13-14 TP391.41

# GIF 图像格式及压缩方法的分析

辽宁大学计算机科学系(110036) 李世清 陈春光 鞠晓光 潘照新

摘 要：GIF 图像文件的格式、压缩方法及原理。

关键词：GIF 图像文件格式 LZX 压缩方法

GIF (Graphics Interchang Format)是由 CompuServe 公司推出的。CompuServeGIF 是目前较复杂的，也是应用较广泛的一种图像文件格式。因为它不是某一种应用软件的副产品(例如：PCX 图像格式文件是属于 PC Paintbrush 的)，这使计算机软件人员对它非常有兴趣。

| |
|---|
| 标识域 GIF87a |
| 水平分辨率 |
| 垂直分辨率 |
| 整体标识 |
| 背景颜色 |
| 结束标志 00H |
| 彩色表 |
| 局部数据 |
| 图像数据 |
| 结束符 21H |

图 1

## 1 GIF 图像文件格式

GIF 图像文件格式的总体结构如图 1。下面介绍每部分的含义。

### 1.1 文件头的数据结构

```
Struct CIFHeader{
    unsigned char Tag[]="GIF87a";    |*标识域 GIF 固定不变*|
int Screen _wideth;                  |*水平分辨率*|
int Screen _Depth;                   |*垂直分辨率*|
unsigned char Global _Flag;          |*全局标识*|
unsigned char Background[3];         |*背景颜色 RBG*|
unsinged char EndFlag;               |*结束标志 00H*|
```

GIF 图像文件格式文件头的文件标志前三个字节固定为"GIF"三个字符，作为测试标志，后三个字节是一些版本信息。一般图像文件阅读器只要识别前三个字节就可以了；从这儿以后到头结束标志(EndFlag)称为整体数据，它规定了文件中图像(可能有多个)的整体信息。屏幕宽(Screen _ width)和屏幕深(Screen _ Depth)规定了屏幕的物理特性，一般这两个数据不使用；背景色(Background)是屏幕图像背景的颜色号，在显示图像前，必须把屏幕及其边框设成该颜色，以达到原图像设计的视觉效果。文件头结束标识(EndFlag)，一方面可以作为文件结束的标识，另一方面，如果图像文件阅读器读到该字节不是 00H，则说明文件结构有错误。

整体标志(Global Flag)较复杂，所以对它的说明独立成段落，可从这个字节中提出几种信息：

1. 将 Global Flag 字节的低三位加上 1，得到 n，将"01H"左移 n 次得到 m，则 m 为该图像所包含的颜色数。

2. 如果 Global Flag 字节最高位为"0"，则说明文件有一个整体"颜色映射"，否则，默认为使用系统缺省彩色表(COLOR Map)。

彩色表(Color Map)长度随 GIF 文件图像中颜色的不同而不同，16 色的是 16×3 字节，24 位彩色的是 256×3 字节……。关于彩色表的其它知识，请参考有关文献。

整体数据之后，被称为局部信息。一个 GIF 图像格式文件可以含有一个或多个图像，这样一个文件就可以含有一个或多个局部信息。每个局部信息又包含几个部分，如图 2。局部信息是一个块(chunk)的集合，每个块可能是图像、扩充、或者文件结束符。而 GIF 以三种不同字符来区分不同的块：","(2CH)表示图像块；"!"(3BH)表示扩充块；";"(21H)表示文件结束符。

| |
|---|
| 图像块标识 2CH |
| 左边界 Left |
| 上边界 top |
| 图像实际宽度 wide |
| 图像实际深度 deep |
| 局部标识 |

left、top、wide、deep 决定实际的显示位置及大小。

图 2

### 1.2 局部表头的数据结构

一个图像块又包括一个局部表头和位于其后的图像压缩数据部分。局部表头的数据结构：

```
Struct Local _Header{
                    char LH _Flag=',';
                    int Image _Left;
                    int Image _Top;
                    int Image _Wide;
                    int Image _Deep;
                    unsigned char Local _Flag;
                    };
```

LH _Flag(",")是局部信息的开始标识，且每一局部信息都是以它作为开始标识，接着的这些数据是实际图像的大小，而非整体数据中的屏幕宽(Screen _ Width)和屏幕深(Screen _ Depth)规定了屏幕的大小。图像左边距(Image _ Left)和图像上边距(Image _ Top)是图像的左上顶点坐标；图像宽(Image _ Wide)

和图像深(Image _ Deep)表示图像的大小(以屏幕上的象素为单位);局部标识(Local _ Flag)与整体标识(Global _ Flag)的作用基本相同,只是如果该字节最高位是"1",表示颜色数由整体标识(Global Flag)决定,局部彩色表由下一字节开始。另外,该字节的第7位表示图像的存储方式。若为"1",表示图像按交替方式存储;若为"0",则未压缩的图像数据的顺序与屏幕上的行相同。

GIF 图像文件的这种结构使其具有较好的扩展性。

图像数据的信息量巨大,对图像数据进行压缩是非常必要的。GIF 图像文件采取 LZW(Lempel Ziv & Welch)压缩方法。

## 2 LZW 压缩方法

LZW 压缩方法是同类压缩方法(标识化方法)较先进和完善的,但也是较复杂的,完全可以独立成章,所以本文只能介绍其基本原理。

LZW 算法是基于转换表的,它能动态地产生标志,压缩程序首先初始化转换表,在转换表中建立 256 项,第 0 页为 0,第 1 项为 1,第 2 项为 2……第 255 项为 255;然后将字符串 S 置为空;这时在图像数据中读入一个字节 C,将 S 与 C 拼接(S+C)(如果该图像数据为第一个,则 S+C 必为 C,且一定在前 256 项中);如果 S+C 在转换表中,则将 S+C 赋于 S;否则,向图像文件中输出与 S 相关的代码,并且在转换表中加入新表项 S+C,使 S=C。重复以上过程,直到最后一个象素值。

表 1 图像原始数据

| 象素点序号 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | … |
|---|---|---|---|---|---|---|---|---|---|
| 象素点值 | 42 | 42 | 42 | 31 | 31 | 42 | 42 | 56 | … |

举例如下:将表 1 所示的第 0 点值 42 读入 C,此时 S 为空,则 S+C 为 42,42 必然在转换表中,将 S 置为 42;读第 1 点值 42 到 C 中,S+C 为 42、42,没在转换表中,则输出 S,且在 256 项加入 S+C,并使 S 等于 C;将第 2 点 42 读入 C,则 S+C 为 42、42,且在转换表中(第 256 项),这样将 S 置为 S+C(42、42);将第 3 点值 31 读入 C,则 S+C 为 42、42、31,不在转换表中,输出与 S 有关的代码 256,且在第 257 项加上 S+C(42、42、31),将 S 置为 C(31);……。重复以上过程,直到最后一个象素值。转换表中的结果如表 2 所示。LZW 算法用 12 位代码,最多可在转换表中处理 4096 项。

表 2 转换表内容

| 像点 | 原始数据 | 输出 | 增加项及内容 |
|---|---|---|---|
| 0 | 42 | | |
| 1 | 42 | 42 | 256:42,42 |
| 2 | 42 | | |
| 3 | 31 | 256 | 257:42,42,31 |
| 4 | 31 | 31 | 258:31,31 |
| 5 | 31 | 31 | 259:31,42 |
| 6 | 42 | | |
| 7 | 56 | 256 | 260:42,42,56 |
| … | … | | |

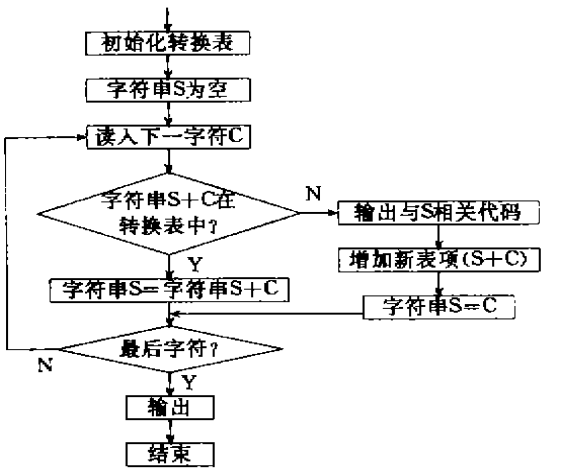


图3 LZW算法流程图

LZW 算法是基于转换表的标识化方法,即把图像原始数据,通过转换表(字符串表)转换成一系列标记,解码程序可以无失真地将压缩数据还原为图像数据。LZW 算法流程如图 3 所示。读者阅读本文后,可以编程读出感兴趣的图像文件。

(收稿日期:1994-05-29)

# 巧用 EDLIN 给程序加行号

中国航天工业总公司燎原无线电厂(636156) 许 杰

由于众多的全屏幕编辑器的出现,DOS 所提供的行编辑 EDLIN 现很少使用,但它所提供的按行号显示功能却能很方便的给程序加上行号,在进行系统开发过程中,尤其是后期系统维护中能提供清晰的源程序文档。方法如下:

1. 建立一文本文件 LINE,内容如下:

```
1,nnnL
E
```

其中 nnn 为大于或等于程序总行数的一整数。

2. 命令行下运行)EDLIN PROG1(LINE )PROG2

PROG1 为欲加行号的源程序,PROG2 为加行号后的文件。

3. 删除 PROG2 中的第一行 End of input file,第二行 * 1,nnn L 和最后一行 * E

将 PROG2 打印出来即为带行号的源程序文档。

(收稿日期:1994-08-10)