长度	

FAT16目录项32个字节的表示定义			
字节偏移(16进制)	字节数	定义	
0x0~0x7	8	文件名	
0x8~0xA	3	扩展名	
0xB	1	属性字节	00000000(读写)
			00000001(只读)
			00000010(隐藏)
			00000100(系统)
			00001000(卷标)
			00010000(子目录)
			00100000(归档)
0xC~0x15	10	系统保留	
0x16~0x17	2	文件的最近修改时间	
0x18~0x19	2	文件的最近修改日期	
0x1A~0x1B	2	表示文件的首簇号	
0x1C~0x1F	4	表示文件的长度	

日期格式：

- Bits 0-4：日期，有效值为1-31。
- Bits 5-8：月份，有效值为1-12。

- Bits 9–15：1980 后经过的年数有效值为 0–127，可以表示的范围是 1980–2107 年.

时间格式：

- Bits 0–4：秒，以 2 秒为一个单位，有效值为 0–29，(实际表示 0 – 58 秒)。
- Bits 5–10：分，有效值为 0–59。
- Bits 11–15：时，有效值为 0–23。

有效的时间范围是 00:00:00 - 23:59:58。

DIR_Name[0]：

文件名的第 0 Byte（DIR_Name[0]）比较特殊，要专门提出来注意一下：

- 如果 DIR_Name[0]等于 0xE5，则表示该目录项是空的，它曾经被使用，但是已经被删除了，现在没有被任何文件或文件夹占用。
- 如果 DIR_Name[0]等于 0x00，则表示该目录项是空的，并且没有任何目录项在这之后了(这之后的所有项 DIR_Name[0]值都会是 0x00)。如果 FAT 系统程序看到某一项 DIR_Name[0]的值为 0x00，就不用再往下读取目录项了，因为它们全都是空的了。
- 如果 DIR_Name[0]等于 0x05，则实际上完全等效为 0xE5。因为 0xE5 在日语字符集中用特殊应用，所以使用 0x05 代替 0xE5，不管日语系统能否处理该想文件名，都不会造成将该项看成是空的项。

如下字符不能出现在 DIR_Name 中的任何位置上：

- 小于 0x20 的字符（0x05 在 DIR_Name[0]除外）
- 0x22, 0x2A, 0x2B, 0x2C, 0x2E, 0x2F, 0x3A, 0x3B, 0x3C, 0x3D, 0x3E, 0x3F, 0x5B, 0x5C, 0x5D, and 0x7C.

下面是一个 FAT32 的根目录：

```
00000000  54 45 53 54 34 20 20 20  54 58 54 20 18 1D 27 5D  TEST4   TXT ..']
00000010  6B 3B 6B 3B 00 00 28 5D  6B 3B 00 00 00 00 00 00  k;k;..(]k;......
00000020  54 45 53 54 35 20 20 20  54 58 54 20 18 32 51 5D  TEST5   TXT .2Q]
00000030  6B 3B 6B 3B 00 00 52 5D  6B 3B 00 00 00 00 00 00  k;k;..R]k;......
00000040  54 45 53 54 20 20 20 20  54 58 54 20 18 5F 8B 5A  TEST    TXT ._嫉
00000050  6B 3B 6C 3B 00 00 EB 5B  6B 3B 07 00 00 08 00 00  k;l;..隱k;......
00000060  44 49 52 34 20 20 20 20  20 20 20 10 08 99 78 5D  DIR4       ..櫝]
00000070  6B 3B 6B 3B 00 00 79 5D  6B 3B 0F 00 00 00 00 00  k;k;..y]k;......
00000080  44 49 52 35 20 20 20 20  20 20 20 10 08 0E BC 5D  DIR5       ...糬
00000090  6B 3B 6B 3B 00 00 BD 5D  6B 3B 10 00 00 00 00 00  k;k;..絽k;......
000000A0  54 45 53 54 32 20 20 20  54 58 54 20 18 2B 64 5B  TEST2   TXT .+d[
000000B0  6B 3B 6B 3B 00 00 D9 5B  6B 3B 05 00 00 04 00 00  k;k;..贓k;......
000000C0  44 49 52 36 20 20 20 20  20 20 20 10 08 89 CB 5D  DIR6       ..壩]
000000D0  6B 3B 6B 3B 00 00 CC 5D  6B 3B 11 00 00 00 00 00  k;k;..蘑k;......
000000E0  44 49 52 37 20 20 20 20  20 20 20 10 08 1F D5 5D  DIR7       ...誡
000000F0  6B 3B 6B 3B 00 00 D6 5D  6B 3B 12 00 00 00 00 00  k;k;..謁k;......
00000100  44 49 52 38 20 20 20 20  20 20 20 10 08 9D D8 5D  DIR8       ..漸]
00000110  6B 3B 6B 3B 00 00 D9 5D  6B 3B 13 00 00 00 00 00  k;k;..贀k;......
00000120  54 45 53 54 33 20 20 20  54 58 54 20 18 8B 24 5C  TEST3   TXT .?\
00000130  6B 3B 6B 3B 00 00 3C 5C  6B 3B 0B 00 00 02 00 00  k;k;..<\k;......
```

```
00000140   44 49 52 31 20 20 20 20  20 20 20 10 08 C6 11 5D   DIR1      ..?]
00000150   6B 3B 6C 3B 00 00 12 5D  6B 3B 0C 00 00 00 00 00   k;l;...]k;......
00000160   44 49 52 32 20 20 20 20  20 20 20 10 08 41 15 5D   DIR2      ..A.]
00000170   6B 3B 6B 3B 00 00 16 5D  6B 3B 0D 00 00 00 00 00   k;k;...]k;......
00000180   44 49 52 33 20 20 20 20  20 20 20 10 08 00 20 5D   DIR3      ... ]
00000190   6B 3B 6B 3B 00 00 20 5D  6B 3B 0E 00 00 00 00 00   k;k;.. ]k;......
000001A0   E5 49 52 39 20 20 20 20  20 20 20 10 08 33 EB 5D   鎲R9      ..3?]
000001B0   6B 3B 6B 3B 00 00 EC 5D  6B 3B 14 00 00 02 00 00   k;k;..?]k;......
000001C0   E5 49 52 31 30 20 20 20  20 20 20 10 08 76 F0 5D   鎲R10     ..v?]
000001D0   6B 3B 6B 3B 00 00 F1 5D  6B 3B 15 00 00 02 00 00   k;k;..?]k;......
000001E0   E5 49 52 31 31 20 20 20  20 20 20 10 08 2A F7 5D   鎲R11     ..*?]
000001F0   6B 3B 6B 3B 00 00 F8 5D  6B 3B 16 00 00 02 00 00   k;k;..?]k;......
```

看第一项（最前面 32 Bytes）

```
00000000   54 45 53 54 34 20 20 20  54 58 54 20 18 1D 27 5D   TEST4   TXT ..']
00000010   6B 3B 6B 3B 00 00 28 5D  6B 3B 00 00 00 00 00 00   k;k;..(]k;......
```

套入上表，偏移 0x0A 数据为 0x20，是一个文件，文件名是 TEST4.TXT，文件长度是 0 字节，空文件。

最后一项：

```
000001E0   E5 49 52 31 31 20 20 20  20 20 20 10 08 2A F7 5D   鎲R11     ..*?]
000001F0   6B 3B 6B 3B 00 00 F8 5D  6B 3B 16 00 00 02 00 00   k;k;..?]k;......
```

第 0 Byte 为 0xE5，表示该项已经被删除。

再看第 11 项：

```
00000140   44 49 52 31 20 20 20 20  20 20 20 10 08 C6 11 5D   DIR1      ..?]
00000150   6B 3B 6C 3B 00 00 12 5D  6B 3B 0C 00 00 00 00 00   k;l;...]k;......
```

偏移 0x0A 数据为 0x10，是一个子目录（文件夹），名字是 DIR1，起始簇号是 0x0C。想要看到 DIR1 的内容，就要跳转到簇 0x0C 的位置。

下面是簇 0x0C 的第一个 Sector 内容：

```
00000000   2E 20 20 20 20 20 20 20  20 20 20 10 00 C6 11 5D   .         ..?]
00000010   6B 3B 6B 3B 00 00 12 5D  6B 3B 0C 00 00 00 00 00   k;k;...]k;......
00000020   2E 2E 20 20 20 20 20 20  20 20 20 10 00 C6 11 5D   ..        ..?]
00000030   6B 3B 6B 3B 00 00 12 5D  6B 3B 00 00 00 00 00 00   k;k;...]k;......
00000040   E5 74 00 00 00 FF FF FF  FF FF FF 0F 00 20 FF FF   鎟...       ..
00000050   FF FF FF FF FF FF FF FF  FF FF 00 00 FF FF FF FF               ..
00000060   E5 B0 65 FA 5E 28 00 59  00 29 00 0F 00 20 20 00   灏e鷁(.Y.)...  .
00000070   87 65 2C 67 87 65 63 68  2E 00 00 00 74 00 78 00   嘆,g嘆ch....t.x.
00000080   E5 C2 BD A8 28 59 7E 31  54 58 54 20 00 22 2E 4F   迂建(Y~1TXT ."?O
00000090   6C 3B 6C 3B 00 00 2F 4F  6C 3B 00 00 00 00 00 00   l;l;../Ol;......
000000A0   E5 31 46 31 20 20 20 20  54 58 54 20 18 22 2E 4F   ?F1   TXT ."?O
000000B0   6C 3B 6C 3B 00 00 51 4F  6C 3B 17 00 00 02 00 00   l;l;..QOl;......
000000C0   42 74 00 78 00 74 00 00  00 FF FF 0F 00 52 FF FF   Bt.x.t...  ..R
```

```
000000D0  FF FF FF FF FF FF FF FF  FF FF 00 00 FF FF FF FF           ..
000000E0  01 73 00 75 00 62 00 64  00 69 00 0F 00 52 72 00  .s.u.b.d.i...Rr.
000000F0  31 00 66 00 69 00 6C 00  65 00 00 00 31 00 2E 00  1.f.i.l.e...1...
00000100  53 55 42 44 49 52 7E 31  54 58 54 20 00 22 2E 4F  SUBDIR~1TXT .".O
00000110  6C 3B 6D 3B 00 00 E9 84  6D 3B 14 00 00 0E 00 00  l;m;..閥m;......
00000120  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000130  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000140  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000150  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000160  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000170  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000180  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000190  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000001A0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000001B0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000001C0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000001D0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000001E0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000001F0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
```

第 1 项：

```
00000000  2E 20 20 20 20 20 20 20  20 20 20 10 00 C6 11 5D  .        ..?]
00000010  6B 3B 6B 3B 00 00 12 5D  6B 3B 0C 00 00 00 00 00  k;k;...]k;......
```

偏移 0x0A 为 0x10，名字为 0x2E(“.”)，表示当前目录。

第 2 项：

```
00000020  2E 2E 20 20 20 20 20 20  20 20 20 10 00 C6 11 5D  ..       ..?]
00000030  6B 3B 6B 3B 00 00 12 5D  6B 3B 00 00 00 00 00 00  k;k;...]k;......
```

偏移 0x0A 为 0x10，名字为 0x2E 0x2E(“..”)，表示上一级目录。

```
00000100  53 55 42 44 49 52 7E 31  54 58 54 20 00 22 2E 4F  SUBDIR~1TXT .".O
00000110  6C 3B 6D 3B 00 00 E9 84  6D 3B 14 00 00 0E 00 00  l;m;..閥m;......
```

偏移 0x0A 为 0x20，是一个文件，名字为“SUBDIR~1.TXT”，文件长度为 0x00000E00，文件起始位置在簇 0x00000014。

为了读取该文件，还需要参照 FAT 得到它的簇链。首先要解决一个问题，就是怎么知道一个簇在 FAT 表中的位置？

给定一个簇号 N，确定它在表中的位置（下面是以 FAT16/FAT32 为例，FAT12 稍微复杂些）：

在 FAT 表中的 Sector 号为：N/(BPB_BytsPerSec/fact)，如果是 FAT16 则 fact 取 2，如果是 FAT32 则 fact 取 4。

在 Sector 内的偏移地址为：N%(BPB_BytsPerSec/fact)，即 N 除以(BPB_BytsPerSec/fact)取余数。

因此，簇 0x00000014 在 FAT 表中的 Sector 为 0x14/(512/4)=0，Sector 内偏移为 0x14%(512/4)=0x50

下面是 FAT1 的第 0 个 Sector：

```
00000000  F8 FF FF FF FF FF FF FF  03 00 00 00 FF FF FF 0F  ?       ....    .
00000010  00 00 00 00 06 00 00 00  FF FF FF 0F 08 00 00 00  ........    .....
00000020  09 00 00 00 0A 00 00 00  FF FF FF 0F FF FF FF 0F  ........    .    .
00000030  FF FF FF 0F FF FF FF 0F  FF FF FF 0F FF FF FF 0F     .    .    .    .
00000040  FF FF FF 0F FF FF FF 0F  FF FF FF 0F FF FF FF 0F     .    .    .    .
00000050  15 00 00 00 16 00 00 00  17 00 00 00 18 00 00 00  ................
00000060  19 00 00 00 1A 00 00 00  FF FF FF 0F 00 00 00 00  ........    .....
00000070  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000080  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000090  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000000A0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000000B0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000000C0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000000D0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000000E0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000000F0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000100  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000110  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000120  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000130  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000140  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000150  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000160  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000170  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000180  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000190  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000001A0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000001B0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000001C0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000001D0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000001E0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000001F0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
```

FAT32 表每簇占用 4 Bytes，从 Sector 内偏移地址 50 读取 4 Bytes 数据为 0x15，表示之后的数据占用簇 0x15，簇 0x15 在 Sector 内偏移地址为 0x54，从地址 0x54 读取 4 Bytes 数据为 0x16，依次类推，最后在偏移地址 0x68 读取 4 Bytes 数据为 0x0FFFFFFF，表示文件在该簇就结束了。所以该文件占用的簇号依次为 0x14，0x15，0x16，0x17，0x18，0x19 和 0x20。

九、 长文件名

长文件名是在原有的 FAT 系统上引入的，在只支持短文件名的系统上，长文件名就像是不存在一样。为了达到这个目标，长文件名通过在原有的目录项中引入新的属性字（Attribute）得以实现。

```
ATTR_LONG_NAME  = ATTR_READ_ONLY |
                  ATTR_HIDDEN |
                  ATTR_SYSTEM |
                  ATTR_VOLUME_ID
```

判断一个目录项是否为长文件名，要通过下面 MASK 实现：

```
ATTR_LONG_NAME_MASK = ATTR_READ_ONLY |
                  ATTR_HIDDEN |
                  ATTR_SYSTEM |
                  ATTR_VOLUME_ID |
                  ATTR_DIRECTORY |
                  ATTR_ARCHIVE
```

长文件名目录项数据结构如下：

Name	**Offset (byte)**	**Size (bytes)**	**Description**
LDIR_Ord	0x00	1	该项在这组长文件名中的序号。 如果第 6 bit 为 1（Mask with 0x40），说明这是该组长文件名的最后一项。每一组有效的长文件名都必须有这个标志位。
LDIR_Name1	0x01	10	长文件名的第 1-5 个字符
LDIR_Attr	0x0B	1	属性字 – 必须为 0x0F（ATTR_LONG_NAME）
LDIR_Type	0x0C	1	如果是 0，则表示这个目录项是长文件名的一部分。 非 0 数值为保留设置。
LDIR_Chksum	0x0D	1	对应的短文件名校验和（Checksum）
LDIR_Name2	0x0E	12	长文件名的第 6-11 个字符
LDIR_FstClusLO	0x1A	2	Must be ZERO. This is an artifact of the FAT "first cluster" and must be zero for compatibility with existing disk utilities. It's meaningless in the context of a long dir entry.
LDIR_Name3	0x1C	4	长文件名的第 12-13 个字符

长文件名字符采用 Unicode 编码。

Checksum 算法，用 C 语言实现如下：

```
//-------------------------------------------------------------------------
//  ChkSum()
//  Returns an unsigned byte checksum computed on an unsigned byte
//  array.  The array must be 11 bytes long and is assumed to contain
//  a name stored in the format of a MS-DOS directory entry.
```

```
//  Passed:  pFcbName    Pointer to an unsigned byte array assumed to be
//                       11 bytes long.
//  Returns: Sum        An 8-bit unsigned checksum of the array pointed
//                       to by pFcbName.
//-----------------------------------------------------------------------
unsigned char ChkSum (unsigned char *pFcbName)
{
    short FcbNameLen;
    unsigned char Sum;

    Sum = 0;
    for (FcbNameLen=11; FcbNameLen!=0; FcbNameLen--) {
        // NOTE: The operation is an unsigned char rotate right
        Sum = ((Sum & 1) ? 0x80 : 0) + (Sum >> 1) + *pFcbName++;
    }
    return (Sum);
}
```

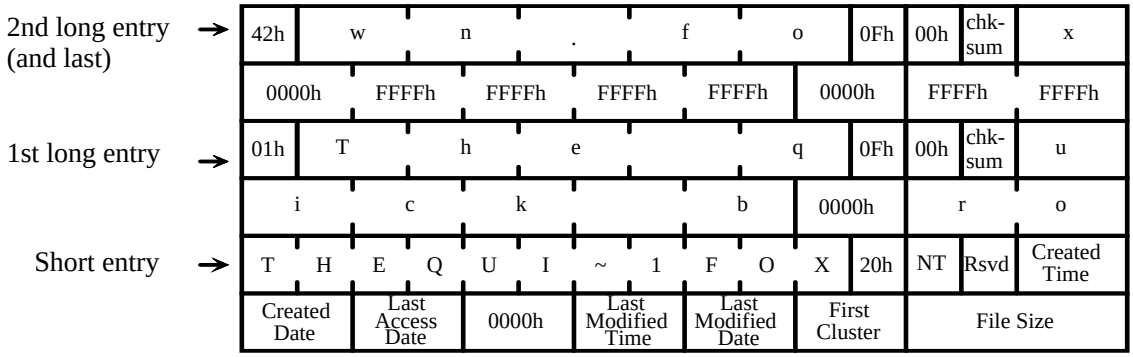
下面是一组长文件名目录项。

```
00799640  43 70 00 71 00 72 00 73  00 74 00 0F 00 52 2E 00  Cp.q.r.s.t...R..
00799650  74 00 78 00 74 00 00 00  FF FF 00 00 FF FF FF FF  t.x.t...  ..
00799660  02 63 00 64 00 65 00 66  00 67 00 0F 00 52 68 00  .c.d.e.f.g...Rh.
00799670  69 00 67 00 6B 00 6C 00  6D 00 00 00 6E 00 6F 00  i.g.k.l.m...n.o.
00799680  01 73 00 75 00 62 00 64  00 69 00 0F 00 52 72 00  .s.u.b.d.i...Rr.
00799690  66 00 69 00 6C 00 65 00  5F 00 00 00 61 00 62 00  f.i.l.e._...a.b.
007996A0  53 55 42 44 49 52 7E 31  54 58 54 20 00 22 2E 4F  SUBDIR~1TXT ".O
007996B0  6C 3B 71 3B 00 00 E9 84  6D 3B 14 00 00 0E 00 00  l;q;..閥m;......
```

可以看出，长文件名为“subdirfile_abcdefghigklmnopqrst.txt”，对应的短文件名为“SUBDIR~1.TXT”。

下面是《Microsoft Extensible Firmware Initiative FAT32 File System Specification—version1.03》中给出的示例：

假设创建一个名为“The quick brown.fox”的文件，系统将为它建立如下的目录项：



参考资料

- Microsoft Extensible Firmware Initiative FAT32 File System Specification--version1.03
- FAT文件系统原理--http://www.sjhf.net