NAME

Xcursor - Cursor management library

SYNOPSIS

#include <X11/Xcursor/Xcursor.h>

DESCRIPTION

Xcursor is a simple library designed to help locate and load cursors. Cursors can be loaded from files or memory. A library of common cursors exists which map to the standard X cursor names. Cursors can exist in several sizes and the library automatically picks the best size.

FUNCTIONAL OVERVIEW

Xcursor is built in a couple of layers; at the bottom layer is code which can load cursor images from files. Above that is a layer which locates cursor files based on the library path and theme. At the top is a layer which builds cursors either out of an image loaded from a file or one of the standard X cursors. When using images loaded from files, Xcursor prefers to use the Render extension CreateCursor request if supported by the X server. Where not supported, Xcursor maps the cursor image to a standard X cursor and uses the core CreateCursor request.

CURSOR FILES

Xcursor defines a new format for cursors on disk. Each file holds one or more cursor images. Each cursor image is tagged with a nominal size so that the best size can be selected automatically. Multiple cursors of the same nominal size can be loaded together; applications are expected to use them as an animated sequence.

Cursor files are stored as a header containing a table of contents followed by a sequence of chunks. The table of contents indicates the type, subtype and position in the file of each chunk. The file header looks like:

magic:	CARD32 "Xcur" (0x58, 0x63, 0x75, 0x72)
header:	CARD32 bytes in this header
version:	CARD32 file version number
ntoc:	CARD32 number of toc entries
toc:	LISTofTOC table of contents

Each table of contents entry looks like:

type:	CARD32 entry type
subtype:	CARD32 type-specific label - size for images
position:	CARD32 absolute byte position of table in file

Each chunk in the file has set of common header fields followed by additional type-specific fields:

header:	CARD32 bytes in chunk header (including type-specific fields)
type:	CARD32 must match type in TOC for this chunk
subtype:	CARD32 must match subtype in TOC for this chunk
version:	CARD32 version number for this chunk type

There are currently two chunk types defined for cursor files; comments and images. Comments look like:

header:	20 Comment headers are 20 bytes
type:	0xfffe0001 Comment type is 0xfffe0001
subtype:	{ 1 (COPYRIGHT), 2 (LICENSE), 3 (OTHER) }
version:	1
length:	CARD32 byte length of UTF-8 string
string:	LISTofCARD8 UTF-8 string

Images look like:

header:	36 Image headers are 36 bytes
type:	0xfffd0002 Image type is 0xfffd0002
subtype:	CARD32 Image subtype is the nominal size
version:	1
width:	CARD32 Must be less than or equal to 0x7fff
height:	CARD32 Must be less than or equal to 0x7fff
xhot:	CARD32 Must be less than or equal to width
yhot:	CARD32 Must be less than or equal to height
delay:	CARD32 Delay between animation frames in milliseconds
pixels:	LISTofCARD32 Packed ARGB format pixels

THEMES

Xcursor (mostly) follows the freedesktop.org spec for theming icons. The default search path it uses is

~/.local/share/icons, ~/.icons, /usr/local/share/icons, /usr/local/share/pixmaps

Within each of these directories, it searches for a directory using the theme name:

- Within the theme directory, it looks for cursor files in the "cursors" subdirectory.

  Xcursor looks for a specific file, which must be one of the cursor *shape* names, e.g., as used in

XcursorLibraryLoadImage or XcursorLibraryShape.

- If it finds no matching cursor file in the "cursors" subdirectory, Xcursor next looks for an "index.theme" file in each theme directory to look for inherited themes. Those are lines in this format:

  Inherits = *theme-name*

  Xcursor uses the first inherited *theme-name*, ignoring others which may exist in a given "index.theme" file. If it finds an inherited theme, Xcursor searches along the path to use that as well. Xcursor ignores other *keys* in the "index.theme" file, including "Name" (i.e., the name which a graphical application may use as the *presentation name*).

  More than one *theme-name* may be listed on the **Inherits=** line. The freedesktop.org spec states that list items are separated by commas. Xcursor also accepts semicolon, but translates both to colon when searching the path. Xcursor expects only one **Inherits=** line; the freedesktop.org spec is unclear whether multiple keys are allowed.

If no theme is set, or if no cursor is found for the specified theme anywhere along the path, Xcursor checks the "default" theme.

When Xcursor finds a cursor file, it stops searching. It always uses the first cursor file found while searching along the path.

DATATYPES

XcursorImage

holds a single cursor image in memory. Each pixel in the cursor is a 32-bit value containing ARGB with A in the high byte.

```
typedef struct _XcursorImage {
    XcursorDim      size;       /* nominal size for matching */
    XcursorDim      width;      /* actual width */
    XcursorDim      height;     /* actual height */
    XcursorDim      xhot;       /* hot spot x (must be inside image) */
    XcursorDim      yhot;       /* hot spot y (must be inside image) */
    XcursorPixel   *pixels;     /* pointer to pixels */
} XcursorImage;
```

XcursorImages

holds multiple XcursorImage structures. They are all freed when the XcursorImages is freed in

XcursorImagesDestroy.

```
typedef struct _XcursorImages {
    int                         nimage;       /* number of images */
    XcursorImage               **images;      /* array of XcursorImage pointers */
} XcursorImages;
```

XcursorCursors

Holds multiple Cursor objects. They are all freed when the XcursorCursors is freed. These are reference counted so that multiple XcursorAnimate structures can use the same XcursorCursors.

```
typedef struct _XcursorCursors {
    Display                    *dpy;          /* Display holding cursors */
    int                         ref;          /* reference count */
    int                         ncursor;      /* number of cursors */
    Cursor                     *cursors;      /* array of cursors */
} XcursorCursors;
```

XcursorAnimate

References a set of cursors and a sequence within that set. Multiple XcursorAnimate structures may reference the same XcursorCursors; each holds a reference which is removed when the XcursorAnimate is freed.

```
typedef struct _XcursorAnimate {
    XcursorCursors             *cursors;      /* list of cursors to use */
    int                         sequence;     /* which cursor is next */
} XcursorAnimate;
```

XcursorFile

Xcursor provides an abstract API for accessing the file data. Xcursor provides a stdio implementation of this abstract API; applications are free to create additional implementations. These functions parallel the stdio functions in return value and expected argument values; the read and write functions flip the arguments around to match the POSIX versions.

```
typedef struct _XcursorFile {
    void *closure;
    int (*read)  (XcursorFile *file, unsigned char *buf, int len);
    int (*write) (XcursorFile *file, unsigned char *buf, int len);
    int (*seek)  (XcursorFile *file, long offset, int whence);
};
```

FUNCTIONS

Object Management

```
XcursorImage *XcursorImageCreate (
                  int                    width,
                  int                    height)
```

```
void XcursorImageDestroy (
                  XcursorImage           *image)
```

Allocate and free images. On allocation, the hotspot and the pixels are left uninitialized. The size is set to the maximum of *width* and *height*.

```
XcursorImages *XcursorImagesCreate (
                  int                    size)
```

```
void XcursorImagesDestroy (
                  XcursorImages          *images)
```

Allocate and free arrays to hold multiple cursor images. On allocation, *nimage* is set to zero.

```
XcursorCursors *XcursorCursorsCreate (
                  Display                *dpy,
                  int                    size)
```

```
void XcursorCursorsDestroy (
                  XcursorCursors         *cursors)
```

Allocate and free arrays to hold multiple cursors. On allocation, *ncursor* is set to zero, *ref* is set to one.

Reading and writing images.

```
XcursorImage *XcursorXcFileLoadImage (
                  XcursorFile            *file,
                  int                    size)
```

```
XcursorImages *XcursorXcFileLoadImages (
        XcursorFile                *file,
        int                         size)


XcursorImages *XcursorXcFileLoadAllImages (
        XcursorFile                *file)


XcursorBool XcursorXcFileLoad (
        XcursorFile                *file,
        XcursorComments            **commentsp,
        XcursorImages              **imagesp)


XcursorBool XcursorXcFileSave (
        XcursorFile                *file,
        const XcursorComments      *comments,
        const XcursorImages        *images)
```

These read and write cursors from an XcursorFile handle. After reading, the file pointer will be left at some random place in the file.

```
XcursorImage *XcursorFileLoadImage (
        FILE                       *file,
        int                         size)


XcursorImages *XcursorFileLoadImages (
        FILE                       *file,
        int                         size)


XcursorImages *XcursorFileLoadAllImages (
        FILE                       *file)


XcursorBool XcursorFileLoad (
        FILE                       *file,
        XcursorComments           **commentsp,
        XcursorImages             **imagesp)


XcursorBool XcursorFileSaveImages (
        FILE                      *file,
        const XcursorImages       *images)


XcursorBool XcursorFileSave (
        FILE                      *file,
        const XcursorComments     *comments,
        const XcursorImages       *images)
```

These read and write cursors from a stdio FILE handle. Writing flushes before returning so that any errors should be detected.

```
XcursorImage *XcursorFilenameLoadImage (
                    const char                *filename,
                    int                        size)


XcursorImages *XcursorFilenameLoadImages (
                    const char                *filename,
                    int                        size)


XcursorImages *XcursorFilenameLoadAllImages (
                    const char                *file)


XcursorBool XcursorFilenameLoad (
                    const char                *file,
                    XcursorComments           **commentsp,
                    XcursorImages             **imagesp)


XcursorBool XcursorFilenameSaveImages (
                    const char                *filename,
                    const XcursorImages        *images)
```

```
    XcursorBool XcursorFilenameSave (
                    const char                 *file,
                    const XcursorComments      *comments,
                    const XcursorImages        *images)
```

These parallel the stdio FILE interfaces above, but take filenames.

Reading library images

```
    XcursorImage *XcursorLibraryLoadImage (
                    const char                 *name,
                    const char                 *theme,
                    int                         size)
```

```
    XcursorImages *XcursorLibraryLoadImages (
                    const char                 *name,
                    const char                 *theme,
                    int                         size)
```

These search the library path, loading the first file found of the desired *size*, using a private function (XcursorScanTheme) to find the appropriate theme:

- If *theme* is not NULL, use that.
- If *theme* is NULL, or if there was no match for the desired theme, use "default" for the theme name.
- If neither search succeeds, these functions return NULL.

The two functions differ by more than the number of images loaded:

- XcursorLibraryLoadImage calls XcursorFileLoadImage but
- XcursorLibraryLoadImages calls XcursorFileLoadImages, and (on success) it calls XcursorImagesSetName to associate *name* with the result.

Library attributes

const char * XcursorLibraryPath (void)

Returns the library search path:

- If the environment variable **XCURSOR_PATH** is set, return that value.
- Otherwise, return the compiled-in search path.

```
int XcursorLibraryShape (
        const char          *library)
```

Search Xcursor's table of cursor font names for the given "shape name" (*library*):

- If found, return the index into that table, multiplied by two (to account for the source- and mask-values used in an X cursor font).
- If not found, return -1.

Cursor APIs

```
Cursor XcursorFilenameLoadCursor (
        Display             *dpy,
        const char          *file)
```

```
XcursorCursors *XcursorFilenameLoadCursors (
        Display             *dpy,
        const char          *file)
```

These load cursors from the specified file.

```
Cursor XcursorLibraryLoadCursor (
        Display             *dpy,
        const char          *name)
```

```
XcursorCursors *XcursorLibraryLoadCursors (
        Display             *dpy,
        const char          *name)
```

These load cursors using the specified library *name*. The theme comes from the display.

```
Cursor XcursorImageLoadCursor(
                    Display                          *dpy,
                    const XcursorImage               *image)
```

This creates a cursor, given the image to display. It calls XcursorSupportsARGB to decide what type of cursor to create:

- XRenderCreateCursor is used if ARGB is supported on the display, and
- XCreatePixmapCursor is used otherwise.

```
Cursor XcursorImagesLoadCursor(
                    Display                          *dpy,
                    const XcursorImages              *images)
```

This provides an interface for creating animated cursors, if the *images* array contains multiple images, and if XcursorSupportsAnim returns true. Otherwise, it calls XcursorImageLoadCursor.

```
XcursorCursors *XcursorImagesLoadCursors(
                    Display                          *dpy,
                    const XcursorImages              *images)
```

This calls XcursorCursorsCreate to create an array of XcursorCursors, to correspond to the XcursorImages *images* array, and uses XcursorImageLoadCursor to load the corresponding cursor data.

Normally it returns the resulting array pointer. On any failure, it discards the result XcursorCursorsDestroy, and returns NULL.

X Cursor Name APIs

```
XcursorImage *XcursorShapeLoadImage (
                    unsigned int                      shape,
                    const char                       *theme,
                    int                               size)
```

```
XcursorImages *XcursorShapeLoadImages (
                unsigned int            shape,
                const char              *theme,
                int                     size)
```

These map *shape* to a library name using the standard X cursor names and then load the images.

```
Cursor XcursorShapeLoadCursor (
                Display                 *dpy,
                unsigned int            shape)
```

```
XcursorCursors *XcursorShapeLoadCursors (
                Display                 *dpy,
                unsigned int            shape)
```

These map *shape* to a library name and then load the cursors.

X Cursor Comment APIs

```
XcursorComment *XcursorCommentCreate (
                XcursorUInt             comment_type,
                int                     length)
```

XcursorXcFileLoad uses this function to allocate an XcursorComment structure for a single cursor. The *comment_type* parameter is used as the *subtype* field, e.g., COPYRIGHT. The *length* is the number of bytes to allocate for the comment text.

```
void XcursorCommentDestroy(
                XcursorComment          *comment)
```

Deallocates the given XcursorComment structure.

```
XcursorComments * XcursorCommentsCreate (
                int                     size)
```

XcursorXcFileLoad uses this function to allocate an index of XcursorComment structure pointers. The *size* parameter tells it how many pointers will be in the index.

```
void XcursorCommentsDestroy (
        XcursorComments         *comments)
```

Deallocates the given XcursorComments structure as well as the XcursorComment structures which it points to.

Animated Cursors

```
XcursorAnimate * XcursorAnimateCreate (
        XcursorCursors          *cursors)
```

Wrap the given array of cursors in a newly allocated XcursorAnimate structure, which adds a sequence number used in XcursorAnimateNext.

```
void XcursorAnimateDestroy (
        XcursorAnimate          *animate)
```

Discards the given *animate* data, freeing both the XcursorCursors array of cursors as well as the XcursorAnimate structure.

```
Cursor XcursorAnimateNext (
        XcursorAnimate          *animate)
```

Cyclically returns the next Cursor in the array, incrementing the sequence number to prepare for the next call.

The caller is responsible for displaying the series of Cursor images. Xcursor does not do that.

Glyph Cursor APIs

The X11 XCreateFontCursor and XCreateGlyphCursor functions use this part of the API to extend the X core cursors feature to use themes.

```
void XcursorImageHash (
        XImage                  *image,
        unsigned char            hash[XCURSOR_BITMAP_HASH_SIZE])
```

Compute a hash of the image, to display when the environment variable XCURSOR_DISCOVER is set.

```
void XcursorImagesSetName (
                XcursorImages                  *images,
                const char                     *name)
```

Associates the given name with the images.

```
void XcursorNoticeCreateBitmap (
                Display                        *dpy,
                Pixmap                          pid,
                unsigned int                    width,
                unsigned int                    height)
```

Check if the display supports either ARGB or themes, and also if the image size fits within the maximum cursor size (64 pixels). If so, create a bitmap of the specified size, and cache the result in Xcursor, identifying it with the Pixmap-id (pid) value.

```
void XcursorNoticePutBitmap (
                Display                        *dpy,
                Drawable                        draw,
                XImage                         *image)
```

Update the image contents in the bitmap specified by the draw value (a Pixmap-id). The bitmap must have been created by XcursorNoticeCreateBitmap.

```
Cursor XcursorTryShapeBitmapCursor (
                Display                        *dpy,
                Pixmap                          source,
                Pixmap                          mask,
                XColor                         *foreground,
                XColor                         *background,
                unsigned int                    x,
                unsigned int                    y)
```

If the display supports either ARGB or themes, try to load a cursor into Xcursor's cache using the *source* parameter as a Pixmap-id. The source may no longer be in the cache. Xcursor uses the hash value to identify the desired image.

```
Cursor XcursorTryShapeCursor (
        Display              *dpy,
        Font                 source_font,
        Font                 mask_font,
        unsigned int         source_char,
        unsigned int         mask_char,
        XColor _Xconst       *foreground,
        XColor _Xconst       *background)
```

If the display supports either ARGB or themes, try to load a cursor into Xcursor's cache using the *source_char* parameter as a shape. Using

- the default size from XcursorGetDefaultSize,
- the default theme from XcursorGetTheme, and
- the *source_char* parameter as a shape,

Xcursor calls XcursorShapeLoadImages to load the cursor images. If successful, Xcursor uses XcursorImagesLoadCursor to load the cursor information.

Display Information APIs

```
XcursorBool XcursorSupportsARGB (
        Display              *dpy)
```

Returns true if the display supports ARGB cursors. Otherwise, cursors will be mapped to a core X cursor.

```
XcursorBool XcursorSupportsAnim (
        Display              *dpy)
```

Returns true if the display supports animated cursors. Otherwise, cursors will be mapped to a core X cursor.

```
XcursorBool XcursorSetDefaultSize (
        Display              *dpy,
        int                  size)
```

Sets the default size for cursors on the specified display. When loading cursors, those whose nominal size is closest to this size will be preferred.

```
int XcursorGetDefaultSize (
        Display                 *dpy)
```

Gets the default cursor size.

```
XcursorBool XcursorSetTheme (
        Display                 *dpy,
        const char              *theme)
```

Sets the current theme name.

```
char *XcursorGetTheme (
        Display                 *dpy)
```

Gets the current theme name.

```
XcursorBool XcursorGetThemeCore (
        Display                 *dpy)
```

```
XcursorBool XcursorSetThemeCore (
        Display                 *dpy,
        XcursorBool              theme_core)
```

Get or set property which tells Xcursor whether to enable themes for core cursors.

ENVIRONMENT VARIABLES

Environment variables can be used to override resource settings, which in turn override compiled-in default values.

Some of the environment variables recognized by Xcursor are booleans, specified as follows:

true for "t", "1", "y" or "on"

false for "f", "0", "n" or "off"

Xcursor ignores other values for these booleans.

HOME
Xcursor interprets "~" in the search list as the home directory, using this variable rather than the password database.

XCURSOR_ANIM
If the display supports the Render CreateCursor request, and the Render feature is enabled, disable *animated* cursors if the environment variable is *false*.

If the environment variable is not given, Xcursor uses the resource **Xcursor.anim**.

XCURSOR_CORE
If the display supports the Render CreateCursor request disable the Render feature if the environment variable is *false*.

If the environment variable is not given, Xcursor uses the resource **Xcursor.core**.

XCURSOR_DISCOVER
If the variable is set, Xcursor turns on a logging feature. It displays the hash value and the image so that users can see which cursor name is associated with each image.

There is no corresponding resource setting.

XCURSOR_DITHER
This variable sets the desired *dither*.

If the environment variable is not given, Xcursor uses the resource **Xcursor.dither**.

If neither environment variable or resource is found, Xcursor uses "threshold"

These are the recognized values:

diffuse

median

ordered

threshold

XCURSOR_PATH

This variable sets the list of paths in which to search for cursors, rather than the compiled-in default list.

Directories in this path are separated by colons (:).

XCURSOR_SIZE

This variable sets the desired cursor size, in pixels.

If the environment variable is not given, Xcursor tries the **Xcursor.size** resource.

If no size is given, whether by environment variable or resource setting, Xcursor next tries the **Xft.dpi** resource setting to guess the size of a 16-point cursor.

Finally, if **Xft.dpi** is not set, Xcursor uses the display height, dividing by 48 (assuming that the height is 768).

XCURSOR_THEME

This variable selects the desired *theme*.

If the environment variable is not given, Xcursor tries the **Xcursor.theme** resource.

If neither environment variable or resource is found, Xcursor uses the *default* theme.

XCURSOR_THEME_CORE

Enables themes for core cursors if the environment variable is *true*.

If the environment variable is not given, Xcursor tries the **Xcursor.theme_core** resource.

An application can enable or disable themes using XcursorSetThemeCore.

SEE ALSO

XCreateRenderCursor(3), XCreatePixmapCursor(3), and XCreateFontCursor(3)

as well as

Icon Theme Specification

https://specifications.freedesktop.org/icon-theme-spec/

RESTRICTIONS

Xcursor will probably change radically in the future; weak attempts will be made to retain some level of source-file compatibility.

AUTHOR

Keith Packard