



## 《振南 znFAT--嵌入式 FAT32 文件系统设计与实现》一书 【上下册】已正式出版发行

全国各渠道全面发售

（在当当、京东、亚马逊、淘宝等网络平台上搜索  
关键字"**znFAT**"即可购买，各地实体书店也有售）

此书是市面上 唯一 一套详细全面而深入讲解嵌入式存储技术、**FAT32** 文件系统、**SD** 卡驱动与应用方面的专著。全套书一共 **25** 章，近 **70** 万字。从基础、提高、实践、剖析、创新、应用等很多方面进行阐述，力求通俗，振南用十年磨一剑的精神编著此书，希望对广大工程师与爱好者产生参考与积极意义。

此书在各大电子技术论坛均有**长期的「抢楼送书活动」**，如 211C、elecfans 等等。

振南的**【ZN-X 开发板】**是市面上唯一全模块化、多元化的开发板，可支持 **51、AVR、STM32 (M0/M3/M4)**

详情请关注 [www.znmcu.cn](http://www.znmcu.cn) （振南个人主页!!）

# 第 4 章

## 自建模型,会意由衷:让我们的思想与 FAT32 接轨

上一章使用 WinHex 找到了 SD 卡上的文件数据,并实现了简易 SD 卡音乐数码相框,但最后也留下了一个问题“文件系统是如何组织数据的?如果文件数据不连续该怎么办?”为了更好地回答这一问题,这一章撇开生搬硬套、条条框框的形式,让大家开动脑筋,自己亲自设计一种较为合理的文件管理方案。振南相信通过这种主动思考的方式会让读者对文件系统和数据管理的认识更为真切。在对方案不断改进的过程中,循序渐进地意识到问题所在,最终将会发现我们的思想已经与微软的 FAT32 文件系统不谋而合了。

### 4.1 文件管理模型

要设计一种文件管理模型,说白了就是要实现一种文件名与文件数据的映射,也就是给定一个文件名后就可以找到它所对应的数据,如图 4.1 所示。

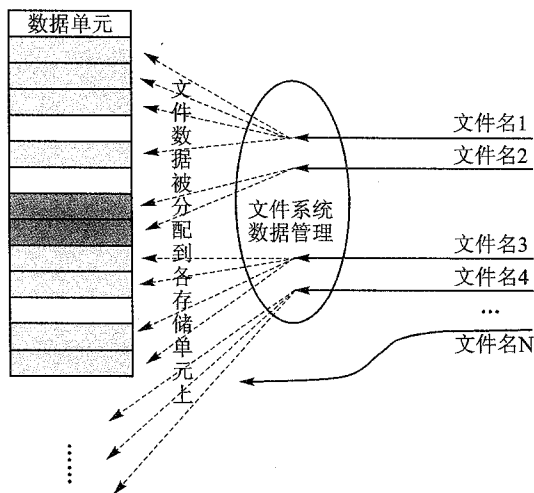


图 4.1 文件名向数据的映射



这看似简单,事实上并不简单,要考虑很多的问题,比如数据的不连续存储、存储空间的有效利用等。下面就来介绍几种振南设计的文件管理模型,看看是不是与读者的构思相似呢? 文件管理模型相应的视频教程可参考《振南带你学单片机视频教程》之《FAT32 文件系统专辑》。

### 4.1.1 原始模型

存储器是由很多顺序分布的扇区构成的,文件数据就存储在这些扇区上。而分布在若干个扇区上的具有某种关联性的数据可以成为一个数据集合,给这个集合取一个名字,这就是一种原始意义上的“文件”了。因此,就有了这样一种最原始的文件管理模型,如图 4.2 所示。

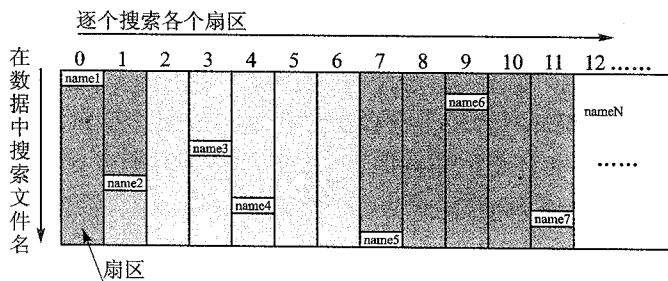


图 4.2 最原始的文件管理模型

从图中可以看到,文件名与数据是直接放在一起存储的,而且数据的存储是顺序的。这种模型的优点显而易见:简单明了,绝不浪费一丝空间,数据存储十分紧凑。但是缺点太多,最主要的就是文件搜索效率太低。比如要找到 name3,我们就要从 0 扇区开始,依次读取扇区数据,并在其中检测文件名,如此一直循环下去,直到发现 name3 为止。也许你会说:“这 name3 前面也没几个扇区,花不了多少时间就能找到。”其实不然,这只是一个例子,实际上它前面可能有成百上千、成千上万的扇区,这样效率就非常低了。最坏的情况就是搜索 nameN,即位于磁盘最末尾的文件,基本上就要遍历整个磁盘的所有扇区了。

### 4.1.2 改进模型

读者可能会想到这样一个方案:如果能把文件名都提到扇区的开始处,那不是可以省去在扇区数据里找文件名的步骤了吗?或许能将搜索文件的效率提高一点。没错,我们来看图 4.3。这样确实可以提高一些效率,但实际上是“换汤不换药”,效率低下的问题并没有从根本上解决。我们还是要着眼于如何摆脱逐个搜索扇区的困境。同时,我们看到上面的改进模型中已经开始出现存储空间的浪费了。其实这也体现了“时空平衡”的原则,要提高效率,则必然要牺牲空间。

其实,原始模型与改进模型都存在着另一个非常棘手的问题:它们的数据存储

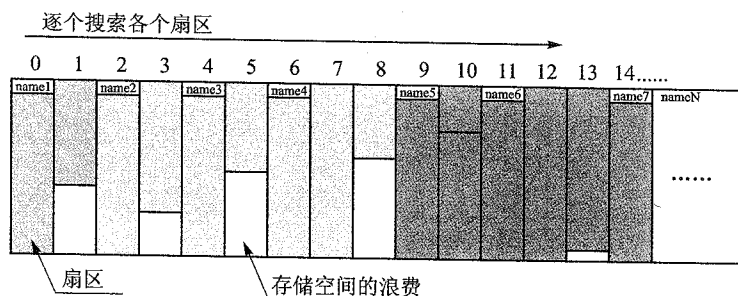


图 4.3 对原始模型的改进

都是顺序的,是不能跨扇区的。如果要进行删除文件的操作,比如把 name2 删除,那么就要把数据清空,于是这些存储空间就会空闲出来(即 2、3 扇区)。我们自然会想到,在写入新文件时,这部分存储空间能否被重新利用起来? 如果新文件的数据量比它小(如图 4.4 中的 nameS),那么它是可以放在这里的。但如果新文件数据量比较大(如图 4.4 中的 nameL),这块存储空间就放不下了。我们只能放弃它,再到后面寻找更大的空闲连续扇区。造成这一问题的根本原因在于:这种文件管理模型无法将数据分为若干部分来分散存储。就算把它强行分散到各个扇区中,我们也无法知道它们是如何分布的,因为各部分数据之间根本毫无关联。

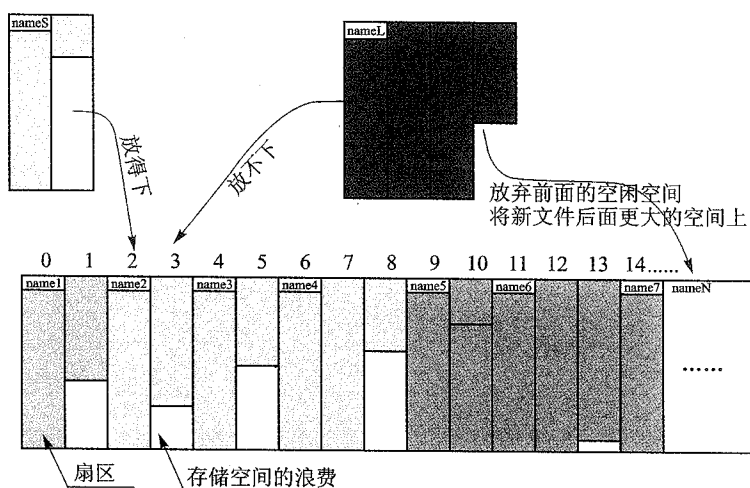



图 4.4 改进模型的棘手问题

要让数据可以不连续存储,又要能够知道各部分数据的具体存储位置,该如何实现呢? 对,就是链表。我们给每一个扇区的头部都设置一个专门的字段来记录扇区所属文件以及下一个扇区的地址,这样在读取某一扇区之后就可以知道记录后续数据的下一扇区在哪里了。它就如一条锁链,环环相扣,直到最后一个扇区。图 4.5 是一个更为形象的示意。



注:  表示扇区间的链接关系  
 表示最后一个扇区

只管到后面去找,最后别忘了把前后两部分空间链起来就可以了,如图 4.6 所示。



文件部分  
数据写入前  
面的空闲空间

[illegible]

形成了一个初步的文件管理的概念了,很快就可以揭开 FAT32 文件系统的“神秘面纱”了。

### 4.2.1 逼近模型

了链式存储的原理,其实这就是 FAT32 的核心思想,但这个模型最多只能算是 FAT32 的雏形,经过进一步改进而得到的逼近模型,与真正的 FAT32 就更加“神似”了。

我们的文件管理模型虽然经过了多次改进,但仍未摆脱遍历扇区的“厄运”,搜索目标文件的效率仍然非常低下。有什么办法可以快速得到文件的位置呢?索引!说白了,就如同书的目录一样,记录了章节的标题和具体位置。我们可以拿出一些扇区专门用来记录文件名及其对应数据的开始位置,这样一来,我们就不用遍历所有扇区,只需要对这块文件索引区进行搜索,就可以找到目标文件了。这样说还是比较抽象,请看图 4.7。

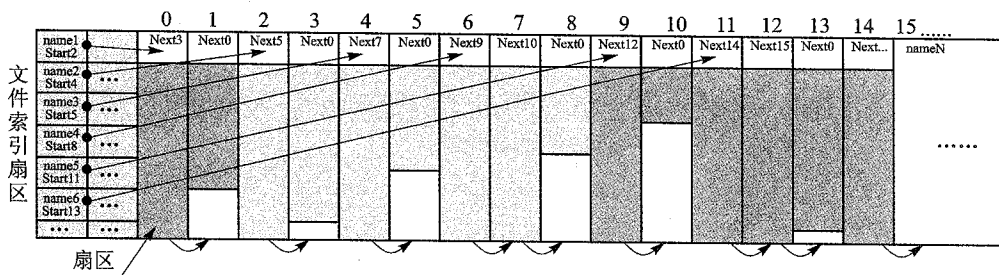


图 4.7 带文件索引的文件管理模型

图 4.7 中可以看到,我们用文件索引扇区记录了文件名与数据开始扇区的对应关系,但是用于描述扇区间链式关系的字段仍然位于各扇区的开始位置,这一点让人有些“不爽”:读/写数据的时候,还要把它跟数据分离开。我们何不将这些字段也提出来,专门以映射表的形式存放在一个地方,统一管理。这个映射表就犹如 FAT32 文件系统核心中的核心——FAT(文件分配表)。如图 4.8 所示,这才算是 FAT32 文件系统的逼近模型。

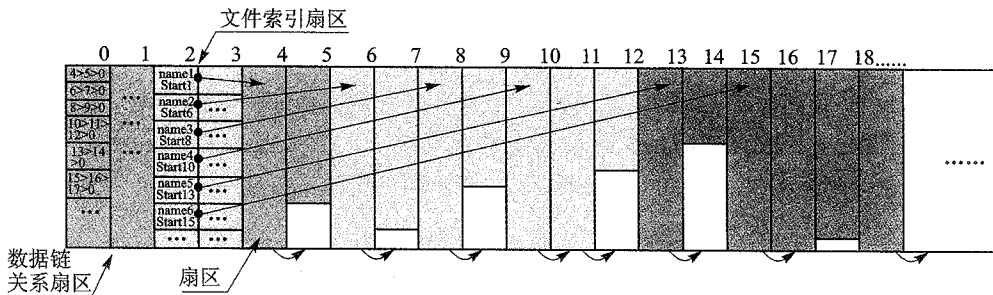


图 4.8 FAT32 文件系统的逼近模型

记得曾经有人问过这样的问题:“记录数据链式关系的映射表固然是一种很好的解决方法,但是磁盘扇区的数量是很大的,把它们之间的链式关系都放到映射表里,这表的体积未免也太大了吧?单单这个表就会占用较大的存储空间,另外数据的读取效率也是问题。”我们来举一个例子:如果一个磁盘的容量为 1 GB,则它的扇区数为 2 097 152,如果扇区地址使用 32 位整型来存储,那么一个扇区最多可以存储 128 个扇区地址,因此需要  $2\,097\,152/128=16\,384$  个扇区,即 8 MB 的存储空间,体积还



是很大的。如果一个文件的数据占用了  $N$  个扇区,那么就要到映射表中查找  $N$  次,而每查找一次,却仅仅可以读到一个扇区的数据(512 字节),这种“一查一读”的方式效率十分低下。这一过程被一些人称为“频转浅行”,意为一辆车频繁地转换行驶方向,但每次仅前行几米,时间都浪费在转向上了,实际上并没有走出多少路程。如何解决这些问题呢?磁盘的扇区数是由硬件物理结构决定的,不可能减少,而要压缩映射表的体积必然要减少地址数量,这可如何是好呢?扇区合并!将几个扇区合并为一个存储单元,以它作为最小单位,而不直接使用扇区。这样一来,地址数量减少了,映射表体积自然也就减小了。同时,每一次在映射表中查找后都可以读到  $n$  个扇区的数据,读取效率也上来了。这个新的存储单元,在后面我们就会知道它叫“簇”,同样是 FAT32 文件系统中非常重要而基础的概念。

有人又问了:“要是文件数据量比较小,存不满一个存储单元(其中包含若干个扇区),那可能会浪费更多的存储空间啊?”没错,如果在磁盘上存储一个只有一个字节的文件,那么浪费确实是太严重了。但对于大文件,那简直就是“效率的天堂”,仅仅查询几次映射表,数据就可以全部读出来了。所以,这正如我们上面所说的,这是由于空间与效率的矛盾而进行权衡的典型例子。

### 4.2.2 FAT32 的轮廓

有了前面的基础,这里简要介绍实际 FAT32 文件系统的整体结构与主要功能部分,这是我们后面继续深入研究的基础,如图 4.9 所示。

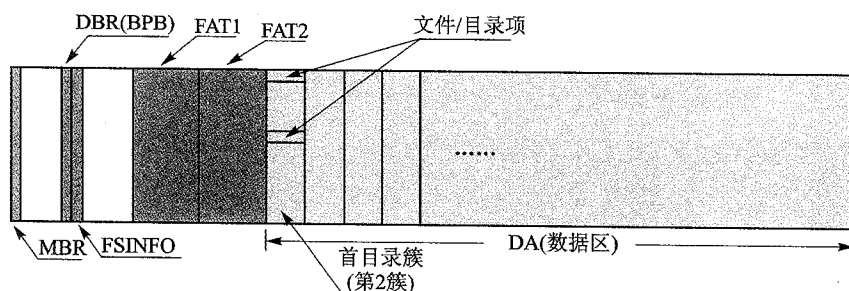


图 4.9 FAT32 文件系统的主要功能部分

MBR: MBR,主引导记录(Master Boot Record)。严格来说,MBR 并不属于 FAT32 文件系统,主要功能是记录各个分区的信息(分区信息由 MBR 中的 DPT 记录)。另外,MBR 也是造成上一章看到的物理 0 扇区与逻辑 0 扇区不相等的根本原因,后面会详细讲解。

DBR: DOS 引导记录(DOS Boot Record)。它对于 FAT32 来说,是极其重要的。DBR 中的 BPB(BIOS Parameter Block,即 BIOS 参数块)记录了很多非常关键的参数,如扇区和簇的大小、FAT 表的大小、根目录的位置等。如果 DBR 被意外破坏,则会直接导致 FAT32 文件系统的崩溃。当然,如果是有意而且合理地对它进行

篡改,则可能创造意想不到的效果。比如可以让一支只有几十 M 的小容量 U 盘摇身一变成为容量高达几 G、甚至上 T 的海量 U 盘,后面会介绍制作“假 U 盘”的具体方法。

FAT:可以把它读成英语 fat 的发音,但它可不是肥胖的意思,而是文件分配表(File Allocation Table)。它就犹如上面逼近模型中的映射表一样,记录了所有数据的链式关系。

首目录(根目录):在操作磁盘上的文件时,起初就是从根目录一层层进入的。不过在 FAT32 文件系统中,更准确的叫法是“首目录”,即第一个目录,是进入磁盘的第一道入口。首目录的扇区里存的是什么呢?还记得上面逼近模型里的文件索引吗?首目录里存的就是文件索引,还有子目录索引。

文件/目录项:振南称之为 FDI(File Directory Item,FAT32 中实际上并没有这个词),我们在计算机上看到一个一个的文件和目录根本上是源自于此。它通常是一个 32 字节的数据段,用来记录文件和目录的相关信息,如名称、大小、时间、数据开始位置等。文件/目录项其实就是上面所说的“索引”,可以通过它们找到数据的开始位置。

数据区:DA(Data Area),是 FAT32 文件系统的数据存储的主体部分。数据区中存储的自然是文件和目录的数据,本身是由很多存储单元(簇)构成的。

上面介绍的是 FAT32 的主要功能部分。当然,还有很多细微的内容,后面的讲解过程中会有所涉及。这些功能部分虽然是分开来介绍的,但实际上是相互关联的。来看看这种关联:从 MBR 的分区信息中可以获取到 DBR 的位置(DBR 所在的扇区就是分区的逻辑口扇区);而用 DBR 的 BPB 得到的参数可以计算得到 FAT、首目录等部分的具体位置和大小;FAT 又记录了数据区中数据之间的链式存储关系;而数据区中数据存储单元(簇)的大小又记录在 DBR 的 BPB 中;文件/目录项作为文件或目录的索引,给出了数据的开始位置,此开始位置也是 FAT 中一个链式关系的开始节点,并可由此找到位于数据区中的后续数据。FAT32 文件系统各功能部分环环紧扣,同时又是错综复杂的,但不要生畏,没有想得那么难,其实它的脉落是十分清晰的。

这一章我们真正引出了 FAT32 文件系统,后面就开始深入到各个部分进行详细研究和具体实现。随着我们在 FAT32 之路上越走越远,读者也会看到更多更加精彩的实验,获得更多的收获。好,继续前进……



