WAV 文件格式

文件是 Windows 标准的文件格式，WAV 文件作为多媒体中使用的声波文件格式之一，它是以 RIFF 格式为标准的。RIFF 是英文 Resource Interchange FileFormat 的缩写，每个 WAV 文件的头四个字节便是“RIFF”。WAV 文件由文件头和数据体两大部分组成。其中文件头又分为 RIFF／WAV 文件标识段和声音数据格式说明段两部分。WAV 文件各部分内容及格式见附表。常见的声音文件主要有两种，分别对应于单声道（11.025KHz 采样率、8Bit 的采样值）和双声道（44.1KHz 采样率、16Bit 的采样值）。采样率是指：声音信号在“模→数”转换过程中单位时间内采样的次数。采样值是指每一次采样周期内声音模拟信号的积分值。对于单声道声音文件，采样数据为八位的短整数（short int 00H-FFH）；而对于双声道立体声声音文件，每次采样数据为一个16位的整数（int），高八位和低八位分别代表左右两个声道。WAV 文件数据块包含以脉冲编码调制（PCM）格式表示的样本。WAV 文件是由样本组织而成的。在单声道 WAV 文件中，声道0代表左声道，声道1代表右声道。在多声道 WAV 文件中，样本是交替出现的。

WAV 文件格式说明表

文件头	偏移地址	字节数	数据类型	内 容
00	H	4	char	"RIFF"标志
04	H	4	long	int 文件长度
08	H	4	char	"WAV"标志
0C	H	4	char	"fmt"标志
10	H	4		过渡字节（不定）
14	H	2	int	格式类别（10H 为 PCM 形式的声音数据)
16	H	2	int	单声道为1，双声道为2通道数
18	H	2	int	采样率（每秒样本数），表示每个通道的播放速度
1C	H	4	long	波形音频数据传送速率，其值为通道数×每秒数据位数×每样 本的数据位数 / 8。播放软件利用此值可以估计缓冲区的大小
22	H	2		每样本的数据位数，表示每个声道中各个样本的数据位数。如果有多 个声道，对每个声道而言，样本大小都一样。 24H 4 char 数据标记符 " data " 28H 4 long int 语音数据的长度

PCM 数据的存放方式:

样本1 样本2

8位单声道 0声道 0声道

8位立体声 0声道（左） 1声道（右） 0声道（左） 1声道（右）

16位单声道 0声道低字节 0声道高字节 0声道低字节 0声道高字节

16位立体声 0声道（左）低字节 0声道（左）高字节 1声道（右）低字节 1声道（右）高字节

PCM 数据的存放方式:

WAV 文件的每个样本值包含在一个整数 i 中，i 的长度为容纳指定样本长度所需 的最小字节数。首先存储低有效字节，表示样本幅度的位放在 i 的高有效位上， 剩下的位置为0，这样8位和16位的 PCM 波形样本的数据格式如下所示。

样本大小	数据格式	最大值	最小值
8位 PCM	unsigned int	225	0
16位 PCM	int	327	67

wav 文件格式分析详解

一、综述

WAVE 文件作为多媒体中使用的声波文件格式之一，它是以 RIFF 格式为标准的。RIFF 是英文 Resource Interchange File Format 的缩写，每个 WAVE 文件的头四个字节便是“RIFF”。

WAVE 文件是由若干个 Chunk 组成的。按照在文件中的出现位置包括：RIFF WAVEChunk，Format Chunk，Fact Chunk(可选)，Data Chunk。具体见下图：

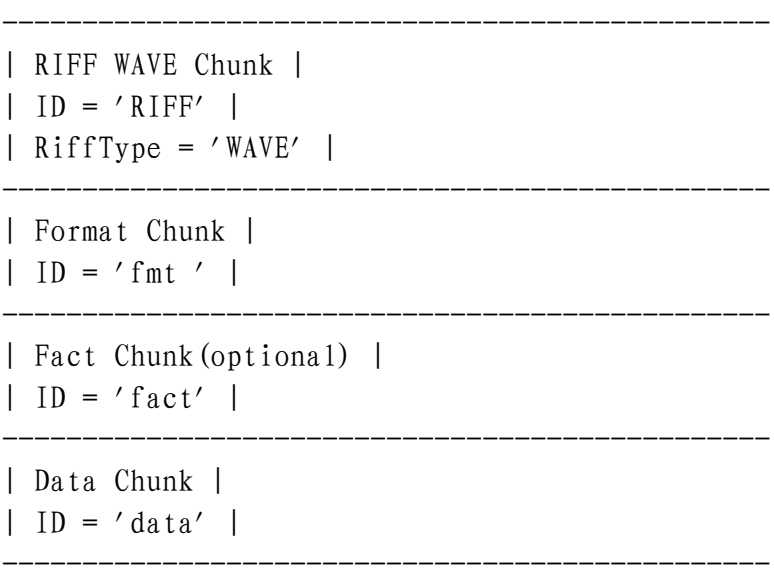

图 1 Wav 格式包含 Chunk 示例

其中除了 Fact Chunk 外，其他三个 Chunk 是必须的。每个 Chunk 有各自的 ID，位于 Chunk 最开始位置，作为标示，而且均为 4 个字节。并且紧跟在 ID 后面的是 Chunk 大小（去除 ID 和 Size 所占的字节数后剩下的其他字节数目），4 个字节表示，低字节表示数值低位，高字节表示数值高位。下面具体介绍各个 Chunk 内容。

PS：所有数值表示均为低字节表示低位，高字节表示高位。

二、具体介绍

RIFF WAVE Chunk

	所占字节数	具体内容
ID	4 Bytes	'RIFF'
Size	4 Bytes	
Type	4 Bytes	'WAVE'

图 2 RIFF WAVE Chunk

以'FIFF'作为标示，然后紧跟着为 size 字段，该 size 是整个 wav 文件大小减去 ID 和 Size 所占用的字节数，即 FileLen－8＝Size。然后是 Type 字段，为'WAVE'，表示是 wav 文件。

结构定义如下:

```
struct RIFF_HEADER
{
  char szRiffID[4]; // 'R','I','F','F'
  DWORD dwRiffSize;
  char szRiffFormat[4]; // 'W','A','V','E'
 };
```

Format Chunk

	字节数	具体内容
ID	4 Bytes	'fmt '
Size	4 Bytes	数值为 16 或 18，18 则最后又附加信息
FormatTag	2 Bytes	编码方式，一般为 0x0001
Channels	2 Bytes	声道数目，1--单声道；2--双声道

SamplesPerSec	4 Bytes	采样频率	
AvgBytesPerSec	4 Bytes	每秒所需字节数	|===> WAVE_FORMAT
BlockAlign	2 Bytes	数据块对齐单位(每个采样需要的字节数)	
BitsPerSample	2 Bytes	每个采样需要的 bit 数	
	2 Bytes	附加信息（可选，通过 Size 来判断有无）	

图 3 Format Chunk

以'fmt '作为标示。一般情况下 Size 为 16，此时最后附加信息没有；如果为 18 则最后多了 2 个字节的附加信息。主要由一些软件制成的 wav 格式中含有该 2 个字节的附加信息。

结构定义如下:

```
struct WAVE_FORMAT
{
  WORD wFormatTag;
  WORD wChannels;
  DWORD dwSamplesPerSec;
  DWORD dwAvgBytesPerSec;
  WORD wBlockAlign;
  WORD wBitsPerSample;
};
struct FMT_BLOCK
{
  char szFmtID[4]; // 'f','m','t',' '
  DWORD dwFmtSize;
  WAVE_FORMAT wavFormat;
};
```

Fact Chunk

	所占字节数	具体内容
ID	4 Bytes	'fact'

Size	4 Bytes	数值为 4
data	4 Bytes	

图 4 Fact Chunk

Fact Chunk 是可选字段，一般当 wav 文件由某些软件转化而成，则包含该 Chunk。结构定义如下:

```
struct FACT_BLOCK
{
  char szFactID[4]; // 'f','a','c','t'
  DWORD dwFactSize;
};
```

Data Chunk

	所占字节数	具体内容
ID	4 Bytes	'data'
Size	4 Bytes	
data		

图 5 Data Chunk

Data Chunk 是真正保存 wav 数据的地方，以'data'作为该 Chunk 的标示。然后是数据的大小。紧接着就是 wav 数据。根据 Format Chunk 中的声道数以及采样 bit 数，wav 数据的 bit 位置可以分成以下几种形式:

单声道	取样 1	取样 2	取样 3	取样 4
8bit 量化	声道 0	声道 0	声道 0	声道 0

```
| 双声道 | 取样 1 | 取样 2 |
| |-----------------------------------------------
| 8bit 量化 | 声道 0(左) | 声道 1(右) | 声道 0(左) | 声道 1(右)
-------------------------------------------------
| | 取样 1 | 取样 2 |
| 单声道 |-----------------------------------------
| 16bit 量化 | 声道 0 | 声道 0 | 声道 0 | 声道 0 |
| | (低位字节) | (高位字节) | (低位字节) | (高位字节) |
-------------------------------------------------
| | 取样 1 |
| 双声道 |-----------------------------------------
| 16bit 量化 | 声道 0(左) | 声道 0(左) | 声道 1(右) | 声道 1(右)
| | (低位字节) | (高位字节) | (低位字节) | (高位字节) |
-------------------------------------------------
```

图 6 wav 数据 bit 位置安排方式

Data Chunk 头结构定义如下:

```
struct DATA_BLOCK
{
  char szDataID[4]; // 'd','a','t','a'
  DWORD dwDataSize;
};
```

三、小结

因此，根据上述结构定义以及格式介绍，很容易编写相应的 wav 格式解析代码。这里具体的代码就不给出了。同时由于采用 4 字节存储文件/数据大小，理论上一个 wave 文件不能大于 2G（有符号）或 4G（无符号）。