放入尽可能多的项。当这个记录也用完后，剩下的行串放在MFT记录108中。这种方式可以用多个MFT记录去处理大的分段存储文件。

有可能会出现这样的问题：如果文件需要的MFT记录太多，以至于首个MTF记录中没有足够的空间去存放所有的索引。解决这个问题的方法是：使扩展的MFT记录列表成为非驻留的（即：存放在其他的硬盘区域而不是在首MFT记录中），这样它就能根据需要而增大。

图11-45表示一个MFT表项如何描述一个小目录。这个记录包含若干目录项，每一个目录项可以描述一个文件或目录。每个表项包含一个定长的结构体和紧随其后的不定长的文件名。定长结构体包含该文件对应的MFT表项的索引、文件名长度以及其他的属性和标志。在目录中查找一个目录项需要依次检查所有的文件名。

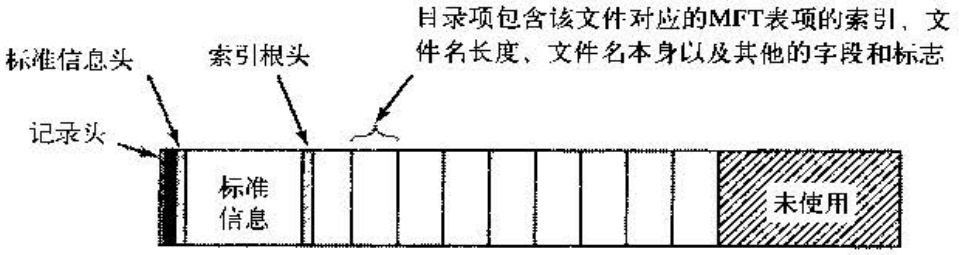

图11-45 描述小目录的MFT记录

大目录采用一种不同的格式，即用B+树而不是线性结构来列出文件。通过B+树可以按照字母顺序查找文件，并且更容易在目录的正确位置插入新的文件名。

现在有足够的信息去描述使用文件名对文件\??\C:\foo\bar的查找是如何进行的。从图11-22可以知道Win32、原生NT系统调用、对象和I/O管理器如何协作通过向C盘的NTFS设备栈（device stack）发送I/O请求打开一个文件。I/O请求要求NTFS为剩余的路径名\foo\bar填写一个文件对象。

NTFS从C盘根目录开始分析\foo\bar路径，C盘的块可以在MFT中的第五个表项中找到（参考图11-41）。然后在根目录中查找字符串“foo”，返回目录foo在MFT中的索引，接着再查找字符串“bar”，得到这个文件的MFT记录的引用。NTFS通过调用安全引用管理器来实施访问检查，如果所有的检查都通过了，NTFS从MFT记录中搜索得到::$DATA属性，即默认的数据流。

找到文件bar后，NTFS在I/O管理器返回的文件对象上设置指针指向它自己的元数据。元数据包括指向MFT记录的指针、压缩和范围锁、各种关于共享的细节等。大多数元数据包含在一些数据结构中，这些数据结构被所有引用这个文件的文件对象共享。有一些域是当前打开的文件特有的，比如当这个文件被关闭时是否需要删除。一旦文件成功打开，NTFS调用IoCompleteRequest，它通过把IPR沿I/O栈向上传递给I/O和对象管理器。最终，这个文件对象的句柄被放进当前进程的句柄表中，然后回到用户态。之后调用ReadFile时，应用程序能够提供句柄，该句柄表明C:\foo\bar文件对象应该包含在传递到C：设备栈给NTFS的读请求中。

除了支持普通文件和目录外，NTFS支持像UNIX那样的硬连接，也通过一个叫做重解析点的机制支持符号链接。NTFS支持把一个文件或者目录标记为一个重解析点，并将其和一块数据关联起来。当在文件名解析的过程中遇到这个文件或目录时，操作就会失败，这块数据被返回到对象管理器。对象管理器将这块数据解释为另一个路径名，然后更新需要解析的字符串，并重启I/O操作。这种机制用来支持符号链接和挂载文件系统，把文件搜索重定向到目录层次结构的另外一个部分甚至到另外一个不同的分区。

重解析点也用来为文件系统过滤器驱动程序而标记个别文件。在图11-22中显示了文件系统过滤器如何安装到I/O管理器和文件系统之间。I/O请求通过调用IoCompleteRequest来完成，其把控制权转交给在请求发起时设备栈上每个驱动程序插入到IRP中的完成例程。需要标记一个文件的驱动程序首先关联一个重解析标签，然后监控由于遇到重解析点而失败的打开文件操作的完成请求。通过用IRP传回的数据块，驱动程序可以判断出这是否是一个驱动程序自身关联到该文件的数据块。如果是，驱动程序将停止处理完成例程而接着处理原来的I/O请求。通常这将引发一个打开请求，但这时将有一个标志告诉NTFS忽略重解析点并同时打开文件。