客户机／服务器计算对人们来说有许多含义。通常，客户机／服务器计算是一种应用，它具有运行于用户本地计算机（如一台 PC 机、Macintosh 或工作站）上的客户部分，以及运行在诸如基于 Unix 的服务器或主机的大型计算机上的服务器部分。客户机／服务器大多是以数据库为中心。本书讨论基于 Intranet 的客户机／服务器应用——在局域网上应用 WWW(World Wide Web)的技术。

最近有一则关于客户机／服务器计算的评论，其中将当前客户机／服务器的状况比做“流行年”，这表明客户机／服务器技术正在迅速发展着。似乎每天都会有与客户机／服务器相关的产品发布新特性或升级的消息，全球几乎每个软件包都在增加对 Web 的支持，或者至少增加 HTML 输出的选项。

Java 正在走向成熟，人们不断将一些新功能加入到这种可视化语言的基础代码或核心代码中，如支持数据库的访问功能，而仅仅在一年前这些功能还需要第三方厂家来提供。在前两年关于客户机／服务器的书中谈论的内容不再是如今的热门话题。

因此，如何使这本书不在写完出版之时就过时呢？具体的答案集中在本书有关基于 Web 的 Intranet 客户机／服务器的章节中。本书介绍的技术和方法强调实用性，开放的解决方案不仅适用于今天，而且适用于今后几年。厂商在不断推出基于客户机／服务器应用的自动开发商业软件包的同时，还需提供特定的客户机／服务器解决方案。本书介绍的基于 Web 的一般方法，能够有效地用于许多客户机／服务器结构的应用中。随着 Web 的剧增，许多公司开始考虑用基于 Web 的工具代替传统工具（如 PowerBuilder 或 Delphi）来开发客户机／服务器应用。基于 Web 的方法有许多优点，它比专用快速开发工具（如原型法或产生式系统）便宜，并有众多的支持 HTML 输出的商业软件包可供选择。本书将讨论如何选择这些工具，还通过几个带有完整源代码的实例研究详细说明如何开发 Intranet／Internet 应用。

本书主要考察 Intranet 客户机／服务器应用，这是那些关心 Web 上的客户机／服务器应用的读者感兴趣的。无论是面临基于 Web 的 Intranet 应用开发的程序员，还是需要具备在基于 Web 的客户机／服务器和传统的客户机／服务器之间进行决策能力的系统管理员都将从书中学到许多有用的东西。

本书不讨论客户机／服务器计算的硬件方面的内容，有关客户机

和服务器平台的选择也不在本书的讨论范围。基于 Web 的客户机 / 服务器应用大多运行在较现代的计算机上。对于 Web 应用的设计稍加注意，就可以继续沿用原有的客户机 / 服务器平台。其它方面的考虑如数据库服务器的选择也会涉及到硬件方面的问题。

本书最后的附录将提供有关 HTML 文件类型定义的信息。

1.1 客户机／服务器方法

可以实现客户机 / 服务器应用的方法有许多，有瘦客户机与胖服务器、胖客户机与任意规模的服务器等，有二层、三层和多层方法等。详细内容将在本书后续章节逐一讨论。

客户机和服务器的瘦与胖实际上是应用处理在客户机和服务器之间进行分配的另一种说法，瘦客户机只有少量的应用处理，主要是处理用户界面；胖客户机包含一些特定的应用处理，如字段核实或与数据库的连接。胖瘦的程度可以有不同的变化，对于较瘦的客户机，大部分应用必须驻留在服务器上。

人们常常用层次的观点来描述客户机 / 服务器的应用，层次既可以对应于硬件也可以对应于软件，但通常是指软件。层次描述了客户机与服务器之间应用的划分。需要特别指出的是，尽管大多数应用都是二层和三层的，但专家认为可以有许多可能的层次。二层只有客户机层和服务器层，三层包括客户机层、应用处理层和服务器层（通常是数据库服务器）。

1.2 词 汇 表

初步看来，客户机 / 服务器应用显得并不太复杂，无非是有一个客户机、一个服务器。客户机负责提供用户界面，而服务器则提供数据服务。但更深入地考察，就会发现客户机 / 服务器计算实际上是很复杂的。仅从概念上讲，就出现了许多全新的词汇、缩略语和时髦话题。

这里所列的许多术语已经非常流行，人人都在使用这些最新的词语，但却很少有人理解它的含义。客户机 / 服务器术语肯定也属于这一类情形。本章剩下部分将列出一些主要的客户机 / 服务器术语，并给出其定义。

API

应用程序接口（Application Program Interface）。

客户（Client）

应用程序中与用户交互作用的部分。

CORBA

通用对象请求代理结构（Common Object Request Broker Architecture）。

DBMS

数据库管理系统（Database Management Systems）。

GUI

图形用户界面（Graphical User Interface）。

Internet

以 Internet 互连网络协议为基础组成的网络，它首先是由 Unix 的 TCP / IP 等协议发展起来的。从 ARPAnet 开始，Internet 已经发展成为一个全球性的、可以进行公共访问的计算机网络。Internet 上较有代表性的工具有 FTP、telnet 以及基于 WWW 的浏览器和服务器。

Intranet

以 Internet 网络协议为基础，通常使用 WWW 工具。Internet 与 Intranet 网络的主要区别是 Intranet 一般通过防火墙将公共访问与内部访问隔离开来。

LAN

局域网（Local Area Network）。

中间件（Middleware）

常常被称为是客户机 / 服务器应用结构中的胶合物，它通常在客户机与服务器之间提供通信接口。

NFS

网络文件系统（Network File System）。

面向对象

一种软件开发方法，它将数据模型化为对象和过程，操作作用于被调用的数据之上。

ODBC

开放数据库链接协议（Open Database Connectivity）。

OLE

对象链接嵌入（Object Linking and Embedding）。

RAD

快速应用开发（Rapid Application Development）。

RPC

远过程调用（Remote Procedure Call）。

服务器（Server）

应用程序中用来响应客户的请求、提供信息和服务的部分，单个服务器通常可以为多个客户机服务。

SQL

结构化查询语言（Structured Query Language）。

SSL

安全套接层协议（Secure Socket Layer）。

瘦客户机

除了处理用户界面外，实际只能完成很少一点应用处理的客户机，用户界面是像 X-Windows 这样的 GUI 程序。

瘦服务器

通常与胖客户机配合、只能完成少量应用处理的服务器。

胖客户机

能完成较多应用处理的客户机。

胖服务器

通常与瘦客户机配合，能完成大量应用处理的服务器。

层次（Tier）

区分不同结构应用程序的抽象模型，它可以是逻辑上的，代表了软件功能的划分和硬件的分配。

1.3 本书的风格与结构

本书的每一章涵盖一个单独的主题，它由简介及相应章节的主题等内容组成，在适当的地方还将列举一些实例。简介将给出所涉及主题的正式定义，某些章节可能会包括其它章节所需的有关主题的完整解释。

实例研究所示的程序一般是从多种语言版本中筛选出来的版本，而讨论可能要用到所有的语言版本或某一语言版本。标识符和保留字用斜体打印。用“-”连接的两个组合键代表一个转换键与一个字母键的组合，转换键包括 Alt、Ctrl 和 Shift。例如，按下 Ctrl-C 退出表示在保持按下 Ctrl 键的同时按下 C 键。

1.4 代码风格

在本书例子中的代码格式清晰明了，尽可能采用 C 语言程序员的代码风格。例如，典型的测试一个 9 个字符的字符串串长的方法是：

```
if(strlen(xxx)==9){ ... }
```

书中实现该功能的代码如下：

```
len = strlen(xxx);
if(len == 9){
 ...
 }
```

代码段包含有少量的注释语言，并与正文的叙述区别开来。现在普遍被人们接受的新观点是，代码的可读性比程序的效率更重要，这一点会体现在所有例子中。

客户机／服务器计算

本章主要讨论什么是客户机／服务器计算，并简要回顾客户机／服务器计算的历史，介绍“层次”体系结构，以及常用的客户机／服务器开发工具，如 Borland 公司的 Delphi 等。

2.1 什么是客户机／服务器计算

客户机／服务器是一个新的计算术语，它意味着“因人而异，各取所需”。一个普遍被人们接受的定义是，客户机／服务器的高层是一个应用实体，通常以数据库为中心。它分为处理用户界面、进行数据核实和错误检查的客户部分，以及响应客户对数据、资源请求的服务器部分。

当客户机／服务器开始成为热点时，文件服务器商就宣称他们的系统是 PC 客户的服务器，应用程序可以访问服务器上的任何文件。但现在，几乎没有人会认为这就是客户机／服务器计算，因为远程文件访问仅仅是局域网及其服务的一个基本内容。

目前，客户机／服务器一般是指包括一个客户部分和一个服务器部分的应用。客户完成以下一个或多个处理：

- 用户界面。
- 命令解释。
- 数据输入。
- 数据核实。
- 在线帮助。
- 错误恢复。

服务器完成以下几个处理：

- 响应客户请求。
- 同时为多个客户服务。
- 执行函数关联操作。
- 解决冲突和记录锁定。
- 操作共享数据。
- 管理共享资源。

服务器通常是被动的，它总是等待客户的服务请求。图 2-1 从概念上说明了客户与服务器的关系，一个服务器常常可以处理多个客户

的服务请求。

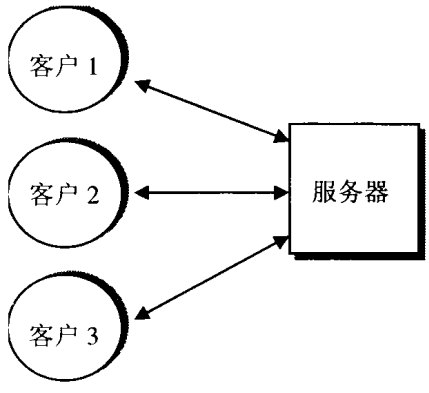

图 2-1 一个服务器处理多个客户请求

2.1.1 用户界面

客户负责处理用户界面，它包括现在的 PC 客户计算机上的屏幕处理或图形用户界面（GUI）。显示视窗、鼠标、键盘等都属于用户界面处理。如果应用程序中有多媒体部分，显示视窗处理中还包括多媒体功能。

2.1.2 命令解释

客户应用程序包含一定的描述功能，因而必须具备命令解释能力。许多客户应用程序在按下操作键或选择了超级链接后，必须能够对参数进行解释并运行相应的过程。这就是命令解释。

2.1.3 数据输入

实际上，所有的客户机 / 服务器应用都需要进行一些数据输入，不仅仅是向一个表单中的变量赋值，还包括应用程序本身所需的数据输入。虽然输入的可能只是简单的 E-mail 地址或一个查询串，但必须用到数据输入过程。

2.1.4 数据核实

当数据输入表单后，常常需要进行初始核实。核查范围从确认输入数据所需的字段到日期的检查和内容的核查等。但在这里核实只涉及到在客户应用程序中进行的检查。

2.1.5 在线帮助

目前，大部分交互式应用系统中都包含了交互式帮助功能，视窗应用程序一般也包含帮助文件以提供上下文含义有关的帮助。HTML、PDF、Envoy 或其它电子文档文件也可以与相关的观察器一起提供给客户。需要注意的是，这些程序都是运行在客户机上，而不是服务器上。

2.1.6 错误恢复

在客户机 / 服务器应用程序运行时，需要做许多错误恢复工作。包括数据错误、通信错

误、硬件故障或其它各种错误，对这些错误必须进行适当处理。

2.2　服务器的种类

服务器有许多种，由于大多数客户机／服务器应用都是以数据库为中心，因此许多人都将“服务器”看作是数据库服务器的同义语。典型的服务器有：

- 数据库服务器。
- 文件服务器。
- 计算服务器。
- 显示服务器。

大多数服务器都有一个共同的特征，即运行于多任务操作系统环境下。由于服务器同时要处理多个客户的应用，因此必须具备多任务处理能力。

2.2.1 数据库服务器

顾名思义，数据库服务器就是为客户提供数据库访问服务的服务器，它总是等待客户发出的访问请求和其它命令，然后对指定的数据库进行操作。因此，数据库服务器也常常称为数据库引擎。Oracle、SQL-Sever、Informix 和 SyBase 是目前流行的用于客户机／服务器应用的数据库服务程序。

2.2.2 文件服务器

文件服务器能为客户提供透明的文件访问服务。其中首先使用客户机／服务器术语的是文件服务器商。典型的低档文件服务器是运行于 Novell NetWare、OS／2 或 Windows NT 的 PC 服务器，中档文件服务器一般能经由 NFS 访问 UNIX 系统，主机系统中最重要的因素是速度和存储能力。客户应用程序可以像访问本地磁盘一样访问文件服务器上的文件，而无须知道两者的差异。

2.2.3 计算服务器

从软件的观点来看，计算服务器是专用的，它能完成某种对计算或资源大量需求的操作。一般情况下，计算服务器需要使用专用的硬件。

2.2.4 显示服务器

显示服务器是非常特殊的系统。常见的 X-Window 服务器通常只能运行在 UNIX 工作站上。

2.3　客户机／服务器的历史

开始，并不存在客户机／服务器计算结构，只是通过专用通信线路将哑终端连接到一个主机上，常用线路如 RS-232，后来也使用 RS-422 线路。哑终端为用户提供键盘和显示器，

显示器具有以字符为基本单位的处理能力，可以对光标进行编址，但所显示的字符必须存储在终端的 ROM 中。所有应用程序相当于主机中一个独立运行的单元，这种计算模式常称为单机模式，如图 2-2 所示，也可以把它看成是一级计算。初期的客户机 / 服务器应用通常需要在客户与服务器之间传送大量的数据，后来的二层和大多数三层应用转向了消息传送方案，使得在任一时刻能以最小的网络连接时间传送少量的数据。毋庸讳言，采用数据传送和消息传送将影响到网络负载的大小。也许在用户较少、网络负载较轻时不易看出这种差别，但当用户增多时会明显降低网络的性能。

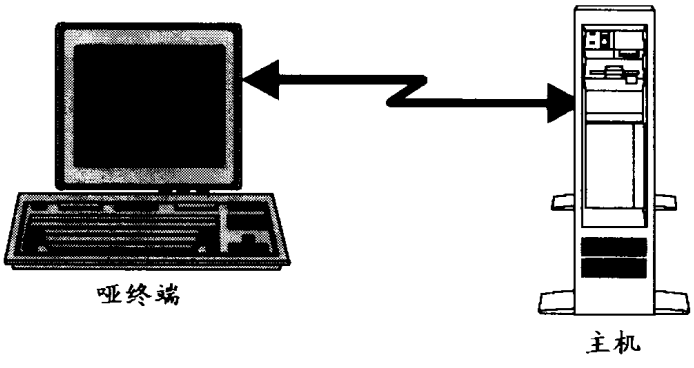

图 2-2　单机计算模型

2.3.1 二层客户机 / 服务器计算

第一代客户机 / 服务器应用通常是用二层逻辑实现的。这一计算模式中具有两级硬件结构，如图 2-3 所示，即客户和服务器不是运行在同一台计算机上。二层的客户机 / 服务器结构将应用程序分成完全不同的两个部分，一部分运行在客户机上，另一部分运行在服务器上。但需要注意的是，客户和服务器的程序代码既不关心也不知道它们是在同一台计算机上运行还是在不同的计算机上运行。因此，应用程序被客户和服务器分割开了。

运行于客户或服务器上的应用逻辑的数量决定了系统的“瘦”与“胖”。瘦是指系统中只有少量的应用进程在运行，而胖是指系统中有大量的应用逻辑存在。瘦与胖的程度可以有不同的变化，一个瘦客户程序通常只能处理很少的应用逻辑，而且主要是通过 GUI 来解决用户间的交互关系。当客户机（通常是老的 PC 机）性能有限时，瘦客户是有吸引力的，如 386 或低档 486 就是这类客户机的典型代表，它们没有足够的 MIPS 来处理应用进程中的 GUI、通信及其它关键部分。

二层的客户机 / 服务器结构在系统开发和维护方面已经很难满足人们原先的期望了，它常常不能很好地升级。当应用系统的复杂性和用户数量增加时，要在客户与服务器之间新增服务器或重新划分应用进程，可能需要对系统进行大量修改。并且，客户机 / 服务器应用开发工具还需时间的考验。现在，许多数据库和工具软件商都能提供支持二层客户机 / 服务器应用系统开发的软件包。Visual Basic、Delphi 和 PowerBuilder 等都是这类开发工具的代表，它们大多正在转向支持三层的客户机 / 服务器结构。

2.3.2 中间件

中间件是用来描述客户与服务器之间通信的“胶合”的一个术语。如图 2-4 所示，应用

系统的客户部分通过 API 与中间件相连，中间件负责与服务器的通信。它避免了应用程序员直接通过低层操作系统和硬件与服务器通信所必须应付的复杂接口问题的麻烦，这对于应用系统是非常重要的。

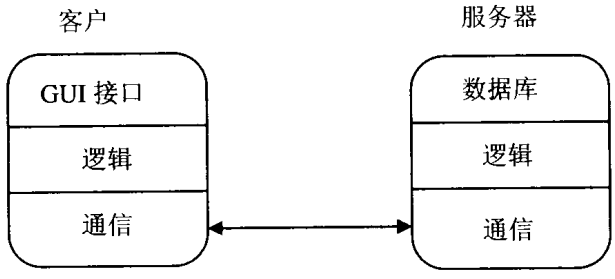

图 2-3　二层客户机 / 服务器模型

一般而言，用中间件来处理通信问题是常见的途径，它可以用于任何应用系统。中间件通常就像是主机中负责客户通信和与服务器软件接口的一种软件，中间件常有以下几种：

- TCP/IP。
- DCE。
- NetBIOS。
- 名字管道。
- LAN Manager。

专用中间件是一般中间件的改进，它能够处理专用服务器的通信。如用于数据库服务器的 ODBC、用于邮件服务器的 SNMP 和用于 Web 服务器的 SSL。这一层次的中间件必须运行在一般中间件的协议之上，像 HTTP 运行于 TCP/IP 之上就是一个典型的例子。

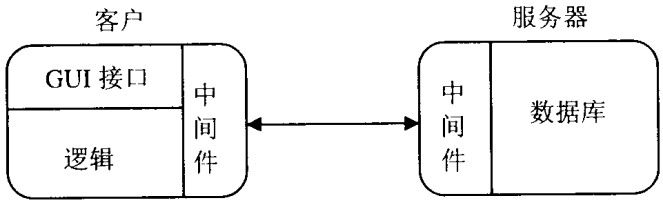

图 2-4　中间件模型

2.3.3 三层客户机 / 服务器计算

客户机 / 服务器计算的最新进展是如图 2-5 所示的三层结构，人们也用多层结构来统称两层以上的逻辑结构。

目前，多级客户机 / 服务器应用的开发可以采用几种不同的方案，最常见的是继续沿用二层结构的工具来开发客户端的 GUI 和进程。第二个方案也许会很快流行起来，尤其是对于大型项目，它集各种客户机 / 服务器应用的开发工具于一体，如 TI 公司的 IEF。第三个方案是在前端采用二层开发工具，如 Visual Basic 或 PowerBuilder，而在后端采用集成的软件包来生成服务器应用代码和建立数据库连接。

二层应用和三层应用之间最主要的区别是服务器上附加的软件层。二层应用侧重于将应用逻辑放入客户端和向数据库中传送记录（胖客户模式），或者在存储过程中向数据库传送

数据并由数据库引擎实现应用逻辑（瘦客户模式）。三层应用侧重于在客户与服务器的应用代码之间传送消息，由服务器部分实现应用逻辑，然后向数据库发送记录。应用逻辑在客户机 / 服务器体系中常被称为是“事务规则”。

由于第三层结构是应用开发中附加的代码，从而使系统更加复杂。服务器代码的开发工具和编程语言更多地依赖于服务器平台。Visual Basic 不适用于 Sun Sparc 服务器的应用层，一个运行于 Windows NT 的基于 Intel 的服务器可以有效地利用 Delphi 进行编程，而一个 UNIX 服务器可能要用 C 或 C++来编程。简而言之，没有哪一种语言适用于客户机 / 服务器应用中的所有服务器，也不能把客户端的语言，强行用于服务器的开发。

三层体系的服务器部分增加了一些应用的复杂性。然而，这对于三层客户机 / 服务器的性能来说是有益的。其中包括：

- 升级能力。
- 低通信流量。
- 可扩充性。

升级能力的提高是由于服务器代码与数据库的分离，它们可以在一个主机上启动然后再分开。多个应用服务器可以同时与中心数据库会话，当系统升级时，应用服务器仍然可以在访问多个数据库的同时为客户提供服务。三层模块的重组比在二层客户机 / 服务器模式下容易得多。

低通信流量是由于应用程序中只传送少量报文而不传送整个数据记录。

可扩充性的增加是由于客户、服务器和数据库系统可以单独更换而不影响其它部分，同时保证接口不变。例如，将数据库从 SyBase 换成 Oracle，只会影响到应用中的服务器部分而不会影响到客户；将客户程序由 Visual Basic 改为 Delphi，不会影响应用的其余部分，而且所提供的接口也是适用的。

现在，许多以数据库为中心的商用应用系统，包括 SAP 公司的 R/3 都使用了三层客户机 / 服务器模式，以获得较好的升级能力和可扩充性。基于 Web 的 Intranet 应用系统也大多是三层的客户机 / 服务器应用。

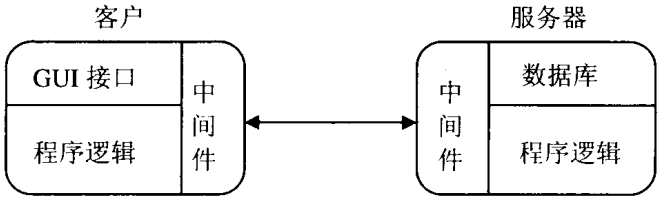

图 2-5　三层客户机 / 服务器模型

2.4　流行的客户机／服务器开发工具

现在的计算机市场上，有大量的客户机 / 服务器开发工具可供选择。表 2-1 例举了几种最流行的非 Web 开发工具，这些工具大多都能完美地用于二层应用的开发。

表 2-1　几种最流行的非 Web 开发工具

语　言	说　明
Borland Delphi	是一种面向对象的“快速应用开发”工具，这种编程语言建立在 Borland 公司的面向对象的 Pascal 的基础上，能生成 PC 机的源代码。开发环境是“可视化”的，这种思想最初用于 Visual Basic。同时还提供数据库和 ODBC 驱动程序
Visual Basic	是最早的可视化编程语言，它提供了建立客户应用程序的工具。第 4 版能够通过 ODBC 访问数据库。Visual Basic 唯一的缺憾是速度慢，因为它是解释性的。第 5 版在新增功能中会提供编译器
PowerBuilder	一种数据库应用系统的 GUI 前端开发工具，是最早的建立客户程序的工具。它具有跨越平台操作的能力，可以运行在 Windows 95、基于 Intel 和 Alpha 的 Windows NT、Solaris(Unix)以及 Macintosh 客户机上，而且从第 5 版开始内置 Netscape 软件。ODBC 及其驱动程序可以用于 Oracle、Sybase System 11、Informix、SQL Server6 和 DB2 数据库
C/C++	是最早的支持客户机 / 服务器结构的编程语言。最新的 PC 机的 C/C++编译程序，如 Microsoft 公司的 Visual C++和 Borland C++都为客户程序的开发提供了类似 Visual Basic 的可视化编程环境，还可通过子程序库提供通信和数据库的能力。在增加可视化功能之前，这类开发工具就已经具有同 PowerBuilder 一样的结构清晰的优点
Developer/2000	是由 Oracle 公司研制的客户机 / 服务器开发工具，它包括设计和开发数据输入表单和由 Oracle 数据库生成报表的模块
Access	是 Microsoft 公司的基于 SQL 的个人数据库产品。由于它内置 ODBC 的功能，常用来作为其它 SQL 数据库系统（如 Oracle 和 SyBase）的前端客户部分
其它语言	其它几种编程语言，主要是面向对象的语言，如 Smalltalk 和 Eiffel 也能成功地用于客户机 / 服务器程序的开发，任何一种能够访问对象库的语言都可以有效地应用

2.5 小　结

客户机 / 服务器计算对于不同的人来说意味着不同的含义，它是一个热门话题，也是一种提高计算应用性能的有效方法。关于客户机 / 服务器应用可以概括为：

- 支持软件的协同处理。
- 客户与服务器可以运行在同一台计算机上。
- 多数通用的应用是网络数据库。
- 客户、服务器二者均有重要的应用。
- 客户一般请求和接受信息，而服务则由服务器提供。
- 服务器提供客户所请求的信息，并服务于客户。
- 客户先动作。
- 服务器对客户作反应。
- 客户处理与用户间的交互关系。
- 客户机 / 服务器应用使服务器和网络环境对用户来说是透明的。

成功地运用客户机 / 服务器的方法需要具有清晰的管理思路，要清楚一个应用打算做什么，以及如何去实现它。还需要大量的前期准备工作，这些工作甚至比普通的软件开发项目还要多。

本章概括了全书的所有内容，目的是试图说明客户机 / 服务器计算的应用范围。本书其余章节将详细论述这些基于 Web 应用的各个专题。

第三章

传统的客户机/服务器与基于Web的客户机/服务器

本章将对传统客户机/服务器和基于 Web 的客户机/服务器的开发方法进行比较。这里所说的传统客户机/服务器是指非 Web 的开发方法，在 Web 开发方法中，有客户的开发和服务器的开发，也可能存在多层应用。在如此众多的选择中，要作出一个合理的决定似乎是困难的。

3.1 传统的客户机/服务器

传统的客户机/服务器应用是以数据库为中心的，大多数的客户机/服务器开发工具也是以此为基础的。因此，如果从顶层开始，基本的客户机/服务器模型可以描述为图 3-1。

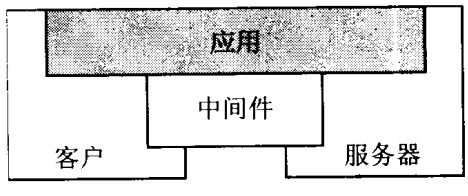

图 3-1 基本的客户机/服务器模型

下面就从图 3-1 开始这一节的讨论。客户机/服务器应用的开发需要进行大量的前期规划和分析，这甚至比一般的软件开发项目还要繁杂。无论是在开发开始之前还是作为初步分析阶段的一部分，都必须进行许多论证和决策。有关客户的问题应侧重以下几点：

- 硬件平台。
- 实现语言。
- 动态链接库（DLL）。
- 瘦或胖。
- 配置。

3.2 客户

图 3-2 说明了客户部分应考虑的主要问题。

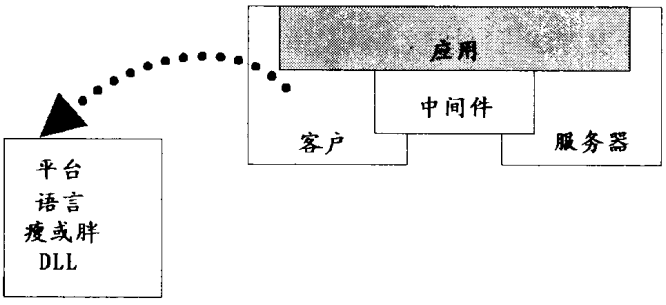

图 3-2 关于客户的主要问题

3.2.1 硬件平台

硬件与软件对于客户来说就是鸡和蛋的问题。在某些情况下，软件会决定客户的硬件；而其它情况下，硬件的选择将影响到软件。两个主要的决定因素是硬件的处理能力（MIPS）和网络带宽。

在当今的计算机界，一个带有 16 MB 内存的 100MHz 奔腾机和以太网对于普通用户来说是最小的基于 Intel 的客户配置。而程序开发员则需要高档的机器，如带有 32MB 或 64MB 内存、2GB 以上的大硬盘和 17 英寸或更大显示器的 200MHz 奔腾机。熟练的程序开发员通常在项目开发过程之外还要花费一些额外的时间，尤其是在项目开发的初期，经常要重新启动已有的客户计算机。要避免使用 386 PC 机或低档 Macintosh 机。一般来说，它们不具备必须的处理速度和硬件资源。落后的客户硬件只会挫伤用户的积极性，并增加开发新系统的阻力。

一些开发工具（如 PowerBuilder）是多平台的，可以工作在不同的客户环境下。多平台开发工具允许根据需要来配置硬件，而不是由特定的开发软件包来决定客户的硬件。最简单的解决方案是基于单位当前的组织结构来选择客户硬件，使用 PC 机的单位可能会选择 PC 客户机，而 Macintosh 或工作站商可能会坚持使用他们觉得最好的客户硬件。

3.2.2 实现语言

客户程序的实现语言或开发工具的选择可能会大大地影响硬件的选择，这也是一个相互关联的复杂问题，需要开发单位权衡利弊。这些影响因素包括：

- 开发人员的经验。
- 开发工具的费用（包括开发人员的许可和培训）。
- 硬件。
- 系统其它部分所选择的软件。

- 客户的瘦与胖。

还要使项目具有一定的可扩展性，可以采用 Visual Basic 来开发一个较小的、具备最基本功能和发展潜力有限的系统。如果事先知道是以客户为起点，最终扩展到单位的每一台桌面计算机上，那么，诸如 Delphi 或 Visual C++这样的功能强大、运行速度较快的编译语言也许会更能胜任这一工作。

语言的选择还应考虑开发人员的经验，绝不要选择一个开发人员不熟悉的语言，即使只有一个有经验的开发人员也会大大缩短全体员工的学习过程。当使用一种新的编程语言或开发工具时，要制订好进行培训的计划。

关于选择客户实现语言的最后一点需要说明的是，一旦选定某种语言或工具就会在整个项目的生命周期内把用户与该软件的开发商联系起来，选择 Visual Basic 意味着直接或间接地与 Microsoft 公司建立了联系，而选择 PowerBuilder 则意味着与 Powersoft 公司联系起来。

3.2.3 动态链接库（DLL）

动态链接库的问题只是在客户运行 Windows 环境（如 Windows3.1、Windows For Workgroups、Windows95 或 Windows NT）时才会出现，过去的几年中 DLL 的增长使得 DLL 的选择成为客户端所要考虑的主要问题。

DDL 一般包括基本系统库，如 Visual Basic 的运行时模块。开发商通常会随应用程序发布运行所需的 DLL，经常提供的 DLL 包括：

- Visual Basic 运行库。
- ODBC 驱动程序。
- Winsock。
- 图形转换库。
- 应用程序模块。

这是一个与配置管理和用户支持有关的问题。假如适当地设定用户 PC 机的功能，并将一个新软件包与一个较新的 ODBC 驱动程序装载在一起，更新的 DLL 将覆盖现有的被其它程序使用的 ODBC 驱动程序。这样，由于 DLL 的不同，一些旧的应用程序就不再起作用了。DLL 往往不能完全向下兼容。应用程序或许在有一个 DLL 错误时能正常工作，而当驱动程序的这个错误被纠正时却常常不能工作了。不同开发商的 DLL 驱动器之间也存在一些微妙的差别，这会对应用程序产生一定的影响。

除了事先了解可能存在的问题外，对于 DLL 的多版本问题没有什么有效的办法。有了这种认识，在出现这一问题时就有希望识别出来。在与开发商一起工作时，常常需要解决 DLL 的不兼容问题。不断更新 Windows 应用程序将有助于减少这类问题。

3.2.4 瘦客户与胖客户

瘦客户与胖客户的问题总是表面上的，由于它影响到开发工具的选择，因此一定要尽早确定。越瘦的客户所要求的最小速度和内存也越低，快速应用开发方法（RAD）就是以性能为代价的，解释型 Visual Basic 属于这类工具（具有代码编辑器的 Visual Basic 5.0 在本书出版时应该面市了）。

越胖的客户所用的最小速度和内存就越高，RAD 和性能之间的折衷左右着系统的性能，客户机 / 服务器开发工具的进一步发展是致力于同时提供快速应用开发方法和提高性能。

影响客户的瘦与胖的因素包括：

- 客户硬件，低档客户硬件对应瘦客户软件。
- 网络，重负荷的网络适合胖客户环境。

无论客户怎样配置，至少事先制订一个目标是很重要的，这样在决定项目开发的其它问题时能把对客户的选择一起考虑进来。

3.2.5 配置

在制订计划的初期就考虑应用系统的配置似乎显得令人费解，但当用户数非常多时它就显得特别重要了：

- 有那些安装需求？
- 怎样进行更新？
- 有没有标准的客户硬件配置？如果有，那么它们是怎样的？
- 标准的客户软件配置是怎样的？
- 是否与已知的 DLL 不兼容？
- 客户的安装能否交叉进行？

客户的自动安装已经实用化，现在有许多实用安装工具可以用于各种规模的应用程序。另一个可能的考虑是网络安装，胖客户也许需要通过网络进行控制以及不断更新其表格文件。此外，事先了解客户的构想能避免在以后的应用开发过程中遇到更多的麻烦。

3.2.6 小结

没有必要事先回答开发过程中可能遇到的所有问题，有些答案是显而易见的，有些则是棘手的。提出这些问题能给人以启发，考虑或者完全忽视配置问题，会使其它一些问题有不同的解决方案。

3.3 服 务 器

有关服务器的问题可用图 3-3 来描述。

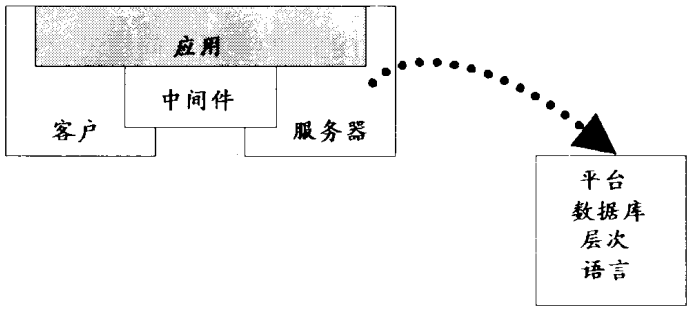

图 3-3 服务器的问题

3.3.1 硬件平台

从某种角度讲，选择服务器的硬件平台是比较容易的。就经验而论，可以选择你所能承受的具有最快的处理器、最大的内存和最大的硬盘的服务器。在实际选择中有许多因素与客户硬件的选择类似。

操作系统和数据库的选择必须与硬件的选择同步进行。操作系统应该是多任务的，其它多任务操作系统（也都是多用户的）包括 DEC 的开放式 VMS、IBM 的 MVS 和 OS/400，在某些情况下也是很好的选择。另外，IBM 的 OS/2 和 Noevll 的 NetWare 是用于一些 PC 应用的操作系统。当今两个最杰出的服务器操作系统是 Unix 和 Windows NT，两者的功能都很强大。表 3-1 列出了可运行 Unix 和 Windows NT 的硬件平台：

表 3-1　　可运行 Unix 和 Windows NT 的硬件平台

硬件平台	Unix	NT
Intel PC	Solaris 86	×
	Linux	
	SCO Unix	
	UnixWare	
Sun Sparc	Solaris	
	Linux	
Dec Alpha	DEC Unix	×
	Linux	
HP RISC	HP / UX	
MIPS RISC		×*
Power PC	IBM AIX	×**
	Linux	

*支持 Mircosoft 开始淘汰的产品。

**正在开发。

桌面计算机与服务器计算机之间有一个很大的不同，这就是 I/O 带宽。对于这一点，Unix 开发商似乎比他们的 PC 竞争对手更清楚，当然，也有个别例外的情况。不要强行将桌面计算机当成服务器来使用，在充当服务器的桌面计算机上，应用程序开始会运行得较好，但最终当系统负载增加时，性能会受到损害。

桌面机的硬盘空间的扩展范围通常是限制最多有两个 IDE 驱动器，而服务器一般具有 SCSI 控制器，能将硬盘驱动器增加到八个。

当把桌面机用作服务器时，遇到的另一个方面的问题常常是缺少内存。在桌面机上，32MB 的内存看起来像是无限的资源；而在服务器中，128MB 或 256MB 的内存还经常被完全用尽。

3.3.2 数据库

数据库的选择也要与硬件的选择同时进行，例如，不能在 Solaris 上运行 Microsoft 的 SQL Server。要记住，即使一个数据库能在若干种平台上运行，每个开发商通常也都只有一个支