第八章　FAT32 文件系统详解

DESCRIPTION：

OS:　MICROSOFT WINDOWS 7、MICROSOFT PROFESSIONAL XP SP3

SOFTWARE:　WINHEX15.2 SR-10

HARDWARE:　MEGA16、KINGSTON 2G SD CARD

AUTHOR:　FGD

TIME:　20090808

8.1　Microsoft

比尔·盖茨于 2008 年 6 月 27 日退休了，他在微软同事的心目中是一个什么形象呢？这个当属与他一起共同执掌了微软 28 年之久的 CEO 鲍尔默最有话语权了。“他是一个比较内向的小伙子，不太爱说话，但浑身充满了活力，尤其是一到晚上就活跃起来。当时的情况是，经常在我早上醒来时，他才准备睡觉。”鲍尔默在最近接受《华尔街日报》采访时，如此形容比尔·盖茨。鲍尔默说的对，也许只有活力才是成功的最关键因素，这是比尔·盖茨留给大家最好的礼物！

1. LIFE IS UNFAIR, YOU WANT TO ADAPT IT.

2. THE WORLD WILL NOT TAKE YOUR SELF-ESTEEM, BUT FOR THE SELF-SATISFACTION BEFORE YOU HAVE SUCCESS.

3. JUST RETURNED FROM THE SCHOOL COME OUT WHEN YOU CAN NOT EARN 60,000 U.S. DOLLARS A MONTH, BUT WILL NOT BECOME ANY COMPANY VICE PRESIDENT, ALSO OWNED A CAR UNTIL YOU HAVE WON THE HAND OF THOSE THAT DAY.

4. IF YOU THINK SCHOOL TEACHERS IS TOO HARSH, THEN YOU HAVE TO THINK BACK TO THE BOSS.

5. SELLING HAMBURGER AND NOT DETRIMENTAL TO YOUR DIGNITY. YOUR GRANDPARENTS HAD A DIFFERENT UNDERSTANDING TO SELL HAMBURGER, THEY CALLED IT "OPPORTUNITY".

6. IF YOU GET INTO DIFFICULTIES, IT IS NOT THE FAULT OF YOUR PARENTS, YOU SHOULD NOT BE THE RESPONSIBILITY ONTO OTHERS, AND TO LEARN TO LEARN FROM IT.

7. BEFORE YOU WERE BORN, YOUR PARENTS DO NOT LIKE SO BORING. THEY LOOK INTO THIS TODAY BECAUSE THESE YEARS HAVE BEEN FOR YOU TO PAY BILLS, TO YOUR LAUNDRY. SO, IN TALKING TO PARENTS, OR WHATEVER CLEANING YOUR OWN HOUSE?

8. YOU MAY NO LONGER HOST SCHOOL HOURS AND POOR HEALTH THEMSELVES, BUT LIFE IS NOT THE CASE. IN SOME SCHOOLS HAD NOT "FAIL" CONCEPT, THE SCHOOL WILL CONTINUE TO GIVE YOU THE OPPORTUNITY FOR YOU TO PROGRESS, BUT REAL LIFE IS NOT LIKE THAT.

9. UNLIKE IN THE LIFE OUT OF SCHOOL AFTER A SEMESTER OF THE SAME SCHOOL HOURS, NOR THAT THE SUMMER. NO BOSS TO HELP YOU FIND SOME SELF-AND YOU MUST RELY ON ITS OWN TO COMPLETE.

10. MANY OF THE SCENES ON TELEVISION IS NOT REAL LIFE. IN REAL LIFE, PEOPLE MUST DO THEIR BURIED IN HIS WORK, NOT LIKE TV WHERE MY DAILY DWELL IN THE CASE OF COFFEE LANE.

11. TREAT YOUR AVERSION TO THE PEOPLE, BECAUSE THERE DAYS YOU WILL WORK FOR SUCH A PERSON.

8.2　先说说硬盘

【笔者按：这部分知识基本是笔者对《数据重现》一书（作者：马林）读书笔记的整理，要了解详更详细的细节，可参考该书或求助互联网。】

8.2.1 硬盘结构

硬盘基本上由两大部分组成：控制电路板和盘体。

1. 控制电路板

控制电路板是由接口、DSP 处理器、ROM、缓存、磁头驱动电路和盘片电机驱动电路等等组成。

2. 盘体

盘体由盘腔、上盖、盘片电机、盘片、磁头、音圈和其他辅助组件组成。

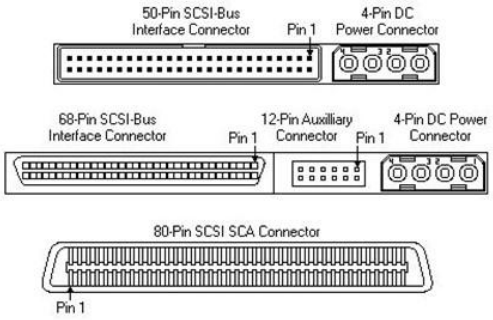
【以上专业名词希望读者可以通过互联网了解】

8.2.2 硬盘接口

1. IDE 接口（Integrated Drive Electronics）

2. SCSI 接口（Small Computer System Interface）

SCSI 硬盘接口有三种，分别是 50 针（N-Narrow）、68 针（W-Wide）和 80 针（SCA-Single Connector Attachment）。



3. SATA 接口（Serial-ATA）

IDE 系列属于 Parallel-ATA（并行），SATA 是一种新的标准，目前已成为硬盘的主流接口。

4. SAS 接口（Serial Attached SCSI）

即串行 SCSI 接口。

5. IEEE1394 接口

IEEE1394 并不是硬盘专用接口，但它却可以方便的链接包括硬盘在内的 63 个不同设备，并支持即插即用和热插拔。在数据传输率方面，IEEE1394 可以提供 100MB/S、400MB/S、1.2GB/S 三档高速传输率，是现在所有硬盘望尘莫及的。

8.2.3 磁道、扇区、柱面

1. 磁道

当磁盘在旋转时，磁头若保持在一个位置上，则每个磁头都会在磁盘表面划出一个圆形轨迹，这些圆形轨迹就叫做磁道。每张盘片上的磁道由外向内依次从“0”开始进行编号。虽然磁道的编号是“由外向内依次从“0”开始进行编号”，但这并不意味着“0”磁道是位于磁盘片的最外沿的。固件区的物理位置有的位于比“0”磁道更靠近磁盘片的外缘的磁道上。有的位于磁盘片的中部。

2. 扇区

磁盘上的每个磁道被等分为若干个弧段，这些弧段便是磁盘的扇区。每个扇区大小为 512 字节。扇区从“1”开始编号。

3. 柱面

磁盘通常由重叠的一组盘片构成。前面提到，每个盘面都被划分为树木相等的磁道，并从外圈的“0”开始编号，具有相同编号的磁道形成一个圆柱，这个圆柱我们称之为磁盘的柱面。磁盘上数据的存取是沿柱面进行的，也就是在一个柱面内依次从低号盘片向高号盘片写入，写满一个柱面后再转到下一个柱面。

磁盘的柱面数与一个盘面上的磁道数相等。由于每一个盘面都有自己的磁头，因此，盘面数等于总的磁头数。所谓硬盘的 CHS，既是 Cylinder（柱面）、Head（磁头）、Sector（扇区）。

8.2.4 硬盘的启动过程

（1）硬盘上电后，DSP 首先运行 ROM 中的程序，部分硬盘会检查各部件的完整性。

（2）然后盘片电机启动，当转速达到预定的转速时，磁头开始运行，定位到盘片的固件区，读取硬盘的固件程序和坏道表，部分硬盘会先将 ROM 中记忆的系列号与盘片上的进行比较，如果不一致，硬盘会终止初始化工作。

（3）当所有必须的固件正确读出后，磁盘即进入就绪状态，等待接收指令进行数据的读写操作。

8.2.5 硬盘的性能指标

1.硬盘的转速

2.硬盘的数据传输率

3.硬盘缓存

缓存是硬盘与外部总线交换数据的场所。

4.平均寻道时间

平均寻道时间指的是从硬盘接到相应指令开始到磁头转移到指定磁道位置为止所用的平均时间。

5.柱面切换时间

6.平均潜伏期

平均潜伏期指的是磁头移动到指定磁道后，还需要多少时间指定的（即要读写的）扇区才会转到磁头下进行读取或写入的相关操作。换算公式：

(60*1000) / (硬盘转速*2) = 平均潜伏期

7.平均方位时间

8.2.6 寻址方式

所谓寻址方式，大哥比方就好比我们看一本书，要从中找到我们要读的章节，那么要怎样才能纸袋它所在的位置呢？通常我们可以目录中找到它所在的页数，然后找到它。还有一种方法就是将所有章节都进行编号，我们只要知道它章节号，也可以找到它所在位置。

同样，要访问硬盘上的数据，也需要合适的寻址方式方可准确读取数据。硬盘上的寻址方式有两种：

& C/H/S 寻址方式

使用 Cylinder（柱面）、Head（磁头）、Sector（扇区）三个参数来定位唯一的扇区地址。

&LBA 寻址方式

LBA 寻址方式即 Logic Block Address（逻辑块地址），又称为“线性寻址模式”。

（插入图片）

8.3 数的存储格式

数的存储格式也就是数字的存储顺序。

& Big-endian：也被称作“大头位序”或“大端模式”。字节由最高位向最低位依次存放，高位在前，地位在后。

& Little-endian：也被称作“小头位序”或“小端模式”。字节由最低位向最高位依次存放，低位在前，高位在后。

例如：有一个十六进制数“00 23 0f 4a”：

Big-endian: 00 23 0f 4a

Little-endian: 4a 0f 23 00

在进行数据操作的时候一定要注意大小端存放格式，否则后果就是，你应该知道的。

8.4 DOC 分区

8.4.1 概述

说明：虽然我们的存储介质没有分区（你的 u 盘，sd 卡等小容量存储介质肯定没有分区的吧）。但是在文件系统初始化的开始我们必须通过 MBR 获取分区表项的数据，以获得 CHS、LBA 参数以及分区大小扇区数，否则就无法知道你文件系统的起始物理扇区号!!!

Microsoft 将使用 DOC 分区体系的磁盘称为“主引导记录（Master Boot recorder， MBR)”磁盘，这是对于使用“全局 ID 分区表（GUID Partition Table， GPT）磁盘”而言的。

很多人一提到“DOC 分区”，就认为只是指 windows 下使用的分区体系，这是对“DOC 分区”的一个理解错误。“DOC 分区”并不是以操作系统的不同而划定的分区体系，而是指使用“主引导记录（MBR)”的分区体系。

8.4.2 主引导记录扇区

使用“DOC 分区”体系时，磁盘的第一个——也就是 0 号扇区被称为主引导记录扇区，也称为主引导记录 MBR（Master Boot recorder，MBR)。

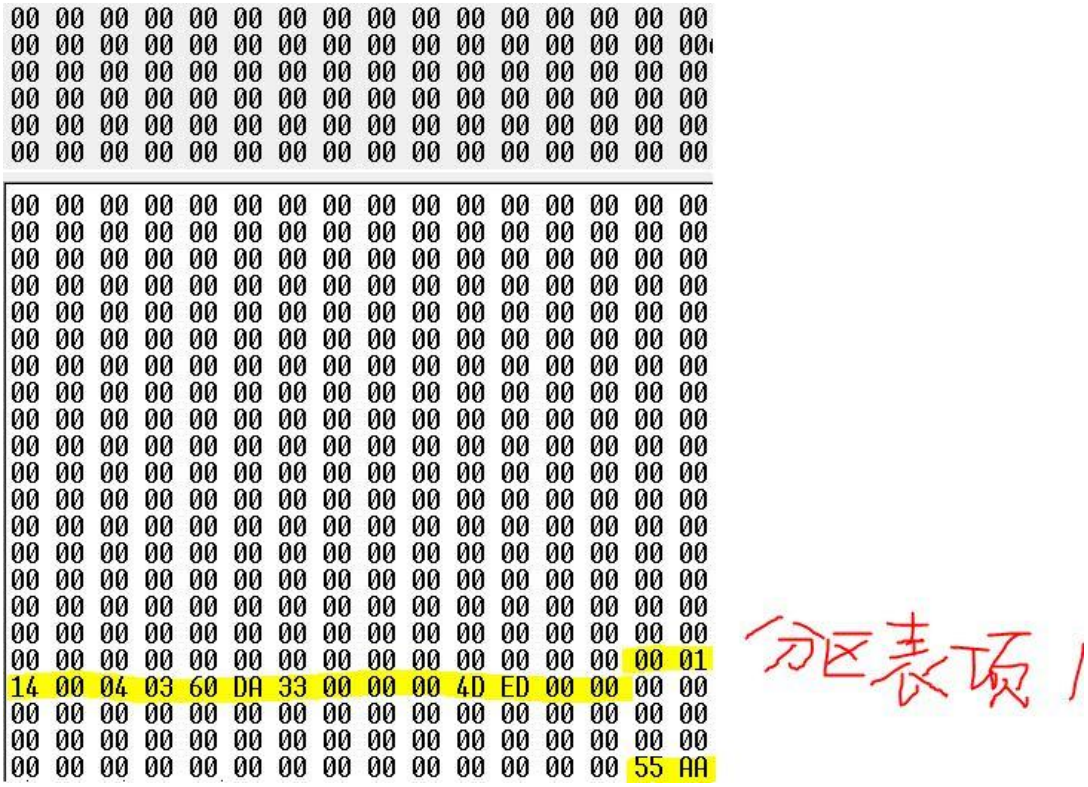
1. MBR 数据结构

MBR 由 446 个字节的引导代码、64 字节的主分区（4 个）表及两个字节的签名值“55 AA”组成。

我们可以用 WinHex 打开一个 SD 卡（Canon 的牌子，32MB，文件系统：FAT16）的 MBR 查看：

（说明，不知为什么，用 WinHex 无法看到 SD 卡文件系统以前的扇区数据，也许是笔者没有发现吧！没有 WinHex，怎么办？好办！我们可以通过读取 SD 卡 0 号扇区的数据，然后用串口发回到计算机的超级终端观察。所以下面的数据是笔者在调试过程中用 uart 获取的）

注：在使用WinHex打开SD卡的时候，要选择物理硬盘那一类而不是逻辑驱动器那一类，才能在物理0扇区看到MBR，



我们可以看到前 446 字节的数据全是 0，怎么回事？我也不知道，可能是因为 SD 卡不存在系统引导问题，所以将所有数据用 0 来填充。

接下来的 64 字节的数据就是我们所关心的重要数据。首先先介绍一下分区表项的数据结构：

分区表项共占用 64 字节数据，共四个分区表项，图示中用黄色长条标记的是第一个分区表项。

分区表项数据结构

偏移（十六进制）	字节数	描述
00~00	1	可引导标志，0x00 不可引导，0x80 可引导
01~03	3	分区起始 CHS 地址
04~04	1	分区类型
05~07	3	分区结束 CHS 地址
08~0B	4	分区起始 LBA 地址（Little-endian 顺序）
0C~0F	4	分区大小扇区数（Little-endian 顺序）

说明：

分区起始 CHS 地址：

第一个字节用于记录分区起始磁头号。

第二个字节的低 6 位用于记录分区起始扇区号

用 10 个 bit 记录分区起始柱面号：第二个字节的高两位作为起始柱面号的高两位，第三个字节的 8 个 bit 作为分区起始柱面号的低 8 位。

现在我们来分析一下上面所示图中的数据：

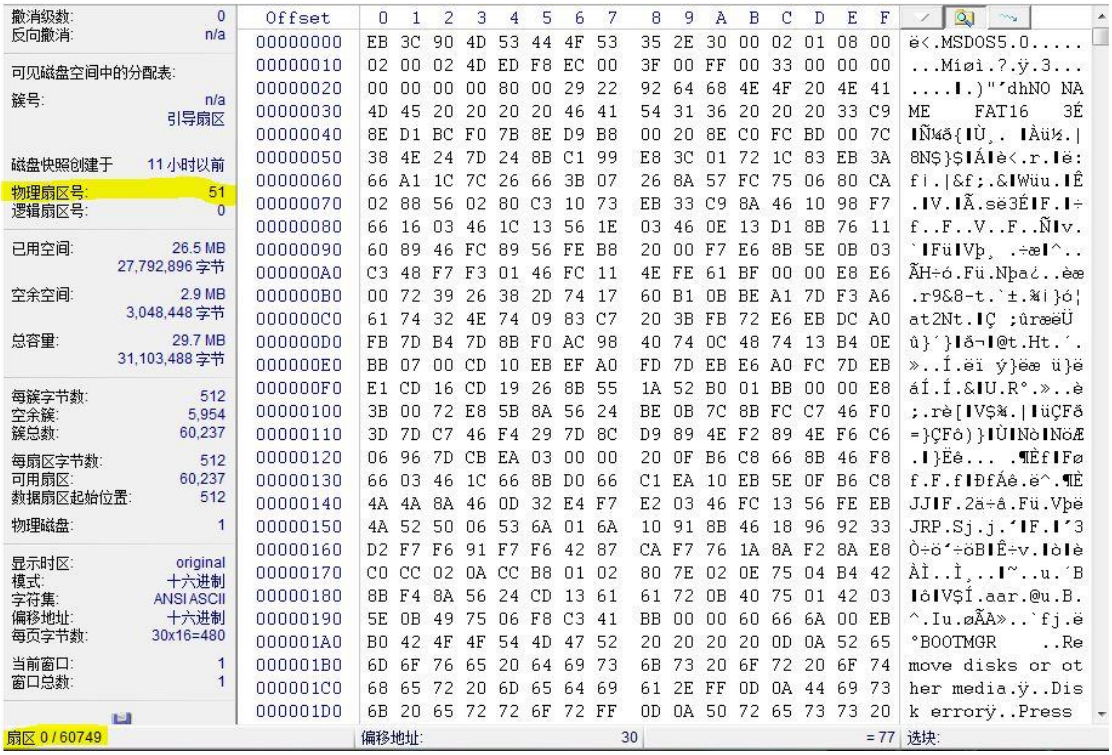
00-00（00）不可引导

01~03（00 14 01） 该处我们不做分析，因为 SD 卡不存在柱面问题，有兴趣的朋友可有用 WinHex 打开一个 FAT32 的硬盘，然后算算 CHS。

04~04（04） 分区标志。这个地方有一张表格（分区标志类型值及其含义），请读者自行上网查询。

05~07（DA 60 03）此处不介绍

08~0B（00 00 00 33）这正是我们要获取的数据，通过计算可知它是 51，它很重要，因为它就是我们 SD 卡上文件系统开始的物理扇区号，我们可以用 WinHex 打开 SD 卡查看，在左边的记录中发现，文件系统的 DBR 起始物理扇区号果真就是 51!!! 如下图所示：



0C~0F（00 00 ED 4D）60749 这就是分区大小的扇区数，可以这么说，SD 卡只有一个分区，我们在 WinHex 中看到的总扇区数即除去 MBR（还有隐藏扇区）剩余的扇区总数，在 WinHex 中同样可知，如上图下方黄色区域标记的一样。

8.4.3 扩展引导记录扇区 EBR

MBR 扇区只包含 4 个分区表项，最多只能描述 4 个分区。大多数用户都会将自己的硬盘分为 4 个分区甚至更多，那怎么办呢？那就交给扩展引导记录扇区 EBR 去处理。

由于这里的知识不在我们的讨论范围，有兴趣的读者可以查看《数据重现》一书，上面有详细的介绍。

好了，DOC 分区的知识就介绍这么多，感兴趣的读者还可以通过互联网获取更多的知识。知识的获取是一个积累的过程，在积累中提高自己，在积累中丰富自己的生活。

接下来就开始我们的文件系统之旅，读者有可能看过一遍后有不理解的地方，没关系，继续看第二遍、第三遍、第四遍（要知道笔者当时连续用了两个通宵来了解它!!!）……上网查找资料，有一天，你总会这样感觉到，原来文件系统就那么回事儿，呵呵。好，Let us go!!!

8.5 认识文件系统

8.5.1 FAT 文件系统简介

初次接触文件系统，很多读者或许都和笔者有同感：到底文件系统是什么东西？？？

文件系统就是对数据进行存储与管理的方式。

怎么样？听上去还好理解吧。就好比在我们自己的电脑上，现在我们的硬盘都是几百个 G 容量，你天天往里面存放各种各样不同的数据，有文档文件，有音乐文件，有视频文件等等等。想象一下，如果你的电脑没有一个管理数据的好家伙，我的天，那将无法想象!!! 你的数据将被乱七八糟的存放着，更要命的是，你根本无法再从硬盘中读取你之前存放的数据，因为你根本不知道它们在哪。有了文件系统，那就简单了，你可以随意将任何文件存放在任何地方，有的文件不想让人一下就看见，行，你把它放到安装 windows 系统的系统盘目录下，再觉得不爽，好，那就隐藏了它。文件系统会在你把文件存放到某个目录下时记录下你文件的信息（文件存放的起始簇号，文件大小，文件创建、修改、访问、保存的时间等等），当你下次要打开那个文件时，文件系统就根据已知的文件信息去寻找它，找到后，你就又可以读，写，修改，移动你的文件了，并且文件系统会同时更新。

上面的描述是希望让读者对文件系统有一个感官的认识，接下来，我们详细的讲解文件系统。

文件系统是为了长久的存储和访问数据而为用户提供的一种基于文件和目录的存储机制。我们都知道，在使用硬盘存储数据之前，首先要进行分区（当然你也可以不分区），然后对分区（或整个硬盘）进行格式化，其实格式化的过程就是在分区内建立文件系统的过程。一个文件系统由系统结构和按一定规则存放的用户数据组成。日常，我们都有这样的经历，在 windows 下当我们要格式化一个分区或是其他存储介质时，windows 会弹出一个对话框，上面有这样一些选择内容：容量、文件系统、分配单元大小、卷标等等。其中文件系统的下拉菜单中就有几种不同的文件系统共供户选择，一般我们都会选择默认、FAT32 或 NTFS 文件系统，当我们按下格式化按钮后，操作系统就开始为这个分区建立你所选择的文件系统。

文件系统种类繁多，但所有的文件系统都有一定的共性

1. 数据单元

数据在写入磁盘或从磁盘读取数据时每次操作的数据量称为数据单元，它的大小在建立文件系统时确定。数据单元在不同的文件系统中有不同的称呼：例如在 FAT 和 NTFS 文件系统中称作“簇（Cluster）”，ExtX 中称作“块（Block）”等。一个数据单元由若干个连续的扇区组成，大小总是 2 的整数次幂个扇区。

2. 坏数据单元

坏数据单元也就是包含缺陷扇区的数据单元。

3. 逻辑文件系统地址

磁盘上的一个扇区在不同的情况下会有不同的地址表达形式。

&每个扇区都会有一个 LBA 地址，也就是物理地址

&每个物理卷内的扇区又有一个物理卷地址

&在逻辑卷内部的扇区会有一个逻辑卷地址

（以上几个概念比较容易混淆，注意区分）

4. 逻辑文件地址

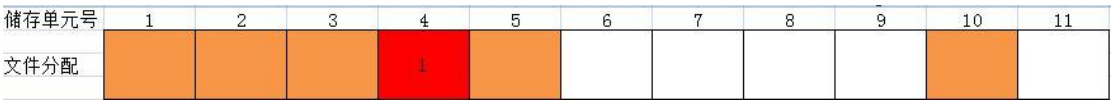
对于每个文件来说，将它按所在文件系统中的数据单元大小为单位进行分割，分割后的每一个部分由 0 开始编号，这个编号就是其对应数据单元的逻辑文件地址。一个文件前后相邻的两个数据单元在物理上的存储地址可能是不连续的，但它的逻辑文件地址一定是连续的。

5. 分配策略

【1】第一可用分配策略，即当为一个文件分配了一个存储单元后还要继续为其分配时，操作系统会重新从文件系统的起始处搜索可以使用的空间。

举例说明，加入我们现在有一个文件需要 4 个储存单元，文件系统内的 1、2、3、5、10 号储存单元已经有数据放在那了，4、6、7、8、9 号储存单元空闲，按照第一分配策略为这个文件的 4 个部分分配空间时步骤如下：

<1>第一部分分配到 4 号储存单元，如下图所示：



<2>为文件第一部分分配好储存位置后，又从 1 号储存单元开始向后寻找空位置，找到 6 号后分配给文件的第二部分，如下图：

<3>为文件第二部分分配好储存位置后，又从 1 号储存单元开始向后寻找空位置，恰好这个时候，2 号储存单元的文件内容被删除了，就把 2 号位置分配给文件的第三部分，如下如：

<4>为文件第三部分分配好储存位置后，又从 1 号储存单元开始向后寻找空位置，找到 7 号后分配给文件的第四部分，如下图：

至此，这个文件的四个部分就分配好了，这四部分的储存位置就是 4-6-2-7。

【2】下一可用分配策略，即为文件分配了一个储存单元后并不再回到卷开始处重新寻找可用空间，而是直接向后进行搜索。以上文件的四个部分利用下一可用分配策略分配的最后结果如下图所示：

【3】最佳分配策略，即在为文件分配空间时，会尽可能找到足够的连续空间以避免其片段化。对于之前的例子使用最佳分配策略时，最后结果如下图：

6. 松弛空间

松弛空间分为两种，一种是数据的结尾与为其分配的空间结束处的未使用部分，还有一种就是位于分区结尾的卷松弛空间。

7. 元数据

任何文件和目录都会有一个名字，我们将其统称为“文件名”。除了文件名外，文件或目录还有其他一些描述信息，如大小，时间信息，是否加密或压缩，储存位置信息等，我们将这些描述信息统称为文件或目录的元数据。

8.6 揭开 FAT 的神秘面纱

这一节，笔者将尽力用最容易让读者理解的语言和例子来阐述，希望读者阅读这一节一定要细心，务必上机操作。

8.6.1 FAT 文件系统概述

FAT（File Allocation Table，文件分配表）文件系统是 windows 操作系统所使用的一种文件系统，它的发展过程经历了 FAT12、FAT16、FAT32 三个阶段。

FAT 文件系统用“簇”作为数据单元。一个“簇”由一组连续的扇区组成，簇所含的扇区数必须是 2 的整数次幂。簇的最大值为 64 个扇区，即 32KB。所有簇从 2 开始进行编号，每个簇都有一个自己的地址编号。用户文件和目录都存储在簇中。

FAT 文件系统的数据结构中有两个重要的结构：文件分配表和目录项：

&文件和文件夹内容储存在簇中，如果一个文件或文件夹需要多余一个簇的空间，则用 FAT 表来描述如何找到另外的簇。FAT 结构用于指出文件的下一个簇，同时也说明了簇的分配状态。FAT12、FAT16、FAT32 这三种文件系统之间的主要区别在与 FAT 项的大小不同。

&FAT 文件系统的每一个文件和文件夹都被分配到一个目录项，目录项中记录着文件名、大小、文件内容起始地址以及其他一些元数据。

在 FAT 文件系统中，文件系统的数据记录在“引导扇区中（DBR）”中。引导扇区位于整个文件系统的 0 号扇区，是文件系统隐藏区域（也称为保留区）的一部分，我们称其为 DBR（DOS Boot Recorder——DOS 引导记录）扇区，DBR 中记录着文件系统的起始位置、大小、FAT 表个数及大小等相关信息。

在 FAT 文件系统中，同时使用“扇区地址”和“簇地址”两种地址管理方式。这是因为只有存储用户数据的数据区使用簇进行管理（FAT12 和 FAT16 的根目录除外），所有簇都位于数据区。其他文件系统管理数据区域是不以簇进行管理的，这部分区域使用扇区地址进行管理。文件系统的起始扇区为 0 号扇区。

逻辑0号扇区

8.6.2 FAT 文件系统的整体布局

如图所示：

说明：

【1】 保留区含有一个重要的数据结构——系统引导扇区（DBR）。FAT12、FAT16 的保留区通常只有一个扇区，而 FAT32 的保留扇区要多一些，除 0 号扇区外，还有其他一些扇区，其中包括了 DBR 的备份扇区。

【2】 FAT 区由来年各个大小相等的 FAT 表组成——FAT1、FAT2，FAT2 紧跟在 FAT1 之后。

【3】 FAT12、FAT16 的根目录虽然也属于数据区，但是他们并不由簇进行管理。也就是说 FAT12、FAT16 的根目录是没有簇号的，他们的 2 号簇从根目录之后开始。而 FAT32 的根目录通常位于 2 号簇。

接下来的内容就是 FAT 文件系统的核心内容了!!!

8.6.3 FAT32 的保留区

FAT32 文件系统的开始部分有一个由若干个扇区组成的保留区，保留区的刀削会记录在 DBR 扇区中，比较常见的为 32、34 或 38 个扇区。

8.6.3.1 引导扇区

引导扇区是 FAT32 文件系统的第一个扇区，也称为 DBR 扇区。它包含这样一些文件系统的基本信息：

【1】 每扇区字节数

【2】 每簇扇区数

【3】 保留扇区数

【4】　FAT 表个数

【5】　文件系统大小（扇区数）

【6】　每个 FAT 表大小（扇区数）

【7】　根目录起始簇号

【8】　其他一些附加信息

（DBR 扇区中记录文件系统参数的部分也称为 BPB（BIOS Parameter Block））

说明：

&引导扇区中的某些参数是至关重要的：【1】、【2】、【3】、【4】、【5】、【6】、【7】等。

我们可以通过每个 FAT 表的大小扇区数乘以 FAT 表的个数得到 FAT 区域的大小；通过保留扇区数和 FAT 区域的大小就可以得知数据区的起始位置，也就得到了文件系统第一簇的位置。由根目录的簇号和第一簇的位置就可以得到根目录的位置。

引导扇区数据结构及实例讲解：

这个小节笔者将通过讲解一个 Kingston 2GB 的 SD 卡的 DBR（FAT32 文件系统），来向读者详细说明引导扇区数据结构各个参数的含义，先给出几张图片：

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00000000	EB	58	90	4D	53	44	4F	53	35	2E	30	00	02	08	C0	02	ëX.MSDOS5.0...À.
00000010	02	00	00	00	00	F8	00	00	3F	00	FF	00	89	00	00	00	.....ø..?.ÿ.‰...
00000020	77	9F	3A	00	A0	0E	00	00	00	00	00	00	02	00	00	00	w‰:. .....
00000030	01	00	06	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000040	80	00	29	C8	6C	9F	54	4E	4F	20	4E	41	4D	45	20	20	€.)Èl‰TNO NAME
00000050	20	20	46	41	54	33	32	20	20	20	33	C9	8E	D1	BC	F4	FAT32 3É‰Ñ¼ô
00000060	7B	8E	C1	8E	D9	BD	00	7C	88	4E	02	8A	56	40	B4	41	{‰Á‰Ù½.|ˆN.ŠV@´A
00000070	BB	AA	55	CD	13	72	10	81	FB	55	AA	75	0A	F6	C1	01	»ªUÍ.r..ûUªu.öÁ.
00000080	74	05	FE	46	02	EB	2D	8A	56	40	B4	08	CD	13	73	05	t.þF.ë-ŠV@´.Í.s.
00000090	B9	FF	FF	8A	F1	66	0F	B6	C6	40	66	0F	B6	D1	80	E2	¹ÿÿŠñf.¶Æ@f.¶Ñ€â
000000A0	3F	F7	E2	86	CD	C0	ED	06	41	66	0F	B7	C9	66	F7	E1	?÷â†ÍÀí.Af.·Éf÷á
000000B0	66	89	46	F8	83	7E	16	00	75	38	83	7E	2A	00	77	32	f‰Føƒ~..u8ƒ~*.w2
000000C0	66	8B	46	1C	66	83	C0	0C	BB	00	80	B9	01	00	E8	2B	f‹F.fƒÀ.».€¹..è+
000000D0	00	E9	2C	03	A0	FA	7D	B4	7D	8B	F0	AC	84	C0	74	17	.é,. ú}´}‹ð¬„Àt.
000000E0	3C	FF	74	09	B4	0E	BB	07	00	CD	10	EB	EE	A0	FB	7D	<ÿt.´.».Í.ëî û}
000000F0	EB	E5	A0	F9	7D	EB	E0	98	CD	16	CD	19	66	60	80	7E	ëå ù}ëà˜Í.Í.f`€~
00000100	02	00	0F	84	20	00	66	6A	00	66	50	06	53	66	68	10	...„ .fj.fP.Sfh.
00000110	00	01	00	B4	42	8A	56	40	8B	F4	CD	13	66	58	66	58	...´BŠV@‹ôÍ.fXfX
00000120	66	58	66	58	EB	33	66	3B	46	F8	72	03	F9	EB	2A	66	fXfXë3f;Før.ùë*f
00000130	33	D2	66	0F	B7	4E	18	66	F7	F1	FE	C2	8A	CA	66	8B	3Òf.·N.f÷ñþÂŠÊf‹
00000140	D0	66	C1	EA	10	F7	76	1A	86	D6	8A	56	40	8A	E8	C0	ÐfÁê.÷v.†ÖŠV@Šèà
00000150	E4	06	0A	CC	B8	01	02	CD	13	66	61	0F	82	75	FF	81	ä..Ì¸..Í.fa.‚uÿ.
00000160	C3	00	02	66	40	49	75	94	C3	42	4F	4F	54	4D	47	52	Ã..f@IuîÃBOOTMGR
00000170	20	20	20	20	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	0D	0A	52	65	..... Re
000001B0	6D	6F	76	65	20	64	69	73	6B	73	20	6F	72	20	6F	74	move disks or ot
000001C0	68	65	72	20	6D	65	64	69	61	2E	FF	0D	0A	44	69	73	her media.ÿ..Dis
000001D0	6B	20	65	72	72	6F	72	FF	0D	0A	50	72	65	73	73	20	k errorÿ..Press
000001E0	61	6E	79	20	6B	65	79	20	74	6F	20	72	65	73	74	61	any key to resta
000001F0	72	74	0D	0A	00	00	00	00	00	AC	CB	D8	00	00	55	AA	rt.....¬ËØ..Uª

第一张图片就是笔者使用的 SD 卡截图，第二章图片显示的就是通过 WinHex 获取的 DBR 数据截图。现在我们来着重分析 DBR，请读者务必在阅读完本章节后自行上机实践查看，以加深影响。

具体分析如下：

（首先说明，数据的存储是以小端模式存储的）

【1】0x00~0x02：3 个字节，跳转指令。

【2】0x03~0x0A：8 个字节，文件系统标志和版本号，这里为 MSDOC5.0。

【3】0x0B~0x0C：2 个字节，每扇区字节数，512（0X02 00）。

【4】0x0D~0x0D：1 个字节，每簇扇区数，8（0x08）。

【5】0x0E~0x0F：2 个字节，保留扇区数，704（0x02 C0）。

【6】0x10~0x10：1 个字节，FAT 表个数，2。

【7】0x11~0x12：2 个字节，根目录最多可容纳的目录项数，FAT12/16 通常为 512。FAT32 不使用此处值，置 0。

【8】0x13~0x14：2 个字节，扇区总数，小于 32MB 时使用该处存放。超过 32MB 时使用偏移 0x20~0x23 字节处的 4 字节存放。笔者的 SD 卡容量为 2GB，所以不使用该处，置 0.

【9】0x15~0x15：1 个字节，介质描述符，0xF8 表示本地硬盘。

【10】0x16~0x17：2 个字节，每个 FAT 表的大小扇区数（FAT12/16 使用，FAT32 不使用此处，置 0）。

【11】0x18~0x19：2 个字节，每磁道扇区数，63（0x00 3F）。

【12】0x1A~0x1B：2 个字节磁头数，255（0x00 FF）。

【13】0x1C~0x1F：4 个字节，分区前已使用扇区数，137（0x00 00 00 89）。（这个数据要尤其的重视，文件系统初始化的第一步要找的就是这玩意儿。因为我们的 SD 卡没有分区，默认就是一个分区，这个数据就是相对于 MBR（关于 MBR 的介绍请读者参看 8.4 小节的 DOC 分区）的地址偏移量，MBR 的扇区地址才是整个 SD 卡的物理扇区号为 0 的那个地址，也就是说文件系统并不是处在整个 SD 卡最开始的地方，它处在 MBR 所处的保留区之后，于是我们可以对使用 FAT32 文件系统的 SD 卡整体布局给出如下图示）

【14】0x20~0x23：4 个字节，文件系统大小扇区数，3841911（0x 00 3A 9F 77）。

【15】0x24~0x27：4 个字节，每个 FAT 表的大小扇区数，3744（0x 00 00 0E A0）。

【16】0x28~0x29：2 个字节，标记。

【17】0x2A~0x2B：2 个字节，版本号。

【18】0x2C~0x2F：4 个字节，根目录簇号，2。（虽然在 FAT32 文件系统下，根目录可以存放在数据区的任何位置，但是通常情况下还是起始于 2 号簇）

【19】0x30~0x31：2 个字节，FSINFO（文件系统信息扇区）扇区号，1。（上图的标注即用黄色条纹的标注有误，请读者注意）该扇区为操作系统提供关于空簇总数及下一可用簇的信息。

【20】0x32~0x33：2 个字节，备份引导扇区的位置，6。（上图的标注即用黄色条纹的标注有误，请读者注意）备份引导扇区总是位于文件系统的 6 号扇区。

【21】0x34~0x3F：12 个字节，未使用。

【22】0x40~0x40：1 个字节，BIOS INT 13H 设备号，0x80。（这个我也不知道什么意思☺）

【23】0x41~0x41：1 个字节，未用。

【24】0x42~0x42：1 个字节，扩展引导标志。0x29。

【25】0x43~0x46：1 个字节，卷序列号。通常为一个随机值。

【26】0x47~0x51：11 个字节，卷标（ASCII 码），如果建立文件系统的时候指定了卷标，会保存在此。笔者当时没有指定卷表，上图中的 YCY 是后来指定的。

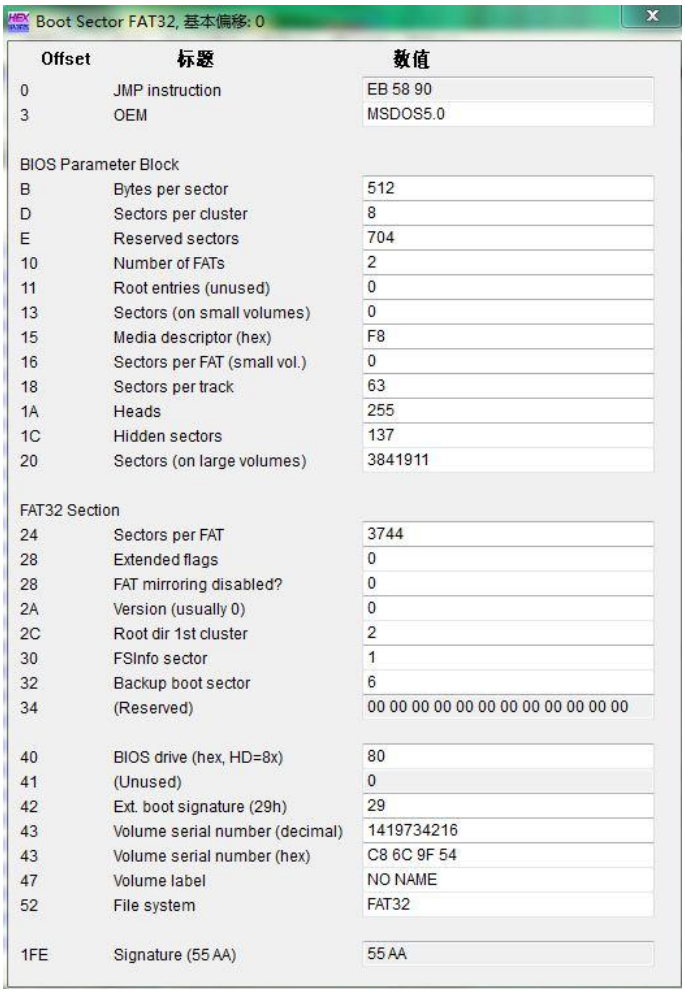
【27】0x52~0x59：8 个字节，文件系统格式的 ASCII 码，FAT32。

【28】0x5A~0x1FD：410 个字节，未使用。该部分没有明确的用途。

【29】0x1FE~0x1FF：签名标志“55 AA”。

好了，一个 DBR 的所有参数都介绍完了，你是否有冲动也看看自己的 SD 卡的 DBR 呢？还等什么呢？赶快去看看吧！

其实 WinHex 中提供了 FAT32 引导扇区的模板，模板名为“Boot Sector FAT32”。如下图所示：



Boot Sector FAT32, 基本偏移: 0

Offset	标题	数值
0	JMP instruction	EB 58 90
3	OEM	MSDOS5.0
	BIOS Parameter Block	
B	Bytes per sector	512
D	Sectors per cluster	8
E	Reserved sectors	704
10	Number of FATs	2
11	Root entries (unused)	0
13	Sectors (on small volumes)	0
15	Media descriptor (hex)	F8
16	Sectors per FAT (small vol.)	0
18	Sectors per track	63
1A	Heads	255
1C	Hidden sectors	137
20	Sectors (on large volumes)	3841911
	FAT32 Section	
24	Sectors per FAT	3744
28	Extended flags	0
28	FAT mirroring disabled?	0
2A	Version (usually 0)	0
2C	Root dir 1st cluster	2
30	FSInfo sector	1
32	Backup boot sector	6
34	(Reserved)	00 00 00 00 00 00 00 00 00 00 00 00
40	BIOS drive (hex, HD=8x)	80
41	(Unused)	0
42	Ext. boot signature (29h)	29
43	Volume serial number (decimal)	1419734216
43	Volume serial number (hex)	C8 6C 9F 54
47	Volume label	NO NAME
52	File system	FAT32
1FE	Signature (55 AA)	55 AA

8.6.3.2 引导代码

FAT 文件系统将引导代码与文件形同数据结构融合在一起，而不像 Unix 文件系统那样各自存在，引导扇区的前三个字节为一个由机器代码构成的跳转指令，以使 CPU 越过跟在后面的配置数据跳转到配置数据后面的引导代码处。

FAT32 文件系统引导扇区的 512 字节中，90~509 字节为引导代码，而 FAT12/16 则是 62~509 字节为引导代码。同时，FAT32 还可以利用引导扇区后的山区空间存放附加的引导代码。

一个 FAT 卷即使不是可引导文件文件系统，也会存在引导代码。

8.6.3.3 FSINFO 信息扇区

FAT32 在保留区中增加了一个 FSINFO 扇区，用以记录文件系统中空闲簇的数量以及下一可用簇的簇号等信息，以供操作系统作为参考。

1. FSINFO 信息扇区结构

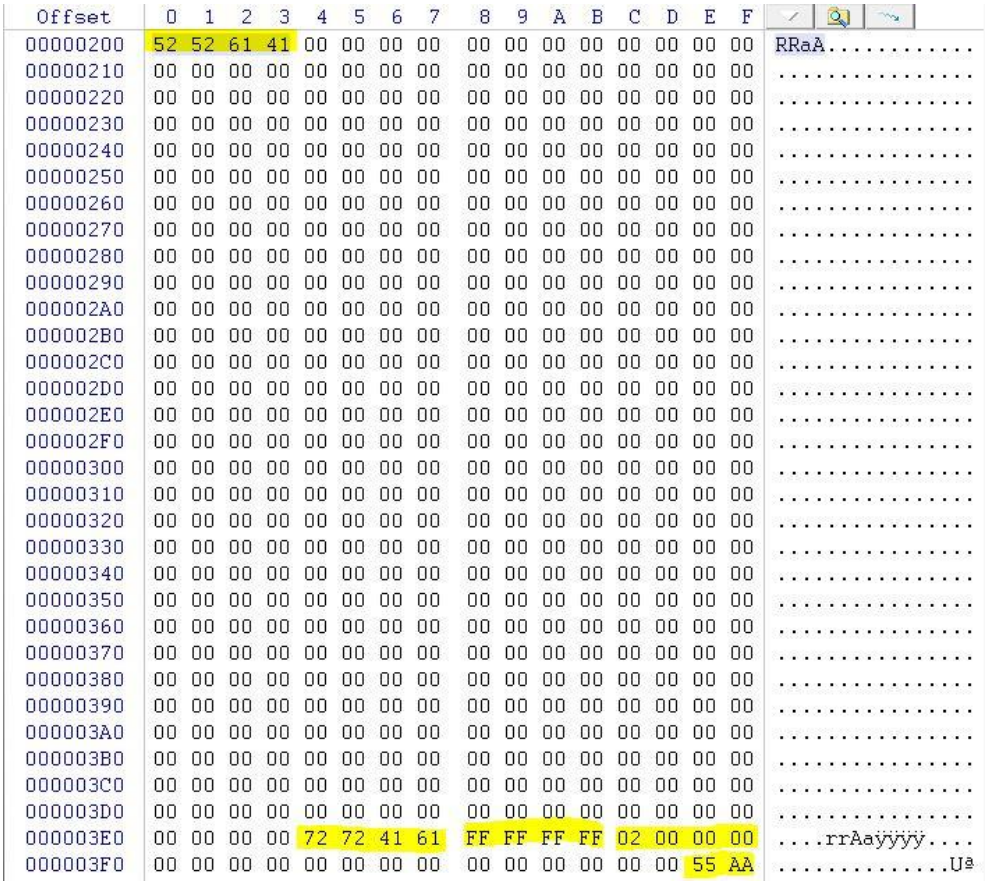
大多数的 FSINFO 信息扇区一般位于文件系统的 1 号扇区，结构非常简单。

字节偏移（十六进制）	字节数	含义
00~03	4	扩展引导标志“52526141”
04~1E3	480	未使用

1E4~1E7	4	FSINFO 签名“72724161”
1E8~1EB	4	空闲簇数
1EC~1EF	4	下一可用簇号
1F0~1FD	14	未使用
1EE~1EF	2	“55 AA”标志

2. FSINFO 信息扇区实例

首先附上一幅图如下：



Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00000200	52	52	61	41	00	00	00	00	00	00	00	00	00	00	00	00	RRaA............
00000210	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00000220	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00000230	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00000240	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00000250	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00000260	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00000270	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00000280	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00000290	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
000002A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
000002B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
000002C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
000002D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
000002E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
000002F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00000300	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00000310	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00000320	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00000330	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00000340	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00000350	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00000360	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00000370	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00000380	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00000390	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
000003A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
000003B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
000003C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
000003D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
000003E0	00	00	00	00	72	72	41	61	FF	FF	FF	FF	02	00	00	00	....rrAaÿÿÿÿ....
000003F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	55	AA	..............Uª

具体分析如下：

【1】0x00~0x03：4 个字节，扩展引导标志“52526141”。

【2】0x04~0x1E3：480 个字节，未使用，全部置 0。

【3】0x1E4~0x1E7： 4 个字节，FSINFO 签名“72724161”。

【4】0x1E8~0x1EB： 4 个字节，文件系统的空簇数，4294967295（0xFF FF FF FF）。

【5】0x1EC~0x1EF： 4 个字节，下一可用簇号，2（0x 00 00 00 02）。

说明：由于笔者的 SD 卡在收集这些数据时格式化过，里面没有东西，所以导致【4】【5】两项的数据为原始数据。下一可用簇号为 2，你知道文件系统数据区的起始扇区号吗？没错，正是 2 号簇，因为 0 号和 1 号 FAT 表项被系统写入特定的值，不用。详细讲解见下一小节。

【6】0x1F0~0x1FD： 14 个字节，未使用。

【7】0x1FE~0x1FF： 2 个字节，“55 AA”标志。

温馨提示：通常情况下，文件系统的 2 号扇区结尾也会被设置“55 AA”标志。6 号扇区也会有一个引导扇区的备份，相应的，7 号扇区应该是一个备份 FSINFO 信息扇区。8 号扇区可以看做是 2 号扇区的备份，

它的结尾也会有一个“55 AA”标志。

8.6.4 FAT32 的 FAT 表

8.6.4.1 FAT 表概述

位于保留区后的是 FAT 区，有两个完全相同的 FAT（File Allocation Table， 文件分配表）表组成，FAT 文件系统的名字也是因此而来。

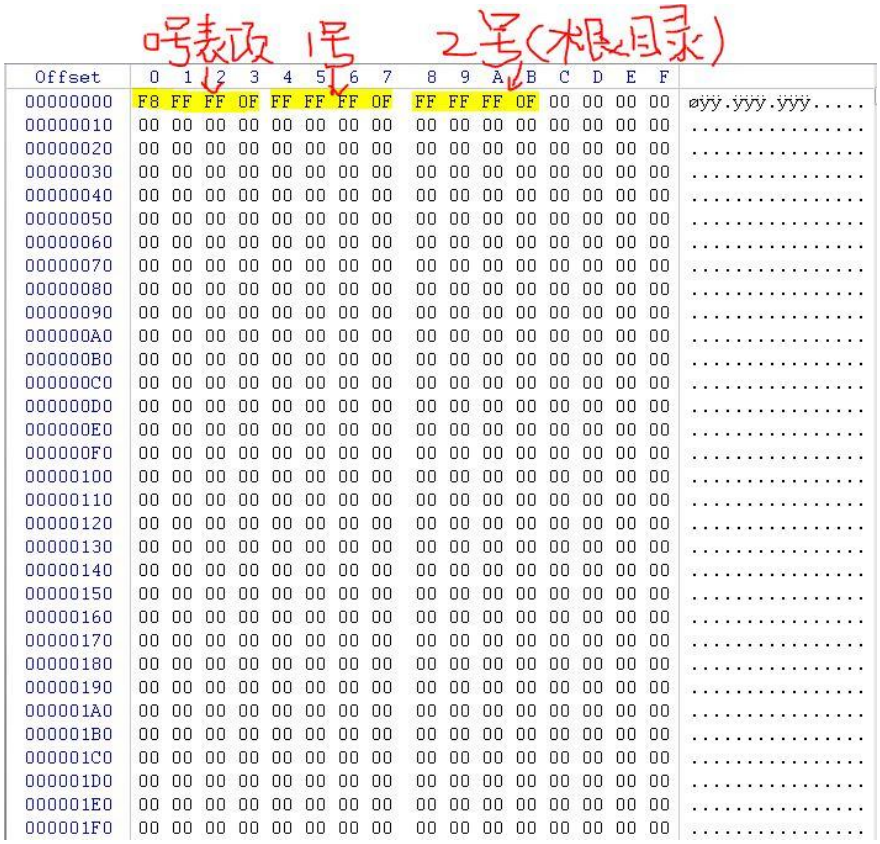
重要说明：

1. 对于文件系统来说，FAT 表有两个重要作用：描述簇的分配状态以及标明文件或目录的下一簇的簇号。
2. 通常情况下，一个 FAT 把文件系统会有两个 FAT 表，但有时也允许只有一个 FAT 表，FAT 表的具体个数记录在引导扇区的偏移 0x10 字节处。
3. 由于 FAT 区紧跟在文件系统保留区后，所以 FAT1 在文件系统中的位置可以通过引导记录中偏移 0x0E~0x0F 字节处的“保留扇区数”得到。
4. FAT2 紧跟在 FAT1 之后，它的位置可以通过 FAT1 的位置加上 FAT 表的大小扇区数计算出来。

8.6.4.2 FAT 表的特性

FAT 表由一系列大小相等的 FAT 表项组成，总的说来 FAT 表有如下特性：

1. FAT32 中每个簇的簇地址，是有 32bit（4 个字节）记录在 FAT 表中。FAT 表中的所有字节位置以 4 字节为单位进行划分，并对所有划分后的位置由 0 进行地址编号。0 号地址与 1 号地址被系统保留并存储特殊标志内容。从 2 号地址开始，每个地址对应于数据区的簇号，FAT 表中的地址编号与数据区中的簇号相同。我们称 FAT 表中的这些地址为 FAT 表项，FAT 表项中记录的值称为 FAT 表项值。
2. 当文件系统被创建，也就是进行格式化操作时，分配给 FAT 区域的空间将会被清空，在 FAT1 与 FAT2 的 0 号表项与 1 号表项写入特定值。由于创建文件系统的同时也会创建根目录，也就是为根目录分配了一个簇空间，通常为 2 号簇，所以 2 号簇所对应的 2 号 FAT 表项也会被写入一个结束标记。如下图所示：



Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00000000	F8	FF	FF	0F	FF	FF	FF	0F	FF	FF	FF	0F	00	00	00	00	øÿÿ.ÿÿÿ.ÿÿÿ.....
00000010	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000030	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000050	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000060	00	00	00	00	00	00	00	00	00	00	00	00	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	.....

3. 如果某个簇未被分配使用，它所对应的 FAT 表项内的 FAT 表项值即用 0 进行填充，表示该 FAT 表项所对应的簇未被分配。
4. 当某个簇已被分配使用时，则它对应的 FAT 表项内的 FAT 表项值也就是该文件的下一个存储位置的簇号。如果该文件结束于该簇，则在它的 FAT 表项中记录的是一个文件结束标记，对于 FAT32 而言，代表文件结束的 FAT 表项值为 0x0FFFFFFF。
5. 如果某个簇存在坏扇区，则整个簇会用 FAT 表项值 0xFFFFFF7 标记为坏簇，不再使用，这个坏簇标记就记录在它所对应的 FAT 表项中。
6. 由于簇号起始于 2 号，所以 FAT 表项的 0 号表项与 1 号表项不与任何簇对应。FAT32 的 0 号表项值总是“F8FFFF0F”。如上图所示。
7. 1 号表项可能被用于记录脏标志，以说明文件系统没有被正常卸载或者磁盘表面存在错误。不过这个值并不重要。正常情况下 1 号表项的值为“FFFFFFFF”或“FFFFFF0F”。
8. 在文件系统中新建文件时，如果新建的文件只占用一个簇，为其分配的簇对应的 FAT 表项将会写入结束标记。如果新建的文件不只占用一个簇，则在其所占用的每个簇对应的 FAT 表项中写入为其分配的下一簇的簇号，在最后一个簇对应的 FAT 表象中写入结束标记。
9. 新建目录时，只为其分配一个簇的空间，对应的 FAT 表项中写入结束标记。当目录增大超出一个簇的大小时，将会在空闲空间中继续为其分配一个簇，并在 FAT 表中为其建立 FAT 表链以描述它所占用的簇情况。
10. 对文件或目录进行操作时，他们所对应的 FAT 表项将会被清空，设置为 0 以表示其所对应的簇处于未分配状态。

8.6.4.3 FAT 表的使用

一个文件的起始簇号记录在它的目录项中，该文件的其他簇则用一个簇链结构记录在 FAT 表中。如果

要寻找一个文件的下一簇，只需要查看该文件的目录项中描述的起始簇号所对应的 FAT 表项，如果该文件只有一个簇，则此处的值为一个结束标记；如果该文件不止一个簇，则此处的值是它的下一个簇的簇号。

下面我们用一个形象的比喻来说明使用 FAT 表寻找簇地址的过程：

我们的寝室楼都有各自的标号，而且每个房间都依次进行了编号，只是没有使用 0 和 1 两个号码，直接从 2 号开始。公寓一楼大厅设立了一个大柜子（FAT 表），大柜子被分成了很多小格子（FAT 表项），所有的小格子（FAT 表项）从 0 开始编号，0 号和 1 号被楼下阿姨保留做特殊之用。从 2 号开始分配给每个房间（簇）作为信箱，信箱（FAT 表项）号与房间（簇）号一一对应，所有的房间（簇）都得到自己的信箱（FAT 表项）后，剩余的小格子就闲置不用。

现在呢，我们要来做一个寻宝的游戏：一个宝物被分成了几份放在了几个房间（簇）内，每个房间（簇）放置一份。然后在每个房间对应的信箱中（FAT 表项）记录了下一份宝物所在的房间（簇）号，现在我们就去把这件宝物找出来。

第一步：首先公寓管理处（目录项）告诉我们一条线索，宝物的第一份所在的房间（簇）号。假设这个房间（簇）号是 5，我们从 5 号房间（簇）取出第一份，然后去 5 号信箱（FAT 表项）查看一下下一份放在哪个房间（簇）。

第二步：5 号信箱（FAT 表项）内记录的数字是 6，我们就去 6 号房间（簇）取出第二份，然后去 6 号信箱（FAT 表项）查看。

第三步：6 号信箱（FAT 表项）内记录的数字是 7，我们就去 7 号房间（簇）取出第三份，然后去 7 号信箱（FAT 表项）查看。

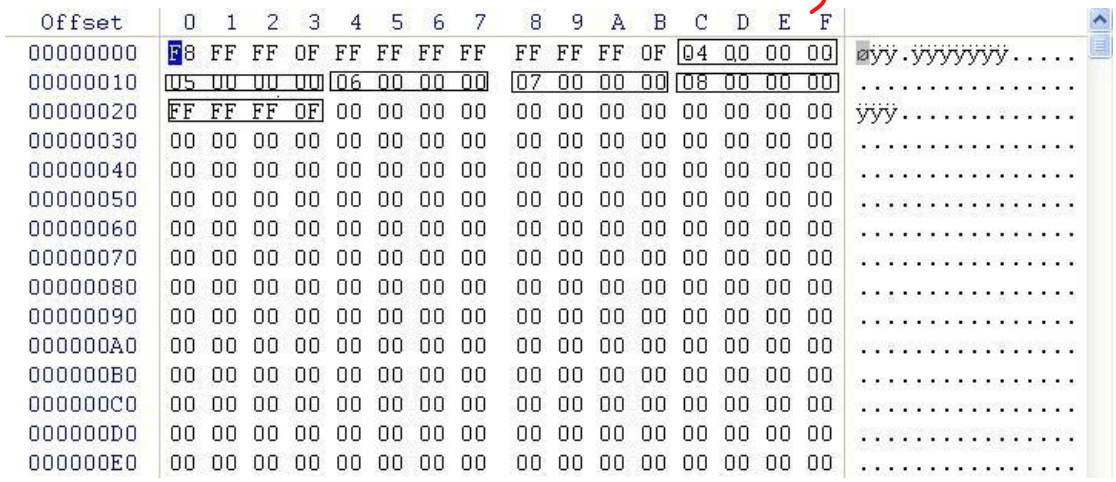
第四步：7 号信箱（FAT 表项）内记录的数字是 8，我们就去 8 号房间（簇）取出第四份，然后去 8 号信箱（FAT 表项）查看。

第五步；8 号信箱（FAT 表项）内记录的数字是 9，我们就去 9 号房间（簇）取出第三份，然后去 9 号信箱（FAT 表项）查看。

第六步：这时发现 9 号信箱（FAT 表项）内的内容是一个结束标记，也就是说后面没有了。这时我们就把所有的宝物找出来了。

下面我们尝试分析一个真实的例子。首先在 SD 卡的根目录下建立一个名为 ycy.txt 的文本文件，如下图所示：

FAT1 图示如下：



Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00000000	F8	FF	FF	0F	FF	FF	FF	FF	FF	FF	FF	0F	04	00	00	00	øÿÿ.ÿÿÿÿÿÿÿÿ....
00000010	05	00	00	00	06	00	00	00	07	00	00	00	08	00	00	00	....
00000020	FF	FF	FF	0F	00	00	00	00	00	00	00	00	00	00	00	00	ÿÿÿ.............
00000030	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00000040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00000050	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00000060	00	00	00	00	00	00	00	00	00	00	00	00	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	....

我们现在来尝试读取起始于 3 号簇的文件：

第一步：由该文件的目录项中得知它的第一簇存储在 3 号簇，到 3 号簇读取它的内容后，查看 3 号 FAT 表项。

第二步：3 号表项内的表项值为 4，即存储文件的下一个簇为 4 号簇，读取 4 号簇中的内容，查看 4 号簇对应的 4 号 FAT 表项。

第三步：4 号表项内的表项值为 5，即存储文件的下一个簇为 5 号簇，读取 5 号簇中的内容，查看 5 号簇对应的 5 号 FAT 表项。

第四步：5 号表项内的表项值为 6，即存储文件的下一个簇为 6 号簇，读取 6 号簇中的内容，查看 6 号簇对应的 6 号 FAT 表项。

第五步：6 号表项内的表项值为 7，即存储文件的下一个簇为 7 号簇，读取 7 号簇中的内容，查看 7 号簇对应的 7 号 FAT 表项。

第六步：7 号表项内的表项值为 8，即存储文件的下一个簇为 8 号簇，读取 8 号簇中的内容，查看 8 号簇对应的 8 号 FAT 表项。

第七步：8 号表项内的表项值为 9，即存储文件的下一个簇为 9 号簇，读取 9 号簇中的内容，查看 9 号簇对应的 9 号 FAT 表项。

第八步：这时发现 9 号 FAT 表项中的值已是结束标志：0x0FFFFFFF，说明 9 号簇已经是最后一簇了。

8.6.4.4 其他说明

要找一个簇的 FAT 表项，只要用它的簇号乘以每个 FAT 表项的字节数即可。Winhex 提供了直接跳转到某个指定 FAT 表项的功能，单击 position|go to FAT Entry，即可弹出转到 FAT 项对话框，在对话框输入目标 FAT 项号码后单击 OK，光标即会在该 FAT 项的第一个字节上闪烁。

文件系统大小的上限取决于 FAT 项的大小。簇链中的每个 FAT 项记录着下一个簇的簇地址，FAT 项所能表示的数字有一个上限，这个上限值也就是文件系统中的最大簇号。FAT32 文件系统的 FAT 项只使用了

32bit 的 28bit，因此只能描述 268435456 个簇（实际上还要少于这个值，因为这其中还包括用作结束标记和坏簇标志的保留值）。

8.6.5 FAT32 的数据区

数据区时真正用于存放用户数据的区域。数据区紧跟在 FAT2 之后，被划分成一个个的簇。所有的簇从 2 开始进行编号。也就是说，2 号簇的起始位置就是数据区的起始位置。

8.6.5.1 根目录

虽然原则上 FAT32 允许根目录位于数据去的任何位置，但通常情况下它都位于 2 号簇。

1. 定位根目录

在 FAT 文件系统中，寻找第一簇（即 2 号簇）的位置也就是寻找数据区的开始位置，这并不是一件容易的事，因为它不是位于文件系统开始处，而是位于数据区。在数据区前面是保留区域和 FAT 区域，这两个区域都不使用 FAT 表进行管理。因此，数据区以前的区域只能使用扇区地址（逻辑卷地址），而无法使用簇地址。

×¢Òâ:
DBRÒ²Ëã±£ÁôÉÈ
ËùÒÔ²»ÓÃÔÙ¼Ó

要想定位一个 FAT32 文件系统的数据起始处，可以通过引导扇区的相关参数计算出来。1.从引导扇区的偏移 0x0E~0x0F 字节处得到保留扇区。2.从偏移 0x10 字节处得到 FAT 表的个数。3.从偏移 0x24~0x27 字节处得到每个 FAT 表的大小扇区数。4.利用如下公式计算：

保留扇区数 + 每个 FAT 表大小扇区数 × FAT 表个数 = 数据区起始扇区号

要想计算其他已知簇号的扇区号，还要由引导扇区的偏移 0x0D 字节处查找到每个簇大小扇区数，并使用如下公式计算：

某簇起始扇区号 = 保留扇区数 + 每个 FAT 表大小扇区数 × FAT 表个数 + （该簇簇号 - 2） × 每簇扇区数

2. 根目录分析

根目录在文件系统建立时即已被创建，其目的就是存储目录（也称文件夹）或文件的目录项。每个目录项的大小为 32 个字节。

文件系统刚被创建时，还没有存储任何数据时，根目录下没有任何内容，文件系统只是为根目录分配了一个簇的空间（通常为 2 号簇），将结束标记写入该簇对应的 FAT 表项，表示该簇已经被分配使用。这时候，为根目录分配的空间没有任何内容。但如果在创建文件系统的时候是定了卷标，则会在根目录下为其建立一个卷标目录项，该目录项占用根目录中的第一个目录项位置。

下图显示了刚刚创建的 FAT32 文件系统的根目录，该文件系统的卷标为“YCY”。

Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
4194304	59	43	59	20	20	20	20	20	20	20	20	08	00	00	00	00	YCY .....
4194320	00	00	00	00	00	00	C0	BD	0B	3B	00	00	00	00	00	00	......À½.;......
4194336	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
4194352	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
4194368	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
4194384	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....

这个属性字节 0x08 表示卷标

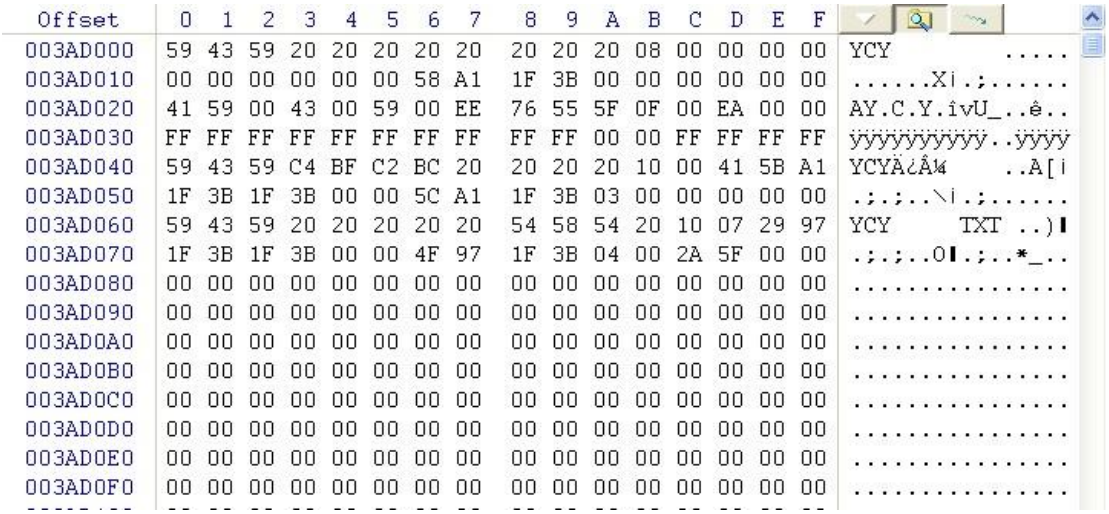
不管是根目录还是子目录下的目录项，都具有以下的基本特性：

1. 为文件或子目录分配的第一个簇的簇号记录在它的目录项中，其他后续簇则由 FAT 表中的 FAT 表链进行跟踪。
2. 目录项中除记录子目录或文件起始簇号外，还记录它的名字、大小（子目录没有大小）、时间值

等信息。

3. 每个子目录或文件除具有一个端文件目录项外，还会有长文件名目录项。
4. 短文件名目录项固定占用 32 字节，长文件名目录项则根据需要占用 1 个或者若干个 32 字节。
5. 对于同一个子目录或文件，它的长文件名目录项存放在它的短文件名目录项之前，如果长文件名目录项占用多个 32 字节，则按倒序存放于段文件名目录项之前。

下面我们在根目录下新建一个子目录“YCY 目录”和一个文本文件“YCY.txt”，看看根目录的内容如下：



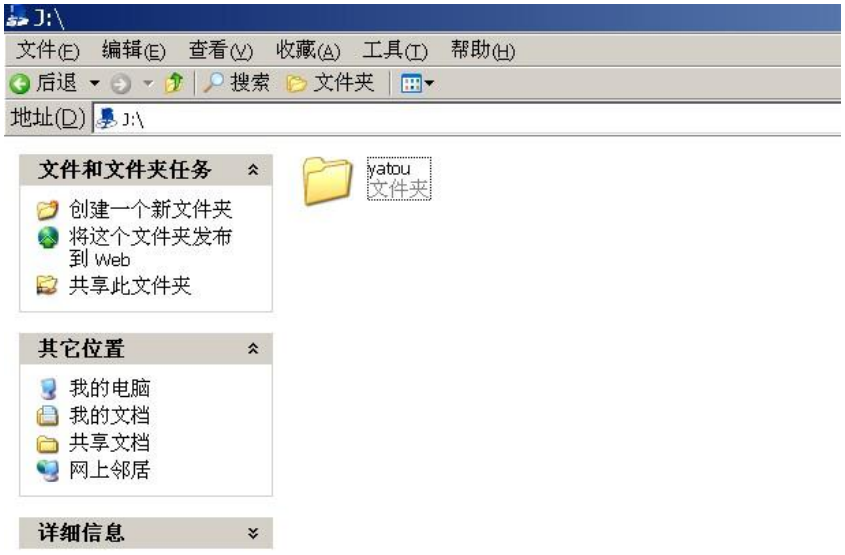
Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
003AD000	59	43	59	20	20	20	20	20	20	20	20	08	00	00	00	00
003AD010	00	00	00	00	00	00	58	A1	1F	3B	00	00	00	00	00	00
003AD020	41	59	00	43	00	59	00	EE	76	55	5F	0F	00	EA	00	00
003AD030	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	00	00	FF	FF	FF	FF
003AD040	59	43	59	C4	BF	C2	BC	20	20	20	20	10	00	41	5B	A1
003AD050	1F	3B	1F	3B	00	00	5C	A1	1F	3B	03	00	00	00	00	00
003AD060	59	43	59	20	20	20	20	20	54	58	54	20	10	07	29	97
003AD070	1F	3B	1F	3B	00	00	4F	97	1F	3B	04	00	2A	5F	00	00
003AD080	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
003AD090	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
003AD0A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
003AD0B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
003AD0C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
003AD0D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
003AD0E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
003AD0F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

8.6.5.2 子目录

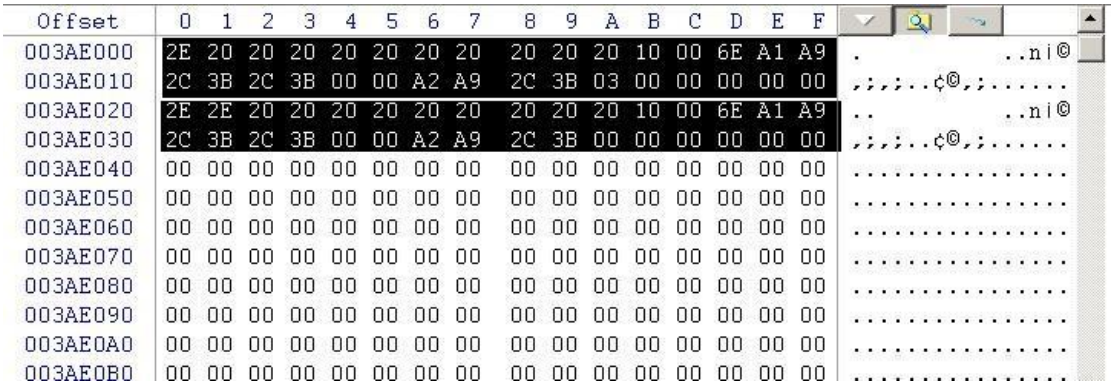
在 FAT32 文件系统中，除根目录在创建文件系统时即被建立并分配空间外，其他所有的子目录都是在使用过程中根据需要建立的。新建一个子目录时，在其父目录中为其建立目录项，在空闲空间中为其分配一个簇并对该簇进行清零操作，同时将这个簇号记录在它的目录项中。如果在根目录下创建一个子目录，我们就称这个子目录为根目录的子目录，称根目录为这个子目录的父目录。

创建子目录时，在为其父目录分配的簇中建立目录项，目录项中描述了这个目录的起始簇号。在为子目录建立目录项的同时，也在为子目录分配的簇中，使用前两个目录项描述它与父目录的关系。

下面我们在根目录下建立一个子目录 yatou。



我们来看看为其分配的簇中记录了那些内容：



Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
003AE000	2E	20	20	20	20	20	20	20	20	20	20	10	00	6E	A1	A9
003AE010	2C	3B	2C	3B	00	00	A2	A9	2C	3B	03	00	00	00	00	00
003AE020	2E	2E	20	20	20	20	20	20	20	20	20	10	00	6E	A1	A9
003AE030	2C	3B	2C	3B	00	00	A2	A9	2C	3B	00	00	00	00	00	00
003AE040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
003AE050	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
003AE060	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
003AE070	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
003AE080	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
003AE090	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
003AE0A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
003AE0B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

我们看到，虽然该子目录下并没有再建立任何下级子目录和文件，但却在其占用的簇空间中建立了两个目录项，第一个是“.”目录项，它描述该子目录本身的一些信息；第二个是“..”目录项，它描述的是该子目录的父目录（也就是根目录）的相关信息。通过这种方式，就在子目录和父目录之间建立起了联系。在子目录中建立下一级子目录或文件时，为下级子目录或文件建立的目录项将从第三个目录项开始写入。

温馨提示：子目录和根目录不同之处只在于根目录是在创建文件系统时就建立了的，如果没有卷标和内容，分配给根目录的簇空间内没有任何内容。

而子目录实在存储过程中根据需要由用户建立的，随着子目录建立的同时，会在为其分配的簇空间开始处建立两个目录项来描述子目录本身和其父目录的信息，以使字符目录间建立联系。

8.6.5.3 目录项

在 FAT32 文件系统中，根据结构不同可以讲目录项大致分为四种：卷标目录项、“.”目录项和“..”目录项、短文件名目录项、长文件名目录项。短文件名目录项是最重要的数据结构，其中存放着有关子目录或文件的短文件名、属性、起始簇号、时间值以及内容大小等信息。在 FAT32 文件系统中，将子目录看作是一种特殊的文件。

1. 短文件名目录项

短文件名目录项是 FAT 文件系统中非常重要的一种数据结构。之所以称其为短文件名目录项，是因为它所记录的文件名延续了 DOS 时代的 8.3 格式，即 8 个字符的名字加上 3 个字符的扩展名：

- 如果文件名不足 8 个字符，用 0x20 进行填充。
- 超过 8 个字符时则会被截短，因为短文件名目录项中没有足够的空间记录超出的部分。截短的方法是取文件名的前 6 个字符加上“~1”（如果有同名文件，则会依次递增该数值），然后加上其扩展名。
- 如果是子目录，则将扩展名部分用“0x20”进行填充。

为了解决长文件名的问题，FAT 文件系统又增加了一种“长文件名”目录项结构。其从 windows95 开始，不管文件名的长度是否超过 8 个字符，都会同时为其创建短文件名目录项和长文件名目录项，因为短文件名不区分大小写，而长文件名则是区分大小写的。

2. 端文件名目录项的特性

每个文件或子目录都分配有一个大小为 32 字节的目录项，用以描述文件或目录的属性。

- 所有的目录项并不是具有相同的地址，要找到一个目录项的位置只能用分配给文件或子目录的全名进行是搜索。
- 目录项结构中有一个属性区域，每个文件可以设置 7 中属性。
- 每个文件或目录还有四个非关键性属性：
  - 只读属性

- 隐藏属性
- 系统属性
- 存档属性

■ 每个目录项包括三个时间值，即建立时间、最后访问时间、最后修改时间：

- 建立时间，精确到十分之一
- 最后访问时间，精确到日
- 最后修改时间，精确到 2 秒

■ 一个目录项是否被分配使用它的第一个字节来描述。对于已经分配使用的目录项，它的第一个字节是文件名的第一个字符，而文件或目录被删除后，它所对应的目录项的第一个字节将被置为 0xE5，这就是为什么有的 FAT 数据恢复工具需要用户自己输入文件名的第一个字符的原因。

3．短文件名目录项的数据结构

每个短文件名目录项占 32 个字节，数据结构如下图所示：

偏移字节	字节数	含义
00~00	1	文件名的第一个ASCII码字符，在某些情况下座位该目录项的分配状态值：如果是0x00，表示该目录项未被使用过；如果是0xE5，说明该目录项曾经被使用过，但是现在已被删除
01~0A	10	文件名的第2至第11个ASCII码
0B~0B	1	文件属性：0x01-只读；0x02-隐藏；0x04-系统文件；0x08-卷标；0x0F-为此值时表示该目录项为长文件名目录项；0x10-目录；0x20-存档
0C~0C	1	保留未使用
0D~0D	1	建立时间（十分之一秒）
0E~0F	2	建立时间（时、分、秒）
10~11	2	建立日期
12~13	2	最后访问日期
14~15	2	文件内容起始簇号（4字节）的高两个字节（FAT12/16文件系统将此两个字节设置为0，因为他们只使用2个字节描述起始簇号，不使用高字节）
16~17	2	最后修改时间
18~19	2	最后修改日期
1A~1B	2	文件内容起始簇号的低两个字节
1C~1F	4	文件内容大小字节数（子目录不适用大小值，设置为0）

具体解释如下：

【1】0x00~0x00：1 个字节，如果该目录项正在使用中，则为文件名或子目录名的第一个字符。

■ 0x00：说明该目录项未被分配使用。

■ 0xE5：说明该目录项曾经被使用过，但是现在已被删除。目前处于未分配状态

【2】0x01~0x0A：10 个字节，文件名的第 2 至第 11 个 ASCII 码，除扩展名外，如果文件的名字不足 8 个字符则用 0x20 进行填充。

【3】0x0B~0x0B：1 个字节，所描述文件的属性

■ 0x01-只读

■ 0x02-隐藏

■ 0x04-系统文件

■ 0x08-卷标

■ 0x0F-为此值时表示该目录项为长文件名目录项

■ 0x10-目录

■ 0x20-存档

【4】0x0C~0x0C：1 字节，保留

【5】0x0D~0x0D：1 个字节，文件穿件的时间，精确到创建时刻的十分之一秒

【6】0x0E~0x0F：2 个字节，文件创建的时间——时分秒

两个字节的 16bit 被划分为 3 个部分：

- 0~4bit 为秒，以 2 秒为单位，有效值为 0~29，可以表示的时刻为 0~58
- 5~10bit 为分，有效值为 0~59
- 11~15bit 为时，有效值为 0~23

下面举例说明：

如上图所示，其子目录项偏移 0x0E~0x0F 字节处的内容为“A1A9”，我们来计算一下。由于 FAT 文件系统数据采用的小端存储方式，因此“A1A9”表示成 16 进制为 0xA9A1，换算成 2 进制就是 1010 1001 1010 0001，我们将其分成三部分并计算它的值，如下图所示：

	时	分	秒
bit位	15~11	10~5	4~0
二进制	10101	001101	00001
十进制	21	13	1
时刻	21时13分2秒		

下面我们再回过头来看看丫头这个目录的属性，点击右键选择属性一项可以看到：

可以看到上面的创建时间是 21：13：03，在误差允许的范围内。

【7】0x10~0x11：2 个字节，文件创立的日期，16bit 也划分为三个部分：

- 0~4bit 为日，有效值为 1~31
- 5~10bit 为月，有效值为 1~12
- 11~15bit 为时，有效值为 0~127，这是一个相对于 1980 年的年数值，也就是说该值加上 1980 即为文件创建的日期值。该部分笔者就不再举例就计算了，原理和计算创建时间是一样的。请读者自己去计算。

【8】0x12~0x13：2 个字节，最后访问日期。

【9】0x14~0x15：2 个字节，文件起始簇号的高两个字节。

【10】0x16~0x17：2 个字节，文件最后修改的时间。

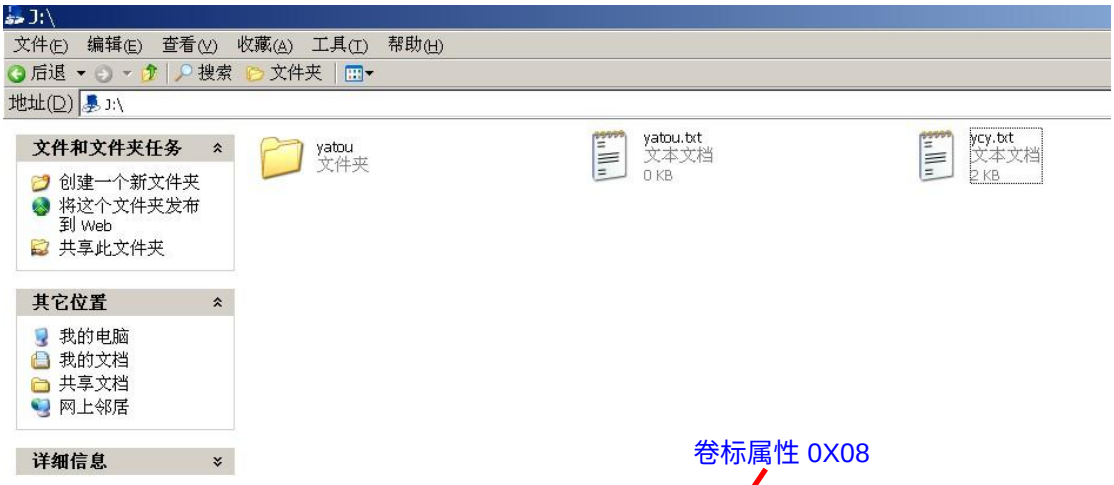
【11】0x18~0x19：2 个字节，文件最后被修改时的日期。

【12】0x1A~0x1B：文件内容起始簇号的低两个字节，与 0x14~0x15 字节处的高两个字节组成文件内容起始簇号。

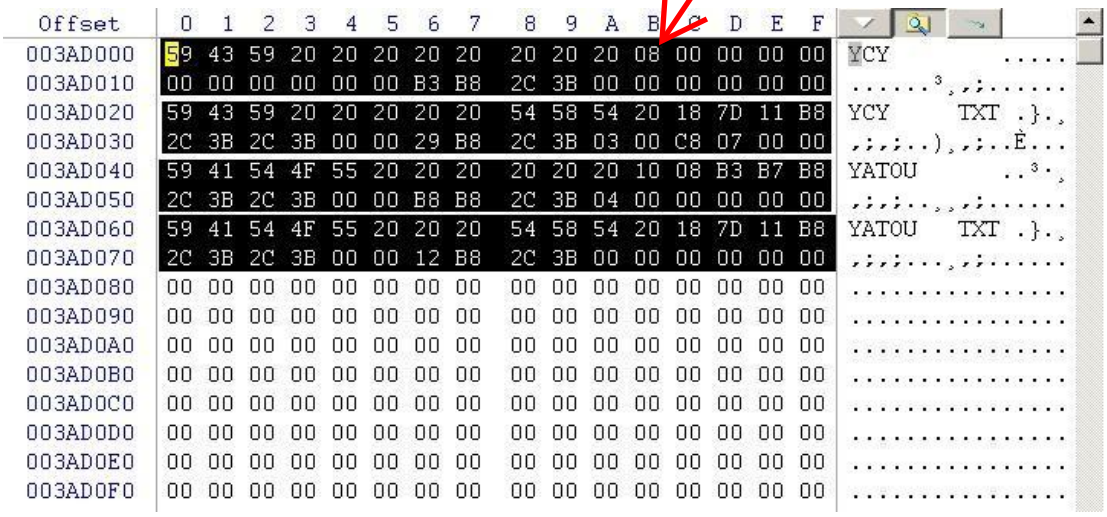
【13】0x1C~0x1F：文件内容大小字节数，只对文件有效，子目录的目录项此处全部设置为 0。

4.短文件名目录项实例分析

上面我们讲解了短文件名目录项的各参数及其含义，下面我们通过一个具体的实例来分析。首先我们在卷标为 ycy 的 SD 卡中再建立一个 yatou.txt 空的文本文件和一个写有数据的 ycy.txt 文本文件。即当前 SD 卡下只有三个文件：yatou（目录）、yatou.txt（空的文本文件）、ycy.txt（写有数据的文本文件）。如下图：



然后用 winhex 打开 ycy，如下图所示：



具体分析如下：

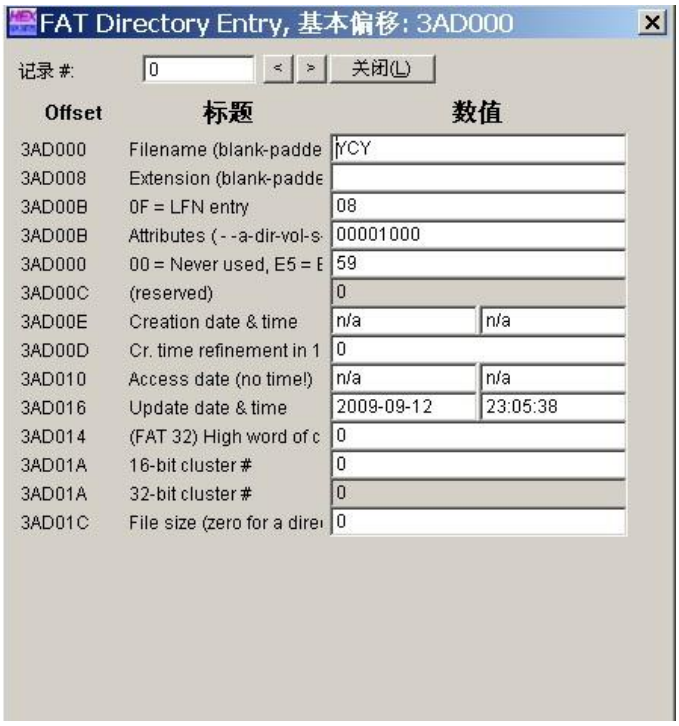
第一个目录项描述的是卷标信息。我们从第二个目录项开始分析，它是 ycy.txt 的目录项。

- 它的名字只有 3 个字符，因此共占用了偏移量 0x00~0x02，非分配给文件名部分的 0x03~0x0A 字节用 0x20 进行填充。这里需要说明的一点是：如果文件名有汉字，则一个汉字占用两个字节。
- 0x0B 字节处的属性值为 20，说明该文件时存档文件。
- 0x14~0x15 字节处的簇号高两位为 0x0000，0x1A~0x1B 字节处的簇号低两位为 0x0003，因此该文件起始于 3 号簇。
- 0x1C~0x1F 字节处描述的是文件的大小，0x07C8，即 1992 字节，即 1.94k，通过文件属性可知

该值是正确。

其他文件和目录的目录项笔者在这里就不再一一分析了，请读者自行研究。

提示：winhex 中提供了 FAT 短文件名的模板，模板名字为“FAT Directory Entry（Nomal/short entry format)”，如下图：



5.长文件名目录项

FAT32 文件系统在为文件分配短文件名目录项的同时会为其分配长文件名目录项。文件系统在为文件创建长文件名（Long File Name，LFN）类型的目录项时，并没有舍弃原有的短文件名目录项，具有 LFN 的文件同时也有一个常规的 SFN（Short File Name，短文件名）类型目录项。之所以仍然需要 SFN，是因为 LFN 目录项只包含文件的名字，而不包括任何有关时间、大小及起始簇号等信息，这些信息仍然需要用 SFN 目录项来记录。

6.长文件名目录项特性

如果一个文件的文件名超过了 8 个字符，则会为其名字截短后为其建立短文件名。将短文件名存储在短文件名目录项中。长文件名则存放在长文件名目录项中。长文件名目录项有以下的特性：

- LFN 和 SFN 目录项结构在相同位置有一个属性标志字节，LFN 目录项使用一个特定的属性值，以说明它是一个长文件名项。
- 项中的其他字节，使用 UTF-16 编码（UTF-16 是 Unicode 的其中一个使用方式。 UTF 是 Unicode/UCS Transformation Format，即把 Unicode 转做某种格式的意思），存储 13 个 Unicode 字符的文件名，每个字符占用两个字节。
- 如果文件名长于 13 个字符，则继续为其分配 LFN 项，知道够用为止。
- 所有 LFN 都包含一个校验和，通过这个校验和将其与相应的 SFN 项关联起来。
- 一个文件的所有 LFN 项按倒序排列在它的 SFN 项前面，即文件名的第一部分距离 SFN 是最近的。

7.长文件名目录项数据结构

偏移字节	字节数	含义
00~00	1	如果目录项使用中则为序列号，如果未分配过则为0x00；如果曾经使用过但已经被删除，则为0xE5
01~1A	10	文件名的第1~5个字符（Unicode），未使用部分先填充两个字节的“00”，然后用0xFF填充
0B~0B	1	长目录项属性标志0x0F
0C~0C	1	保留未使用
0D~0D	1	校验和
0E~19	12	文件名的第6~11个字符（Unicode），未使用部分先填充两个字节的“00”，然后用0xFF填充
1A~1B	2	保留
1C~1F	4	文件名的第12~13个字符（Unicode），未使用部分先填充两个字节的“00”，然后用0xFF填充

详细解释如下：

【1】0x00~0x00：1 个字节，长文件名目录项的序列号，一个文件的第一个长文件名序列号为 1，然后依次递增。如果是该文件的最后一个长文件名目录项，则将该目录项的序号与 0x40 进行“或（OR）运算”的结果写入该位置。如果该长文件名目录项对应的文件或子目录被删除，则将该字节设置成删除标志 0xE5。

【2】0x01~0x0A：5 个字节，长文件名的第 1~5 个字符。长文件名使用 Unicode 码，每个字符需要两个字节的空间。如果文件名结束但还有未使用的字节，则会在文件名后先填充两个字节的“00”，然后开始使用 0xFF 填充。

【3】0x0B~0x0B：1 个字节，长目录项的属性标志，一定是 0x0F。

【4】0x0C~0x0C：保留。

【5】0x0D~0x0D：1 个字节，校验和。如果一个文件的长文件名需要几个长文件名目录项进行存储，则这些长文件名目录项具有相同的校验和。

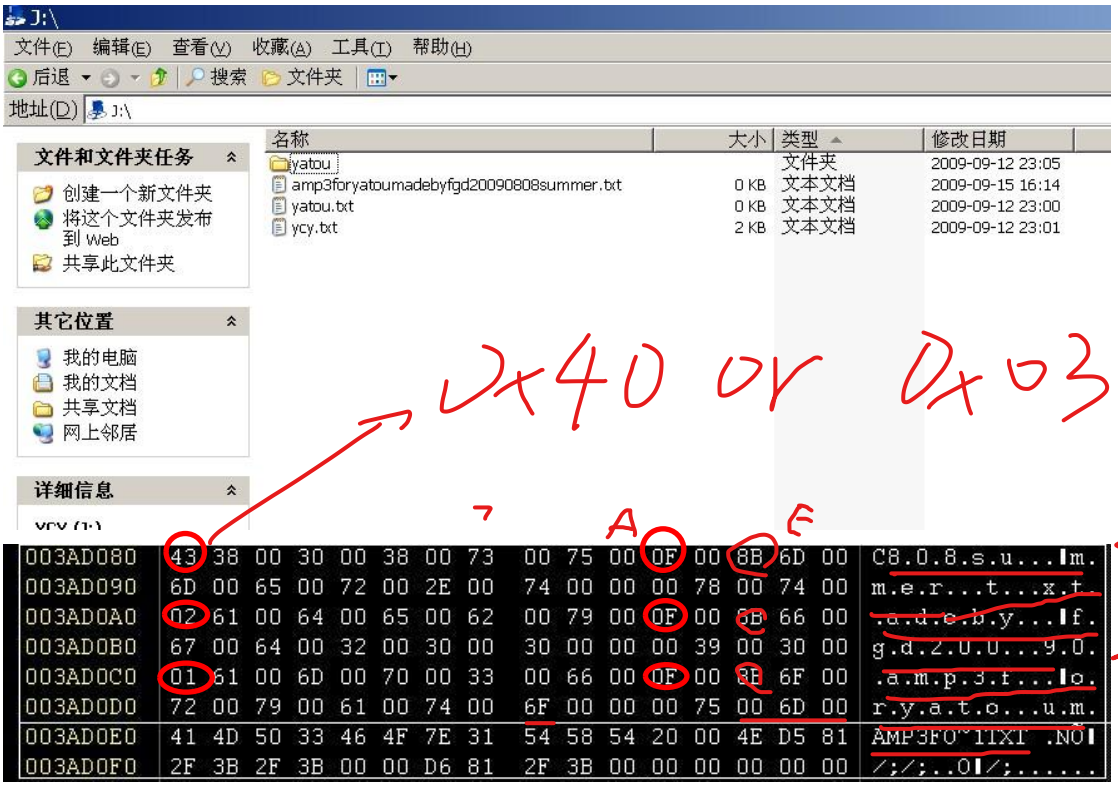
【6】0x0E~0x19：12 个字节，文件名的第 6~11 个字符，未使用的字节用 0xFF 填充。

【7】0x1A~0x1B：2 个字节，保留。

【8】0x1C~0x1F：4 个字节，文件名的第 12~13 个字符，未使用的字节用 0xFF 填充。

8. 长文件名目录项实例分析

首先我们在根目录下建立一个名字为“amp3foryatoumadebyfgd20090808summer.txt”的文件，然后用 winhex 来看看它的长文件名目录项，如下图：



短目录项问题紧跟在长目录项后面

上面看的第二幅图就是由“amp3foryatoumadebyfgd20090808summer.txt”文件的短文件名目录项和长文件名目录项组成。最下面的一个目录项就是该文件的短文件名目录项，我们可以看到，“amp3foryatoumadebyfgd20090808summer.txt”文件名被截断，取出前六个字符“AMP3FO”（注意短文件名是不分大小写的），然后加上“~1”，最后加上它的扩展名。由短文件名向上，是它的长文件名目录项。“amp3foryatoumadebyfgd20090808summer.txt”共有 39 个字符，每个长文件名目录项可以记录 13 个字符，因此需要 3 个长文件名目录项。短文件名目录项向上第一个小方框卷定的目录项是它的第一个长文件名目录项，向上依次为 2、3 号，从每个长文件名目录项的第一个字节可以看出他们的序号。

【1】我们先来看第一个内容：

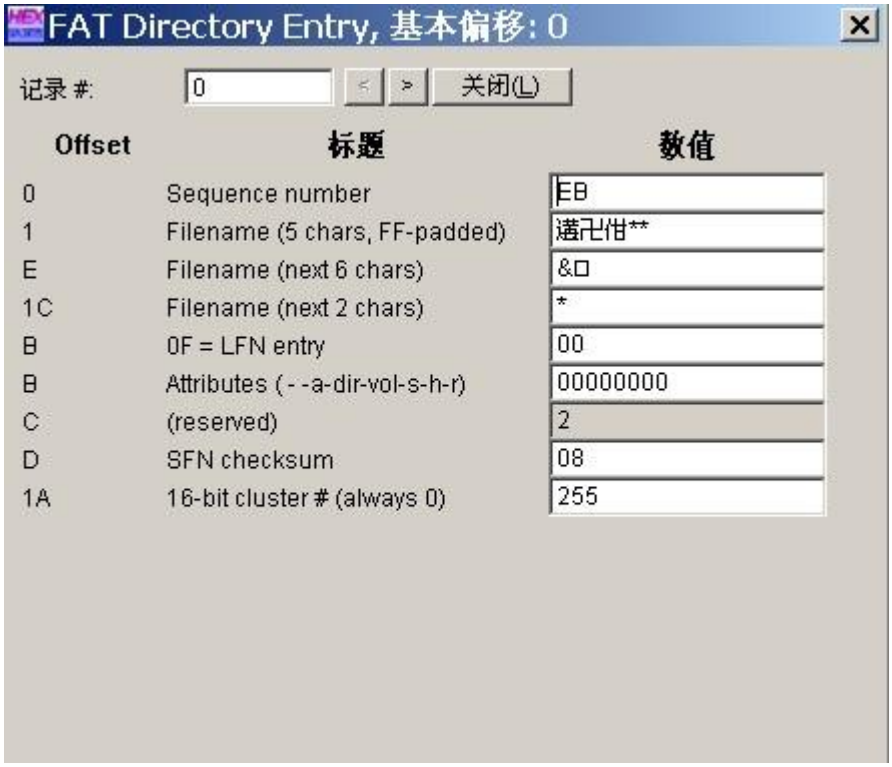
- 0x0B 字节处的“0F”表示这是一个长文件名目录项。
- 0x00 字节处的“01”表示这是该文件的第一个长文件名目录项。
- 0x01~0x0A 字节处的 10 个字节为文件名的第 1~5 个字符“amp3f”。0x0E~0x19 子接触的 12 个字节是文件名的第 6~11 个字符“oryato”，0x1C~0x1F 处的 4 个字节是文件名的第 12~13 个字符“um”。

【2】第二个长文件名目录项的第一个字节“02”表示这是该文件的第二个长文件名目录项，各部分字节含义由读者自行分析。

【3】第三个长文件名目录项的第一个字节为“43”，是 0x40 和 0x03 进行或运算的结果。说明这是该文件的第 3 个长文件名目录项，也是最后一个。

【4】 我们分别看 3 个长文件名目录项 0x0D 字节处的值——0x8B，这是长文件名目录项的校验和。说明这 3 个目录项同属一个长文件名目录项。

【5】 Winhex 也提供了 FAT 长文件名目录项的模板，如下图：



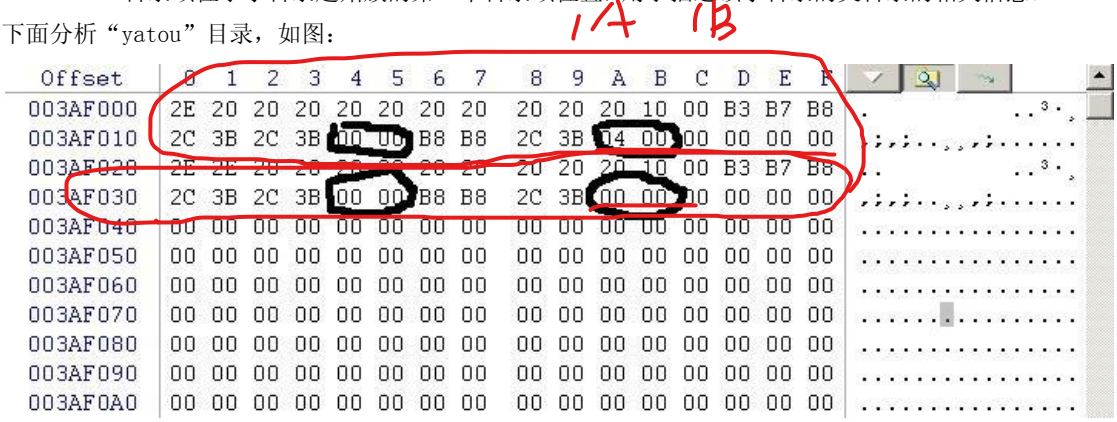
（注：以上这张图片似乎有点问题，可能与笔者使用的软件版本有关）

9. “.”目录项和“..”目录项

前面曾经介绍过，一个子目录的起始簇，前两个目录为“.”目录项和“..”目录项，子目录通过这两个目录项及它在父目录中的目录项建立起父子目录的联系。

- “.”目录项位于子目录起始簇的第一个目录项位置，它用以表明该簇是一个子目录的起始簇。另外，该目录项实际上是对目录自身的描述，它记录了该子目录时间信息、起始簇号等。需要注意的是，它所记录的起始簇号也就是该子目录目前所处的位置。
- “..”目录项位于子目录起始簇的第二个目录项位置，用于描述该子目录的父目录的相关信息。

下面分析“yatou”目录，如图：



我们重点看两个目录项的簇号描述部分（被标记部分）

- 第一个目录项也就是“.”目录项，记录的簇号为 4 号簇，也正是本子目录所在的簇。
- 第二个目录项记录的簇号为这个子目录的父目录的起始簇号，如果父目录是根目录，则簇号位置全部设置为 0。

10. 卷目录项

如果创建文件系统时制定了卷标，则会在根目录下第一个目录项项的位置建立一个卷标目录项：

- 卷标名最多允许占用长度为 11 个字节，也就是为短文件名分配的 11 个文件名区域，如果卷标

名不足 11 个字节，则用 0x20 填充。（由于每个汉字占用 2 个字节空间，而卷标最多允许 11 个字节，所以用汉字命名卷标时，卷标的长度不能超过 5 个汉字）。

- 卷标目录项结构与普通短文件名目录项结构完全相同，但没有创建时间和访问时间，只有一个最后修改时间。
- 另外，卷标目录项也没有歧视簇号和大小值，这些字节位置全部这只为 0，0x0B 字节处的属性值为 0x08.

11. 目录项中时间值的更新

目录项中有三个之间值：最后访问时间、最后写入时间和建立时间。

【1】 建立时间。当 windows 为一个“新”文件分配目录项时设置建立时间。

【2】 最后写入时间。当 windows 向文件中写入新的内容时，最后写入时间会被更新。

【3】 最后访问日期。这个时间只精确到日期，会经常被更新。

写到这里，FAT32 文件系统的面纱已被我们窥见大半，剩下的功课就由读者自行去做了。关于 FAT12/16 文件笔者在文中就不再阐述了，其原理和 FAT32 基本相同，因为 FAT12/16 是 Microsoft 早期的文件系统版本，FAT32 是从 FAT12/16 发展而来，只要大家掌握了 FAT32，FAT12/16 文件系统就很容易了。

下面，我们来总结一下在 FAT 文件系统下建立和删除一个文件的步骤，来说明 FAT 文件系统的工作过程。

1. 建立文件

假设现在有一个子目录，它的名字是“smart monkey”，我们要在其下建立一个文件“yatou.txt”。使用的文件系统为 FAT32，簇大小为 4096 字节，我们要建立的文件大小为 5000 个字节。

步骤 1：读取位于卷 0 号扇区的引导扇区，根据引导扇区中的信息定位 FAT 表、数据区和根目录的位置。

步骤 2：遵照“smart monkey”的位置。查看根目录下的每个目录项，寻找名字为“smart monkey”且具有目录属性的目录项。找到后，查看它的起始簇号为 3。

步骤 3：读取 smart monkey 的起始簇（3 号簇）的内容，查找每个目录项，直到找到一个为分配的目录项。

步骤 4：找到可用项后写入文件名“yatou.txt”，并将文件大小和当前时间写入相应的位置。

步骤 5：为文件内容分配簇空间。转到 FAT 表，寻找空闲的位置。发现 4 号 FAT 表项未使用，这就说明 4 号簇是空闲的。将 4 号簇分配给文件，并在 4 号簇的 FAT 表项内写入结束标记。

步骤 6：将簇号 4 写入文件目录项的起始簇号区域。将文件的钱 4096 字节写入到 4 号簇中，还剩下 904 字节，所以还需要再为其分配一个簇。

步骤 7：在 FAT 表中继续寻找为分配簇，找到 5 号簇为空闲未使用（因其 FAT 表项为 0）。

步骤 8：将文件第一簇（即 4 号簇）的 FAT 表项值改写为 5，将文件的最后 904 字节写入 5 号簇。

步骤 9：在 5 号簇的 FAT 表项内写入结束标记。

2. 删除文件

现在我们将前面例子中建立的“smart monkey\yatou.txt”文件删除。

步骤 1：从卷 0 号扇区读取引导扇区，根据引导扇区中的信息定位 FAT 表、数据区和根目录的位置。

步骤 2：在根目录下寻找名字为“smart monkey”且具有目录属性的目录项。

步骤 3：由“smart monkey”的目录项中获取它的起始簇号为 3，到 3 号簇查看“smart monkey”的内容，从中找到文件“yatou.txt”的目录项，提取出它的起始簇号，为 4 号簇。

步骤 4：到 FAT 表中找到该文件的簇链，确定他飞存储位置为 4 号簇和 5 号簇。

步骤 5：将 4 号簇和 5 号簇的 FAT 项设置为 0.

步骤 6：将文件“yatou.txt”的目录项的第一个字节改为 0xE5。

由于笔者水平有限，出错在所难免。

gdfgd0218@126.com