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. It uses the first cursor file found along the path.

If necessary, Xcursor also looks for a "index.theme" file in each theme directory to find inherited themes and searches along the path for those themes as well.

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

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’re all freed when the XcursorImages is freed.

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

XcursorCursors

Holds multiple Cursor objects. They’re 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 (FILE *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. If 'theme' is not NULL, these functions first try appending -theme to name and then name alone.

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.

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.

Display Information APIs

XcursorBool XcursorSupportsARGB (Display *dpy)
Returns whether the display supports ARGB cursors or whether 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.

ENVIRONMENT VARIABLES

XCURSOR_PATH

This variable sets the list of paths to look for cursors in. Directories in this path are separated by colons (:).

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