AT&T

UNIX 系统 V 第 4 版
程序员指南: XWIN 图形窗口系统
Xlib-C 语言界面

UNIX® SYSTEM V
RELEASE 4

Programmer's Guide: XWIN™
Graphical Windowing System
Xlib-C Language Interface

UNIX Software Operation

電子工業出版社

目　录

第一章　Xlib 简介

1·1　Xlib 简介

X 窗口系统是一个网络透明的窗口系统，由 MIT 设计，运行于 4·3BSD UNIX，ULTRIX-32 等许多 UNIX 变种以及 VAX/VMS 和 MS/DOS 等操作系统之上。

AT&T 的 XWIN™ 版本 3·0 是在 MIT X 窗口系统 X11R3 基础上开发出的产品。它是基于流的，在性能上有所增强，运行于 UNIX 系统 V 版本 3·2 和 SVR4·0 之上。

XWIN 显示服务方在带有单色或彩色位映像硬显示设备的计算机上运行。服务方负责将用户输入分布到各客户程序中并接收各客户程序的输出要求，这些客户程序或者在同一机器上或者在网络上的其它地方。Xlib 是一个 C 的子例程库。应用程序(客户)通过流连接手段使用 Xlib 与窗口系统打交道。尽管客户常常和 Xwin 服务方在同一台机器上运行，但并不要求非要这样。

《Xlib-C 语言 X 界面》是有关低层 C 语言与 X 系统协议接口的参考指南。它既不是 X 窗口系统程序设计的初级教材，也不是用户指南。相反，它不仅详细介绍了库中的每个函数，而且给出了相关的背景信息。《Xlib-C 语言 X 界面》以图形窗口系统和 C 程序设计语言为基础，其它高层描述(如 X 工具箱提供的描述)是建筑在 Xlib 库之上的。有关高层库的详细信息，请参见相关的工具箱文档资料。《X 窗口系统协议》中给出了 X 行为的明确说明，虽然这里也出现了一些协议信息，但它不是有关协议的主要文档。

为了介绍 X 程序设计，本章将讨论如下内容：

- XWIN系统概貌
- 错误
- 命名和参数约定
- 程序设计时应注意的问题
- 本文使用的约定

1·2　XWIN 系统概貌

本书引用了很多术语，其中有些只在 XWIN 系统中使用，有些在其它窗口系统中也用，但与在 XWIN 中所用的含义不同，书后附有词汇表，对理解这些术语很有帮助。

XWIN 服务方将所有窗口都组织在严格的层次结构中，每一层的顶端均为根窗口。根窗口占据了显示屏幕，它部分或全部地被其子窗口所覆盖。除了根窗口外，所有窗口都有父窗口。通常情况下，一个应用程序至少打开一个窗口，子窗口还可以打开自己的子窗口。这样，应用程序就可以建立起一棵深度任意的窗口树。XWIN 系统还提供了窗口的图形、正文和光栅操作。

子窗口可以大于父窗口，也就是说，子窗口的部分或全部可以延伸到父窗口的边界之外，但子窗口的所有输出均被其父窗口所裁剪。如果几个子窗口间有重叠部分，则必有其中一个显示在最上面，覆盖了其余几个。倘若被覆盖的窗口没有后备存贮，那么对被覆盖区域的输出就由窗口系统自动遮掩住了。若第一个窗口覆盖了第二个窗口，则第一个窗口覆盖的那些祖先窗口，既是第一个窗口的祖先，又是第二个窗口的祖先。

一个窗口的边可以有零个或多个像素宽度，像素可以为任何模式(像素映像)和颜色。一个窗口通常但并非总是有一个背景映像模式。当窗口不再被覆盖时，该模式要由窗口系统重画。每一窗口都有自己的坐标系统。如果子窗口拥有背景模式，则它们能遮住其父窗口，而且这时父窗口中的图形操作通常由其子窗口裁剪。

XWIN 服务方不确保窗口内容的完整。当一个窗口的部分或全部被遮盖并再显示到屏幕上时。其内容可能会丢失。这时，服务方向客户程序发送一个 Expose 事件，以提示它窗口的一部分或全部需要重画。客户程序必须根据要求准备重新生成窗口内容。

XWIN 服务方还提供图形对象的脱屏存贮，该图形对象称为像素映像。单一的平面像素映像(深度为 1)有时也指位映像。像素映像可互换地用于窗口间的大多数图形函数中，也可以在各种图形操作中用于定义模式和瓦片。窗口和像素映像一起称为可绘对象。

Xlib 中的大多数函数仅仅是在输出缓冲区中增加几个请求，这些请求以后在 XWIN 服务方上异步执行。有些函数要求返回存贮在服务方中的信息值。这些函数要到收到一个显式的应答或出现错误时才将值返回(即，这些函数被锁住)。用户可以提供一个错误处理器，在报告错误时调用它。

若客户不希望其请求异步执行，则在请求后跟着调用 XSync。XSync 一直锁住，直到以前缓冲区中的异步事件均已发出并起作用时才解除封锁。注意一个严重的副作用：对于从服务方中返回值或等待输入的函数来讲，每次调用函数一次，Xlib 中的输出缓冲区总要被刷新一次。

许多 Xlib 函数会返回一个整数资源 ID，它允许用户引用存贮于 XWIN 服务方中的对象，ID 可以为 Window，Font，Pixmap，Colormap，Cursor 和 GContext 类型，在 <X11/X.h> 文件中加以定义。

这些资源根据请求创建，并根据请求释放，当连接关闭时，资源也被释放。多数资源在应用程序间潜在共享，事实上，所有窗口都是由窗口管理程序显式地操纵的。字体根据需要装入装出，由多个客户共享。在 XWIN 服务方中，字体通常放在高速缓存内。Xlib 不支持应用程序间图形正文的共享。

客户程序由事件唤醒，事件可能是某一请求的副作用(例如，重建窗口的堆栈时要产生 Expose 事件)或完全是异步的(例如，从键盘上接收的事件)。客户程序要求由事件触发，因为其它应用程序可以对之发送事件，所以客户程序必须准备处理(或放弃)各种类型的事件。

输入事件(例如，按键或移动鼠标)从服务方异步传来并排好队，等待一个显式调用(如 XNextEvent 或 XWindowEvent)请求接收它们。此外，一些库函数(例如，

XRaiseWindow)也产生 Expose 和 ConfigureRequset 事件，这些事件也异步到达，但可以显式地等待。例如客户调用了一个函数，该函数能导致服务方产生事件，这时，客户在调用该函数之后调用 Xsync，便可以显式地等待这些 Expose 事件或 ConfigureRequest 事件的到达。

注意：< >的含义在C编译器中由#include语句定义，< >内的文件位于一个公认的目录下，对XWIN系统而言，该目录为/usr/X/include。

1·3 错误

某些函数会返回一个整型错误标识 status。若函数执行不成功，则函数返回 0，这时若错误标识为 0，说明函数没有修改返回参数。由于 C 不提供多个返回值，所以许多函数通过将结果写入客户传来的存贮区来返回多个值。

默认情况下，错误由标准库函数处理，或者由用户提供的函数处理。若函数要求返回字符串指针，则当该串不存在时，返回 NULL 指针。

XWIN 服务方随时检测协议错误并随时报告，对一个给定的请求，若产生了多于一个错误，服务方能报告其中任一个。

由于 Xlib 通常先将请求放在缓冲区中，而不是立即将之送到服务方去，所以报告错误通常比出现错误晚得多。为了调试方便，我们可以用 Xlib 提供的机制，强行执行同步操作(参见第八章"支持或禁止同步"一节)。一旦设置了同步方式，错误便随时产生随时报告。

当 Xlib 检测到错误时，它就调用用户程序提供的错误处理器，如果用户程序不提供错误处理器，它就将错误打印出来并终止用户程序的运行。

1·4 Xlib 中的命名和参数约定

Xlib 遵循如下命名约定和函数语法规定。记住函数要求的这些信息，有助于预测函数的语法关系。

主要的命名约定有：

- 为了区别XWIN符号和其它符号，Xlib库用混合大小写表示外部符号，用小写表示变量，用全部大写表示用户自定义的宏，这与已有约定相同。
- 所有Xlib函数以大写的X开头。
- 所有函数名和符号的开头字母均大写。
- 所有用户可见的数据结构名均以大写的X开头，更一般地，任何用户可能间接引用的符号均以大写的 X 开头。
- 宏和其它符号不以大写X开头，为了将它们同用户符号分开，宏中的每个单词

均大写。

- 数据结构中的所有元素或变量用小写形式，复合单词用下划线(_)连接。
- 若使用显示器参数，则总是将它放在参数表的最前面。
- 若使用资源对象，则将它放在参数表中紧跟显示器参数之后。
- 若一图形上下文资源与另一类型的资源(最常见的为一可绘对象)同时出现，则图形上下文放在参数表中其它资源之后。可绘对象比所有其它资源级别高。
- 参数表中，源参数总是出现在目标参数的前面。
- 参数表中，x参数总是出现在y参数的前面。
- 参数表中，宽度参数总是出现在高度参数的前面。
- 当x参数、y参数、宽度参数和高度参数一起使用时，x和y参数总是出现在宽度和高度参数的前面。
- 参数表中，若掩码和结构一起使用时，掩码总是出现在指向结构的指针前面。

1·5 程序设计时需要注意的问题

程序设计时主要要注意以下问题：

- 不同厂家生产的键盘有很大区别，若想使自己的程序可移植，在这时应该特别留心。
- 许多显示器系统限制了脱屏存贮器的数量，编程时要尽可能少用像素映像和后备存贮。
- 用户应能控制其屏幕的实际状况。所以，用户需要编写自己的应用程序与窗口管理打交道，而不要假想已能控制整个屏幕。用户上层窗口内所做的一切取决于用户的应用程序。进一步的讨论请参见第九章。
- XWIN系统中的坐标和规格实际上是16位长的量，在界面上通常说明为"int"类型(某些机器上 int 为 16 位)。长于 16 位的值自动被截断。规格(宽度和高度)是无符号量。采用这种策略，对于给定的性能级别，可以使所需的带宽的最小。

1·6 《Xlib—C 语言 X 界面》中使用的约定

本文使用如下约定：

- 《Xlib—C语言X界面》中的全称符号用等宽字体表示。这些全称符号可能是函数名，包含文件中定义的符号，或结构名。参数用斜体表示。
- 每一函数由一概要介绍引入，该介绍将此函数与其它函数区别开来。随后是函数声明和每一参数的特别说明。如果需要，参数说明后有一函数的总体讨论，其最后一段列出此函数可能产生的 Xlib 错误码。在第八章"使用默认的错误处

理程序”一节中，有 Xlib 错误码的完整讨论。

- 为了区分函数的传入参数和传出参数，对于用户传入函数的参数，在参数说明前冠以单词指定，若有多个传入参数，也冠以指定。对于函数返回的参数，在参数说明前冠以单词返回，若有多个传出参数，也冠以返回。对于既是传入又是传出的参数，在参数说明前冠以单词指定并返回。
- 任何用于返回值的结构指针，以名中的后缀*–return*标识，其它传入函数的指针只用于读取。只有很少几个函数，采用指向结构的指针，既要输入又要输出，这种指针以名中的后缀*–in–out* 标识。
- Xlib定义了布尔值`True`和`False`。

第二章　显示器函数

2·1　引言

在用户程序使用显示器之前，必须先同 XWIN 服务方建立连接。只有在这种连接建立起来之后，用户才能使用本章介绍的 Xlib 宏和函数，得到有关显示器的信息。本章讨论如下内容：

- 打开(连接)显示器
- 获取有关显示器、图像格式和屏幕的信息
- 释放客户建立的数据
- 关闭显示器(与显示器切断连接)

本章对关闭与 XWIN 服务方的连接时会出现的情况进行了概括讨论，并给出结论。

2·2　打开显示器

使用 `XOpenDisplay` 函数，可以打开与 XWIN 服务方的连接，控制显示器。

```
Display *XOpenDisplay (display_name)
    char * display_name;
```

display_name　指定硬件显示器名，以确定要用的显示器和通信范围。在基于UNIX的系统中，如果该显示器名为 NULL，则采用 DISPLAY 环境变量作默认值。

在基于 UNIX 的系统中，显示器名或 DISPLAY 环境变量是如下形式的字符串：

hostname: number·screen_number

hostname　指定显示器实际所在的主机名。其后可接单冒号(:)，也可接双冒号(::)。

number　指定该台主机上显示器的序号。序号后面可以跟句号(·)，也可以不跟。单 CPU 可以有多个显示器，多个显示器通常从零开始编号。

screen_number　指定该显示器上要用的屏幕。此屏幕号设置了一个内部变量，如果使用的是 C 以外的程序设计语言，这个内部变量可以用宏 `DefaultScreen` 或函数 `XDefaultScreen` 存取。

例如，下列串将指定名为 mit-athena 主机上的显示器 0 的第 2 个屏幕：

```
mit-athena: 0.2
```

函数 `XOpenDisplay` 返回 `Display` 类型的结构，它负责与 XWIN 服务方的连接，并包含了 XWIN 服务方的所有信息。`XOpenDisplay` 将用户应用程序同 XWIN 服务方连接起来，这种连接是通过 TCP、UNIX domain 或 StarLan(只对 SVR3·2)通信协议实现