

\$SPAD/src/lib pixmap.c

The Axiom Team

July 31, 2014

**Abstract**

## Contents

|   |                            |   |
|---|----------------------------|---|
| 1 | MAC OSX zopen redefinition | 3 |
| 2 | License                    | 3 |

## 1 MAC OSX zopen redefinition

On the [[MAC OSX]] platform they defined [[zopen]]. Since the function is only used in this file we simply rename it to [[zzopen]].

— mac zopen redefinition 1 —

```
FILE *  
zzopen(char *file, char * mode)
```

—————

— mac zopen redefinition 2 —

```
file = zopen(filename, "r");
```

—————

## 2 License

```
/*  
Copyright (c) 1991-2002, The Numerical ALgorithms Group Ltd.  
All rights reserved.
```

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of The Numerical ALgorithms Group Ltd. nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

```

*/

      ____ * ____

#include <X11/Xlib.h>
#include <X11/Xutil.h>
#include <X11/Xos.h>
#include <stdlib.h>
#include <stdio.h>
#include <netinet/in.h>

#define yes 1
#define no 0

#include "spadcolors.h"

#include "pixmap.h1"
#include "halloc.h1"
#include "spadcolors.h1"

/* returns true if the file exists */

int
file_exists(char *file)
{
    FILE *f;

    if ((f = fopen(file, "r")) != NULL) {
        fclose(f);
        return 1;
    }
    return 0;
}

\getchunk{mac zopen redefinition 1}
{
    char com[512], zfile[512];

    if (file_exists(file))
        return fopen(file, mode);
    sprintf(zfile, "%s.Z", file);
    if (file_exists(zfile)) {
        sprintf(com, "gunzip -c %s.Z 2>/dev/null", file);
        return popen(com, mode);
    }
}

```

```

    }
    return NULL;
}
#endif OLD

/*****
KF 6/14/90
write_pixmap_file(display, filename, pm, width, height)
    and
write_pixmap_file_xy(display, filename, pm, x, y, width, height)
has been merged into one function.

INPUT: display dsp, screen s, file name fn to write the file in,
       window id wid where pixmap is,
       upper left corner x, y of original pixmap,
       width and height of pixmap
OUTPUT: binary file with data
PURPOSE: write_pixmap_file gets the image structure of the input
         pixmap, convert the image data with the permutation color
         vector, writes the image structure out to filename.

Note that writing out a Z pixmap is 8x faster than XY pixmap.
This is because XY writes out each pixel value per plane, thus
number of bits; Z writes out each pixel, or 8 bits at a time.

The XY format may have been chosen for a reason -- I don't know.

*****/
void
write_pixmap_file(Display *dsp, int scr, char *fn,
                  Window wid, int x, int y, int width, int height)
{
    XImage *xi;
    FILE *file;
    int *permVector;
    int num;
    int num_colors;

    /* get color map and permutation vector */
    if ((num_colors = makePermVector(dsp, scr, (unsigned long **)&permVector)) < 0) {
        printf("num_colors < 0!!\n");
        exit(-1);
    }

    /* reads image structure in ZPixmap format */
    xi = XGetImage(dsp, wid, x, y, width, height, AllPlanes, ZPixmap);
    file = fopen(fn, "wb");
    if (file == NULL) {
        perror("opening pixmap file for write");
        exit(-1);
    }

```

```

    }

#define PUTW(a,b) putw(htonl(a),b)

    PUTW(xi->width, file);
    PUTW(xi->height, file);
    PUTW(xi->xoffset, file);
    PUTW(xi->format, file);
    PUTW(xi->byte_order, file);
    PUTW(xi->bitmap_unit, file);
    PUTW(xi->bitmap_bit_order, file);
    PUTW(xi->bitmap_pad, file);
    PUTW(xi->depth, file);
    PUTW(xi->bytes_per_line, file);
    PUTW(xi->bits_per_pixel, file);
    PUTW(xi->red_mask, file);
    PUTW(xi->green_mask, file);
    PUTW(xi->blue_mask, file);

    num = xi->bytes_per_line * height; /* total number of pixels in pixmap */

    /* store value from permutation */
    {
        int ii, jj;

        for (ii = 0; ii < width; ii++)
            for (jj = 0; jj < height; jj++) {
                XPutPixel(xi, ii, jj, permVector[(int) XGetPixel(xi, ii, jj)]);
            }
    }
    fwrite(xi->data, 1, num, file);
    fclose(file);
}

/*****
KF 6/14/90

INPUT: display, screen, filename to read the pixmap data from,
OUTPUT: ximage structure xi, width and height of pixmap
PURPOSE: read_pixmap_file reads an Ximage data structure from
the input file.
This routine can handle pixmaps of both XYPixmap and
ZPixmap. If a pixmap has ZPixmap format, then the image
data, read in as spadColor index, is converted to the
pixel value using spadColor.

Note that reading in Z format takes less space and time too.
*****/

```

```

int
read_pixmap_file(Display *display, int screen, char *filename,
                  XImage **xi, int *width, int *height)
{
    FILE *file;
    int wi, h, num, num_colors, read_this_time, offset;
    Colormap cmap;
    int ts;
    unsigned long *spadColors;

    /* colormap is necessary to call makeColors */
    cmap = DefaultColormap(display, screen);
    if ((num_colors = makeColors(display, screen, &cmap, &spadColors, &ts)) < 0) {
        return(-1);
    }
    \getchunk{mac zopen redefinition 2}
    if (file == NULL) {
        printf("couldn't open %s\n", filename);
        return BitmapOpenFailed;
    }
    #define GETW(f) ntohl(getw(f))
    *width = wi = GETW(file);
    *height = h = GETW(file);
    (*xi) = XCreateImage(display, DefaultVisual(display, screen),
                        DisplayPlanes(display, screen),
                        ZPixmap, 0, NULL, wi, h, 16, 0); /* handles both XY & Z */
    if ((*xi) == NULL) {
        fprintf(stderr, "Unable to create image\n");
        return(-1);
    }
    (*xi)->width = wi;
    (*xi)->height = h;
    (*xi)->xoffset = GETW(file);
    (*xi)->format = GETW(file);
    (*xi)->byte_order = GETW(file);
    (*xi)->bitmap_unit = GETW(file);
    (*xi)->bitmap_bit_order = GETW(file);
    (*xi)->bitmap_pad = GETW(file);
    (*xi)->depth = GETW(file);
    (*xi)->bytes_per_line = GETW(file);
    (*xi)->bits_per_pixel = GETW(file);
    (*xi)->red_mask = GETW(file);
    (*xi)->green_mask = GETW(file);
    (*xi)->blue_mask = GETW(file);

    /* program will bomb if XYPixmap is not allocated enough space */
    if ((*xi)->format == XYPixmap) {
        /* printf("picture is in XYPixmap format.\n"); */
        num = (*xi)->bytes_per_line * h * (*xi)->depth;
    }
}

```

```

else                                     /* ZPixmap */
    num = (*xi)->bytes_per_line * h;
    (*xi)->data = (void*)malloc(num, "Ximage data");

    offset = 0;
    while (offset < num) {
        read_this_time = fread((*xi)->data + offset, 1, num - offset, file);
        offset = offset + read_this_time;
    }
    fclose(file);

/*
 * pixmap data in ZPixmap format are spadColor indices; pixmap data in
 * XYPixmap format are pixel values
 */
if ((*xi)->format == ZPixmap) {

    int ii, jj;

    for (ii = 0; ii < wi; ii++)
        for (jj = 0; jj < h; jj++) {
            XPutPixel(*xi, ii, jj, spadColors[(int) XGetPixel(*xi, ii, jj)]);
        }

}

return 0;
}

#else /*OLD*/

#include "xpm.h"

int
read_pixmap_file(Display *display, int screen, char *filename,
                  XImage **xi, int *width, int *height)
{
    XpmAttributes attr;
    XImage *xireturn;

    attr.valuemask = 0;

    attr.bitmap_format=ZPixmap;           /* instead of XYPixmap */
    attr.valuemask |= XpmBitmapFormat;
    attr.valuemask |= XpmSize;             /* we want feedback on width,height */
    attr.valuemask |= XpmCharsPerPixel;    /* and cpp */
    attr.valuemask |= XpmReturnPixels;     /* and pixels, npixels */

```



```

attr.valuemask |= XpmReturnAllocPixels; /* and alloc_pixels, nalloc_pixels */
attr.exactColors = False;
attr.valuemask |= XpmExactColors;      /* we don't want exact colors*/
attr.closeness = 30000;
attr.valuemask |= XpmCloseness;        /* we specify closeness*/
attr.alloc_close_colors = False;
attr.valuemask |= XpmAllocCloseColors; /* we don't allocate close colors*/


XpmReadFileToImage(display,filename,xi,&xireturn, &attr );
*width= (*xi)->width;
*height=(*xi)->height;
#ifdef DEBUG
fprintf(stderr,"image file:%s\n",filename);
fprintf(stderr,"\twidth:%d\theight:%d\tcpp:%d\n",attr.width,attr.height,attr.cpp);
fprintf(stderr,"\tused/alloc'ed color pixels:%d/%d\n",attr.npixels,attr.nalloc_pixels);
#endif
return 0;
}

void
write_pixmap_file(Display *dsp, int scr, char *fn,
                  Window wid, int x, int y, int width,int height)
{
    XImage *xi;

    /* reads image structure in ZPixmap format */
    xi = XGetImage(dsp, wid, x, y, width, height, AllPlanes, ZPixmap);
    if (xi==0) return ;
    XpmWriteFileFromImage(dsp,fn,xi,0,0);
}

#endif

```

---

## References

- [1] nothing