X服务器总是位于用户的计算机内部，而X客户有可能在远方的远程计算服务器上，这看起来也许有些不可思议，但是X服务器的主要工作是在屏幕上显示位，所以让它靠近用户是有道理的。从程序的观点来看，它是一个客户，吩咐服务器做事情，例如显示文本和几何图形。服务器（在本地PC中）只是做客户吩咐它做的事情，就像所有服务器所做的那样。

对于X客户和X服务器在不同机器上的情形，客户与服务器的布置如图5-37所示。但是当在单一的机器上运行Gnome或者KDE时，客户只是使用X库与相同机器上的X服务器进行会话的某些应用程序（但是通过套接字使用TCP连接，与远程情形中所做的工作相同）。

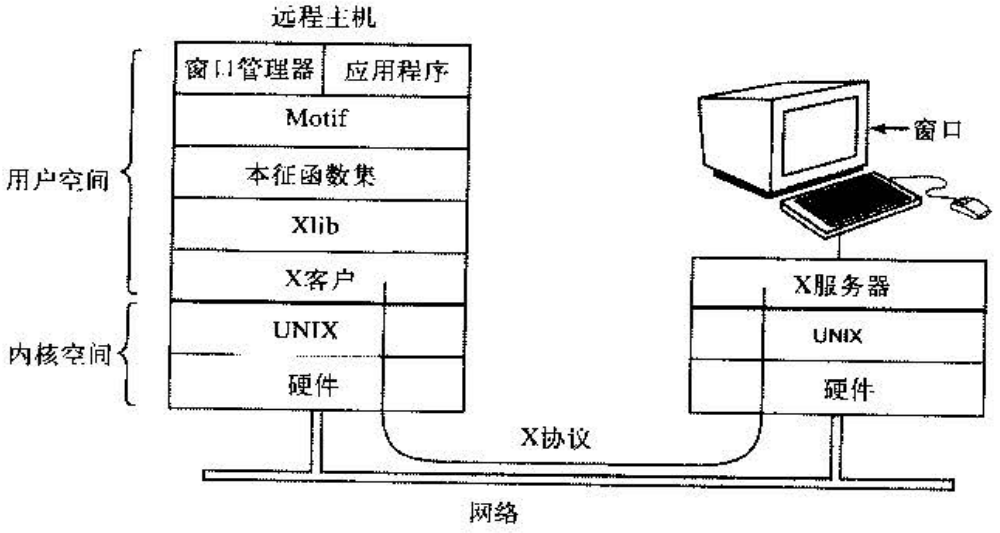

图5-37 MIT X窗口系统中的客户与服务器

在单机上或者通过网络在UNIX（或其他操作系统）之上运行X窗口系统都是可行的，其原因在于X实际上定义的是X客户与X服务器之间的X协议，如图5-37所示。客户与服务器是在同一台机器上，还是通过一个局域网隔开了100m，或者是相距几千公里并且通过Internet相连接都无关紧要。在所有这些情况下，协议与系统操作都是完全相同的。

X只是一个窗口系统，它不是一个完全的GUI。为了获得完全的GUI，要在其上运行其他软件层。一层是Xlib，它是一组库过程，用于访问X的功能。这些过程形成了X窗口系统的基础，我们将在下面对其进行分析，但是这些过程过于原始了，以至于大多数用户程序不能直接访问它们。例如，每次鼠标点击是单独报告的，所以确定两次点击实际上形成了双击必须在Xlib之上处理。

为了使得对X的编程更加容易，作为X的一部分提供了一个工具包，组成了Intrinsics（本征函数集）。这一层管理按钮、滚动条以及其他称为窗口小部件（widget）的GUI元素。为了产生真正的GUI界面，具有一致的外观与感觉，还需要另外一层软件（或者几层软件）。一个例子是Motif，如图5-37所示，它是Solaris和其他商业UNIX系统上使用的公共桌面环境（Common Desktop Environment）的基础。大多数应用程序利用的是对Motif的调用，而不是对Xlib的调用。Gnome和KDE具有与图5-37相类似的结构，只是库有所不同。Gnome使用GTK+库，KDE使用Qt库。拥有两个GUI是否比一个好是有争议的。

此外，值得注意的是窗口管理并不是X本身的组成部分。将其遗漏的决策完全是故意的。一个单独的客户进程，称为窗口管理器（window manager），控制着屏幕上窗口的创建、删除以及移动。为了管理窗口，窗口管理器要发送命令到X服务器告诉它做什么。窗口管理器经常运行在与X客户相同的机器上，但是理论上它可以运行在任何地方。

这一模块化设计，包括若干层和多个程序，使得X高度可移植和高度灵活。它已经被移植到UNIX的大多数版本上，包括Solaris、BSD的所有派生版本、AIX、Linux等，这就使得对于应用程序开发人员来说在多种平台上拥有标准的用户界面成为可能。它还被移植到其他操作系统上。相反，在Windows中，窗口与GUI系统在GDI中混合在一起并且处于内核之中，这使得它们维护起来十分困难，并且当然是不可移植的。

现在让我们像是从Xlib层观察那样来简略地看一看X。当一个X程序启动时，它打开一个到一个或