

# Xlib

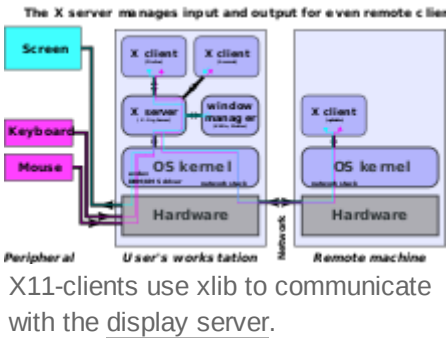
**Xlib** (also known as **libX11**) is an X Window System protocol client library written in the C programming language. It contains functions for interacting with an X server. These functions allow programmers to write programs without knowing the details of the protocol. Few applications use Xlib directly; rather, they employ other libraries that use Xlib functions to provide widget toolkits:

- X Toolkit Intrinsics (Xt)
- Athena widget set (Xaw)
- Motif
- FLTK
- GTK
- Qt (X11 version)
- Tk
- SDL (Simple DirectMedia Layer)
- SFML (Simple and Fast Multimedia Library)

Xlib appeared around 1985, and is currently used in GUIs for many Unix-like operating systems. The XCB library is an attempt to replace Xlib. While Xlib is still used in some environments, modern versions of the X.org server implement Xlib on top of XCB.<sup>[1]</sup>

<b>Contents</b>
<u>Data types</u>
<u>Protocol and events</u>
<u>Functions</u>
<u>Example</u>
<u>Other libraries</u>
<u>References</u>
<u>External links</u>

<b>Xlib</b>	
<b>Developer(s)</b>	X.Org Foundation
<b>Initial release</b>	~1985
<b>Repository</b>	gitlab.freedesktop.org/xorg/lib/libx11.git (https://gitlab.freedesktop.org/xorg/lib/libx11.git)
<b>Written in</b>	C
<b>Type</b>	Library
<b>Website</b>	www.x.org (https://www.x.org), documentation: www.x.org/releases/current/doc/libX11/libX11/libX11.html (https://www.x.org/releases/current/doc/libX11/libX11/libX11.html)



## Data types

The main types of data in Xlib are the `Display`<sup>[2]</sup> structure and the types of the identifiers.

Informally, a display is a physical or virtual device where graphical operations are done. The `Display` structure of the Xlib library contains information about the display, but more importantly it contains information relative to the channel between the client and the server. For example, in a Unix-like operating system, the `Display` structure contains the file handle of the socket of this channel (this can be retrieved using the `ConnectionNumber` macro.) Most Xlib functions have a `Display` structure as an argument

because they either operate on the channel or are relative to a specific channel. In particular, all Xlib functions that interact with the server need this structure for accessing the channel. Some other functions need this structure, even if they operate locally, because they operate on data relative to a specific channel. Operations of this kind include for example operations on the event queue, which is described below.

Windows, colormap, etc. are managed by the server, which means that the data about their actual implementation is all stored in the server. The client operates on these objects by using their *identifiers*. The client cannot directly operate on an object, but can only request the server to perform the operation specifying the identifier of the object.

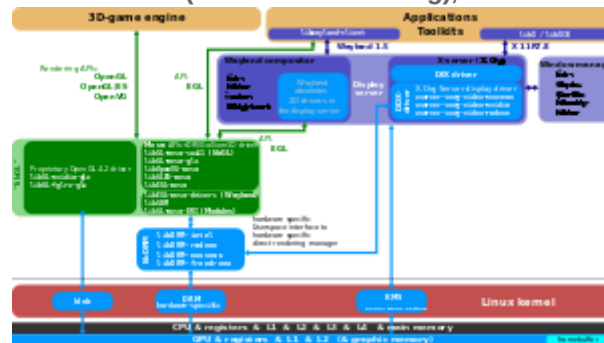
The types `Window`, `Pixmap`, `Font`, `Colormap`, etc. are all identifiers, which are 32-bit integers (just as in the X11 protocol itself). A client 'creates' a window by requesting that the server create a window. This is done via a call to an Xlib function that returns an identifier for the window, that is, a number. This identifier can then be used by the client for requesting other operations on the same window to the server.

The identifiers are unique to the server. Most of them can be used by different applications to refer to the same objects. For example, two applications connecting with the same server use the same identifier to refer to the same window. These two applications use two different channels, and therefore have two different `Display` structures; however, when they request operations on the same identifier, these operations will be done on the same object.

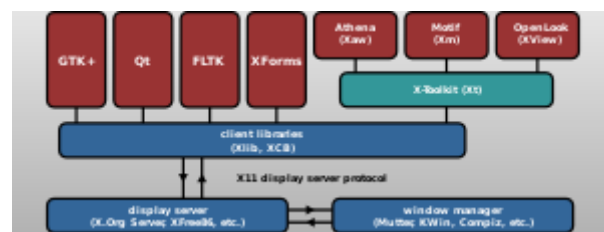
## Protocol and events

The Xlib functions that send requests to the server usually do not send these requests immediately but store them in a buffer, called the *request buffer*. The term *request* in this case refers to the request from the client that is directed to the server: the request buffer can contain all kinds of requests to the server, not only those having a visible effect on the screen. The request buffer is guaranteed to be flushed (i.e., all requests done so far are sent to the server) after a call to the functions `XSync` or `XFlush`, after a call to a function that returns a value from the server

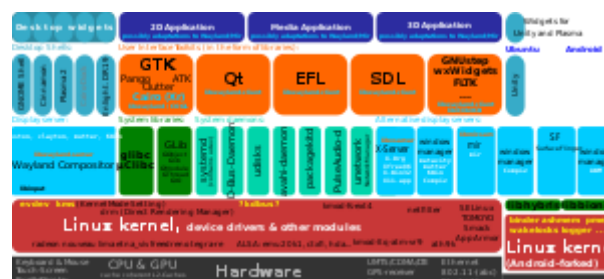
## The role of KMS (Kernel mode-setting), Linux example



Illustrates the Linux graphics stack current as of 2013-08-24



XCB and Xlib are client libraries which implement a display server communications protocol



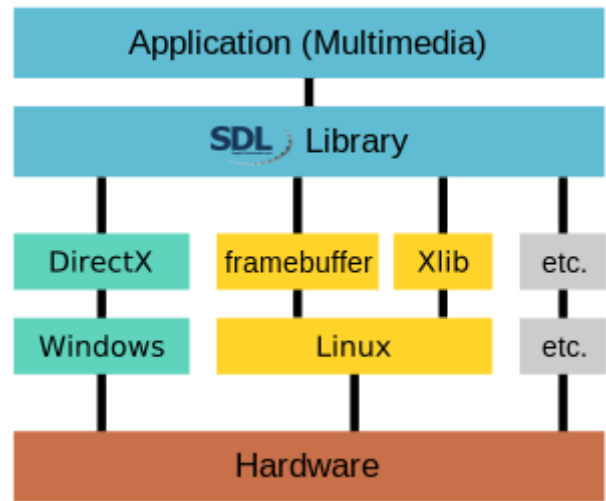
The display server sits between the kernel (*here: Linux kernel*) and its clients. It communicates with its clients over a given protocol.

(these functions block until the answer is received), and in some other conditions.

Xlib stores the received events in a queue. The client application can inspect and retrieve events from the queue. While the X server sends events asynchronously, applications using the Xlib library are required to explicitly call Xlib functions for accessing the events in the queue. Some of these functions may block; in this case, they also flush the request buffer.

Errors are instead received and treated asynchronously: the application can provide an error handler that will be called whenever an error message from the server is received.

The content of a window is not guaranteed to be preserved if the window or one of its parts are made not visible. In this case, the application are sent an `EXPOSE` event when the window of one part of it is made visible again. The application is then supposed to draw the window content again.



## Functions

The functions in the Xlib library can be grouped in:

1. operations on the connection (`XOpenDisplay`, `XClosedDisplay`, ...);
2. requests to the server, including requests for operations (`XCreateWindow`, `XCreateGC`, ...) and requests for information (`XGetWindowProperty`, ...); and
3. operations that are local to the client: operations on the event queue (`XNextEvent`, `XPeekEvent`, ...) and other operations on local data (`XLookupKeysym`, `XParseGeometry`, `XSetRegion`, `XCreateImage`, `XSaveContext`, ...)

## Example


The following program creates a window with a little black square in it:

```
/*
   Simple Xlib application for creating a window and drawing
   a box in it.
   gcc input.c -o output -lX11
*/

#include <X11/Xlib.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main(void)
{
    Display *display;
    Window window;
    XEvent event;
    char *msg = "Hello, World!";
    int s;

    // open connection to the server
    display = XOpenDisplay(NULL);
```



Hello, World!

Simple Xlib application drawing a box and text in a window. Without window manager decorations.

```

if (display == NULL)
{
    fprintf(stderr, "Cannot open display\n");
    exit(1);
}

s = DefaultScreen(display);

// create window
window = XCreateSimpleWindow(display, RootWindow(display,
s), 10, 10, 200, 200, 1,
                                BlackPixel(display, s),
                                WhitePixel(display, s));

// select kind of events we are interested in
XSelectInput(display, window, ExposureMask |
KeyPressMask);

// map (show) the window
XMapWindow(display, window);

// event loop
for (;;)
{
    XNextEvent(display, &event);

    // draw or redraw the window
    if (event.type == Expose)
    {
        XFillRectangle(display, window,
DefaultGC(display, s), 20, 20, 10, 10);
        XDrawString(display, window, DefaultGC(display,
s), 50, 50, msg, strlen(msg));
    }
    // exit on key press
    if (event.type == KeyPress)
        break;
}

// close connection to the server
XCloseDisplay(display);

return 0;
}

```



Simple Xlib application drawing a box and text in a window. With [IceWM window manager](#) decorations.

The client creates a connection with the server by calling `XOpenDisplay`. It then requests the creation of a window with `XCreateSimpleWindow`. A separate call to `XMapWindow` is necessary for mapping the window, that is, for making it visible on the screen.

The square is drawn by calling `XFillRectangle`. This operation can only be performed after the window is created. However, performing it once may not be enough. Indeed, the content of the window is not always guaranteed to be preserved. For example, if the window is covered and then uncovered again, its content might require being redrawn. The program is informed that the window or a part of it has to be drawn by the reception of an `Expose` event.

The drawing of the window content is therefore made inside the loop handling the events. Before entering this loop, the events the application is interested in are selected, in this case with `XSelectInput`. The event loop waits for an incoming event: if this event is a key press, the application exits; if it is an expose event, the window content is drawn. The function `XNextEvent` blocks and flushes the request buffer if there is no event in the queue.

## Other libraries

Xlib does not provide support for buttons, menus, scrollbars, etc. Such widgets are provided by other libraries, which in turn use Xlib. There are two kinds of such libraries:

- libraries built atop of the X Toolkit Intrinsics library (Xt), which provides support for widgets but does not provide any particular widget; specific widgets are provided by widget set libraries that use Xt, such as Xaw and Motif;
- libraries that provide widget sets using Xlib directly, without the Xt library, such as the X versions of GTK, Qt, FLTK and fpGUI.

Applications using any of these widget libraries typically specify the content of the window before entering the main loop and do not need to explicitly handle `Expose` events and redraw the window content.

The XCB library is an alternative to Xlib. Its two main aims are: reduction in library size and direct access to the X11 protocol. A modification of Xlib has been produced to use XCB as a low-level layer.

## References

---

1. "Adoption" (<https://xcb.freedesktop.org/adoption/>).
2. "Display Structure on freedesktop CVS" (<http://webcvs.freedesktop.org/xorg/lib/X11/include/X11/Xlib.h?revision=1.6&view=markup>). *Tip search for: `typedef struct _XDisplay Display`.*

## External links

---

- X.Org Foundation's official programming documentation (<http://xorg.freedesktop.org/wiki/ProgrammingDocumentation/>), including most recent version of Xlib - C Language X Interface (<http://www.x.org/releases/current/doc/libX11/libX11/libX11.pdf>) in several formats.
  - A short tutorial on Xlib (<https://tronche.com/gui/x/xlib-tutorial/>)
  - Manual pages for all Xlib functions (<https://tronche.com/gui/x/xlib/function-index.html>)
  - Kenton Lee's pages on X Window and Motif (<http://www.rahul.net/kenton/bib.html>)
  - A longer tutorial on Xlib (<https://web.archive.org/web/20060923165147/http://users.actcom.co.il/~choo/lupg/tutorials/xlib-programming/xlib-programming.html>)
  - Using Xlib for creating a screensaver module (<http://www.dis.uniroma1.it/%7eliberato/screensaver>)
- 

Retrieved from "<https://en.wikipedia.org/w/index.php?title=Xlib&oldid=993341122>"

---

This page was last edited on 10 December 2020, at 02:46 (UTC).

Text is available under the Creative Commons Attribution-ShareAlike License; additional terms may apply. By using this site, you agree to the Terms of Use and Privacy Policy. Wikipedia® is a registered trademark of the Wikimedia Foundation, Inc., a non-profit organization.