FAT 表中查出链接簇号，转换为逻辑扇区，定位磁头进行读写等操作的。

5.1.2.3 介质格式字节（Media Format Byte）

介质格式字节是 FAT 的头一个字节，所以习惯性地称为 FAT 表头（见图 5-3）。在早一些版本的 MS-DOS 时是用来标识卷的格式，但是现在已经被 BIOS 参数块所代替。介质格式字节仅在格式化磁盘的时候写入，以后基本对其忽略不计。

表项	长度	说明
1	1 字节	FF：每道 8 扇区的双面软盘 FE：每道 8 扇区的单面软盘 FD：每道 9 扇区的双面软盘 FC：每道 8 扇区的单面软盘 F8：硬盘
2	1 字节	恒为 FF
3	1 字节	恒为 FF

表 5-3　介质格式字节

5.1.3 簇

5.1.3.1 簇的概念

簇是磁盘上的最小存储分配单位，每一个簇包含一个或以上扇区，每簇包含的扇区数必须是 2 的整数幂，通常的取值范围是 1~128。

簇在 FAT 文件系统中有重要的作用：FAT 表实际上是一个以簇号为下标的一维数组，数组纪录的内容是下一簇的簇号。在 FAT 中采用簇号来管理磁盘有效的达到了以下两个目的：

- 规格化目录结构。由于有 FAT 专门记录文件的分配情况，在目录项中只用记录文件的首簇号，即使很大的文件也只需要一个目录项。
- 文件的易扩充性。可以在磁盘的任意位置存储文件的任意部分而不用考虑物理存放位置的连续性，文件的可存放长度只受磁盘容量的限制。

5.1.3.2 簇的查找

一个文件占用的存储空间链映像在 FAT 中。根据簇号与逻辑扇区的关系，可以使用 FAT 查找文件，这个过程对于 FAT16 来说相对简单：一个表项占两个字节，只要将簇号乘以 2，该乘积处一个字的内容就是下一个簇的簇号。下面以 FAT12 为例，在已知簇号的情况下如何得到下一个簇号。整个过程分为 5 个步骤：

1．将已知的十进制簇号乘以 1.5。

2．乘积的整数部分是到 FAT 的相对位移，在该位置存放着下一簇的簇号。

3．从该位移处取出一个字，低位在前，高位在后。

4．若已知的簇号位偶数，这保留这个字节的高 12 位，否则保留低 12 位。

5．若保留的 12 位是 FF8~FFF，则表示是最后一簇。

知道了簇号，就可以通过下面公式计算出逻辑扇区，进行扇区定位：

扇区号=每簇扇区数*（簇号-2）+数据区开始的扇区号

簇的查找涉及的另一个问题就是分配文件是的空闲 FAT 表项的查找，这是关系到文件系统效率的一个重要因素。文件系统采用两种方法保证空闲 FAT 表项查找的高效率：

- 保存 FAT 表的第一个空闲表项并随时更新。
- 采用“就近双项查找”算法。

5.1.3.3 簇的分配

簇的分配有两种情况：存储新文件和在旧文件中加入数据。

存储新文件时，系统顺序查找 FAT，跳过所有已分配的簇，找到第一个可用簇，把它作为文件的起始簇，然后继续查找第二个可用簇，如此直到找到满足新文件字节数的最后一个簇空间，在该簇中作“最后一簇”的标记。只有找到足够的空间时才进行分配。

在向已有文件中加入数据时，首先要测算新文件的长度，如果原有的存储空间足够，则就在原有空间中加入，否则就按前面的方法分配一个新簇。

5.1.3.4 簇的释放

响应删除文件的系统调用时，现在该文件的目录项的第一个字节中写入一个“E5”作为删除标志，然后沿 FAT 提供的簇链，将各个簇号的内容清 0，使之成为可重新分配的空闲项。

响应删除子目录的系统调用时，除了要先判断子目录是否为空以外，其余细节和删除文件相同。

5.1.3.5 簇的大小

簇作为存储单元分配的最小单位，它的大小关系到一个文件系统的空间分配效率和文件系统的执行效率。

簇一旦被分配，不管利用多少，所剩余的部分都无法在被其他文件使用，一个文件的最后一簇的利用率是随机的，没有被该文件使用的区域就浪费掉了。这些被浪费掉的空间称为碎片。利用概率计算，设最后一簇被浪费掉的空间为 50%，这样磁盘上一共被浪费掉的空间为：

文件总数*一个簇的空间大小*50%

显然减少簇的大小可以可以提高磁盘的利用率，但会增加 FAT 所占的空间，同时增加文件所占的簇链数也会增加，每读一个块都会有寻道延时和等待延时，因此，读许多有许多小块组成的文件会造成访问时间增加和文件系统效率降低。

假设磁盘每道有 32768 个字节，其旋转时间为 16.67ms，平均寻道时间为 30ms，以 ms 为单位的读取一个 k 字节块的所需时间是：

$$30+8.3+(k/32768)*16.67$$

图 5-4 的实线表示一个磁盘的数据读取速率与块大小之间的关系。如果粗略假设所有文件都是 1k 字节（Mullender&Tanenbaum,1984），则磁盘空间的利用率如图中虚线所示。从图中可以看到，时间效率和空间效率本质上相冲突。

在 FAT 文件系统中，有它的一些限制：

	FAT12	**FAT16**	**FAT32**
FAT 每项长度	12	16	28
最大簇数	**4086**	**65,526**	**268,435,456**
簇大小	0.5K~4k	2k~32k	4k~32k
最大分区大小	16M	2G	About10^7T

表 5-4 FAT 的块大小

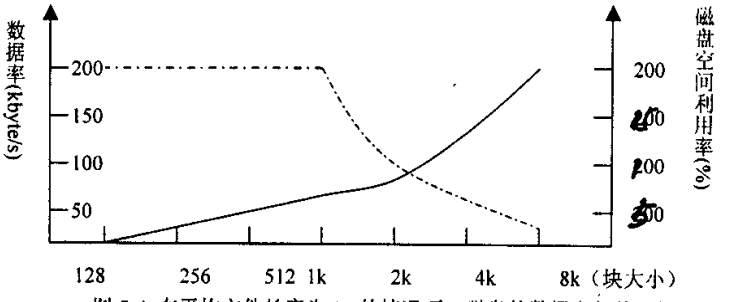

图 5-4 在平均文件长度为 1k 的情况下，磁盘的数据率与使用效率

所有这些给系统提出的要求是：根据磁盘的大小，自动寻找空间和效率的最佳结合点。我们的做法是：在不违反 FAT 的“最大簇数”规定的情况下，使得簇的大小尽量的小，以满足嵌入式系统的特殊需要。

5.2 文件系统调用和算法举例

5.2.1 文件系统调用

现在的文件系统版本提供了以下文件系统系统调用：

函数名称	函数原型	函数功能
dfs_open	int dfs_open(char *path, int flag)	打开一个已存在的文件
dfs_close	int dfs_close(int fd)	关闭一个已打开的文件
dfs_creat	int dfs_creat(char *path, int attrib)	创建一个新文件
dfs_delete	int dfs_delete(char * path)	删除指定路径下的文件
dfs_read	int dfs_read(int fd, void * buffer, unsigned int count)	函数从 fd（文件描述符）所指向的文件中读取长度为 count 的数据内容，并放到 buffer 所指向的目的缓冲区去
dfs_rename	int dfs_rename(char * path1,	修改文件名，将 path1 所在路径的文