GIF 图像文件的格式分析及显示

贵州省贵阳市花溪 89776 部队一室（550025） 孙维颖

摘 要 本文对 GIF 图像文件的格式进行了详细的分析，并详细介绍了解压缩图像数据方法，最后给出了完整的示例程序，实现了对 GIF 格式图像文件的显示。

关键词 GIF，解压缩，调色板

1. GIF 文件格式

GIF 文件由表头、通用调色板、图像数据区、文件尾等部分组成。

·表头格式：

```
typedef struct
{
   char gif _flag [6]; //文件标志
   int screen _width; //图像水平分辨率
   int screen _depth; //图像垂直分辨率
   char global _flag; //图像整体标志
   char ground _color; //图像背景颜色
   char byte0; //原始图像的像素纵横比，在 87a 版中设定为 0
} global _header
```

在 GIF 文件头，可读出“GIF87a”文件标志，表示为 87a 版的 GIF 文件。

·图像整体标志的意义如图 1 所示：

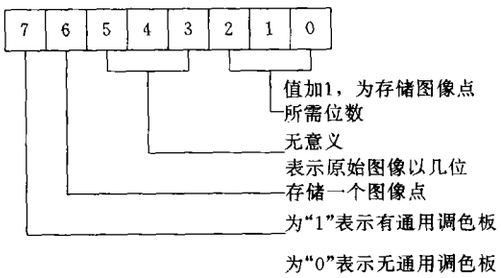


图 1

·通用调色板：GIF 图像文件可以存储多幅图像，通用调色板即提供给这些图像的颜色数据，每幅图像可用专供自己使用的局部调色板数据。调色板数据的设置正确与否对图像文件的显示至关重要，否则将面目全非。

调色板数据由红、绿、蓝（RGB）三原色混合而成，即三个字节代表一种颜色，第一字节为红色，第二字节为绿色，第三字节为蓝色，依次下去。例如显示 256 色图像，则调色板数据由 256*3＝768 个字节表示。

·图像数据区：在通用调色板后，将读到以“0x2c”开头的、经过压缩处理的图像数据，它包括局部表头和图像数据块。

·局部表头格式：

```
typedef struct {
            int image _left; //图像左值
            int image _top; //图像顶值
            int image _wide; //图像宽度
            int image _deep; //图像深度
            char local _flag; //图像局部标志
          } local _header
```

·图像局部标志字节的意义如图 2 所示：

如果有局部调色板，其结构同通用调色板。

如果无局部调色板，则局部表头后紧跟为被压缩的图像数据块。

·图像数据块：图像数据块的第一个字节表示存储一个图像点所需位数，其后为被压缩的图像数据。GIF 文件的图像数据是分块存储的，每块不超过 256 个字节，每块的第一个字节表示该块字节长度（不包括每块的第一个字节）。在所有图像块最后，将读入“0x00”字节，表示图像数据结束。至此，一幅图像的数据就处理完了。如果再读入“,”(0x2c)，表示另一幅图像开始，重复以上过程；如果读入“;”(0x3b)，为文件结束符，表示全部文件处理完。

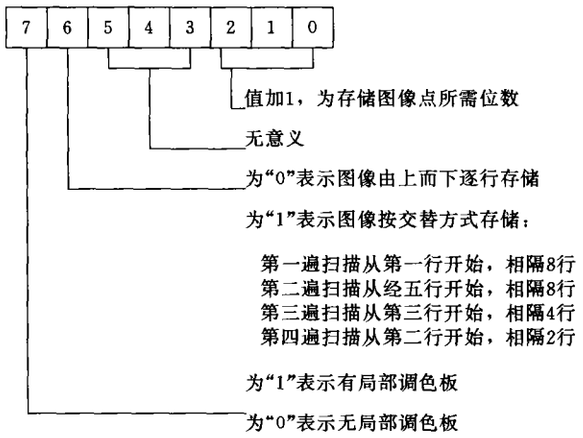
2. GIF 图像数据解压缩

GIF 图像数据是经 GIF-LZW 方法压缩的，它采用不定数据长度压缩数据，长度可以从 3 位到 12 位不定。

在图像数据块中读入的第一个字节在解压缩过程中扮演着很重要的角色，它表示存储一个图像点所

需的位数，由它可计算出清位码（clearcode）、结束码（eofcode）、字符对代码（freecode）的起始值和压缩数据长度（codesize），例如：

```
codesize=0x08;    //图像数据区第一个字节 0x08
clearcode= (1<<codesize);    //清位码为 0x100
eofcode=clearcode+1;    //结束码为 0x101
freecode = firstcode = clearcode + 2;    //字符对代码起始值为 0x102
codesize++;    //压缩数据长度从 9 位开始
initcodesize=codesize;    //压缩数据长度赋值
maxcode= (1<<codesize);    //最大代码为 512
```



在解压缩图像数据过程中，要动态产生三份表格，分别是字头表（prefix）、字尾表（suffix）、输出代码表（outcode）；字头表和字尾表设定为 4096 个表项，输出代码表设定为 1024。压缩数据长度（codesize）会随着字符对代码（freecode）的增加而增加。如上例，当字符对代码从 258 开始增至 512 时，压缩数据长度就从 9 位增至 10 位；当增至 1024 时，压缩数据长度增至 11 位；当增至 2048 时，压缩数据长度增至 12 位。

```
freecode=maxcode=512    codesize=10
freecode=maxcode=1024    codesize=11
freecode=maxcode=2048    codesize=12
```

若读入代码为清位码（clearcode），则重新初始化代码表：

```
if (code==clearcode)
  { codesize=initcodesize;
    maxcode= (1<<codesize);
    freecode=firstcode;
  }
```

• 解压缩数据实例：

〈0x100〉〈0xbd〉〈0x3a〉〈0x09〉〈0x09〉〈0x0d〉〈0x106〉〈0x104〉... 为一串压缩数据，其解码过程如下：

（1）读入第一个代码〈0x100〉，因 code = clearcode，所以进行字符串初始化。

输入数据	输出代码	保留代码	字符对代码	字头表	字尾表
〈0x100〉	—	—	〈0x000〉	0x00	
			〈0x000〉	0x01	
			〈0x000〉	0x02	
			...	...	...
			〈0x0fe〉	0xfe	
			〈0x0ff〉	0xff	
			〈0x100〉	clearcode	
			〈0x101〉	eofcode	

（2）读入第二个代码〈0xbd〉，因代码在字串表中，则将其对应字串输出，并拷贝至保留代码（oldcode）。

输入数据	输出代码	保留代码	字符对代码	字头表	字尾表
〈0xbd〉	〈0xbd〉	〈0xbd〉	—	—	—

（3）读入第三个代码〈0x3a〉，因代码在字串表中，则将其对应字串输出，将保留代码对应字串和输入代对应字串第一个字符分别存入字头表和字尾表，并将输入代码拷贝至保留代码（oldcode）。

输入数据	输出代码	保留代码	字符对代码	字头表	字尾表
〈0x3a〉	〈0x3a〉	〈0x3a〉	〈0x102〉	〈0xbd〉	〈0x3a〉

（4）读入第四个代码〈0x09〉，因代码在字串表中，则将其对应字串输出，将保留代码对应字串和输入代对应字串第一个字符分别存入字头表和字尾表，并将输入代码拷贝至保留代码（oldcode）。

输入数据	输出代码	保留代码	字符对代码	字头表	字尾表
〈0x09〉	〈0x09〉	〈0x09〉	〈0x103〉	〈0x3a〉	〈0x09〉

（5）读入第五个代码〈0x09〉，因代码在字串表中，则将其对应字串输出，将保留代码对应字串和输入代对应字串第一个字符分别存入字头表和字尾表，并将输入代码拷贝至保留代码（oldcode）。

输入数据	输出代码	保留代码	字符对代码	字头表	字尾表
0x09〉	〈0x09〉	〈0x09〉	〈0x104〉	〈0x09〉	〈0x09〉

（6）读入第六个代码〈0x0d〉，因代码在字串表中，则将其对应字串输出，将保留代码对应字串和输入代对应字串第一个字符分别存入字头表和字尾表，并将输入代码拷贝至保留代码（oldcode）。

输入数据	输出代码	保留代码	字符对代码	字头表	字尾表
〈0x0d〉	〈0x0d〉	〈0x0d〉	〈0x105〉	〈0x09〉	〈0x0d〉

（7）读入第七个代码〈0x106〉，因代码不在字串表中，则将保留代码对应字串加上该代码对应字串的第一个字符作为输出代码输出，然后将输出代码配对分别存入字头表和字尾表，产生新字串，将输入代码拷贝至保留代码（oldcode）。

输入数据	输出代码	保留代码	字符对代码	字头表	字尾表

〈0x106〉 〈0x0d〉〈0x0d〉 〈0x106〉 〈0x106〉 〈0x0d〉〈0x0d〉

(8) 读入第八个代码〈0x104〉，因代码在字串表中，则将其对应字串输出，将保留代码对应字串和输入代码对应字串第一个字符分别存入字头表和字尾表，并将输入代码拷贝至保留代码（oldcode）。

输入数据 输出代码 保留代码 字符对代码 字头表 字尾表
〈0x104〉 〈0x09〉〈0x09〉 〈0x104〉 〈0x107〉 〈0x106〉〈0x09〉
……

·解码过程框如图 3 所示：

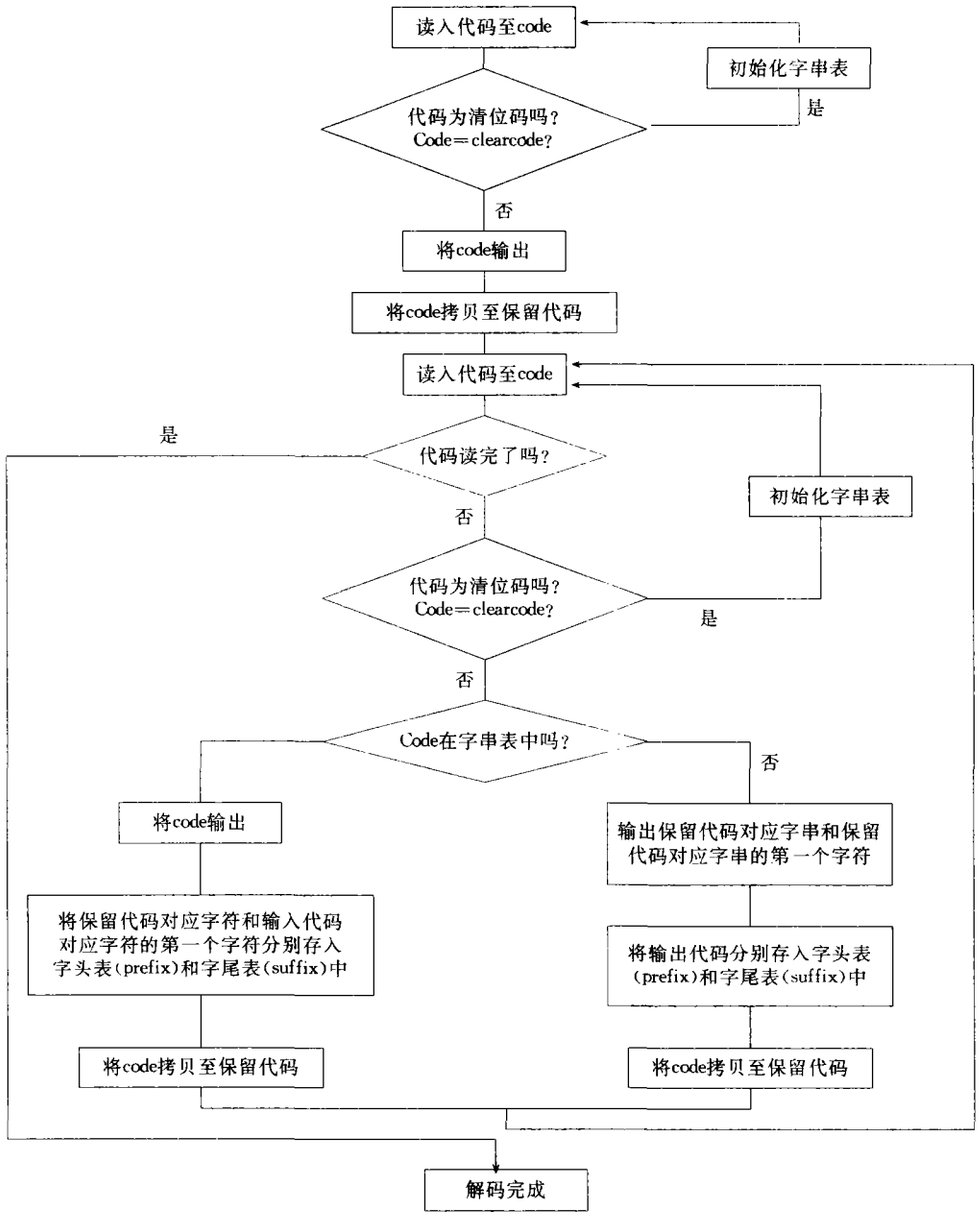


图 3

3. GIF 文件的显示

本文为了快速显示 GIF 图像文件，将压缩图像数据经解压缩后存储于一数据文件中，然后直接传送给视频缓冲区显示（起始地址 0xa0000000）。

本文使用的显示卡是 s3 Trio64v＋VGA 显示卡，在其支持的 VESA 标准扩展 VGA 模式下，采用直接对显示卡编程，设置为 640×480×256 的显示模式（模式号为 0x101）。源程序在 P/133MHZ、16M 内

存微机上通过，是完整的演示程序，可显示 87a 或 89a 版的 GIF 图像文件。

```
    #include " stdio. h"
#include " stdlib. h"
#include " string. h"
#include " conio. h"
#include " dos. h"
#include " math. h"

#define sj " c: \\sj. dat"
#define wj " c: \\wj. dat"

long int bitoffset=0;
int readmask=0;
int codesize=0;
int initcodesize=0;
int maxcode=0, clearcode=0, eofcode=0;
int freecode=0, firstfree=0;
int bitmask=0;
int raw=1;
int local _flag _bit6=0;
int wide=0, deep=0;
long filepoint=0;
char pal [768];
FILE *fp;
FILE *fp1;
FILE *fp2;
int CUR _PAGE=0;

void readglobalhead (void);    //读文件整体表头信息
void readlocalhead (void);    //读文件局部表头信息
int readcode (void);    //读文件代码
void restore (void);    //转储图象数据
void exchange (void);    //转换图象数据
void showgif ();    //显示 GIF 图象文件
void drawgif (int, int, int);    //将 GIF 图象数据送视频缓冲区
void changepage (int);    //选择显示页面
void setdispmode (int);    //设置显示模式
void setVGApalette (char *pal);    //设置调色板信息

void main (void)
{
  char *fname;
  char ch;
  int i;
  int begin _end;
  int display=1;
  char yn;

while (display)
{
  bitoffset=0; readmask=0; codesize=0; initcodesize=0;
  maxcode=0; clearcode=0; eofcode=0; freecode=0;
  firstfree=0; bitmask=0; raw=1; local _flag _bit6=0;
  wide=0; deep=0; filepoint=0; CUR _PAGE=0;
  for (i=0; i <768; i++) pal [i] =0;

  printf (" please input GIF filename: ");
  scanf ("%s", fname);

fp=fopen (fname," r+b");
if (fp==NULL)
 { printf (" cann't open %s file!", fname);
  exit (0);
}

readglobalhead ();

fseek (fp, filepoint, 0);
fread (&begin _end, 1, 1, fp);
begin _end&=0x0ff;
i=0;
while (begin _end! =0x3B)
 {
  i++;
  printf ("%d please waiting... ! \n", i);
  if (begin _end==0x2c)
   {
    readlocalhead ();
    restore ();
    exchange ();
    printf (" please press anykey to display! \n");
    getch ();
    setdispmode (0x101);
    setVGApalette (pal);
    showgif ();
   }
 fseek (fp, filepoint, 0);
 fread (&begin _end, 1, 1, fp);
 begin _end&=0x0ff;
 getch ();
 fclose (fp);
 fclose (fp1);
 fclose (fp2);
 setdispmode (0x03);
 }
 printf (" continue to display GIF file? (Y/N) ");
 yn=getch ();
 printf ("%c\n", yn);
 if (yn=='y'||yn=-'Y') display-1;
 if (yn=='n'||yn--'N') display-0;
}
 printf (" BYE BYE!");
 fclose (fp);
 fclose (fp1);
 fclose (fp2);
} //end

void readglobalhead ()
{
char ch;
int i;
typedef struct {
     char gif _flag [6];
     int screen _width;
     int screen _depth;
     char global _flag;
```

```
    char ground_color;
    char byte0;
   } global_header;
global_header g;
int global_color_flag;
int color_num;
int fh, c, fh1;

  filepoint=0;
  fseek (fp, filepoint, 0);
  fgets (g.gif_flag, 7, fp);
  fh=strcmp (g.gif_flag," GIF89a");
  fh1=strcmp (g.gif_flag," GIF87a");
  if (fh==0||fh1==0)
   {
    printf (" VER %s file open success! \n", g.gif_flag);
  }
  else {printf (" open fail SORY! \n"); exit (0);}

  fread (&g.screen_width, 2, 1, fp);
  fread (&g.screen_depth, 2, 1, fp);
  fread (&g.global_flag, 1, 1, fp);
  fread (&g.ground_color, 1, 1, fp);
  fread (&g.byte0, 1, 1, fp);

  filepoint+=13;

  g.global_flag&=0x0ff;
  global_color_flag=g.global_flag & 0x80;
  g.ground_color&=0x0ff;
  g.byte0&=0x0ff;
  if (g.byte0! =0x00)
   {
    printf (" the file is error!");
    exit (0);
  }

  if (global_color_flag)
   {
    c=g.global_flag & 0x07;
    c+=1;
    color_num= (1 << (c);
    bitmask=color_num-1;
    color_num*=3;
    fread (pal, color_num, 1, fp);
    for (i=0; i <color_num;) pal [i++])) =2;
    filepoint+=color_num;
  }
}

void readlocalhead (void)
{
typedef struct {
    int image_left;
    int image_top;
    int image_wide;
    int image_deep;
    char local_flag;
  } local_header;
local_header l;
int local_color_flag;
int color_num;
int c, i;

  fread (&l.image_left, 2, 1, fp);
  fread (&l.image_top, 2, 1, fp);
  l.image_left&=0x0ffff;
  l.image_top&=0x0ffff;

  fread (&l.image_wide, 2, 1, fp);
  fread (&l.image_deep, 2, 1, fp);
  fread (&l.local_flag, 1, 1, fp);
  l.image_wide&=0x0ffff;
  l.image_deep&=0x0ffff;
  l.local_flag&=0x0ff;
  wide=l.image_wide;
  deep=l.image_deep;
  filepoint+=10;

  local_flag_bit6= ( (l.local_flag&0x40)? 1: 0);

  local_color_flag=l.local_flag&0x80;
  if (local_color_flag)
   {
    c=l.local_flag & 0x07;
    c+=1;
    color_num= (1 << (c);
    bitmask=color_num-1;
    color_num*=3;
    fread (pal, color_num, 1, fp);
    for (i=0; i <color_num;) pal [i++])) =2;
    filepoint+=color_num;
  }

  fread (&codesize, 1, 1, fp);
  codesize&=0x0ff;
  filepoint++;
//init
  clearcode= (1 << (codesize);
  eofcode=clearcode+1;
  freecode=firstfree=clearcode+2;
  bitmask=clearcode-1;

  codesize++;
  initcodesize=codesize;
  maxcode= (1 << (codesize);
  readmask=maxcode-1;
}

void restore (void)
{
int bytelongs=0;
int i;
char ch;

  fp1=fopen (s,"w+b");
  if (fp1==NULL)
   { printf (" cann't open source file!");
```

```
    exit (0);
  }
  fseek (fp, filepoint, 0);
  fread (&bytelongs, 1, 1, fp);
  bytelongs&=0x0ff;
  while (bytelongs! =0x00)
   {
    for (i=0; i <bytelongs; i++)
     {
      fread (&ch, 1, 1, fp);
      fwrite (&ch, 1, 1, fp1);
     }
    filepoint+= (bytelongs+1);
    fread (&bytelongs, 1, 1, fp);
    bytelongs&=0x0ff;
  }
  fwrite (&bytelongs, 1, 1, fp1);
  filepoint++;
}

void exchange (void)
{
char ch;
int i;
int prefix [4096], suffix [4096];
int outcode [1024], outcount=0;
int curcode, oldcode, code, finchar;

  fp1=fopen (sj," r+b");
  if (fp1==NULL)
   {
    printf (" cann't open source file!");
    exit (0);
  }
  fp2=fopen (wj," w+b");
  if (fp1==NULL)
   {
    printf (" cann't open targeter file!");
    exit (0);
  }
code=readcode ();
while (! feof (fp1))
   {
    if (code==clearcode) {
       codesize=initcodesize;
       maxcode= (1 < <codesize);
       readmask=maxcode-1;
       freecode=firstfree;
       curcode=oldcode=code=readcode ();
       finchar=curcode&bitmask;
       fwrite (&finchar, 1, 1, fp2);
    }
  else {
       curcode=code;
       if (curcode) =freecode)
       {
       curcode=oldcode;
       outcode [outcount++] =finchar;
       }
       while (curcode> bitmask)
       {
       outcode [outcount++] =suffix [curcode];
       curcode=prefix [curcode];
       }

       finchar=curcode&bitmask;
       outcode [outcount++] =finchar;

       for (i=outcount-1; i) =0; i--)
       }
       fwrite (&outcode [i], 1, 1, fp2);
       }
       outcount=0;

       prefix [freecode] =oldcode;
       suffix [freecode] =finchar;
       oldcode=code;

       freecode++;
       if (freecode) =maxcode) {
        if (codesize <12)
         {
          codesize++;
          maxcode= (1 < <codesize);
          readmask=maxcode-1;
         }
       }
     } //else
     code=readcode ();
   } //while
}

int readcode ()
{
   long int rawcode, byteoffset;

   byteoffset= (bitoffset/8);
   fseek (fp1, byteoffset, 0);
   if (codesize <8)
     {
       fread (&rawcode, 2, 1, fp1);
       rawcode&=0x0ffff;
     }
   if (codesize> =8)
     {
       fread (&rawcode, 3, 1, fp1);
       rawcode&=0x0ffffff;
     }
   rawcode>> = (bitoffset%8);
   bitoffset+=codesize;
   return (rawcode&readmask);

}
void showgif ()
{
int color;
int xc=0, yc=0;
fp2=fopen (wj," r+b");
```

```
if (fp2==NULL) {printf (" cann't open targeter file!"); exit (0);
}
while (! feof (fp2))
{
  fread (&color, 1, 1, fp2);
  color&=0x0ff;
  drawgif (xc, yc, color);
  if (++xc==wide)
   {
   xc=0;
   if (! local_flag_bit6) yc++;
   else {
     switch (raw) {
       case 1:
         yc+=8;
         if (yc) =deep) {
           raw++;
           yc=4;
          }
         break;
       case 2:
         yc+=8;
         if (yc) =deep) {
           raw++;
           yc=2;
          }
         break;
       case 3:
         yc+=4;
         if (yc) =deep) {
           raw++;
           yc=1;
         }
         break;
       case 4:
         yc+=2;
         break;
       default:
         break;
       }
     }
   }
}
}

void drawgif (int xc, int yc, int col)
{
char far *vedio= (char far *) (0xa0000000);
int cur_page=CUR_PAGE;
int pos;

pos=yc*640+xc;
asm jnc lable1
_DX++;
lable1: {
  if (_DX==cur_page) vedio [pos] =col;
  else changepage (_DX);
  }
}

void setdispmode (int mode)
{
  union REGS regs;
  if (mode) =0x100)
     {
     regs. x. ax=0x4f02;
     regs. x. bx=mode;
     int86 (0x10, &regs, &regs);
     }
  else
   {
     regs. h. ah=0;
     regs. h. al=mode;
     int86 (0x10, &regs, &regs);
   }
}

void setVGApalette (char *pat)
   {
  union REGS r;
  struct SREGS s;

  r. x. ax=0x1012;
  r. x. bx=0;
  r. x. cx=256;
  r. x. dx=FP_OFF (pat);
  s. es=FP_SEG (pat);
  int86x (0x10, &r, &r, &s);
  }

void changepage (int page)
{
CUR_PAGE=page;
asm {
     mov bx, 0
     mov dx, page
     mov ax, 0x4f05
     int 0x10
   };
}
```

参考文献

1. 李振辉，李仁和编著. 探索图像文件的奥妙. 清华大学出版社

2. 林福宗改编. 图像文件格式下—windows 编程. 清华大学出版社

3. 宫小玉. GIF 文件格式分析. 计算机世界月刊 (1993. 12)

（收稿日期：1998 年 4 月 15 日）