XPM 文件格式及 XPM 库函数

BG6RDF

随着 Linux 应用开发的普及，越来越多的开发者接触到了 X Window 系统。X Window 的开发又往往涉及图标等图像文件的操作。本文介绍了 X Window 系统中常用的 XPM 图像文件格式，及 XPM 函数库的使用。

XPM 是 X Window 系统中常用的图像文件格式，特别适合于图标（ICON）等小图像的使用。而 XPM 函数库（Library）定义了读写和显示 XPM 文件的函数。

一．XPM 的基本结构

一个 XPM 文件的基本结构如下：

```
/* XPM */
static char* <pixmap_name>[] = {
“Values-string”,
“Colors-strings”,
“Pixels-strings”,
“Extensions-strings”,
};
```

XPM 文件遵守 C 语言语法，可包括 C 语言风格的注释。正因如此可在 C/C++语言源程序中直接引用(include)XPM 文件。XPM 文件必须有一个如上所示的/* XPM */文件头。

其中“Values-string”的格式是：“width height ncolors cpp [x_hotspot y_hotspot] [XPMEXT]”。Width 是图像宽度，height 是图像高度(width 和 height 的单位都是像素-pixel)，ncolors 是颜色数，cpp（characters per pixel）是颜色定义中每个像素的字符数。X_hotspot 和 y_hotspot 是可选的，它们指定热点的坐标。如果存在”Extensions-String”才需要 XPMEXT。

“Colors-strings”的格式是”chars key value…”。Chars 是代表颜色的字符，它的大小由 cpp 指定。Key 用以指定 value 的类型，包括下列几种：m-单色，c-彩色，g4-四级灰度，g-多于 4 级的灰度，s-标识符。value 用以指定颜色（颜色名可在*****中查询），它可以是颜色名或以“#”开头的 RGB 值或以“%”开头的 HSV 值(，其中 None 表示透明。

“Pixels-strings”包含了图像的实际定义，它包含的字符串数量由 height 指定，每个字符串的有 width*cpp 个字符，而每个 cpp 长度的字符必须在“Colors-strings”中已定义。

“Extensions-strings”存放由应用程序定义的扩展信息。

下面是一个非常简单的 XPM 文件的例子：

```
/* XPM */
static char * bullet_xpm[] = {
/* width height number_of_colors chars_per_pixel */
/* colors */
  “25 25 2 1”,
  “X c black”,
  “. c red”,
/* “.”表示红色，“X”表示黑色。*/
/* pixels */
  “XXXXXXXXXXXXXXXXXXXXXXXXX”,
```

“XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX”,

“XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX”,

“XXXXXXXXXX.....XXXXXXXXXX”,

“XXXXXXXX.....XXXXXXXX”,

“XXXXXX.....XXXXXX”,

“XXXXX.....XXXXX”,

“XXXXX.....XXXXX”,

“XXXX.....XXXX”,

“XXXX.....XXXX”,

“XXX.....XXX”,

“XXX.....XXX”,

“XXX.....XXX”,

“XXX.....XXX”,

“XXXX.....XXXX”,

“XXXX.....XXXX”,

“XXXXX.....XXXXX”,

“XXXXX.....XXXXX”,

“XXXXXX.....XXXXXX”,

“XXXXXXXX.....XXXXXXXX”,

“XXXXXXXXXX.....XXXXXXXXXX”,

“XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX”,

“XXXXXXXXXXXXXXXXXXXXXXXXX”,

“XXXXXXXXXXXXXXXXXXXXXXXXX”};

这个 XPM 文件图形是：

但是这个图标在单色显示器上显示成完全黑色。因为红色在单显上会被转换为黑色。为避免这个问题，可通过修改“.”的颜色定义强行将红色转换为白色，即：“. c red m white”。这里用到了多个 key/value 对，即彩显上是红色，单显上是白色。

在XPM文件中，也可以定义颜色标识符名。例如我们把“.”的颜色定义成“. c red s Foreground”，表示Foreground是这个颜色的标识符名，就可以动态地将Foreground的颜色定义为任何颜色。如果我们把“X”的颜色定义成“X c none”，那么上图中黑色的部分就成为透明的。具体用法下面将讨论。

二．XPM库(library)

XPM库提供了Xlib级操作XPM文件的C语言函数和数据结构。使用这个库可以实现显示XPM文件中定义的图像、将图像保存到XPM文件中。使用XPM库编写程序时，应包含头文件xpm.h，在link时应指定-lXpm参数。

1.显示XPM文件

Xlib中图像处理通过Pixmap，XPM库中提供了由XPM文件生成Pixmap的函数。常用的是int XpmReadFileToPixmap(*display, d, filename, pixmap_return, shapemask_return, attributes*)。这个函数。Pixmap生成后就可以使用Xlib中所有处理Pixmap的函数进行图像处理了，如XcopyArea等。

2.Pixmap属性

XpmReadFileToPixmap函数的最后一个参数是struct XpmAttributes类型的Pixmap属性，我们可在创建前通过该参数设置Pixmap的一些属性，或在创建Pixmap后取得它的一些属性，如长宽等。如同Xlib中GC(Graphics Context图形上下文)一样，用该参数设置Pixmap的属性，必须提供一个掩码，但区别是这个掩码不是一个参数，而是attributes参数中的成员变量valuemask。在调用XpmReadFileToPixmap函数时，如果最后一个参数是0或空指针，表示不使用attributes参数，但如果一旦使用该参数，那么valuemask成员必须初始化。

3.颜色的重载和透明

如果在XPM文件中定义了颜色标识符名或透明色，就可以通过XpmAttributes参数进行颜色的重载或设置透明色。方法是通过XpmColorSymbol结构设置颜色标识符的实际色彩，接着将这个结构赋予XpmAttributes结构的colorsymbol变量，设置valuemask后，调用XpmReadFileToPixmap函数。

4．更多的功能

XPM库除了上面提到的几个功能外，还有诸如定义不规则（非矩形）窗口，进行图像转换等功能，本文仅涉及上述几个基本的功能。如果需要进一步了解，文后的参考网站中有更详细的介绍。当然，对能够对XPM文件进行处理的函数库也有很多，例如Imlib不仅能处理XPM文件，而且能够处理很多别的图像文件格式，但它们的使用都比XPM库复杂。

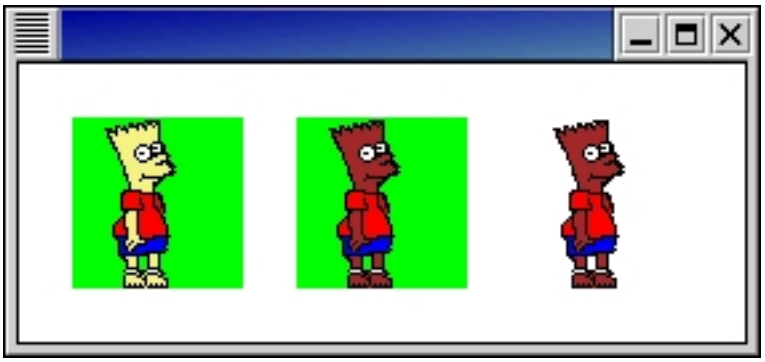
三．举例

为演示XPM库对XPM文件的基本操作，本文列出了一个简单的例子。这个程序演示了

XPM文件的显示、颜色重载、透明色处理。首先通过XPM库将XPM文件装入第一个Pixmap中，并根据返回的XpmAttributes确定图像和窗口的大小；将图像人物的肤色换成棕色后装入第二个Pixmap，完成颜色的重载；将背景色换成窗口用的背景色白色，再次装入第三个Pixmap。在绘制窗口时通过Xlib的XCopyArea函数将前面创建的三个Pixmap复制到窗口中。

如果在C语言源程序中用#include指令引用XPM文件，可以通过XpmCreatePixmapFromData函数根据图像数据生成Pixmap。Pixmap的释放应使用用Xlib提供的XfreePixmap函数。

这个程序并完全基于Xlib和XPM库，在Redhat 6.2和Mandrake 8.1下调试通过。编译命令是：gcc -o xpmtest -L/usr/X11R6/lib -lX11 -lXpm xpmtest.c。输出结果见下图。



参考网站：http://koala.ilog.fr/ftp/pub/xpm/

程序源码：

```
/* XPM Library test program */
/* Written by Tony Zheng     */

#include <X11/Xlib.h>
#include <X11/xpm.h>
#include <stdio.h>

#define filename "bart.xpm"

int main(int argc, char ** argv)
{
  Display *display;
  int screennum;
  Window win;
  XEvent report;
  GC gc;
  Pixmap color1pixmap, color2pixmap, transparentpixmap;
  XpmAttributes attributes;
  int status;
  unsigned int pixmapwidth, pixmapheight, width, height;
  XGCValues values;
  XpmColorSymbol symbol;
  XpmColorSymbol symbols[2];
```

```
 /*  连接 X server */
 display=XOpenDisplay("");
 if (display==NULL) {
  fprintf(stderr, "%s:cannot open display.\n", argv[0]);
  exit(-1);
 }
 screennum=DefaultScreen(display);

/*生成第一个 Pixmap */
 attributes.valuemask=0;
 status=XpmReadFileToPixmap(display, RootWindow(display, screennum),
                            filename, &color1pixmap, 0, &attributes);
 if (status!=XpmSuccess) {
      fprintf(stderr, "Pixmap 1 XpmError:%s\n", XpmGetErrorString(status));
      exit(-1);
 }

 /*  获得 pixmap 尺寸，计算窗口尺寸  */
 pixmapwidth=attributes.width;
 pixmapheight=attributes.height;
 width=pixmapwidth*3+80;
 height=pixmapheight+40;

/*替换肤色，生成第二个 pixmap */
 symbol.name="Skin";
 symbol.value="brown";
 attributes.colorsymbols=&symbol;
 attributes.numsymbols=1;
 attributes.valuemask=XpmColorSymbols;
 status=XpmReadFileToPixmap(display, RootWindow(display, screennum),
                            filename, &color2pixmap, 0, &attributes);

 if (status!=XpmSuccess) {
      fprintf(stderr, "Pixmap 2 XpmError:%s\n", XpmGetErrorString(status));
      exit(-1);
 }

/*替换肤色和背景色，生成第三个 pixmap */
 symbols[0].name="Skin";
 symbols[0].value="brown";
 symbols[1].name="None";
 symbols[1].value="white";
 attributes.colorsymbols=symbols;
```

```
attributes.numsymbols=2;
attributes.valuemask=XpmColorSymbols;
status=XpmReadFileToPixmap(display, RootWindow(display, screennum),
                        filename, &transparentpixmap, 0, &attributes);

if (status!=XpmSuccess) {
      fprintf(stderr, "Pixmap 3 XpmError:%s\n", XpmGetErrorString(status));
      exit(-1);
}

/* 创建白色背景窗口 */
win=XCreateSimpleWindow(display, RootWindow(display, screennum),
       100, 100, width, height, 4, BlackPixel(display, screennum),
       WhitePixel(display, screennum));

/* 选择希望处理的窗口事件 */
XSelectInput(display, win, ExposureMask | KeyPressMask |
                    ButtonPressMask);

/* 创建画图时使用的图形上下文 */
gc=XCreateGC(display, win, 0, &values);

/* 显示窗口 */
XMapWindow(display, win);

/* 事件循环 */
while(1) {
   XNextEvent(display, &report);
   switch (report.type) {
      case Expose:
            /* 仅当收到最后一个 Expose 事件时 */
            if (report.xexpose.count!=0) break;
        XCopyArea(display, color1pixmap, win, gc, 0, 0,
            pixmapwidth, pixmapheight, 20, 20);
        XCopyArea(display, color2pixmap, win, gc, 0, 0,
            pixmapwidth, pixmapheight, 40+pixmapwidth, 20);
        XCopyArea(display, transparentpixmap, win, gc, 0, 0,
            pixmapwidth, pixmapheight, 60+pixmapwidth*2, 20);
            break;
           /*如果任何一个键按下，或鼠标按钮按下，程序退出 */
      case ButtonPress:
            /* Trickle down into KeyPress(no break) */
      case KeyPress:
            XFreeGC(display, gc);
```

```
                XCloseDisplay(display);
                exit(1);
            default:
                break;
        } /* end switch */
    } /* end while */
  return (1);
}    /* end of main() */
```