建议在多个网站上尝试上述操作，以了解对网站的请求数量。你可能会对传输了多少内容感到吃惊！

设计37：运行自己的网络服务器

在本设计中，你将把Raspberry Pi设置为一个网络服务器。你将使用Python 3完成这项工作，所以你可以在任何安装了Python 3的设备上按照步骤进行实际操作，虽然这些步骤是考虑Raspberry Pi的情况编写的。我们的简单网站在收到请求后，将返回文件内容。

打开一个终端窗口，创建一个目录来存放网站将要提供的文件，然后把这个新目录设置为当前目录：

```
$ mkdir web
$ cd web
```

当向网站的根目录发出请求时，网络服务器软件会查找文件index.html，并向客户端返回该文件的内容。让我们创建一个非常简单的index.html文件：

```
$ echo "Hello, Web!" > index.html
```

这个命令创建一个名为index.html的文本文件，其中包含文本Hello，Web！。你可以用文本编辑器打开文件，查看这个文本文件的内容以确保创建成功，还可以像下面这样在终端显示文件内容：

```
$ cat index.html
```

文件就位后，我们用Python内置的网络服务器向所有的连接者提供消息Hello，Web!：

```
$ python3 -m http.server 8888
```

这个命令告诉Python在端口8888上运行一个HTTP服务器。让我们测试一下，看看它是否按预期工作。在Raspberry Pi上打开另一个终端窗口。在

第二个终端窗口中，输入如下命令，以向新网站的根目录发出GET请求：

```
$ curl http://localhost:8888
```

curl实用程序可以用于发出HTTP GET请求，localhost是主机名，引用当前正在使用的计算机。这个命令告诉curl实用程序，在本地计算机的8888端口上执行HTTP GET。你应该看到返回的文本Hello，Web！。此外，在初始终端中，你应该看到有GET请求到达。

现在，让我们尝试从网络浏览器连接到网站。在Raspberry Pi桌面上打开Chromium网络浏览器。在地址栏中输入http://localhost:8888。你应该看到来自网站的文本出现在浏览器中，如图12-13所示。

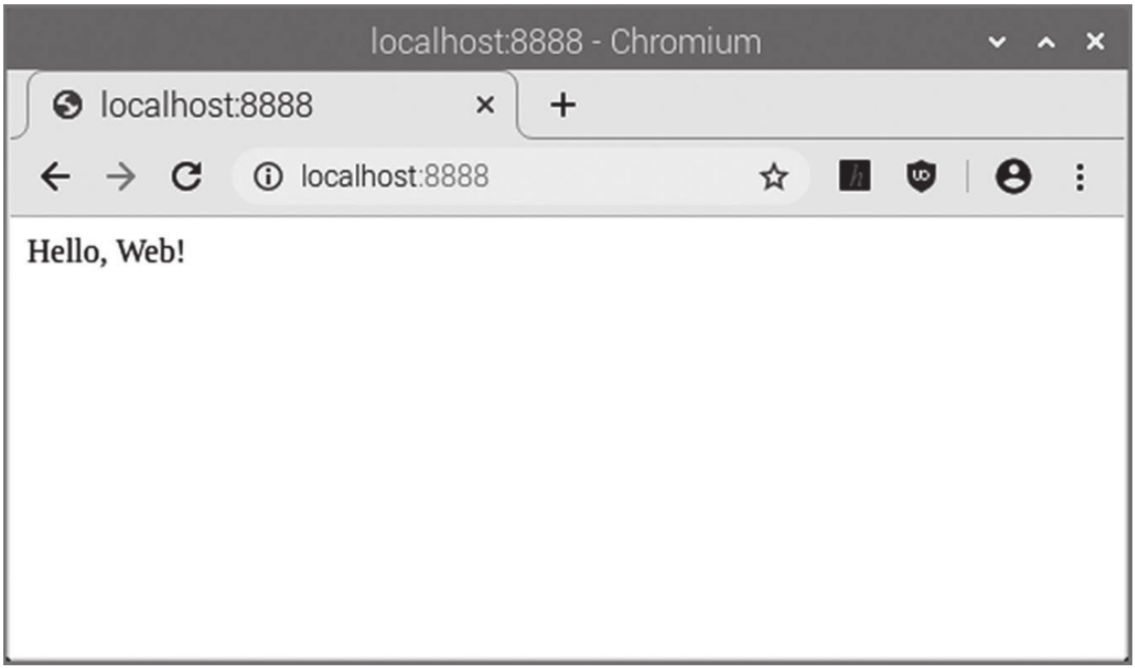

图12-13 用Chromium浏览器连接到本地网络服务器

现在尝试从另一个设备连接网站。为此，第二个设备必须和Raspberry Pi在同一个网络上，例如，它们都在同一Wi-Fi网络上。如果Raspberry Pi有公网IP地址（参见设计34），那么网站就可以被互联网上的所有设备使用！首先，在第二个终端窗口中运行如下命令以获取Raspberry Pi的IP地址：

```
$ ifconfig | grep inet
```

这可能会返回多个IP地址。从远程设备连接时，不能使用127.0.0.1，所以请选择分配给Raspberry Pi的其他IP地址。获得IP地址后，在另一个设备上打开浏览器。这个设备可以是智能手机、笔记本计算机，还可以是网络上的任何带浏览器的设备。在浏览器窗口的地址栏中输入http://*w.x.y.z*:8888（用设备IP地址替换其中的*w.x.y.z*）。按下回车键或浏览器中相应的按钮，导航到该地址。你应该看到Hello，Web！显示在浏览器中。

如果上面的步骤不起作用，并且Raspberry Pi没有公网IP地址，那么请确保两个设备在同一个物理本地网络上。此外，有时Python网络服务器对新请求没有响应。如果网络服务器停止响应，你可以重启它。要停止网络服务器，请转到执行服务器命令的终端，按下键盘上的<Ctrl+C>快捷键。然后再次运行python3-m http.server 8888命令以重启服务器（按下键盘的向上箭头获取最后一条指令）。

网站开始工作后，尝试编辑index.html文件并修改消息以显示想要的内容。可以使用文本编辑器来完成这个操作。index.html文件更新后，在网络浏览器中重新加载网页以查看变化！

如果你不希望其他设备访问你的网站，则可以设置限制，只让来自Raspberry Pi自身的请求得到响应。用--bind选项运行Python网络服务器可以完成此操作，如下所示：

```
$ python3 -m http.server 8888 --bind 127.0.0.1
```

要用--bind选项运行网络服务器，首先停止网络服务器上所有正在运行的实例（按下键盘上的<Ctrl+C>快捷键）。

设计38：从网络服务器返回HTML

前提条件：设计37。

在本设计中，你将更新本地网络服务器以返回HTML，而不是简单的文本。使用文本编辑器打开index.html（在设计37中创建的），并用如下HTML替换文件的所有文本。它与本章讨论的HTML代码相同。你不用担心每行的缩进，因为HTML中的多余空格是无关紧要的。

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>A Cat</title>
  </head>
  <body>
    <h1>Thoughts on a Cat</h1>
    <p>This is a cat.</p>
    <img src="cat.jpg" alt="cat photo">
  </body>
</html>
```

文件更新后，你将再次使用Python内置的网络浏览器。如果它还没有运行，用下面的命令启动它。在运行这个命令之前，请确保终端窗口当前位于Web目录中。

```
$ python3 -m http.server 8888
```

现在，用网络浏览器连接网络服务器，就像在设计37中所做的那样。你应该看到呈现出来的页面，但是没有猫咪的照片。如果你查看运行Python网络服务器命令的终端窗口，你应该会看到尝试获取猫咪照片失败了，如下所示：

```
192.168.1.123 - - [31/Jan/2020 17:38:56] "GET /cat.jpg HTTP/1.1" 404 -
```

404错误码表示没有发现资源，这是有道理的，因为在这个目录中没有名为cat.jpg的文件！为什么网络浏览器要求提供猫咪照片？如果你回头查看页面的HTML，你会看到一个HTML的<img>标签，它引导浏览器去显示cat.jpg图像。浏览器请求该图像，但检索失败，因为没有这个文件。

让我们来修复猫咪图像的问题。你需要下载一个JPEG格式的猫咪图像（或其他任何东西的图像），并将其保存为 ~/web/cat.jpg。为了简单起

见，你可以用如下命令下载本章使用的图像。在运行这个命令之前，请确保终端窗口当前位于Web目录中。

```
$ wget https://www.howcomputersreallywork.com/images/cat.jpg
```

现在，你应该把cat.jpg保存到Web目录中了。重新在网络浏览器中加载页面，以便在页面中查看猫咪图像。请注意，如果网络服务器看上去像是卡住了，请按照设计37中所述的那样重启它。

值得注意的是，你不仅可以在页面中查看猫咪图像，而且还可以直接从服务器请求图像，因为它有自己的URL。尝试把浏览器指向下面的URL（用你网站使用的主机名或IP地址替换SERVER）：http://SERVER:8888/cat.jpg。你应该在网页外部的浏览器中看到猫咪图像。网页上引用的每个资源都有自己的URL，都能被直接访问！

设计39：为网站添加CSS

前提条件：设计38。

在本设计中，你将用CSS为网站设置样式。首先，使用文本编辑器在Web目录中创建名为style.css的文件。这个文件将包含CSS规则。要保证文件名为style.css，且与index.html和cat.jpg文件一起保存在Web目录中。style.css的内容应如下所示：

```
p {
   font-family: Arial, Helvetica, sans-serif;
   font-size: 11pt;
   margin-left: 10px;
   color: DimGray;
}
h1 {
   font-family: 'Courier New', Courier, monospace;
   font-size: 18pt;
   font-weight: bold;
}
```

创建了style.css之后，打开index.html进行编辑，如同在前面的设计中所做的那样。保留现有的HTML。我们只想在head部分添加一行代码，如下

所示：

```
<head>
  <meta charset="utf-8">
  <title>A Cat</title>
  <link rel="stylesheet" type="text/css" href="style.css">❶
</head>
```

把这个更新到index.html❶后，启动网络服务器（如果它还未运行的话），重新在网络浏览器中加载页面。你应该看到更新的页面样式。请注意，如果网络服务器看上去像是卡住了，请按照设计37中所述的那样重启它。

任意编辑style.css以尝试不同的样式。也许你想让段落字体变大或用不同的颜色！按自己的喜好编辑样式并保存style.css，然后重新在浏览器中加载页面。

如果你没有看到更新反映在浏览器中，那么可能的原因是网络浏览器正在加载网站缓存的副本，而不是下载的最新版本。尝试在新选项卡中打开页面或者重新启动浏览器。你还可以告诉浏览器在重新加载时绕过其本地缓存。为此，先导航到该页面，然后按住<Ctrl+F5>快捷键强制重新加载页面。这种方法适用于Windows和Linux上的大多数浏览器。对于Mac，你可以在Chrome和Firefox中用<CMD+Shift+R>快捷键强制刷新。有时候，在浏览器呈现最新内容之前，需要多次刷新。

设计40：为网站添加JavaScript脚本

前提条件：设计39。

在本设计中，你将使用JavaScript使网站具有交互性。首先，使用文本编辑器在Web目录中创建文件cat.js。这个文件将包含JavaScript代码。请确保该文件名称为cat.js，并与index.html和cat.jpg文件一起保存在Web目录中。cat.js的内容应该是这样的：

```
document.addEventListener('DOMContentLoaded', function() {
    document.getElementById('cat-photo').onclick = function() {
        document.getElementById('cat-para').innerHTML += ' Meow!';
    };
});
```

与在前面的设计中所做的一样，保存cat.js后，打开并编辑index.html。保留现有的HTML，做如下修改：

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>A Cat</title>
    <link rel="stylesheet" type="text/css" href="style.css">
    <script src="cat.js"></script>❶
  </head>
  <body>
    <h1>Thoughts on a Cat</h1>
    <p id="cat-para"❷>This is a cat.</p>
    <img id="cat-photo"❸ src="cat.jpg" alt="cat photo">
  </body>
</html>
```

这些修改引用了脚本❶，为段落❷和图像❸提供了ID。

把这些内容更新到index.html后，启动网络服务器（如果它还未运行的话），在网络浏览器中重新加载页面。现在，你应该能单击（或触摸）猫咪照片，并看到Meow！这个词附加到段落。请注意，如果网络服务器看上去像是卡住了，请按照设计37中所述的那样重启它。

第13章

现代计算机

本章将对现代计算机的几个精选领域进行概要介绍。考虑到计算机的多样性和广泛性，可选择的主题有很多。我所选择的领域并不是当今计算机所发生趣事的详尽清单。相反，它们代表了我认为值得考虑的一些主题。本章将介绍app、虚拟化、云计算、比特币等。我们以一个最终设计作为结束，这个设计汇集了本书介绍的许多主题。

13.1 app

从计算机发展的早期开始，人们就把被用户直接使用的软件程序称为应用程序（application）。为了方便，将这个词缩写为app。在过去，这两个词是可以互换的。不过，随着苹果在2008年开设iPhone应用商店以来，app一词已经有了不同的含义。虽然没有标准技术定义把软件程序定义为app，但app通常具有许多共同特征。

app是为最终用户设计的。app一般面向的是移动设备，比如智能手机或平板电脑。app通常通过基于互联网的数字商店（应用商店）——比如苹果应用商店、Google播放商店或者Microsoft商店分发。app对运行其的系统只有有限的访问权限，而且常常必须声明它们需要哪些特定功能才能操作。app倾向于使用触摸屏作为用户输入的主要方式。在单独使用的时候，app一词通常是指安装在直接使用操作系统API的设备上的软件。换句话说，app这个词一般意味着“本机app”，即为特定操作系统构建的app。反过来，网络app是用网络技术（HTML、CSS和JavaScript）设计的app，不与特定的操作系统绑定。图13-1提供了本机app和网络app的高级视图。

如图13-1所示，本机app通常从应用商店安装，旨在利用特定操作系统的功能。网络app通常从网站运行，旨在使用网络技术。网络app在浏览器

或其他呈现网络内容的进程中运行。现在，让我们更细致地看看本机app和网络app。

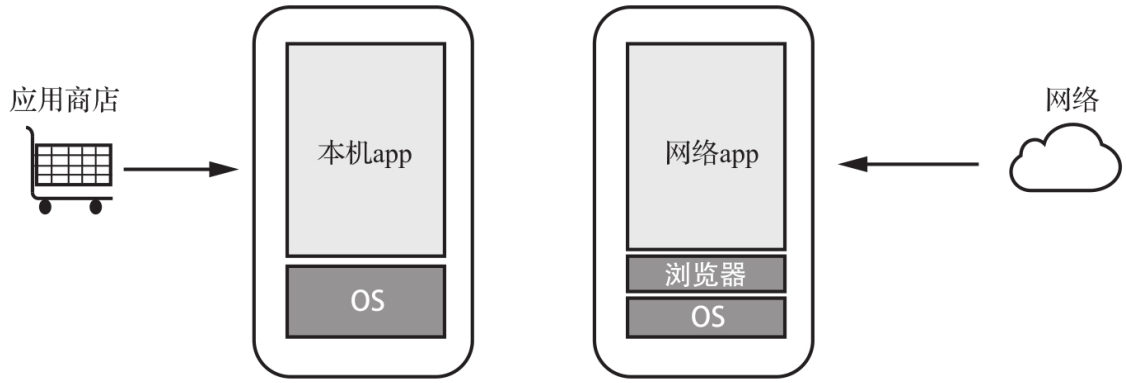

图13-1 本机app为特定操作系统构建，网络app用网络技术构建

13.1.1 本机app

如前所述，本机app为特定操作系统而构建。苹果应用商店和之后类似的应用商店开创了本地软件开发的新时代，为开发人员提供了新的目标平台、新的软件分发方法，以及用软件赚钱的新方法。本机app的开发目前主要集中在两个平台上：iOS和Android。当然，仍然要为其他操作系统开发软件，但一般这样的软件不具备app的典型特性（移动友好、基于触摸屏的输入、通过应用商店分发等）。

Android和iOS的区别在于它们的编程语言和API等。因此，编写在Android和iOS上都运行的app需要维护独立的代码库，或者使用跨平台框架，比如Xamarin、React Native、Flutter、Unity。这些跨平台解决方案把每个操作系统API的底层细节进行了抽象，使得开发人员可以编写能在多个平台上运行的代码。许多本机app也依赖于网络服务，这意味着app开发人员不仅必须为iOS和Android编写并维护代码，而且还必须构建或集成网络服务。

开发跨平台的、网络连接app需要大量的工作和专业知识！过去，开发人员一般只关注一个平台，比如Windows PC或Mac。对于针对多个平台和

网络的开发人员来说，现在的工作当然更为复杂。平台竞争对于用户而言通常是好事，但对于开发人员而言则意味着更多的工作。

有意思的是，app开发的当前状态本可能会完全不同。当2007年1月发布iPhone时，关于iPhone上的第三方app开发，Steve Jobs（苹果当时的CEO）是这样说的：

iPhone内部有完整的Safari引擎。你可以编写令人惊叹的Web 2.0和Ajax app，它们在外观和行为上与iPhone的app完全一样。这些app可以与iPhone服务完美地集成。它们可以打电话，可以发邮件，可以在Google地图上查找位置。你猜怎么着？你不需要SDK!

注意

SDK（Software Development Kit，软件开发工具包）是开发人员用于为特定平台构建应用程序时使用的软件集合。

根据这段话，苹果对于第三方app开发的初始计划是简单地让开发人员构建能使用iPhone功能的类app网站。本机app的开发将局限于苹果开发并包含在iPhone中的app，比如相机、电子邮件、日历app。

当时，把网络用作应用程序开发的平台并不常见。苹果的立场是具有前瞻性的。可惜的是，2007年网络的底层技术还不够成熟，不能让网络成为真正的app平台。到2007年10月，苹果改变了它的说法，宣布苹果将允许开发人员为iPhone构建本机app。2008年，苹果开放了应用商店，作为向用户分发本地iPhone app的唯一受支持的机制。

苹果策略的逆转使公司受益，因为应用商店成为苹果收入的来源。注册为应用商店开发人员需要付费，苹果还从每一笔销售额中收取一定比例的费用。应用商店和本机iPhone开发也为独家内容打开了大门，app只能在苹果设备上运行。

应用商店还为最终用户带来了好处。带有评级的精选app列表很有帮助，且商店提供了对用户信任度的度量。进入应用商店的app必须符合特定的质量标准。集中式支付服务意味着用户不必把自己的支付信息提供给多

个公司。app自动更新，这相对于传统PC软件来说是一个优势，不过对于网络来说不算优势，因为网络app也能在没有用户参与的情况下更新。

随着苹果应用商店的成功，其他公司也创建了类似的数字商店来分发软件。Google播放商店、Microsoft商店和Amazon应用商店的运营模式都与苹果商店类似，而且提供的好处也类似。尽管这个系统对这些公司和最终用户都有很好的效果，但它也给开发人员创造了一个复杂的环境：多个商店、多个平台以及各种技术。每个数字市场都有自己的需求，而app开发人员必须满足这些需求，而且每个商店都占销售收入的一定份额。

13.1.2 网络app

伴随本机app的兴起，网络逐渐成熟，成为有相当能力运行app的平台。一种被称为HTML5的成熟HTML版本被引入其中，网络浏览器在处理内容方面变得更加强大，更加一致。浏览器开发人员使其JavaScript实现符合ECMAScript 5标准，为JavaScript代码提供了更好的基础。在浏览器更新之外，网络开发人员社区接受一种被称为响应式网络设计的概念，即不论显示屏幕的大小如何都能确保网络内容得到很好呈现的一种方法。使用响应式设计技术，网络开发人员可以维护一个跨不同设备仍然运行良好的网站，而不是针对不同设备创建不同的网站。此外，近年来还发布了多个网络开发框架，比如Angular和React。这些框架让开发人员更加轻松地编写和维护网络app——行为类似于app的网站。

开发人员已经意识到现代网络技术可以用于构建与本机app非常相似的体验，许多开发人员搭建了充当app的网站。一些开发人员选择完全放弃本机app，而只构建网络app。这个方法的好处是网络app可以在具有现代网络浏览器的任意设备上运行，而代码只需要编写一次。不过，网络app也有一些缺点。网络app不能访问全部的设备功能，往往比本机app要慢，要求用户在线，而且通常不会出现在应用商店中。

为了解决网络app的一些不足，渐进式网络app（Progressive Web App，PWA）提供了一组技术和指南来帮助弥补本机app和网络app之间的差距。PWA只是一个具有一些额外功能的网站，这些

功能有助于使它更像app。渐进式网络app必须：通过HTTPS提供服务，在移动设备上恰当地呈现，下载后能离线加载，向浏览器提供描述app的清单，在页面之间快速转换。对最终用户而言，运行PWA与运行本机app应该是一样的响应迅速且自然。如果网站符合PWA的标准，那么现代网络浏览器会给用户提供为其主屏幕或桌面添加PWA图标的选项。这表示用户可以像启动本机app一样启动网络app。app在自己的窗口而不是浏览器窗口中打开，而且一般表现得就像本机app一样。

PWA可以为那些希望把网络技术用于其app，但又不想针对不同平台构建多个app的开发人员提供非常大的好处。不过，PWA仍然存在一些缺点。一个重要问题就是，PWA不会出现在应用商店中。多年以来，移动操作系统一直使用户认为：app应该通过应用商店获得。用户不习惯通过浏览网页获得app。在撰写本书的时候，只有Microsoft商店允许PWA直接发布到商店。其他平台希望从浏览器安装PWA，或者将其重新打包成呈现网络内容的本机app，然后把这个重新打包的app提交到商店。另一个潜在的缺点是PWA可能看起来不像本机app，它们在所有平台上的外观通常都基本一样，尽管有些人可能认为这是件好事。PWA仍然不具有本机app的性能，也不能访问底层平台的所有功能，但根据app的需求，这不一定是个问题。

13.2 虚拟化和仿真

什么时候计算机不是物理设备？当然是在它是虚拟计算机的时候！虚拟化是使用软件创建计算机虚拟表示的过程。相关的仿真技术允许为某种类型的设备设计的应用程序运行在完全不同的设备上。本节将探讨虚拟化和仿真。

13.2.1 虚拟化

虚拟计算机被称为虚拟机（Virtual Machine，VM），它像物理计算机一样运行操作系统。应用程序可以在这个操作系统上运行。从应用程序的角度来看，虚拟化硬件就像物理计算机一样。虚拟化支持几个有用的场景。运行一个操作系统的计算机可以在虚拟机中运行另一个操作系统。例

如，运行Windows的计算机可以在虚拟机中运行Linux的实例。虚拟机还允许数据中心在一个物理服务器上托管多个虚拟服务器。这为互联网托管公司提供了一种方法：只要客户愿意使用虚拟服务器，就能轻松迅速地为客户提供专用服务器。VM易于备份、还原和部署。

管理程序（hypervisor）是运行虚拟机的软件平台。管理程序有两种类型，如图13-2所示。

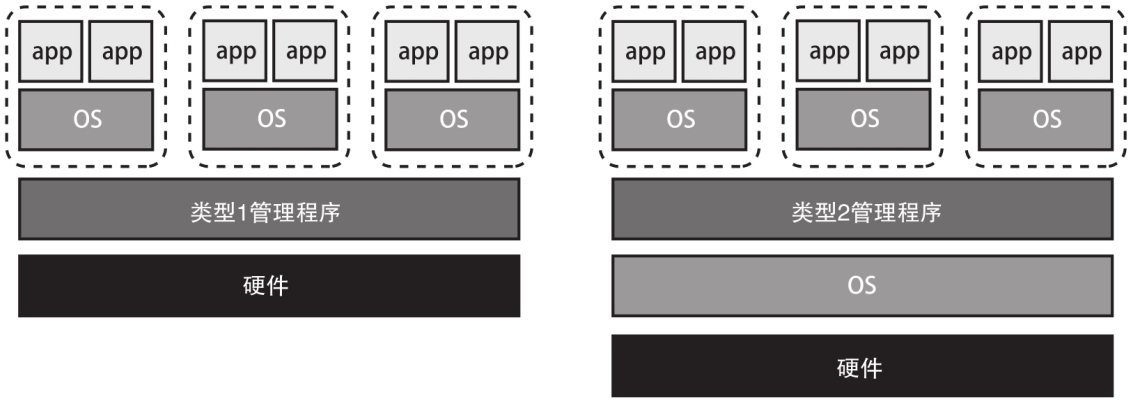

图13-2 类型1管理程序和类型2管理程序

如图13-2左侧所示，管理程序可以直接与底层硬件交互，实际上管理程序放置在技术栈中内核的下面。管理程序与物理硬件通信，并把虚拟化硬件呈现给OS内核，它被称为类型1管理程序。与之相比，图13-2右侧所示的类型2管理程序则作为应用程序运行在操作系统上。微软的Hyper-V和VMware ESX是类型1管理程序，而VMware Player和VirtualBox是类型2管理程序。

另一种常见的虚拟化方法是使用容器。容器提供隔离的用户模式环境，应用程序在这个环境中运行。与虚拟机不同，容器与主机操作系统以及在同一台计算机上运行的其他容器共享内核。在容器中运行的进程只能看到物理计算机上部分可用资源。例如，每个容器都能被授予自己的独立的文件系统。容器可隔离VM，无须为每个VM运行单独的内核。通常，由于内核是共享的，容器被限制运行于主机相同的操作系统。像OpenVZ一样的容器技术被用于虚拟化操作系统的整个用户模式部分，而其他的容器技术（比如Docker）则被用于在独立的容器中运行单个应用程序。

注意

你可能还记得第10章将操作系统进程描述为“容器”，这与虚拟化容器不是一回事。

13.2.2 仿真

仿真是用软件让一种类型的设备的行为类似于另一种类型的设备的行为的过程。仿真和虚拟化的相似点在于，它们都为软件运行提供虚拟环境，但虚拟化提供的是底层硬件的一部分，而仿真提供的虚拟硬件则与使用的物理硬件不同。例如，在x86处理器上运行的虚拟机或容器直接使用物理CPU运行为x86编译的软件。相比之下，在x86处理器上运行的仿真器（执行仿真的程序）可以运行为完全不同的处理器编译的软件。除了处理器之外，仿真器通常还提供其他虚拟硬件。

例如，世嘉Genesis（20世纪90年代的电子游戏系统）的完整仿真器可以模拟摩托罗拉68000处理器、雅马哈YM2612声音芯片、输入控制器，以及世嘉Genesis中的所有其他硬件。运行时，这样的仿真器会把最初设计在世嘉Genesis上运行的CPU指令转换为在x86代码中实现的功能。这带来了巨大的开销，因为每条CPU指令都必须进行转换，但足够快的现代计算机仍然可以全速仿真速度慢得多的世嘉Genesis。其结果就是能在完全不同的平台上运行针对某个平台设计的软件，如图13-3所示。

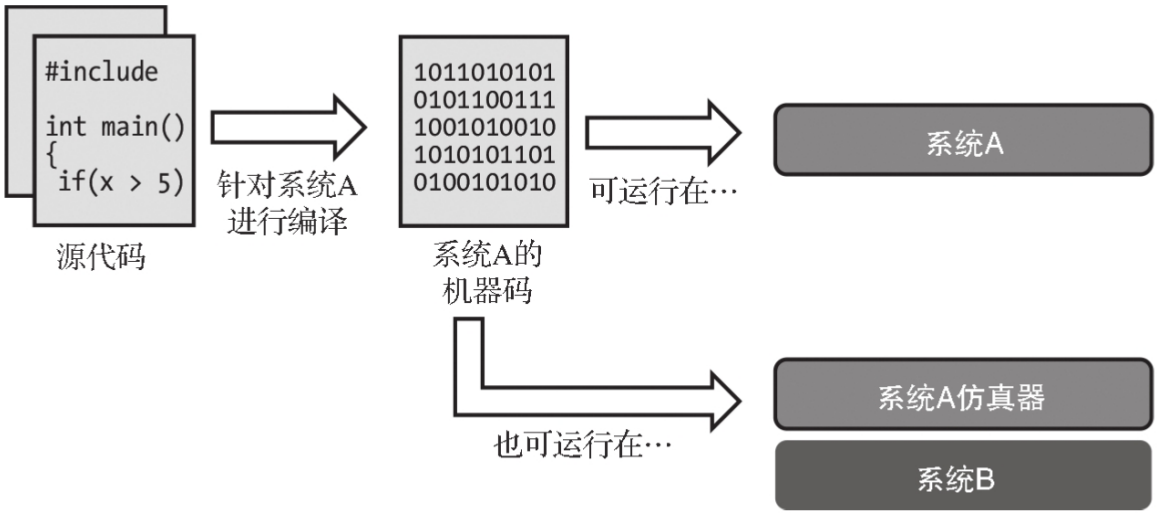

图13-3 为系统A编译的代码可以在系统A的仿真器上运行

仿真在保留为过时平台所设计的软件方面起着重要作用。随着计算机平台的老化，找到能工作的硬件变得越来越难。那些希望把旧软件引入现代平台的软件开发人员往往就使用仿真技术。最初的源代码可能会丢失，或者对其进行现代化的任务可能会很繁重。在这种情况下，使用仿真器能让原始已编译的代码无须修改就可在新平台上运行。

进程虚拟机

还有另外一类虚拟机，它与仿真器有一些共同特性。进程虚拟机运行应用程序的执行环境是对底层操作系统细节的抽象。它与仿真器类似，因为它提供了一个与运行它的硬件和操作系统分离的执行平台。但是，与仿真器不同的是，进程VM不会尝试模拟真实硬件。相反，它提供的环境旨在运行平台无关的软件。如同我们在第9章讨论的一样，Java和.NET利用了运行字节码的进程虚拟机。

13.3 云计算

云计算是指通过互联网提供计算服务。本节将介绍各种类型的云计算，但首先让我们快速回顾一下远程计算的历史。

13.3.1 远程计算的历史

从有计算以来，我们就可以观察到一个钟摆模式：从远程集中式计算（从终端访问的服务器）到本地计算（桌面计算机），现在再回到从智能本地设备（如智能手机）访问的远程计算（网络）。当前许多应用程序同时依赖于远程计算和本地计算。对于网络，有些代码在浏览器上运行，有些代码在网络服务器上运行。与早期那些像房间那么大的计算机相比，今天我们在口袋里揣着的设备要强大得多，但是很多我们想在这些设备上做的工作都涉及与其他计算机的通信，因此在本地设备与远程服务器之间分担处理责任是有意义的。