Windows 常用文件格式详解

万小利　等编

北京理工大学出版社

WINDOWS 常用文件格式详解

万小利　孙　萍　编

北京理工大学出版社

内 容 简 介

本书在收集和整理大量微型计算机文件格式相关资料的基础上，精选并详细分析讲解 Windows 环境下操作系统和应用程序常用的 22 种数据文件格式。对于每一种数据文件中的内容均采用逐字节注释的方法进行说明，特别是不能用文本编辑软件显示其内容的神秘复杂的数据文件变得简单明了。作为软件开发人员的参考资料，本书含有许多读写这些数据文件的操作要点，并附有一些 C 语言程序示例，可供应用软件开发者参考使用。本书也可以作为初学者的入门教材。

图书在版编目(CIP)数据

Windows 常用文件格式详解/万小利，孙萍编. —北京：北京理工大学出版社，2001.3
ISBN 7-81045-793-4

Ⅰ. W… Ⅱ. ①万… ②孙… Ⅲ. 窗口软件，Windows-数据文件 Ⅳ. TP316.7

中国版本图书馆 CIP 数据核字(2001)第 08735 号

责任印制：刘京凤　责任校对：郑兴玉

北京理工大学出版社出版发行
(北京市海淀区中关村南大街 5 号)
邮政编码 100081　电话(010)68912824
各地新华书店经售
北京房山先锋印刷厂印刷
*
787 毫米×1092 毫米　16 开本　16 印张　387 千字
2001 年 3 月第 1 版　2001 年 3 月第 1 次印刷
印数：1—5000 册　定价：21.00 元

前　言

与目前飞速发展的信息时代相比较，早期的微型计算机只能处理一些简单的字符、数字等类型的信息。那时，微型计算机文件的格式除了.COM(可执行文件)、.BIN(二进制代码文件)和 .TXT(文本文件)等一些简单的文件格式外，其他文件格式可以说是屈指可数。

如今计算机技术以近似几何级数的增长速度飞速发展。微型计算机硬件从简单的 PC 机到 286、386、486、586、PENTIUM、PENTIUM MMX、PENTIUM Ⅱ、PENTIUM Ⅲ…；操作系统也从 DOS 到 WINDOWS3X、WINDOWS9X、WINDOWS2000…等等，应用软件层出不穷。需要处理的信息越来越多，也越来越复杂。除了字符、数字等类型的信息外，人们还要处理大量的图形图像、声音、视频等各种复杂信息。

微型计算机文件的格式如此之多，对微型计算机文件的兼容性和共享性形成了很大的障碍。虽然链接和嵌入技术使我们可以采用黑箱式方法对一些文件进行操作，但这些操作毕竟需要依赖第三方软件。况且，某些系统文件本身就没有提供原始的处理软件。

了解各种文件格式的标准，对于软件开发十分重要。本书在全力收集微型计算机文件格式相关资料的同时，对常用的计算机文件格式做了深入分析，以期能达到如下目的：

1. 为促进计算机集成技术的发展提供信息；
2. 为推动应用软件设计的标准化提供依据；
3. 促使更好、更全面的计算机文件格式不断涌现；
4. 对软件开发者提供帮助；
5. 引发初学者的兴趣。

这项工作十分艰巨。其一，虽然相对应用软件发展来说，文件格式的变化相对较慢。但随着计算机技术的迅猛发展，同类文件的格式毕竟会有所变化。同时，各种新的计算机文件格式仍在不断涌现，要想囊括所有不是一件易事。其二，由于软件商的技术保密，文件格式的技术与详细文档披露总是滞后。其三，计算机相关文献来源繁杂，并且大多散落各处；而且外文资料居多，给整理和分析工作带来难度。鉴于以上原因，本书主要涉及 WINDOWS 环境下的常用文件格式。作为冰山一角，仅仅作为初学者的入门教材和软件开发人员的参考资料。读者在确切了解现有文件的格式之前，应该谨慎，尤其是对操作系统相关文件。

编　者

目　　录

1　.ANI ____动态光标文件格式 …… 1
2　.AVI ____音频/视频文件格式 …… 5
3　.BMP ____图像文件格式 …… 18
4　.CLP ____剪贴板文件格式 …… 30
5　.COM ____ DOS 环境下的可执行文件格式 …… 32
6　.DAT ____注册文件格式 …… 33
7　.EMF ____ Win32 中的增强图元文件格式 …… 46
8　.EXE ____ DOS 环境下的可执行文件格式 …… 58
9　.EXE ____ WINDOWS 环境下的可执行文件格式 …… 64
10　.FON ____字体文件格式 …… 74
11　.GIF ____图形图像交换文件格式及压缩算法 …… 81
12　.ICO ____图标文件格式 …… 89
13　.INF ____安装信息文件格式 …… 105
14　.INI ____应用程序初始化文件格式 …… 108
15　.JPG ____静态图像文件格式 …… 110
16　.REG ____注册表项目文件格式 …… 126
17　.RTF ____多媒体文本文件格式 …… 130
18　.SCR ____屏幕保护程序文件格式 …… 135
19　.TTF ____字体文件格式 …… 136
20　.WAV ____音频文件格式 …… 140
21　.WMF ____图元图形文件格式 …… 152
22　.WRI ____书写器文件格式 …… 158

附录：WINDOWS 文件格式应用实例 动态光标文件编辑器 ANIEDIT …… 169

1 .ANI___动态光标文件格式

.ANI 动态光标文件是微软公司 1985 年推出的 WINDOWS 视窗操作系统下的一种光标文件。曾在很多动画和动态光标应用软件中使用，目前仍是广泛使用的一种动画和动态光标文件格式之一。大部分多媒体软件包都支持这种文件格式。以下是动态光标文件的格式说明。

这种动态光标文件本质上是不断重复的光标文件。它的文件参数头由一系列“关键字+域值+数据”的块组成所谓“域”。

“关键字”是域名，由 4 个字符字节组成，不同的“关键字”表明不同的域；

域值表明该域的属性，均占 4 个字节，属于 long 长整数型数字类型，某些域无值；

域的数据紧跟在域值后，某些域无数据。

下面是.ANI 动态光标文件格式中各种域的关键字、值、数据和它们的意义。

关键字：RIFF　　值：有　　数据：无

说　明：该域是文件长度域，其值表明该动态光标文件的总字节数。

关键字：ACON　　值：无　　数据：无

说　明：该域说明文件是动态光标文件，无值无数据。

关键字：LIST　　值：有　　数据：无

说　明：该域说明数据块，其值说明数据块所占字节数。通常后跟“FRAM”域或“INFO”域。说明该数据块的类型。

关键字：INFO　　值：无　　数据：无

说　明：该域说明 LIST 域的数据是文件信息数据块，无值无数据。

关键字：FRAM　　值：无　　数据：无

说　明：该域说明 LIST 数据块是动态光标数据块，无值无数据。

关键字：INAM　　值：有　　数据：有

说　明：该域说明标题，其值是标题数据所占的字节数，紧跟的数据是标题字符串。

关键字：IART　　值：有　　数据：有

说　明：该域说明作者名字，其值是作者名所占的字节数，紧跟的数据是作者名字符串。

关键字：ICON　　值：有　　数据：有

说　明：该域说明一个图标，其值是该图标所占的字节数，紧跟着的数据是图标数据。

关键字：ANIH　　值：有　　数据：有

说　明：该域说明动态光标参数块，其值是动态光标参数块字节数(一般是 36 字节)，紧跟着的数据是动态光标参数块，请参见本节后面的动态光标参数块说明。

关键字：RATE　　值：有　　数据：有

说　明：该域说明显示速度，其值是显示速度值所占字节数(一般是 4 字节)，紧跟着的数据

是显示速度值(1 到动态光标参数块中的 cSteps 参数值，其中 cSteps 参数可参见本节后面的动态光标参数块说明)。

关键字：SEQ␣　　值：有　　数据：　有

说　明：该域说明显示序数，其值是显示序数值所占字节数(一般是 4 字节)，紧跟着的数据是显示序数值(1 到动态光标参数块中的 cFrames 参数，其中 cFrames 参数可参见本节后面的动态光标参数块说明)。注意：关键字包括 SEQ 后面的一个空格。

任何域(比如“ACON”，“ANIH”，“RATE”域)可以任何顺序出现，但一般“RATE”域通常出现在“ANIH”域之后。因为“RATE”域需要从“ANIH”域中读取的数据。一般情况下，域出现的顺序是“RIFF”域、“ACON”域、“INAM”域、“IART”域、“ANIH”域、“RATE”域、“LIST”域、“INFO”域、“FRAM”域、“ICON”域。

在这些域中，“LIST”域可以重复。“INFO”域和“FRAM”通常紧跟在“LIST”域后，“ICON”域通常是重复的并紧跟在“LIST”域或“FRAM”后。“ICON”域是标准的.ICO 光标文件格式，可以参考本书的其他章节。

下面是 ANIH 域中数据所表示的动态光标参数块定义和说明：

```
struct aniheader{
        dword   cbsizeof;       // 动态光标参数块字节数
                                // (一般是 4 字节)。
        dword   cFrames;        // 此光标中定义的图标帧数。
        dword   cSteps;         // 在动态光标循环之前闪烁次数。
        dword   cx，cy;         // 保留，必须是零。
        dword   cBitCount，cPlanes;    // 保留，必须是零。
        dword   jifRate;        // “RATE”域没有提供时的显示速度
                                // 缺省值(每秒 1/60 次)。
        dword   flag;           // 动态光标格式标记(对于 WINDOWS，
                                // flag = 0x003d0011)。
    }
```

下面是读取.ANI 动态光标文件的一个例子：

```
int handle;                     // 文件句柄
unsigned char keyword[4];       // 关键字
unsigned long value;            // 关键字的值
unsigned long filesize;         // 文件字节数
unsigned char *ptitlename;      // 指向标题字符串
unsigned char *partname;        // 指向作者名字符串
unsigned char *prate;           // 指向显示速度字符串
struct aniheader *phead;        // 指向动态光标参数块

    .                           // 进行文件打开等操作，文件句柄在变量
    .                           // handle 中
    .
```

```
while (read(handle, &keyword, 4)>0)          // 读取关键字
{
    toupper(keyword);                        // 忽略字母的大小写
    if(keyword=="RIFF")                      // “RIFF”域
    {
        read(handle, &value, 4);             // 文件长度的字节数
        filesize=value;
    }
    else if(keyword=="INAM")                 // 标题域
    {
        read(handle, &value, 4);             // 标题字符串长度
        ptitlename=maloce(value);
        read(handle, ptitlename, value);     // 标题
                .                  // 标题处理程序
                .
                .
    }
    else if(keyword=="IART")                 // 作者名域
    {
        read(handle, &value, 4);             // 作者名长度
        partname=maloce(value);
        read(handle, partname, value);       // 作者名
                .                  // 作者名处理程序
                .
                .
    }
    else if(keyword=="ANIH")                 // 动态光标参数块域
    {
        read(handle, &value, 4);             // 动态光标参数块长度
        phead=maloce(value);
        read(handle, phead, value);          // 动态光标参数块
                .          // 动态光标参数处理程序
                .
                .
    }
    else if(keyword=="RATE")                 //显示速度域
    {
        read(handle, &value, 4);             //显示速度值所占字节数
        prate=maloce(value)
```

```
        read(handle，prate，value);              //显示速度值
                .          //显示速度参数转换及处理程序
                .
                .
    }
    else if(keyword=="ICON")                     //图标数据域
    {
        read(handle，&value，4);                 //图标数据所占字节数
        picon=maloce(value);
        read(handle，picon，value);              //图标数据(参见.ICO 文件格式)
                .          //图标数据处理程序
                .
                .
    }
}
    .
    .
    .
```

2 .AVI___音频/视频文件格式

.AVI 文件是一种 RIFF 规范，这种规范被应用于获得、编辑和回放音频/视频片段。总的来说，AVI 文件包含多种类型的数据流。大部分 AVI 片断不仅将使用音频而且还将使用视频数据流。对于一个 AVI 片段来说，一个微小的差异在于使用视频数据而不需要音频流。某些特殊的 AVI 片段还可能包含一个控制路径或者 MIDI 路径作为一种辅助的数据流。控制路径能够控制像 MCI 影蝶机这样的外部设备，MIDI 路径能够为此片断播放背景音乐。一个特殊的片段还需要一个特殊的控制程序，以便尽其所能。读和播放 AVI 片段的应用程序在一个特殊的片段中仍然能读和播放 AVI 片段（在这些特殊文件中的应用程序中均忽略掉了非 AVI 数据）。本节主要描述仅含有音频和视频数据的.AVI 文件。

本节包括以下主要内容：

① AVI 文件的必须部分

② AVI 文件的可选部分

③ 正在发展中的写 AVI 文件的方法

（1）AVI 文件的 RIFF 格式

AVI 文件使用 AVI-RIFF 格式。这种 AVI-RIFF 格式通过四字符代码“AVI”来识别。所有的 AVI 文件都包括两个必须的 LIST 块。这些块定义了流和流数据的格式。AVI 文件也可能包括一个索引块。可选块则具体指明了文件中数据块的位置。包含这些块的一个 AVI 文件有以下的格式：

```
RIFF ('AVI '
    LIST ('hdrl'
            .
            .
            .
        )
    LIST ('movi'
            .
            .
            .
        )
    ['idx1'<AVI Index>]
    )
```

LIST 块和索引块为 AVI-RIFF 格式块的第二部分，'AVI'块将此文件识别为一种 AVI-RIFF 格式文件。LIST 'hdrl' 块定义了数据的格式，是第一个必须的块。而 LIST 'movi'块包含了 AVI 片段的数据，是第二个必须的块。'idxl'块是可选的索引块。AVI 文件中的这三块必须保持正确的顺序。

LIST 'hdrl' 块和 LIST 'movi'块使用子块作为它们的数据。下面的例子说明了含有完成 LIST 'hdrl'和 LIST'movi'块所需要的扩展块的 AVI-RIFF 格式：

```
RIFF ('AVI '
        LIST ('hdrl'
                'avih'(<Main AVI Header>)
                LIST ('strl'
                        'strh'(<Stream header>)
                        'strf'(<Stream format>)
                        'strd'(additional header data)
                            .
                            .
                            .
                )
                .
                .
                .
    )
LIST ('movi'
                {SubChunk | LIST('rec '
                                    SubChunk1
                                    SubChunk2
                                        .
                                        .
                                        .
                                )
                    .
                    .
                    .
                }
            .
            .
            .
    )
    ['idx1'<AVIIndex>]
)
```

下面的段落说明包含于 LIST 'hdrl'和 LIST'movi'块中的这些块。"idx1"块也一样。

（2）AVI 文件的数据结构

用于 RIFF 块中的数据结构，已经在 AVIFMT.H 头文件中被定义了。本节末的参考部分说明可以被用于 AVI 文件参数头结构中的数据结构、AVI 索引和调色板块。

（3）AVI 头 LIST 块

文件以主文件参数头开始。在 AVI 文件中，参数文件头以四字节代码"avih"定义。参数文件头包含有关文件的主要信息，例如文件中流的数量和 AVI 序列的宽度和高度。主文件参数头由以下的数据结构定义：

```
typedef struct {
    DWORD   dwMicroSecPerFrame;
    DWORD   dwMaxBytesPerSec;
    DWORD   dwReserved1;
    DWORD   dwFlags;
    DWORD   dwTotalFrames;
    DWORD   dwInitialFrames;
    DWORD   dwStreams;
    DWORD   dwSuggestedBufferSize;
    DWORD   dwWidth;
    DWORD   dwHeight;
    DWORD   dwScale;
    DWORD   dwRate;
    DWORD   dwStart;
    DWORD   dwLength;
} MainAVIHeader;
```

dwMicroSecPerFrame 域明确说明了视频帧框架之间的时段。此值说明了全部的时间安排。

dwMaxBytesPerSec 域明确说明了文件大致最大的数据速率。此值说明了每秒钟系统必须处理的二进制数量，以提供 AVI 序列，该序列是由主文件参数头和流参数头中的其它参数说明的。

dwFlags 域为文件提供标志符，以下标识符被定义：

AVIF_HASINDEX　表示 AVI 文件包含'idxl'块。

AVIF_MUSTUSEINDEX　表示必须用索引来决定数据显示序列。

AVIF_ISINTERLEAVED　表示 AVI 文件是隔行扫描的。

AVIF_WASCAPTUREFILE 表示 AVI 文件是一个捕获实时视频的特殊配置文件。

AVIF_COPYRT GHTED 表示 AVI 文件包含有版权资料。

AVIF_HASINDEX 和 AVIF_MUSTUSEINDEX 标志适用于带有索引块的文件。AVI_HASINDEX 标志表示提供了索引。

AVIF_MUSTUSEINDEX 标志表明索引被用于确定数据显示序列。当标志被设置时，它表示文件中块的物理序列与显示序列不对应。

AVIF_ISINTERLEAVED 标志表示 AVI 文件时隔行扫描的。系统在 CD-ROM 中通过隔行扫描比逐行扫描更有效率。

AVIF_WASCAPTUREFILE 标志表示 AVI 文件是一个捕获实时视频的特殊配置文件。典型的，此文件已经由用户整理碎片，所以能很有效的进入文件中。如果标志被设置，在覆盖写入文件时，应用程序应该提醒用户。

AVIF_COPYRIGHTED 表示 AVI 文件包含有版权的资料。当标志被设置时，应用程序不应该让用户复制文件或文件中的数据。

dwTotalFrames 详细说明了文件中帧数据的总数量。

dwInitialFrames 用于隔行扫描文件。如果用户正在生成一个隔行扫描文件，在 AVI 文件序列帧结构初始化之前应该详细说明文件中帧结构的数量情况。

dwStreams 域明确说明了文件中数据流的数目。比如，一个具有音频和视频数据的文件含有两个流。

dwSuggestedBufferSize 域表明读文件时建议的缓冲区大小。通常，尺寸的大小应该能足够包含文件中的最大块的内容。如果设置为零或太小，回放软件在执行回放时会重新分配内存，这将降低执行效率。对于逐行扫描文件，缓冲区大小应该足够大到读入整个记录而不仅仅是一个块。

dwWidth 和 **dwHeight** 域详细说明了 AVI 文件宽度和高度（字节数）。

dwScale and **dwRate** 域用来表明文件要使用的时间缩放比例。除了时间缩放比例之外，每一条数据流可以有它们自己的时间缩放比例。以每秒采样数位单位的时间缩放比例可根据 dwRate 除以 dwScale 确定。

dwStart 和 **dwLength** 域表明了 AVI 文件启动时间和文件长度。根据 dwRate 和 dwScale 确定数值的单位。dwStart 域通常设置到 0。

（4）流参数块（Strl）

主文件参数头后面跟随一个或更多数量的"strl"块。（需要为每一条数据流提供一个"strl"块）。这些块中含有关于文件中数据流的信息。流参数块由四个字符代码"strh"识别。流的格式块由"strf"识别。除了流参数块和流格式块外，"strl"块也可能包括流数据块。流数据块由四字节字符"strd"识别。

流参数头的数据结构具有下面的定义：

```
typedef struct {
    FOURCC    fccType;
    FOURCC    fccHandler;
    DWORD     dwFlags;
    DWORD     dwReserved1;
    DWORD     dwInitialFrames;
    DWORD     dwScale;
    DWORD     dwRate;
```

```
    DWORD    dwStart;
    DWORD    dwLength;
    DWORD    dwSuggestedBufferSize;
    DWORD    dwQuality;
    DWORD    dwSampleSize;
} AVIStreamHeader;
```

此数据流参数头通过四个字符代码的方式识别，具体说明了流所包含的数据类型，例如音频或视频。如果它所说明的流包含视频数据，fccType 域被设定为"vids"，如果它包含音频数据，就设定为"auds"。

fccHandler 域含四个字节的特征代码，它描述了数据使用的可安装压缩器和解压器。

dwFlags 域包含了数据流提供的任何标志。AVISE-DISABLED 标志表明流数据只能在被用户明确核准时才能交付。AVISF_VIDEO_PALCHANGES 标志表明调色板颜色的改变被嵌入到了该文件。

剩下的区域描述了流的回放特性。这些系数包括回放速率（dwScale 和 dwRate）、序列的开始时间（dwStart），序列的长度(dwLength)，回放缓冲区的大小(dwSuggestedBuffer)，数据质量特性的指示器(dwQuality)和采样尺寸(dwSampleSize)。

流参数头结构中的一些域也出现在主文件参数头结构中，其中的数据适用于整个文件，而流参数头结构中的数据却只适用于此数据流。

流格式块必须跟在流参数头"strh"块之后。流格式块描述了流中的数据格式。该块中的信息是一个 BITMAPINFO 结构（若合适也包括调色板信息）。对于音频数据流中的信息是一个 WAVEFORMATEX 或 PCMWAVEFORMAT 结构（WAVEFORMATEX 结构是 WAVEFORMAT 的扩展形式）。

"strl"块也可能含一个流数据块。如果它被使用，便要跟在流格式块后。该块的格式和内容由安装型压缩器和解压器驱动程序定义。通常驱动程序使用这些信息进行配置。读写 RIEF 文件的应用程序不需要对这些信息解码。它们会把这些数据作为内存块从驱动程序存取。

一个 AVI 放映器用指令把 LIST"hdrl"块中的数据流与 LIST "movi"块中的数据流使用"strl"块中的序列联系起来。第一个"strl"适用于 0 流程，第二"strl"适用于 1 流程，以此类推。例如，如果第一个"strl"块描述了波形音频数据，则此波形音频数据便包含在 0 流程中；相似的，如果第二个"strl"块描述了视频数据，那么视频数据便包含在 1 流程中。

（5）LIST 'movi'块

紧跟着主参数头信息的是一个 LIST 'movi'块，它包含了文件中实际数据块，就是图片和声音本身。数据块可直接存在 LIST 'movi'块中，或者被归入"rec "块中。"rec "块组表示组中的数据块应该一次将所有组中的数据块读入。这只用于从 CD-ROM 中播放隔行扫描的文件。

与任何 RIFF 块一样，数据块也包含一个四字节代码以识别块的类型。这个可识别的四字节代码包括数据流的数量和一个代码定义了块中压缩信息的类型。例如，一段波形由一个二字节的代码'wb'来识别。如果一个波形块对应于第二个 LIST "hdrl"描述，它会具有四字节代码“01wb”。

因为所有的格式信息都是在参数头中，在数据块中的声音数据不含有它自己格式的任何

信息。声音数据块具有下面的格式：

```
WAVE   Bytes    '##wb'
       BYTE     abBytes[];
```

视频数据能或不能被压缩放在 DIBS 中。一个没有被压缩的 DIB 在与之相联系的 BITMAPINFO 结构中的 biCompression 域具有 BI_RGB 属性。一个压缩的 DIB 具有一个其它的 BI-RGB 值。

一个没有压缩的 DIB 数据块含有 RGB 视频数据。这些块由 2 字节字符代码'db'识别(`db'是无压缩 DIB bits 的缩写)。压缩的 DIB 数据块由 2 字节字符代码'dc'识别(`dc'是压缩 DIB bits 的缩写)。数据块都不含有关 DIB 的任何参数头信息。无压缩 DIB 的数据块具有下面形式：

```
DIB   Bits    '##db'
      BYTE    abBits[];
```

对于一个压缩 DIB 的数据块具有下列形式：

```
typedef struct {
    BYTE            bFirstEntry;
    BYTE            bNumEntries;
    WORD             wFlags;
    PALETTEENTRY   peNew;
} AVIPALCHANGE;
```

bFirstEntry 域定义要改变的第一个入口，bNumEntries 域定义要改变的入口数量。PeNew 域含新颜色入口。

如果在视频数据流中包含调色板变化，可以设置流参数头中 dwFlags 域为 AVITF_VIDEO_PALCHANGES。该标志表明这个视频数据流含调色板变化，提醒回放软件需要动态调色板。

（6）"idx1"块

在 AVI 文件中的 LIST "movi"块后面是一个索引块。索引块含有数据块索引列表和数据块在文件中的位置。它提供了随机进入文件中任意数据块的一种方法，应用程序不必扫描整个文件就可以在一个大型 AVI 文件中确定某一特定的声音序列和视频图像。

索引块采用四字节代码“idxl”标识。索引块的数据结构如下：

```
typedef struct {
    DWORD   ckid;
    DWORD   dwFlags;
    DWORD   dwChunkOffset;
    DWORD   dwChunkLength;
} AVIINDEXENTRY;
```

在 AVI 文件中，ckid、 dwFlags、 dwChunkOffset 和 dwChunkLength 对应所索引的数据块而重复。如果是逐行扫描文件，每一个"rec"块都在索引块中有对应的这些参数。"rec"项目应该设置 AVIIF_LIST 标志，在 ckid 域具有列表类型。

ckid 域标识数据块，采用 4 字节字符代码。

dwFlags 域详细说明了数据的所有标志。AVIIF-KEYFRAME 代表图像序列中的主帧。主帧不需要预先解压缩视图信息。AVIIF-NOTIME 代表不影响视频流的块。举个例子，调色板块表明的改变调色板项目应该在放映图像帧的间隔中进行。因此，如果一个应用程序需要确定图像序列的长度，它不应使用带有 AVIIF_NOTIME 标志的块。在这种情况下，程序将忽略调色板块。AVIIF_LIST 代表当前块是列表块。可使用 ckid 域识别这些列表块类型。

dwChunkOffset 和 **dwChunkLength** 详细说明了块的位置和长度。dwChunkOffset 域详细说明了相对'movi'列表的块在文件中的位置。dwChunkLength 域说明块长度，不包括 RIFF 参数头的 8 个字节。

如果在 RIFF 文件中含有一个索引，应在 AVI 文件参数头中的 dwFlags 域设置 AVIF-HASINDEX 标志（这个文件参数头用“avih”标识来识别）。这个标志表示文件有一个索引。

（7）其它数据块

如果要在 AVI 文件中对齐数据，可以添加“JVNK”块。（这种块是 RIFF 标准类型）。应用程序读取这些数据时忽略其内容。光盘中采用这些块去连接数据，所以可以更有效地读取。甚至可以用这些块连接 2 KB 的光盘边界数据。“JVNK”块具有以下的格式：

AVI Padding　'JUNK'

　　Byte　　data[]

对于其它 RIFF 文件，所有读 AVI 文件的应用程序都应该忽略不能识别的非 AVI 数据块。保存已装载的文件时，读写 AVI 文件的应用程序应该预先保存这些非 AVI 数据块。

（8）有关隔行扫描文件的专门信息

从光盘回放隔行扫描文件需要一些特殊的处理。这些文件象别的 AVI 文件被读取时一样，需要特殊的处理才可最终生成。

声音被分割成了许多小的独立的帧，每一帧的声音和图像要被编组在一个“rec”块中。纪录块应被处理成大小都是 2 KB 的倍数，这样在文件中，列表块中的实际数据开始在 2 KB 的边界。（这是指列表块在 2 KB 边界前的 12 字节处开始）。

为了给声音驱动程序提供足够的声音信息，声音数据必须相对视图数据有所偏移。典型的是，声音数据超前足够使帧每一次预载大约 0.75 秒。主文件参数头中的 dwInitialRecords 域和声音流参数头中的 dwInitialFrames 域应放置声音数据偏移的数值。

此外，要保证光盘驱动程序能够尽快地读取数据，以支持 AVI 序列。非 MPC 光驱可以小于 150 KB/S 的速度读取数据。

（9）使用 AVI 文件编辑软件 VidEdit

VidEdit 使用户可以创建和编辑由一系列帧组成的序列，这些帧包含声音数据和视频数据。用户也可以用它的创建和编辑包含一个声音流和一个视图流的 AVI 文件。文件中的每个流都要在文件的开始。（也就是说，每个流参数头中的 dwStart 域必须是 0）。

（10）写 AVI 文件的实例代码

MSDN Library Visual Studio 中的示例程序 WRITEAVI.C 文件中有读写 AVI 文件的源代

码例子。为简单，例子假定所有视图帧被解压成相同大小的 DIB。DIB 可以有任何分辨率位：8、16、24 位是流行的。

这些例子也假定所有的波形数据是在内存中的。这些例子不限于 PCM 波形数据，应该工作在各种格式下。无论波形数据是在内存还是在磁盘，通用程序都应该运行良好。

（11）写 AVI 文件

象其它的 RIFF 文件一样，AVI 文件也是由 mmioOpen、 mmioCreateChunk、mmioWrite、mmioAscend 和 mmioClose 函数生成的。这些函数有以下的定义：

mmioOpen 为读或写打开一个文件并且返还一个句柄给这个打开的文件。

mmioCreateChunk 在一个RIFF文件中生成一个新的块。

mmioWrite 对打开的文件进行具有确定字节数的写操作。

mmioAscend 从一个RIFF文件块跳到文件中的下一个块。

mmioClose 关闭一个打开的文件。

除了这些函数外，用户可以用 mmioFOURCC 函数将四个单独的字符转化为 4 字符代码。要想知道关于这些函数和特性的更多信息，请参考其它资料。

MS Visual C++中的 AVIFMT.H 文件包含着为生成所描述的这种两个或四个字符代码的定义。它也定义了 aviTWOCC 和 TWOCCFromFOURCC。这些定义可以由独立字符或由四字符代码生成两字符代码。与许多 RIFF 文件不一样，AVI 文件使用许多嵌套块和子块，这使得它们比大多数 RIFF 文件更复杂。用下面的内容作为一个整体来帮助决定何时生成一个块，何时对块写入数据，何时从块中跳出。这些写内容不包括有关将非 AVI 数据块写到一个文件的信息。这些写内容中的块信息反映了前面“AVI RIFF 格式”章节中的例子。

（12）生成 AVI 文件块

“AVI”文件块是文件中的第一个文件块，直到其它所有的块生成之后才可以从这个块中跳出。

如何处理块：

RIFF ('AVI'使用mmioOpen函数打开文件，用mmioSeek函数查找文件开头部分，用mmioCreateChunk函数生成AVI块。(使用“AVI”四字符代码和MMIO-CREATERIFF标志) 在准备写其它块时不要从这个块中跳出。

（13）生成 LIST "hdrl" 和 "avih"块

LIST "hdrl"块包括流格式参数头块。因为它包括其它的块，必须到其它的参数头块生成之后才能从这个块中跳出。"avih"块包含主文件参数头列表。它被作为一个完整的块写入。

如何处理块：

LIST ('hdrl'用mmioCreateChunk函数生成LIST "hdrl"块。 (采用4字符代码"hdrl"和MMIO-CREATELIST标志)

'avih'(<AVI 主文件参数头>)用 mmioCreateChunk 函数生成 AVI 主文件参数头块。(采用 4 字符代码"avih") 用 mmioWrite 函数写参数头信息。用 mmioAscend 函数从"avih" 块中跳出。