NAME

XLoadFont, XQueryFont, XLoadQueryFont, XFreeFont, XGetFontProperty, XUnloadFont, XCharStruct, XFontProp, XChar2b, XFontStruct - load or unload fonts and font metric structures

SYNTAX

Font XLoadFont(Display *_display_, _Xconst char *_name_);

XFontStruct *XQueryFont(Display *_display_, XID _font_ID_);

XFontStruct *XLoadQueryFont(Display *_display_, _Xconst char *_name_);

int XFreeFont(Display *_display_, XFontStruct *_font_struct_);

Bool XGetFontProperty(XFontStruct *_font_struct_, Atom _atom_, unsigned long *_value_return_);

int XUnloadFont(Display *_display_, Font _font_);

ARGUMENTS

atom	Specifies the atom for the property name you want returned.
display	Specifies the connection to the X server.
font	Specifies the font.
font_ID	Specifies the font ID or the **GContext** ID.
font_struct	Specifies the storage associated with the font.
gc	Specifies the GC.
name	Specifies the name of the font, which is a null-terminated string.
value_return	Returns the value of the font property.

DESCRIPTION

The **XLoadFont** function loads the specified font and returns its associated font ID. If the font name is not in the Host Portable Character Encoding, the result is implementation-dependent. Use of uppercase or lowercase does not matter. When the characters "?" and "" are used in a font name, a pattern match is performed and any matching font is used. In the pattern, the "?" character will match any single character, and the "*" character will match any number of characters. A structured format for font

names is specified in the X Consortium standard *X Logical Font Description Conventions*. If **XLoadFont** was unsuccessful at loading the specified font, a **BadName** error results. Fonts are not associated with a particular screen and can be stored as a component of any GC. When the font is no longer needed, call **XUnloadFont**.

XLoadFont can generate **BadAlloc** and **BadName** errors.

The **XQueryFont** function returns a pointer to the **XFontStruct** structure, which contains information associated with the font. You can query a font or the font stored in a GC. The font ID stored in the **XFontStruct** structure will be the **GContext** ID, and you need to be careful when using this ID in other functions (see **XGContextFromGC**). If the font does not exist, **XQueryFont** returns NULL. To free this data, use **XFreeFontInfo**.

XLoadQueryFont can generate a **BadAlloc** error.

The **XLoadQueryFont** function provides the most common way for accessing a font. **XLoadQueryFont** both opens (loads) the specified font and returns a pointer to the appropriate **XFontStruct** structure. If the font name is not in the Host Portable Character Encoding, the result is implementation-dependent. If the font does not exist, **XLoadQueryFont** returns NULL.

The **XFreeFont** function deletes the association between the font resource ID and the specified font and frees the **XFontStruct** structure. The font itself will be freed when no other resource references it. The data and the font should not be referenced again.

XFreeFont can generate a **BadFont** error.

Given the atom for that property, the **XGetFontProperty** function returns the value of the specified font property. **XGetFontProperty** also returns **False** if the property was not defined or **True** if it was defined. A set of predefined atoms exists for font properties, which can be found in **X11/Xatom.h**. This set contains the standard properties associated with a font. Although it is not guaranteed, it is likely that the predefined font properties will be present.

The **XUnloadFont** function deletes the association between the font resource ID and the specified font. The font itself will be freed when no other resource references it. The font should not be referenced again.

XUnloadFont can generate a **BadFont** error.

STRUCTURES

The **XFontStruct** structure contains all of the information for the font and consists of the font-specific

information as well as a pointer to an array of **XCharStruct** structures for the characters contained in the font. The **XFontStruct**, **XFontProp**, and **XCharStruct** structures contain:

```
typedef struct {
        short lbearing;   /* origin to left edge of raster */
        short rbearing;   /* origin to right edge of raster */
        short width;      /* advance to next char's origin */
        short ascent;     /* baseline to top edge of raster */
        short descent;    /* baseline to bottom edge of raster */
        unsigned short attributes;          /* per char flags (not predefined) */
} XCharStruct;

typedef struct {
        Atom    name;
        unsigned long card32;
} XFontProp;

typedef struct {  /* normal 16 bit characters are two bytes */
    unsigned char byte1;
    unsigned char byte2;
} XChar2b;

typedef struct {
        XExtData *ext_data;     /* hook for extension to hang data */
        Font fid;          /* Font id for this font */
        unsigned direction;       /* hint about the direction font is painted */
        unsigned min_char_or_byte2;    /* first character */
        unsigned max_char_or_byte2;    /* last character */
        unsigned min_byte1;     /* first row that exists */
        unsigned max_byte1;     /* last row that exists */
        Bool all_chars_exist;     /* flag if all characters have nonzero size */
        unsigned default_char;  /* char to print for undefined character */
        int n_properties;          /* how many properties there are */
        XFontProp *properties; /* pointer to array of additional properties */
        XCharStruct min_bounds;         /* minimum bounds over all existing char */
        XCharStruct max_bounds;         /* maximum bounds over all existing char */
        XCharStruct *per_char; /* first_char to last_char information */
        int ascent;         /* logical extent above baseline for spacing */
        int descent;        /* logical decent below baseline for spacing */
} XFontStruct;
```

X supports single byte/character, two bytes/character matrix, and 16-bit character text operations. Note that any of these forms can be used with a font, but a single byte/character text request can only specify a single byte (that is, the first row of a 2-byte font). You should view 2-byte fonts as a two-dimensional matrix of defined characters: byte1 specifies the range of defined rows and byte2 defines the range of defined columns of the font. Single byte/character fonts have one row defined, and the byte2 range specified in the structure defines a range of characters.

The bounding box of a character is defined by the **XCharStruct** of that character. When characters are absent from a font, the default_char is used. When fonts have all characters of the same size, only the information in the **XFontStruct** min and max bounds are used.

The members of the **XFontStruct** have the following semantics:

- The direction member can be either **FontLeftToRight** or **FontRightToLeft**. It is just a hint as to whether most **XCharStruct** elements have a positive (**FontLeftToRight**) or a negative (**FontRightToLeft**) character width metric. The core protocol defines no support for vertical text.

- If the min_byte1 and max_byte1 members are both zero, min_char_or_byte2 specifies the linear character index corresponding to the first element of the per_char array, and max_char_or_byte2 specifies the linear character index of the last element.

  If either min_byte1 or max_byte1 are nonzero, both min_char_or_byte2 and max_char_or_byte2 are less than 256, and the 2-byte character index values corresponding to the per_char array element N (counting from 0) are:

  byte1 = N/D + min_byte1
  byte2 = N\D + min_char_or_byte2

  where:

  D = max_char_or_byte2 - min_char_or_byte2 + 1
  / = integer division
  \\ = integer modulus

- If the per_char pointer is NULL, all glyphs between the first and last character indexes inclusive have the same information, as given by both min_bounds and max_bounds.

- If all_chars_exist is **True**, all characters in the per_char array have nonzero bounding boxes.

- The default_char member specifies the character that will be used when an undefined or nonexistent character is printed. The default_char is a 16-bit character (not a 2-byte character). For a font using 2-byte matrix format, the default_char has byte1 in the most-significant byte and

byte2 in the least significant byte. If the default_char itself specifies an undefined or nonexistent character, no printing is performed for an undefined or nonexistent character.

- The min_bounds and max_bounds members contain the most extreme values of each individual **XCharStruct** component over all elements of this array (and ignore nonexistent characters). The bounding box of the font (the smallest rectangle enclosing the shape obtained by superimposing all of the characters at the same origin [x,y]) has its upper-left coordinate at:

  [x + min_bounds.lbearing, y - max_bounds.ascent]

  Its width is:

  max_bounds.rbearing - min_bounds.lbearing

  Its height is:

  max_bounds.ascent + max_bounds.descent

- The ascent member is the logical extent of the font above the baseline that is used for determining line spacing. Specific characters may extend beyond this.

- The descent member is the logical extent of the font at or below the baseline that is used for determining line spacing. Specific characters may extend beyond this.

- If the baseline is at Y-coordinate y, the logical extent of the font is inclusive between the Y-coordinate values (y - font.ascent) and (y + font.descent - 1). Typically, the minimum interline spacing between rows of text is given by ascent + descent.

For a character origin at [x,y], the bounding box of a character (that is, the smallest rectangle that encloses the character's shape) described in terms of **XCharStruct** components is a rectangle with its upper-left corner at:

[x + lbearing, y - ascent]

Its width is:

rbearing - lbearing

Its height is:

ascent + descent

The origin for the next character is defined to be:

[x + width, y]

The lbearing member defines the extent of the left edge of the character ink from the origin. The rbearing member defines the extent of the right edge of the character ink from the origin. The ascent member defines the extent of the top edge of the character ink from the origin. The descent member defines the extent of the bottom edge of the character ink from the origin. The width member defines the logical width of the character.

DIAGNOSTICS

BadAlloc The server failed to allocate the requested resource or server memory.

BadFont A value for a Font or GContext argument does not name a defined Font.

BadName A font or color of the specified name does not exist.

SEE ALSO

XCreateGC(3), XListFonts(3), XSetFontPath(3)
Xlib - C Language X Interface