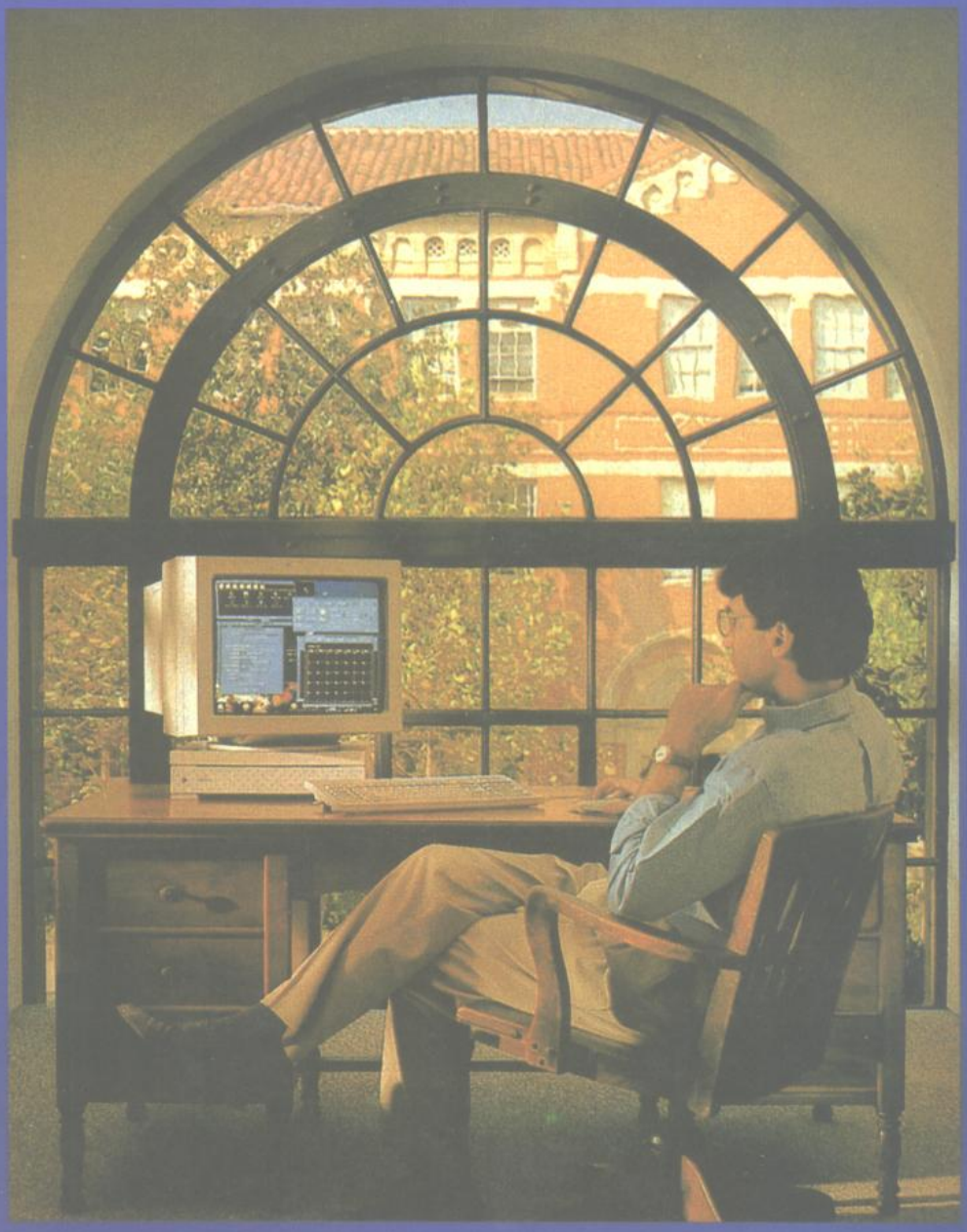
Sun



新型计算机

工作站编程环境

伍颖文　朱文水　张　洁
王　祥　　于长云　编

南　开　大　学　出　版　社

新型计算机

工作站编程环境

伍颖文　朱文水　张　洁
王　祥　　于长云　编

南开大学出版社

[津]新登字 011 号

新型计算机

工作站编程环境

伍颖文 朱文水 张洁 王祥 于长云 编

南开大学出版社出版
（天津八里台南开大学内）
邮编 300071 电话 3358542
新华书店天津发行所发行
南开大学出版社照排中心排版
天津宝坻第二印刷厂印刷

1995 年 3 月第 1 版　1995 年 3 月第 1 次印刷
开本：787×1092 1/16　印张：19.75 插页：2
字数：499 千　印数：1—5000

ISBN 7-310-00681-X
TP·21　定价：26.00 元

内容提要

本书是作者在使用和开发工作站应用软件的基础上编写而成的，它从如何开发 X 窗口系统的角度精要地介绍了工作站编程思想、Xlib 编程方法、XToolkit 编程方法和 XView 编程方法。本书叙述精炼，内容丰富，是开发工作站 GUI 应用程序的极好工具。

本书可作为高等学府计算机专业高年级本科生和研究生的教科书，也可供广大工作站用户、科技工作者、软件开发人员及其它有关人员学习参考。

工作站将以其强大的图形功能和良好的窗口界面在 90 年代大显身手，相信本书的出版必将受到广泛的欢迎！

前　言

八十年代以来，工作站作为一种功能强大的新机种在我国异军突起，时至今日，其发展如日中天。大多数工作站使用 UNIX 操作系统，对于 DOS 用户来讲，UNIX 魅力独具却又难以掌握，为此我们组织了几位长期从事 UNIX 开发的研究人员，编写出一套工作站使用及编程的技术读物。

有关工作站的内容复杂而繁浩，国内资料奇缺。考虑到读者知识层次不一，我们选材时从最基础的 UNIX 使用过渡到 X 窗口系统的编程，重点突出 X 窗口系统的机制和基本开发，以适应国际上软件窗口化的潮流。

这套丛书的上卷《工作站使用环境》已于 1993 年秋由南开大学出版社出版发行。该书的阅读对象主要是 UNIX 初级用户，主要内容为工作站简介、UNIX 使用、OpenWindows 使用和 SunView 使用。

本书是《工作站使用环境》的后续篇，主要内容为 X 窗口系统的基础编程。X 窗口系统是一个基于 Client/Server 模式的网络窗口系统，它的出现彻底改变了 UNIX 难以使用的局面，几乎所有的工作站厂商已接受其为 GUI 工业标准，微机系统上也已出现基于 UNIX 的 X 窗口系统软件产品。

本书第一章简要介绍 Xlib 的编程方法。Xlib 的 C 语言同 X 低层协议的接口，其功能十分强大。考虑到 Xlib 过于繁琐，同时建立在其上的 Xt 和 XView 提供了更为简洁而强大的开发手段。本章只对 Xlib 作了精要的介绍。

第二章主要介绍 Xt 的编程方法。X Toolkit 是一个建立在 Xlib 之上的工具包，而 Xt 则是它的核心 C 语言库。Xt 为图形接口编程提供了一套简便的方法，它采用了可重入接口组件的机制，使得应用界面保持一致的感观。本章只对 Xt 作了精要的介绍。

第三章是本书的重点，主要介绍 XView 的开发方法。XView 也在 Xlib 上开发，它保证对象的功能和外观遵循 OPEN LOOK 规范。XView 是使用最为普遍而功能最为集成的 X 工具包。

参加编写的作者参阅了大量外文资料，并对材料作了合理的取舍，力求突出简洁精练的特点。我们希望这套丛书筑造起广大读者步入工作站及 X 窗口世界的桥梁。

本书的作者如下：

第一章《X 协议编程思想》：张结；第二章《Xlib 编程方法》：张结；第三章《Xt 编程方法》：朱文水；第四章《XView 编程方法》：伍颖文，于长云。

伍颖文老师承担了写作任务，同时集中审校整理了全稿。王祥副教授初校了原稿。另外，于长云副教授及黄国胜老师对校对工作给予了协作。

在编写过程中，南开大学计算机与系统科学系的朱瑞香教授、陈有祺教授、韩维恒教授等给予了大力指导。

南开大学出版社的李江卫编辑及许多其它人员为本书的出版付出了大量的心血，在此深表谢意。

由于时间仓促，无法面面俱到，书中难免谬漏，恳请读者斧正。如果本(套)书能对读者有所裨益，编者即可聊以自慰。

编者识

1994 年 10

目　录

第一章

X 协议编程思想

X Window 系统支持层次窗口结构及与设备无关的图形，X 的界面是可以制作的，这一点与其它 UNIX 系统的内置界面不同。X 不是采用系统调用而是采用网络协议作为一级界面。这种网络协议有以下优点：

- 从用户角度看，本地机和网络连接的操作是相同的。
- X 协议可在不同语言和不同操作系统下使用。
- X 协议的形式是流格式。
- X 协议十分可靠。

值得注意的是，网络和窗口系统应该一同使用，因为本地与网络没有区别，应用程序只要提供网络接口即可。

X 协议在异种机之间的可移植性很强。无论是微机还是巨型机，也不管硬件和系统有什么差别，应用程序使用 X 都是一样的。

1.1 客户和服务器

X 中的客户/服务器与其它系统中的有区别。对 X 来说，服务器指的是管理显示器、键盘及鼠标的软件。客户程序指的是显示在屏幕上接受键盘和鼠标输入的程序。客户程序向服务器发送画图请求，服务器再把回应信息发向客户程序的输入。客户程序与服务器可在同一机器上，也可在网络的不同机器上。

X 并不限于一个客户程序与服务器通讯。它允许有多个客户（如显示在屏幕上的程序）向服务器发送请求；同样，也允许单个客户与几个服务器通讯，这一点对一个程序在多个用户的屏幕上显示时极为有用。

窗口管理程序也是客户程序，除了脱离窗口外观的并具有一定特权外，它与其它客户程序几乎没有区别（因为它使用相同的协议与服务器通讯）。

X 程序可用通用的 C 语言和 Lisp 语言编制。它的编程库包含称为 Xlib 的低层协议接口和面向对象的叫做 Xt 工具的高层接口。Xt 工具用预定义的组件编制用户接口。MIT 提供了一种称为 A-

thena 的组件。

Xlib 和 Xt 工具都是 MIT 开发的。MIT 之外的其它组件有用 C 和 C++开发的，如 Andrew、InterViews 和 XView 等。

1.2 X 协议

X 协议是 X 窗口系统的实定义，任何实现它的语言代码是 X 的实现，它采用异步 8 位流在窗口系统间通讯。

在不与服务器客户冲突的情况下可使用 X 之下的任何双向流来工作。当客户与服务器在本地机上时，这种连接采用的是内部进程通讯(IPC)、共享内存和 UNIX 插板，但它仍是 X 协议。

该协议采用异步工作方式，以求得高效率。同步执行的速度受回路的限制，在目前已有的网络上是在 5 至 50 微秒之间，比网络间无需回答的请求速度慢。服务器还异步发送事件，因为它应允许输入排队。这通常发生在屏幕显示的同时还不断有用户输入的情况下。

许多 UNIX 下的窗口系统采用文件和管道通讯，这与 X 用单一的网络通讯相比有许多缺点。文件的描述往往是受限的，并且不能被不同机器甚至同一机器的不同客户程序共享。

通常，客户程序使实现 X 协议的编程库采用单一的网络协议，例如 TCP/IP 或 DECnet。MIT 提供的 C 语言实现程序叫做 Xlib，它采用 Berkeley UNIX 的接插板。

服务器通常能理解不只一种的协议，以使它能与不同的网络连接。例如 DEC 窗口的服务器能接受来自 TCP/IP 或 DECnet 的客户程序请求。当前只有这两种网络协议支持 X 服务器。

1.3 消息类型

X 协议定义了四种类型的消息用于网络通讯。请求由客户程序发向服务器，回答、事件和错误由服务器发向客户。

- 请求

由客户程序产生并发向服务器。请求协议可包含许多信息，如画线、改变调色板颜色值和询问窗口大小等。请求协议长度可以是 4 字节的任何倍数长。

- 回答

回答来自服务器，它对应一定的客户请求。不是所有的请求都有回答，只有请求信息的请求才有。例如画线请求没有回答，而询问窗口大小的请求却有回答。回答协议的长度是 4 字节的任何倍数长，但大于 32 字节。

- 事件

事件由服务器发向客户程序，它包含设备动作或以前请求的副作用信息。事件中包含的数据多种多样，接收事件是客户程序获得信息的基本途径。所有的事件都是 32 字节的结构，以便统一处理。

- 错误

错误与事件类似，但在客户程序中的处理不同。错误是发向客户方面的错误处理信息。为了处理方便，错误信息与事件具有同样大小。

需要回答的请求叫 Round _ trip 请求。这种请求应尽量少用。因为它会产生网络等待。

之所以所有的协议都是 4 字节的倍数，是为了简化需要 16 或 32 位对齐的结构下的处理。消息中的 16 或 32 倍值均需经过对齐。

1.4 职责分配

在 X 协议的设计过程中，许多思想都归结到服务器和客户的划分，因为这决定了什么信息在请求、回答和事件之间传递。

首先，服务器应设计成尽可能隐藏应用程序的不同。服务器管理窗口，控制画图，与设备驱动程序交互以接受键盘和光标输入。服务器也管理屏幕外观、窗口、字体、光标和调色板。一般的服务器代码分为设备无关和设备有关两部分。与设备有关的部分应能适合不同的硬件设置。

让服务器管理窗口系统的树形结构和叠次关系有许多缺点。但这对客户程序来讲似乎可节省网络时间，因为客户程序请求不可见窗口的图形，一般不知道它的位置和栈次序。这种情况只在一部分区域由不可见变为可见时，才由客户程序做相应处理。在 X 中，发送绘画新显露部分的请求是客户程序的事。

有一些硬件的参数隐藏在服务器中是不可能或不明智的。X 服务器试图将客户程序与屏幕参数（如屏幕大小、彩色还是单色、颜色数等）分离开来，但这都要花代价。隐藏屏幕大小对屏幕分辨率需要固定大小应用的请求来说变得较为方便，但它不好定义手工的贴图，从而加长了服务器的代码。隐藏屏幕是否是彩显以及颜色深度对使用简单颜色的客户程序来说变得方便些，但却很难处理高要求的调色板以及分叠次的颜色轨迹。所以决定让客户程序复杂些以求高效能，目标是在工具库中隐藏复杂度。

服务器力图统一处理不同机器的键盘，但仍不能完全隐藏它们的不同，因为无法知道键帽上是什么符号。例如，并不是所有的键盘都有 Control 和 Meta 键。X 解决这个问题的方法是提供不同层次的名词。每一物理键有服务器设备有关层提供的代码，用于表示每一个键事件。服务器同时还提供一张表，用于表示每一个键组合及其意义。例如，当按住“shift”键，再按下“a”时，表中的键符号表示或“A”。这张表由服务器保存，但客户程序库中通常也有一备份，以便自己能快速解决键事件。服务器不加解释地把键事件发送给客户程序，由客户程序用不同语言或其它技术处理（例如，客户程序可把它换成音乐按键）。X 提供请求 ChangeKeyboardMapping）来改变键符号表，并引起 MappingNotify 事件发向所有的客户程序。

有些划分是为了简化客户编程。例如，画图请求所用的坐标是相对于窗口而不是相对于屏幕的，这实际上提供了一个虚拟的窗口界面，从而简化了编程，因为客户程序无需不断地检查窗口的位置和计算绘图坐标。当然这样做加重了服务器和基于窗口位置作图时的负担。相反地，也允许服务器独立地管理窗口层次，而不必向客户程序通告每一变化。

另外，进行划分是为了提高效力。一个典型的例子是图形内容。X 的图形上下文（GC）允许服务器缓存图形请求如何翻译的信息，所以这些信息不必在每次图形请求时都要由客户程序发向服务器。这就提高了网络传输的效率，尤其是在网络速率较低时更为明显。同样，由服务器保存 GC 可方

便快速地在它们中间切换。最后，作为副效应，GC 的使用也简化了编程，因为它减少了画图请求的参数。

GC 只是 X 在服务器中保留的句柄之一，其它重要的句柄有窗口、三维像点图（屏幕外的绘图平面，只有被拷贝到屏幕上时才变为可见）、调色板和字体。客户程序在请求协议中使用服务器分配的唯一 ID 对每一句柄进行访问。每一 ID 是一个 29 位的整数。ID 由客户方库函数选取，但在连接时要使用服务器指定的一个范围，以便与使用同一服务器的其它客户程序产生的 ID 相区别。ID 不由服务器产生的原因是，产生句柄不必由服务器回答。这对客户程序初始化十分有用，它可以减少时间，因为每一句柄的创建要花掉至少一个来回的时间。

窗口句柄允许服务器管理屏幕的哪一部分显示窗口的哪一部分以及对每一窗口指定窗口属性（如边框和背景）。X 包括获取句柄信息的请求协议，所以客户程序并不是一无所知。但是，从客户角度看并不是每一细节都是需要的，因为许多信息的编码是从方便服务器的角度考虑的。用户不能随意使用，也没有必要知道，例如窗口的许多属性（如位重心，它用于重画优化）和 GC 的不少值都不可访问的。无论是窗口的，还是 GC 的，这些不可访问的值都受其它客户程序的影响。因此它们不会加重客户程序不继续监视它们的负担。

1.5 编程简析

下面说明一个最小的应用程序运行时网络上的情况，它包括创建窗口、申请颜色、等待事件、窗口画图和退出。它使用了四种协议中的三种（假设它不使用错误协议）。

以下是程序正常执行时网络上的事件：

- 客户程序打开与服务器的连接，发送描述自己的信息。
- 服务器回应描述服务器的信息或拒绝连接。
- 客户程序请求创建窗口（无需回应）。
- 客户程序请求分配颜色。
- 服务器回应描述分配颜色的信息。
- 客户程序请求创建图形内容，供以后使用。
- 客户程序发送请求描述需要的事件。它们是 Expose 和 ButtonPress 事件。
- 客户程序请求映射创建的窗口。
- 客户程序等待 Expose 事件。
- 服务器发送 Expose 事件，表明窗口已被显示。
- 客户请求绘画及使用图形内容。
- 循环等待 Expose 事件。

其中客户请求在客户队列中排队，先不发向服务器，在读事件时才激发发送。这实际上不是客户库函数的特性，但由于它利用了协议的异步特性而提高了效力。Xlib 就是这样工作的，它允许客户程序不必停下来等待网络访问（除非它自己要停下来）。

这里，客户的动作应按固定的次序完成。例如，Expose 事件必须在窗口映射前选取，否则没有事件通知客户程序绘图。这在有窗口管理程序管理屏幕时尤为重要。许多窗口管理程序允许用户在窗口映射前定义其大小和位置，指定从客户请求（映射）到实际映射的延迟。只有 Expose 事件才

告诉客户程序什么时候可以画图。

为了提高协议请求的效率须在申请图形内容前申请颜色。客户程序先告诉服务器需要什么颜色，然后服务器回应"象点值"——它表示调色板中最接近的颜色号。当创建图形内容时，可用此象点值指定画笔的前景。也可先创建缺省图形内容再申请颜色，但在已有的图形信息中设置前景值要多加一条请求。

一、打开连接

客户程序允许用户用指定主机和显示器号的方法来描述服务器。在个人工作站上，显示器号是0，因为只有一个键盘、光标、显示器与一个主机相连，从而只有一个服务器。现在尚不多见多用户工作站和支持图形终端的分时系统，但X提供了这种可能，一个主机可支持两个或多个服务器。

客户号的库函数应能提供便捷的方法指定与哪个服务器连接。在UNIX下，通过Xlib读取环境变量DISPLAY。用户用DISPLAY和服务器号指定服务器，中间用"："分开，例如ghost：0。而在Berkeley UNIX系统下，则用文件/etc/hosts(其它系统中，用yellow page daemon)将主机名翻译成网络地址。

对TCP的连接，指定主机上的显示器从0开始编号，显示器N的服务器接受来自6000+N口的连接。对DECnet的连接来说，同一主机上的显示器从0开始编排，服务器名包括"X$X"和数字，例如X$X0和X$X1。

一旦要建立连接，客户程序便发送描述自己的字节。然后服务器如果接受连接则回应自己的信息，如果拒绝连接则回应错误信息。

# off2K Bytes	Type	Values	Description
1		102(MSB first) 154(LSB first)	byte-order
1			unused
2	unsigned integer		protocol-major-version
2	unsigned integer		protocol-minor-version
2		n	length of authorization-protocol-name
2		d	length of authorization-protocol-data
2			unused
n	list of unsigned integers		authorization-protocol-name
p			unused,p=pad(n)
d	list of unsigned integers		authorization-protocol-data
q			unused,q=pad(d)

数据的第一个字节表示客户机采用的位模式。102(ASCⅡ码B)表示高位优先，154(ASCⅡ码l)表示低位优先。除了图象数据无论16位还是32位都以这两种次序传送。图象数据将在后面介绍。然后，客户程序告诉服务器希望它采用什么样的协议版本。主号为11，次号为0(X版11从1到4都是这样)。版本号对以后协议升级很有用。通常，主号在发生不可移植升级时增长，次号在不影响移植的小改动时升级。服务器回应自己支持的协议版本号，它可能与客户程序发送的不一样。服务器可以(但不必须)因版本号不一样而拒绝连接。服务器也可以(但不必须)同时支持几个版本。

权限名表示客户程序希望服务器采用的权限以及协议数据的权限。指定权限不是基本 X 协议的一部分。

因为 X 的协议长度是 4 字节的整数倍，16 位和 32 位数据都以 16 位、32 位边界对齐，所以存在多余字节。但这有助于结构化。

为了理解返回的连接信息，必须知道 X 服务器可同时支持几个屏幕。X 服务器可能以不同的屏幕向单个用户发送信息。使用两个屏幕的例子是用户可同时在单色和彩色屏幕上调试程序，但屏幕由一个服务器控制。彩色系统下的典型服务器可在同一物理屏幕上支持两个逻辑屏幕，一个是彩色的，一个是单色的(速度快)。用户可通过光标移过边界来切换屏幕。协议中不包含屏幕尺寸。光标的切换由服务器控制。

连接信息分别描述每一个屏幕。因为屏幕可有任意多个，所以这种信息来回重复。甚至每一屏幕还可以不同方式使用。例如，彩色屏幕也可显示黑白窗口。这种使用方法叫做 visual。对每一屏幕都有可使用的不同方法的信息。

visual 概念的提出有许多优点。如果知道某一窗口准备在彩色屏幕上以黑白显示(例如，终端窗口)，就可以这种方式对待它，从而可提高效应，因为对每一象点只采用 1 位，而不是 24 位。许多服务器可利用这种特点来提高效应。

我们将在 CreateWindow 请求中看到，窗口是以一定的 visual 创建的，且不可改变。因为可以不同方式使用同一彩色屏幕，一部分信息必须重复以表示每种方式。每一屏幕的描述信息包括多少象点宽、多少象点高、多少微米宽、多少微米高的根窗口(不可改变)。

每一屏幕还包含一个缺省的调色板，它至少有两个固定的元素：黑象点和白象点，这可在单色和彩色屏幕上实现单色应用。许多屏幕的实际黑象点和白象点的 RGB 值是可设置的，有些情况下不是实际的黑和白。当客户程序产生的 ID 以及不同客户程序产生的 ID 都不相同时，resource _in _mask 和 resource _in _base 是必须的。因为服务器统一管理 ID，所以无论是否是同一客户程序采用的或者是否是同种类型的，都应有区别。resource _in _mask 为 32 位，至少应有 18 位被设置。客户程序访问 ID 时用这些位的一个字节与 resource _in _base 进行或运算。要访问下一个 ID，只需增加 resource _in _mask 子集的号码即可实现。这种局部 ID 访问很重要，因为它模拟了创建时的来回应答，从而加快了起动时间。

Maximum _request _lentgth 表示请求的最大长度，以 4 字节为单位，这受限于服务器的可用内存。当请求的最大长度超出此长度时，会产生 BadLength 错误，服务器将消除该请求。通常最大长度为 16384 字节。

X 服务器应能按不同机器次序交换数据字节，尤其是在交换图象数据时更需这样。客户程序打开服务器发送的请求后其第一字节表明本机的字节次序。

图象数据来去服务器一般产生服务器的字节次序，因为图象数据是按坐标增长的。客户程序在收到服务器的回应时得知服务器的字节次序，于以后处理图象时使用。

二、创建窗口

一但与服务器建立连接，客户程序要做的第一件事情就是创建窗口。

CreateWindow 请求是所有协议中最复杂的，但具有同样的结构：包括一个表示操作码的数据块以及一些定长参数和变长参数。所有请求均以 8 位主操作码开头，后跟 16 位的长度字节，并以 4 字节为单位。长度域包括操作码和长度域的请求长度。它必须等于请求所要求的最小长度，否则将

发生错误。

从128到255的主操作码被保留供扩充使用。每个扩充可用于多个请求，一个特定扩充的所有主操作码相同，因此扩充需要使用一个附加的次操作码，它紧跟在长度域的后面。

GreateWindow的有趣特征是它的数据传输需要可变长。CreateWindow的定长部分包括父窗口ID，本窗口ID、窗口大小、位置、边框、窗口类型(InputOutput或InputOnly)以及窗口的visual(由服务器产生的ID，表示如何使用屏幕)。定长部分的最后是一个位码，它表示可变长部分的使用情况。可变长部分一般为位信息，服务器对没有设置的位使用缺省值。位码表示剩下的将出现什么条目。

可变部分一般是窗口属性，用于控制：

- 背景色，边框色或模板。
- 是否在窗口改变大小时要存贮内容以及重心放在何处。
- 窗口改变大小时，子窗口是否移动以及如何移动。
- 是否要求服务器备份存贮窗口的内容。
- 是否要求服务器存贮临时窗口以便在重新显示时加速重画。
- 什么事件在窗口中发生时要通知客户程序。
- 不需向高层窗口传送的事件。
- 是否要求窗口管理程序将其覆盖重定向。
- 使用哪个调色板。
- 使用什么光标。

所有客户程序都要设置的属性是事件表征码。

服务器可向客户程序发送许多种事件，每一事件都包括一种用户动作和请求的副效应。但客户程序并不是对所有的事件都感兴趣，所以窗口具有描述需要什么事件的属性。

事件表征码可在CreateWindow中设置，也可作为ChangeWindowAttributes的一部分来设置。然而，如果在创建窗口时知道需要什么样的事件，最好是在创建时指定。注意创建窗口与单独指定事件的时间延迟不是一个问题，因为在窗口还没被显示时无法发生事件。窗口直到映射时才显示。

三、带回应的请求

有些请求要求服务器立刻作出回应，因为客户程序缺少这些信息将无法继续工作。大部分这样的请求是要从服务器得到如窗口、字体和特性的句柄。另一些是要求报告成功与否以决定客户程序能否安全继续运行。也有一些请求有这样的双重要求。回应通常是立即发生的。

作为例子在此说明申请颜色的情况。客户程序用调色板中的元素指定颜色，称其为象点值。客户程序要为一颜色向服务器询问它的象点值，因为只有服务器知道每一元素的颜色。X服务器能保留几种调色板，根据硬件情况它可安装一个或多个调色板。每个调色板具有一个ID以及一些是否可读写的信息。只读调色板具有固定颜色，可被共享，因为没有客户程序能修改它。通常客户程序在调色板中找不到精确的颜色。读写颜色板的元素为客户程序私有。X用红、绿、蓝三元色值指定颜色，每个为16位。

AllocColor请求从只读调色板中申请颜色，因此它可作用于任何调色板。但是，有时不一定存

在客户程序所要求的颜色，在这种情况下，它将用服务器调色板中最接近的颜色代替之。如果调色板是单色的，则只有黑白两色是可用的。AllocColor 的回应将告诉客户程序最接近的颜色，以及实际存贮的三元色值。客户程序依此决定它是否已足够接近所申请的颜色。

AllocColor 请求

# of Bytes	Type	Values	Description
1		84	opcode
1			unused
2		4	request length
4	COLORMAP		colormap ID
2	unsigned integer		red
2	unsigned integer		green
2	unsigned integer		blue
2			unused

AllocColor 请求指定要使用哪个调色板以及希望颜色的三元色。服务器回应调色板中的只读色素及实际的三元色值。

回应的操作码通常为 1，错误时为 0。内核事件从 2 到 34。系列号是服务器保留的上次发送此请求的计数。服务器发送的所有信息，包括回应、事件和错误，都有一个系列号域。

注意，象请求一样，回应即使是定长时也有一个长度域。这使得客户库函数能容易地正确处理请求和回应，因为无需通过查表得到每一操作码的长度。从而简化了客户库函数在网络上传送无用字节的编程。

服务器回应

# of Bytes	Type	Values	Description
1		1	reply opcode
1			unused
2	unsigned integer		sequence number
4		0	reply length
2	unsigned integer		red
2	unsigned integer		green
2	unsigned integer		blue
2			unused
4	unsigned integer		pixel value
12			unused

和请求中的情况一样，长度域以 4 字节为单位。无用字节不必为 0。

下面列出所有需要回应的请求：

AllocColor	GetSelectionOwner	QueryBestSize
GetAtomName	GetWindowAttributes	QueryColors

GetGeometry
GetImage
GetKeyboardControl
GetKeyboardMapping
GetModifierMapping
GetMotionEvents
GetPointerControl
GetPointerMapping
GetProperty
GetScreenSaver
GrabKeyboard
GrabPointer
InternAtom
ListExtensions
ListFonts
ListHosts
ListInstalledColormaps
ListProperties
LookupColor
QueryExtension
QueryFont
QueryKeymap
QueryPointer
QueryTextExtents
QueryTree
SetModifierMapping
SetPointerMapping
TranslateCoordinates

四、创建图形上下文

图形上下文(GC)是控制服务器如何解释图形请求的句柄。GC 控制线宽、线如何连接、如何结束、使用什么颜色、显示器回响什么层、与屏幕上的已有内容如何运算、区域如何填充和拼贴等。

GC 应尽早一些创建，以减少用户的影响次数。

CreateGC 与 CreateWindow 请求相比要简单得多，其中的一个成员是位码，定义请求的长度和剩余部分的组成。只有要设置而不采用的值在请求中占用空间。

五、映射窗口

映射使得窗口成为可见的。通常情况下，应用程序工作后，映射实现显示窗口。但窗口是否显示还依赖于以下条件：

- 窗口应由 MapWindow 映射过。
- 所有窗口的祖先已被映射。
- 窗口必须在不被遮住的位置。如果窗口被遮住，则它是否被显示依赖于窗口的栈次序。栈次序可用 ConfigureWindow 调整。
- 客户库函数请求的缓存必须清空。
- 顶层窗口的初始映射是个特例，因为窗口的可见性被窗口管理程序延迟。更复杂地，客户必须等待第一个 Expose 到来才能确定窗口被映射并可向其中画图。

六、Expose 事件

从客户的角度看，唯一表明窗口可见的是服务器向其发送 Expose 事件。客户程序只有收到 Expose 事件后才能向窗口绘画。当窗口遇到上述情况时，服务器可发送一个或多个 Expose 事件。之所以需要不只一个 Expose 事件是因为每一个 Expose 描述一个要显示的矩形，其中有一些表示没有被重叠的窗口。

下面是 Expose 事件的信息。都是 32 字节长。

# of Bytes	Type	Values	Description
1		12	code
1			unused
2	unsigned integer		sequence number
4	WINDOW		window
2	unsigned integer		x
2	unsigned integer		y
2	unsigned integer		width
2	unsigned integer		height
2	unsigned integer		count
14			unused

code 表是事件类型。系列号是服务器指定的最近的请求号，用于跟踪错误。窗口域表示哪个窗口暴露，x、y、width 和 height 域表示窗口中显露的区域。计算域指的是由同一请求引起多少暴露事件。

在窗口程序中，事件在一个封闭的循环中收集和处理。Expose 事件也在其中处理。客户程序可以但不必提供退出手段，因为有一个客户程序(xkill 可用来杀掉所执行的程序)。

七、画图

Expose 事件告诉客户程序"可以继续画"，它并非只是在窗口每一次显示时才有，在以后窗口由不可见变为可见时都会有，即仍然要象第一次一样重画。

画连续线的协议请求 PolyLine 如下：

# of Bytes	Type	Values	Description
1		65	opcode
1			coordinate-mode
		0(Origin)	
		1(Previous)	
2		3+n	request length
4	DRAWABLE		drawable
4	GCONTEXT		gc
4n	LISTofPOINT(pair of signed integers		points

opcode 表示该请求是 PolyLine。相关码表示坐标是相对于窗口的 Origin 还是表中前面的线。下面是请求长度，表示表中有多少点。再下面是表示画向什么可画物(窗口或三维象点图)ID 以及用于解释请求的图形内容 ID。最后是点的列表。每一个点用一对 16 位有符号的整数表示，因为 8 位不足以表示屏幕的所有点(1000 左右)。