MP3文件格式 | 2007-9-2 22:10:00

**一．概述：**   
MP3 文件是由帧()构成的，帧是MP3 文件最小的组成单位MP3的全称应为MPEG1 Layer-3 音频  
文件，MPEG(Moving Picture Experts Group) 在汉语中译为活动图像专家组，特指活动影音压缩标准，MPEG   
音频文件是MPEG1 标准中的声音部分，也叫MPEG 音频层，它根据压缩质量和编码复杂程度划分为三层，即  
Layer-1Layer2Layer3， 且分别对应MP1MP2MP3 这三种声音文件，并根据不同的用途，使用不同层  
次的编码MPEG 音频编码的层次越高，编码器越复杂，压缩率也越高，MP1 和MP2 的压缩率分别为4：1 和  
6：1-8：1，而MP3 的压缩率则高达10：1-12：1，也就是说，一分钟CD 音质的音乐，未经压缩需要10MB   
的存储空间，而经过MP3 压缩编码后只有1MB 左右不过MP3 对音频信号采用的是有损压缩方式，为了降  
低声音失真度，MP3采取了感官编码技术，即编码时先对音频文件进行频谱分析，然后用过滤器滤掉  
噪音电平，接着通过量化的方式将剩下的每一位打散排列，最后形成具有较高压缩比的MP3 文件，并使压  
缩后的文件在回放时能够达到比较接近原音源的声音效果

**二．整个MP3 文件结构：**   
MP3 文件大体分为三部分：TAG\_V2(ID3V2)，, TAG\_V1(ID3V1)   
  
ID3V2 包含了作者，作曲，专辑等信息，长度不固定，扩展了ID3V1 的信息量   
一系列的帧，个数由文件大小和帧长决定   
. 每个 的长度可能不固定，也可能固定，由位率bitrate 决定   
. 每个 又分为帧头和数据实体两部分   
. 帧头记录了mp3 的位率，采样率，版本等信息，每个帧之间相互独立   
  
ID3V1 包含了作者，作曲，专辑等信息，长度为128BYTE   
  
**三．MP3的 格式：**每个 都有一个帧头HEADER，长度是4BYTE（32bit）,帧头后面可能有两个字节的CRC 校  
验，这两个字节的是否存在决定于HEADER 信息的第16bit， 为0 则帧头后面无校验，为1 则有校验,  
校验值长度为2 个字节，紧跟在HEADER 后面，接着就是帧的实体数据了，格式如下：   
HEADER   
CRC（free）   
MAIN\_DATA   
4 BYTE   
0 OR 2 BYTE   
长度由帧头计算得出   
  
1．帧头HEADER 格式如下：   
AAAAAAAA AAABBCCD EEEEFFGH IIJJKLMM   
13 个帧头字符的含义如下：   
Sign Length   
(bits)   
Position   
(bits)   
Deion   
A 11 (31-21) sync (all bits set)   
B 2 (20,19) MPEG Audio version   
00 - MPEG Version 2.5   
01 - reserved   
10 - MPEG Version 2   
11 - MPEG Version 1   
C 2 (18,17) Layer deion   
00 - reserved   
01 - Layer III   
10 - Layer II   
11 - Layer I   
D 1 (16) Protection bit   
0 - Protected by CRC (16bit crc follows header)   
1 - Not protected   
E 4 (15,12) Bitrate index   
bits V1,L1 V1,L2 V1,L3 V2,L1 V2,L2 V2,L3   
0000 free free free free free free   
0001 32 32 32 32 32 8 (8)   
0010 64 48 40 64 48 16 (16)   
0011 96 56 48 96 56 24 (24)   
0100 128 64 56 128 64 32 (32)   
0101 160 80 64 160 80 64 (40)   
0110 192 96 80 192 96 80 (48)   
0111 224 112 96 224 112 56 (56)   
1000 256 128 112 256 128 64 (64)   
1001 288 160 128 288 160 128 (80)   
1010 320 192 160 320 192 160 (96)   
1011 352 224 192 352 224 112 (112)   
1100 384 256 224 384 256 128 (128)   
1101 416 320 256 416 320 256 (144)   
1110 448 384 320 448 384 320 (160)   
1111 bad bad bad bad bad bad   
NOTES: All s are in kbps   
V1 - MPEG Version 1   
V2 - MPEG Version 2 and Version 2.5   
L1 - Layer I   
L2 - Layer II   
L3 - Layer III   
"free" means variable bitrate.   
"bad" means that this is not an allowed   
The s in parentheses are from different sources which   
claim that those s are valid for V2,L2 and V2,L3. If   
anyone can confirm please let me know.   
F 2 (11,10) Sampling rate frequency index (s are in Hz)   
bits MPEG1 MPEG2 MPEG2.5   
00      44100       22050        11025   
01      48000       24000        12000   
10      32000       16000        8000   
11      reserv.       reserv.      reserv.   
G 1 (9) Padding bit   
0 - is not padded   
1 - is padded with one extra bit   
H 1 (8) Private bit (unknown purpose)   
I 2 (7,6) Channel Mode   
00 - Stereo   
01 - Joint stereo (Stereo)   
10 - Dual channel (Stereo)   
11 - Single channel (Mono)   
J 2 (5,4) Mode extension (Only if Joint stereo)   
Intensity stereo MS stereo   
00       off off   
01       on off   
10       off on   
11       on on   
K 1 (3) Copyright   
0 - Audio is not copyrighted   
1 - Audio is copyrighted   
L 1 (2) Original   
0 - Copy of original media   
1 - Original media   
M 2 (1,0) Emphasis   
00 - none   
01 - 50/15 ms   
10 - reserved   
11 - CCIT J.17   
1）每帧的播放时间：无论帧长是多少，每帧的播放时间都是26ms；   
2）数据帧大小:   
Size = (((MpegVersion == MPEG1 ? 144 : 72) \* Bitrate) / SamplingRate) + PaddingBit   
例如: Bitrate ＝ 128000, a SamplingRate ＝44100, and PaddingBit ＝ 1   
Size = (144 \* 128000) / 44100 + 1 = 417 bytes   
  
2．MAIN\_DATA：   
MAIN\_DATA 部分长度是否变化决定于HEADER 的bitrate是否变化，一首MP3 歌曲，它有三个版本：96Kbps（96 千比特位每秒）128Kbps 和192KbpsKbps （比特位速率）， 表明了音乐每秒的数据量，Kbps 值越高，音质越好，文件也越大，MP3标准规定，不变的bitrate 的MP3 文件称作CBR，大多数MP3 文件都是CBR 的，而变化的bitrate 的MP3 文件称作VBR， 每个 的长度都可能是变化的下面是CBR 和VBR 的不同点：   
1）CBR：固定位率的 的大小也是固定的（公式如上所述），只要知道文件总长度，和帧长即可由播放每帧需26ms 计算得出mp3 播放的总时间，也可通过计数帧的个数控制快进快退慢放等操作   
2）VBR：VBR 是XING 公司推出的算法，所以在MP3 的 里会有XING"这个关键字（现在很多流行的小软件也可以进行VBR 压缩，它们是否遵守这个约定，那就不得而知了），它存放在MP3 文件中的第一个有效 里，它标识了这个MP3 文件是VBR 的同时第一个 里存放了MP3 文件的 的总个数，这就很容易获得了播放总时间，同时还有100 个字节存放了播放总时间的100 个时间分段的 的INDEX，假设4 分钟的MP3 歌曲，240S， 分成100 段，每两个相邻INDEX 的时间差就是2.4S， 所以通过这个INDEX，只要前后处理少数的，就能快速找出我们需要快进的 头，可参考下文：   
This system was created to minimize file lengths and to preserve sound quality.   
Higher frequencies generally needs more space for encoding (thats why many codecs cut all   
frequencies above cca 16kHz) and lower tones requires less. So if some part of song doesnt consist  
of higher tones then using eg. 192kbps is wasting of space. It should be enough to use only eg.   
96kbps.   
And it is the principle of VBR. Codec looks over and then choose bitrate suitable for its  
sound quality.   
It sounds perfect but it brings some problems:  
If you want to jump over 2 minutes in song, it is not a problem with CBR because you are able   
simply count amount of Bytes which is necessary to skip. But it is impossible with VBR.   
lengths should be arbitrary so you have to either go by and counts (time consuming   
and very unpractical) or use another mechanism for approximate count.   
If you want to cut 5 minutes from the middle of VBR file (all we know CDs where last song takes  
10 minutes but 5 minutes is a pure silence, HELL!) problems are the same.   
Result? VBR files are more difficult for controlling and adjusting. And I dont like feeling that  
sound quality changes in every moment. And AFAIK many codecs have problems with creation VBR in good quality.   
Personally I cant see any reason why to use VBR -I dont give a \*\*\*\* if size of one CD in MP3   
is 55 MB with CBR or 51 MB with VBR. But everybody has a different taste... some people prefer   
VBR.   
VBR File Structure is the same as for CBR. But the first doesnt contain audio data and it is used for special information about VBR file.   
Structure of the first (the table as follow):  
  
Byte       Content  
0-3      Standard audio header (as deed above). Mostly it contains s FF  
FB 30 4C, from which you can count Len = 156 Bytes. And thats exactly enough   
space for storing VBR info.   
This header contains some important information valid for the whole   
-MPEG (MPEG1 or MPEG2)   
-SAMPLING rate frequency index   
-CHANNEL (JointStereo etc.)   
4-x Not used till string "Xing" (58 69 6E 67). This string is used as a main VBR file   
identifier. If it is not found, file is supposed to be CBR. This string can be placed   
at different locations according to s of MPEG and CHANNEL (ya, these from a   
few lines upwards):   
36-39 "Xing" for MPEG1 and CHANNEL != mono (mostly used)   
21-24 "Xing" for MPEG1 and CHANNEL == mono   
21-24 "Xing" for MPEG2 and CHANNEL != mono   
13-16 "Xing" for MPEG2 and CHANNEL == mono   
After "Xing" string there are placed flags, number of s in file and a size   
of file in Bytes. Each of these items has 4 Bytes and it is stored as ''int'' number   
in memory. The first is the most significant Byte and the last is the least.   
Following schema is for MPEG1 and CHANNEL != mon   
40-43 Flags   
Name Deion   
00 00 00 01 s Flag set if for number of s in file is stored   
00 00 00 02 Bytes Flag set if for filesize in Bytes is stored   
00 00 00 04 TOC Flag set if s for TOC (see below) are stored   
00 00 00 08 VBR Scale Flag set if s for VBR scale are stored   
All these s can be stored simultaneously.   
44-47 s   
Number of s in file (including the first info one)   
48-51 Bytes   
File length in Bytes   
52-151 TOC (Table of Contents)   
Contains of 100 indexes (one Byte length) for easier lookup in file. Approximately   
solves problem with moving inside file.   
Each Byte has a according this formula:   
(TOC[i] / 256) \* fileLenInBytes   
So if song lasts eg. 240 sec. and you want to jump to 60. sec. (and file is 5 000  
000 Bytes length) you can use:   
TOC[(60/240)\*100] = TOC[25]   
and corresponding Byte in file is then approximately at:   
(TOC[25]/256) \* 5000000   
If you want to trim VBR file you should also reconstruct s, Bytes and TOC   
properly.   
152-155 VBR Scale   
I dont know exactly system of storing of this s but this item probably doesnt   
have deeper meaning.   
  
**四．ID3v1**   
ID3V1 比较简单，它是存放在MP3 文件的末尾，用16 进制的编辑器打开一个MP3 文件，查看其末尾  
的128 个顺序存放字节，数据结构定义如下：   
typedef struct tagID3V1   
{   
char Header[3]; /\*标签头必须是"TAG"否则认为没有标签\*/   
char Title[30]; /\*标题\*/   
char Artist[30]; /\*作者\*/   
char Album[30]; /\*专集\*/   
char Year[4]; /\*出品年代\*/   
char Comment[28]; /\*备注\*/   
char reserve; /\*保留\*/   
char track;; /\*音轨\*/   
char Genre; /\*类型\*/   
}ID3V1,\*pID3V1;   
ID3V1 的各项信息都是顺序存放，没有任何标识将其分开，比如标题信息不足30 个字节，则使用''\0''  
补足，否则将造成信息错误Genre使用原码表示，对照表如下：   
/\* Standard genres \*/   
0="Blues";   
1="ClassicRock";   
2="Country";   
3="Dance";   
4="Disco";   
5="Funk";   
6="Grunge";   
7="Hip-Hop";   
8="Jazz";   
9="l";   
10="NewAge";   
11="Oldies";   
12="Other";   
13="Pop";   
14="R&B";   
15="Rap";   
16="Reggae";   
17="Rock";   
18="Techno";   
19="Industrial";   
20="Alternative";   
21="Ska";   
22="Deathl";   
23="Pranks";   
24="Soundtrack";   
25="Euro-Techno";   
26="Ambient";   
27="Trip-Hop";   
28="Vocal";   
29="Jazz+Funk";   
30="Fusion";   
31="Trance";   
32="Classical";   
33="Instrumental";   
34="Acid";   
35="House";   
36="Game";   
37="SoundClip";   
38="Gospel";   
39="Noise";   
40="AlternRock";   
41="Bass";   
42="Soul";   
43="Punk";   
44="Space";   
45="Meditative";   
46="InstrumentalPop";   
47="InstrumentalRock";   
48="Ethnic";   
49="Gothic";   
50="Darkwave";   
51="Techno-Industrial";   
52="Electronic";   
53="Pop-Folk";   
54="Eurodance";   
55="Dream";   
56="SouthernRock";   
57="Comedy";   
58="Cult";   
59="Gangsta";   
60="Top40";   
61="ChristianRap";   
62="Pop/Funk";   
63="Jungle";   
64="NativeAmerican";   
65="Cabaret";   
66="NewWave";   
67="Psychadelic";   
68="Rave";   
69="Showtunes";   
70="Trailer";   
71="Lo-Fi";   
72="Tribal";   
73="AcidPunk";   
74="AcidJazz";   
75="Polka";   
76="Retro";   
77="Musical";   
78="Rock&Roll";   
79="HardRock";   
/\* Extended genres \*/   
80="Folk";   
81="Folk-Rock";   
82="NationalFolk";   
83="Swing";   
84="FastFusion";   
85="Bebob";   
86="Latin";   
87="Revival";   
88="Celtic";   
89="Bluegrass";   
90="Avantgarde";   
91="GothicRock";   
92="ProgessiveRock";   
93="PsychedelicRock";   
94="SymphonicRock";   
95="SlowRock";   
96="BigBand";   
97="Chorus";   
98="EasyListening";   
99="Acoustic";   
100="Humour";   
101="Speech";   
102="Chanson";   
103="Opera";   
104="ChamberMusic";   
105="Sonata";   
106="Symphony";   
107="BootyBass";   
108="Primus";   
109="PornGroove";   
110="Satire";   
111="SlowJam";   
112="Club";   
113="Tango";   
114="Samba";   
115="Folklore";   
116="Ballad";   
117="PowerBallad";   
118="RhythmicSoul";   
119="Freestyle";   
120="Duet";   
121="PunkRock";   
122="DrumSolo";   
123="Acapella";   
124="Euro-House";   
125="DanceHall";   
126="Goa";   
127="Drum&Bass";   
128="Club-House";   
129="Hardcore";   
130="Terror";   
131="Indie";   
132="BritPop";   
133="Negerpunk";   
134="PolskPunk";   
135="Beat";   
136="ChristianGangstaRap";   
137="Heavyl";   
138="Blackl";   
139="Crossover";   
140="ContemporaryChristian";   
141="ChristianRock";   
142="Merengue";   
143="Salsa";   
144="Trashl";   
145="Anime";   
146="JPop";   
147="Synthpop";   
  
**五．ID3V2**   
ID3V2 到现在一共有4 个版本，但流行的播放软件一般只支持第3 版， 既ID3v2.3由于ID3V1 记录  
在MP3 文件的末尾，ID3V2就只好记录在MP3 文件的首部了(如果有一天发布ID3V3，真不知道该记录在哪  
里)也正是由于这个原因，对ID3V2 的操作比ID3V1 要慢而且ID3V2 结构比ID3V1 的结构要复杂得多，  
但比前者全面且可以伸缩和扩展   
下面就介绍一下ID3V2.3   
每个ID3V2.3 的标签都一个标签头和若干个标签帧或一个扩展标签头组成关于曲目的信息如标题作者  
等都存放在不同的标签帧中，扩展标签头和标签帧并不是必要的，但每个标签至少要有一个标签帧标签  
头和标签帧一起顺序存放在MP3 文件的首部   
1标签头   
在文件的首部顺序记录10 个字节的ID3V2.3 的头部数据结构如下：   
char Header[3]; /\*必须为"ID3"否则认为标签不存在\*/   
char Ver; /\*版本号ID3V2.3 就记录3\*/   
char Revision; /\*副版本号此版本记录为0\*/   
char Flag; /\*存放标志的字节，这个版本只定义了三位，稍后详细解说\*/   
char Size[4]; /\*标签大小，包括标签头的10 个字节和所有的标签帧的大小\*/   
1）.标志字节   
标志字节一般为0，定义如下：   
abc00000   
a -- 表示是否使用Unsynchronisation(这个单词不知道是什么意思，字典里也没有找到，一般不设置)   
b -- 表示是否有扩展头部，一般没有(至少Winamp 没有记录)，所以一般也不设置   
c -- 表示是否为测试标签(99.99%的标签都不是测试用的啦，所以一般也不设置)   
2）.标签大小   
一共四个字节，但每个字节只用7 位，最高位不使用恒为0所以格式如下   
0xxxxxxx 0xxxxxxx 0xxxxxxx 0xxxxxxx   
计算大小时要将0 去掉，得到一个28 位的二进制数，就是标签大小(不懂为什么要这样做)，计算公式如  
下：   
int total\_size;   
total\_size = (Size[0]&0x7F)\*0x200000   
+(Size[1]&0x7F)\*0x400   
+(Size[2]&0x7F)\*0x80   
+(Size[3]&0x7F)   
2标签帧   
每个标签帧都有一个10 个字节的帧头和至少一个字节的不固定长度的内容组成它们也是顺序存放在文件  
中，和标签头和其他的标签帧也没有特殊的字符分隔得到一个完整的帧的内容只有从帧头中的到内容大  
小后才能读出，读取时要注意大小，不要将其他帧的内容或帧头读入   
帧头的定义如下：   
char ID[4]; /\*用四个字符标识一个帧，说明其内容，稍后有常用的标识对照表\*/   
char Size[4]; /\*帧内容的大小，不包括帧头，不得小于1\*/   
char Flags[2]; /\*存放标志，只定义了6 位，稍后详细解说\*/   
1）.帧标识   
用四个字符标识一个帧，说明一个帧的内容含义，常用的对照如下：   
TIT2=标题 表示内容为这首歌的标题，下同   
TPE1=作者   
TALB=专集   
TRCK=音轨 格式：N/M 其中N 为专集中的第N 首，M为专集中共M 首，N和M 为ASCII 码表示的数字   
TYER=年代 是用ASCII 码表示的数字   
TCON=类型 直接用字符串表示   
COMM=备注 格式："eng\0备注内容"，其中eng 表示备注所使用的自然语言   
2）.大小   
这个可没有标签头的算法那么麻烦，每个字节的8 位全用，格式如下   
xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx   
算法如下：   
int FSize;   
FSize = Size[0]\*0x100000000   
+Size[1]\*0x10000   
+Size[2]\*0x100   
+Size[3];   
3）.标志   
只定义了6 位，另外的10 位为0，但大部分的情况下16 位都为0 就可以了格式如下：   
abc00000 ijk00000   
a -- 标签保护标志，设置时认为此帧作废   
b -- 文件保护标志，设置时认为此帧作废   
c -- 只读标志，设置时认为此帧不能修改(但我没有找到一个软件理会这个标志)   
i -- 压缩标志，设置时一个字节存放两个BCD 码表示数字   
j -- 加密标志(没有见过哪个MP3 文件的标签用了加密)   
k -- 组标志，设置时说明此帧和其他的某帧是一组   
值得一提的是winamp 在保存和读取帧内容的时候会在内容前面加个''\0''，并把这个字节计算在帧内容的  
大小中   
附：帧标识的含义   
4）. Declared ID3v2 s   
The following s are declared in this draft.   
AENC Audio encryption   
APIC Attached picture   
COMM Comments   
COMR Commercial   
ENCR Encryption method registration   
EQUA Equalization   
ETCO Event timing codes   
GEOB General encapsulated object   
GRID Group identification registration   
IPLS Involved people list   
LINK Linked information   
MCDI Music CD identifier   
MLLT MPEG location lookup table   
OWNE Ownership   
PRIV Private   
PCNT Play counter   
POPM Popularimeter   
POSS Position synchronisation   
RBUF Recommended buffer size   
RVAD Relative volume adjustment   
RVRB Reverb   
SYLT Synchronized lyric/text   
SYTC Synchronized tempo codes   
TALB Album/Movie/Show title   
TBPM BPM (beats per minute)   
TCOM Composer   
TCON Content type   
TCOP Copyright message   
TDAT Date   
TDLY Playlist delay   
TENC Encoded by   
TEXT Lyricist/Text writer   
TFLT File type   
TIME Time   
TIT1 Content group deion   
TIT2 Title/songname/content deion   
TIT3 Subtitle/Deion refinement   
TKEY Initial key   
TLAN Language(s)   
TLEN Length   
TMED Media type   
TOAL Original album/movie/show title   
TOFN Original filename   
TOLY Original lyricist(s)/text writer(s)   
TOPE Original artist(s)/performer(s)   
TORY Original release year   
TOWN File owner/licensee   
TPE1 Lead performer(s)/Soloist(s)   
TPE2 Band/orchestra/accompaniment   
TPE3 Conductor/performer refinement   
TPE4 Interpreted, remixed, or otherwise modified by   
TPOS Part of a set   
TPUB Publisher   
TRCK Track number/Position in set   
TRDA Recording dates   
TRSN Internet radio station name   
TRSO Internet radio station owner   
TSIZ Size   
TSRC ISRC (international standard recording code)   
TSSE Software/Hardware and settings used for encoding   
TYER Year   
TXXX User defined text information   
UFID Unique file identifier   
USER Terms of use   
USLT Unsychronized lyric/text tranion   
WCOM Commercial information   
WCOP Copyright/Legal information   
WOAF Official audio file webpage   
WOAR Official artist/performer webpage   
WOAS Official audio source webpage   
WORS Official internet radio station homepage   
WPAY Payment   
WPUB Publishers official webpage   
WXXX User defined URL link