

**NAME****libxdot** – parsing and deparsing of xdot operations**SYNOPSIS**`#include <graphviz/xdot.h>`

```

typedef enum {
    xd_none,
    xd_linear,
    xd_radial
} xdot_grad_type;

typedef struct {
    float frac;
    char* color;
} xdot_color_stop;

typedef struct {
    double x0, y0;
    double x1, y1;
    int n_stops;
    xdot_color_stop* stops;
} xdot_linear_grad;

typedef struct {
    double x0, y0, r0;
    double x1, y1, r1;
    int n_stops;
    xdot_color_stop* stops;
} xdot_radial_grad;

typedef struct {
    xdot_grad_type type;
    union {
        char* clr;
        xdot_linear_grad ling;
        xdot_radial_grad ring;
    } u;
} xdot_color;

typedef enum {
    xd_left, xd_center, xd_right
} xdot_align;

typedef struct {
    double x, y, z;
} xdot_point;

typedef struct {
    double x, y, w, h;
} xdot_rect;

typedef struct {
    int cnt;
    xdot_point* pts;

```

```

} xdot_polyline;

typedef struct {
    double x, y;
    xdot_align align;
    double width;
    char* text;
} xdot_text;

typedef struct {
    xdot_rect pos;
    char* name;
} xdot_image;

typedef struct {
    double size;
    char* name;
} xdot_font;

typedef enum {
    xd_filled_ellipse, xd_unfilled_ellipse,
    xd_filled_polygon, xd_unfilled_polygon,
    xd_filled_bezier, xd_unfilled_bezier,
    xd_polyline,      xd_text,
    xd_fill_color,    xd_pen_color, xd_font, xd_style, xd_image,
    xd_grad_fill_color, xd_grad_pen_color
} xdot_kind;

typedef enum {
    xop_ellipse,
    xop_polygon,
    xop_bezier,
    xop_polyline,    xop_text,
    xop_fill_color,  xop_pen_color, xop_font, xop_style, xop_image,
    xop_grad_fill_color, xop_grad_pen_color
} xop_kind;

typedef struct _xdot_op xdot_op;
typedef void (*drawfunc_t)(xdot_op*, int);
typedef void (*freefunc_t)(xdot_op*);

struct _xdot_op {
    xdot_kind kind;
    union {
        xdot_rect ellipse;    /* xd_filled_ellipse, xd_unfilled_ellipse */
        xdot_polyline polygon; /* xd_filled_polygon, xd_unfilled_polygon */
        xdot_polyline polyline; /* xd_polyline */
        xdot_polyline bezier; /* xd_filled_bezier, xd_unfilled_bezier */
        xdot_text text;        /* xd_text */
        xdot_image image;      /* xd_image */
        char* color;           /* xd_fill_color, xd_pen_color */
        xdot_color grad_color; /* xd_grad_fill_color, xd_grad_pen_color */
        xdot_font font;        /* xd_font */
        char* style;           /* xd_style */
    };
};

```

```

    } u;
    drawfunc_t drawfunc;
};

#define XDOT_PARSE_ERROR 1

typedef struct {
    int cnt;
    int sz;
    xdot_op* ops;
    freefunc_t freefunc;
    int flags;
} xdot;

xdot* parseXDotF (char*, drawfunc_t opfns[], int sz);
xdot* parseXDot (char*);
char* sprintXDot (xdot*);
void fprintfXDot (FILE*, xdot*);
void freeXDot (xdot*);

xdot_grad_type colorType (char*);
xdot_color* parseXDotColor (char*);
void freeXDotColor (xdot_color*);

```

## DESCRIPTION

*libxdot* provides support for parsing and deparsing graphical operations specified by the *xdot* language.

### Types

#### **xdot**

This encapsulates a series of *cnt* xdot operations, stored in the array pointed to by *ops*. The *sz* indicates the size of each item stored in *ops*. If the user sets the *freefunc* field, this function will be called on each item in *ops* during *freeXDot* before the library does its own clean up of the item. This allows the user to free any resources stored in the item by using an expansion of the *xdot\_op* structure.

#### **xdot\_op**

A value of this type represents one xdot operation. The operation is specified by the *kind* field. The corresponding data is stored in the union *u*, with the subfield associated with a given *kind* indicated by the comments.

The *drawfunc* field allows the user to attach a drawing-specific function to the operation, providing an object-based interface. These functions can be automatically attached during parsing by providing a non-NULL second argument to **parseXDotF**.

#### **xop\_kind**

This type provides an enumeration of the allowed xdot operations. See <http://www.graphviz.org/doc/info/output.html#d:xdot> for the specific semantics associated with each operation.

#### **xdot\_rect**

This represents a rectangle. For ellipses, the *x* and *x* fields represent the center of the rectangle, and *w* and *h* give the half-width and half-height, respectively. For images, (*x*,*y*) gives the lower left corner of the rectangle, and *w* and *h* give the width and height, respectively.

#### **xdot\_polyline**

This type encapsulates a series of *cnt* points.

#### **xdot\_text**

A value of this type corresponds to printing the string *text* using the baseline point (*x*,*y*). The *width* field gives an approximation of how wide the printed string will be using the current font and font size. The

*align* field indicates how the text should be horizontally aligned with the point (x,y).

#### **xdot\_image**

This denotes the insertion of an image. The image source is given by *name*. The image is to be placed into the rectangle *pos*.

#### **xdot\_font**

The fields give the name and size, in points, of a font.

#### **xdot\_align**

This enumeration type corresponds to the xdot alignment values -1, 0 and 1 used with the text operator, or '\l', '\n' and '\r' used in dot text.

### **Functions**

#### **xdot\* parseXDotF (char \*str, drawfunc\_t\* opfns, int sz)**

Parses the string *str* as a sequence of xdot operations and returns a pointer to the resulting *xdot* structure. The function parses as many xdot operations as it can. If some unknown or incorrect input was encountered in *str*, the *ops* and *cnt* fields will reflect the operations parsed before the error, and the *XDOT\_PARSE\_ERROR* bit will be set in the *flags* field. The function returns NULL if it cannot parse anything.

If *sz* is non-zero, it is assumed to be the size of some structure type containing *xdot\_op* as a prefix. In this case, the elements in the array pointed to by *ops* will each have size *sz*.

If *opfns* is non-zero, it is taken to be any array of functions indexed by *xop\_kind*. During parsing, the *drawfunc* member of *xop\_op* will be set to the corresponding function in *opfns*.

#### **xdot\* parseXDot (char \*str)**

This is equivalent to *parseXDotF(str, 0, 0)*.

#### **void freeXDot (xdot\* xp)**

This frees the resources associated with the argument. If *xp* is NULL, nothing happens.

#### **extern char\* sprintXDot (xdot\* xp)**

#### **extern void fprintfXDot (FILE\* fp, xdot\* xp)**

These two functions deparse the argument xdot structure, producing a string representation. *fprintXDot* writes the output onto the open stream *fp*; *sprintXDot* returns a heap-allocated string.

The color string with fill and draw operations can encode linear and radial gradients. These values are parsed automatically by **parseXDotF** or **parseXDot**, with *xdot\_op* having kind *xd\_grad\_pen\_color* or *xd\_grad\_fill\_color* and the value is stored in *grad\_color*.

For an application that handles its own parsing of xdot, the library provides three helper functions.

#### **xdot\_grad\_type colorTypeXDot (char \*str)**

returns the color type described by the input string.

#### **char\* parseXDotColor (char \*str, xdot\_color\* clr)**

attempts to parse the string *str* as a color value, storing the result in *clr*. It returns NULL on failure.

#### **void freeXDotColor (xdot\_color\* cp)**

This frees the resources associated with a value of type *xdot\_color*.

### **BUGS**

Although some small checking is done on the *sz* argument to *parseXDotF*, it is assumed it is a valid value from *sizeof* applied to some structure type containing *xdot\_op* as its first field. There can be no validation of the *opfns* argument.

### **AUTHORS**

Emden R. Gansner (erg@research.att.com).