第一部分　安装Linux操作系统作为服务器软件

第1章　Linux发行版本与Windows NT的技术异同

除非是一直在某个荒无人烟的小岛上打发光阴，或者不接触新闻媒体并拒绝商业宣传，那么读者一定已经非常了解了什么是Linux以及为什么要对它感兴趣。因此这里我不再不厌其烦地介绍什么“Linux操作系统的历史”了，读者可以在其他材料中找到类似的内容。我们来看看Linux发行版本和Windows NT（读者可能正在考虑使用Linux替代这个系统平台）在技术方面的异同。这一章还将解释什么是GNU（GNU不是UNIX）许可证，这部分内容可能会帮助读者了解Linux操作系统为什么会是现在这个样子。

1.1　Linux操作系统和Linux发行版本

人们大都知道Linux是由开发工具软件、编辑器软件、GUI图形用户界面、网络工具软件等组成的一个完整的软件包。更正规一些的说法是：这些软件包就是发行版本。读者可能已经听说过Red Hat、Caldera和SuSE等称呼的Linux发行版本，而这些发行版本都已经受到了广泛的关注并且已经销售安装了几千份。Linux操作系统的非商业性发行版本如Slackware和Debian就不那么出名，也没有达到类似的流行程度。

因此如果我们说某个发行版本包括了“Linux用户所需要的全部东西”的时候，到底什么才是Linux操作系统呢？Linux本身是这个操作系统的核心部分，也就是操作系统的内核。而内核是完成那些最基本的操作的程序。它负责其他程序（比如文本编辑器程序）的启动与终止、内存申请处理、硬盘访问、网络连接管理等方面的工作。

操作系统内核就是人们所说的非凡程序，也就是从“Linux”操作系统本身形成好几种Linux发行版本的那个核心部分。各种发行版本使用的都是同一个内核，各种Linux发行版本最基础的操作都是一样的。

各种发行版本之间彼此产生差异的主要原因是它们各自附带的不同的“增值”工具软件。举例来说，Red Hat发行版本中有一个非常有用的软件工具Xconfigurator，它可以把配置图形界面的工作变得相当简单明了。如果读者打算提问“哪一种发行版本更好？”，那就好比是在提问“可口可乐和百事可乐哪一个更好？”一样。因为几乎所有的可乐都含有一些相同的成分——碳酸水、咖啡因以及高甜度玉米糖浆等，所以它们在解渴和引起几个“咖啡加糖”味道的嗝方面的效果也都差不多。说到底，这个问题纯属个人爱好。

1.2　“自由”软件和GNU许可证

在20世纪80年代初期，Richard Stallman在软件业引发了一场革命。他坚持（而且至今仍然认为）的观点是：软件应该是“自由”的。在这里，“自由”一词并没有什么价格方面的含义，而是代表着与“平等自由”一词相近的意义。这个词覆盖的范围不仅仅是软件产品的流

通领域，它还包括了整个程序源代码方面的范畴。

Stallman的想法对80年代早期软件开发人员的心态来说无疑是离经叛道，因为当时的人们只知道销售预包装好的软件；但是“自由软件”的概念从贝尔实验室最早发表UNIX操作系统原始版本的时候就已经得到了体现。早期的UNIX系统确实是附带着源代码的（到了70年代末期，源代码通常都不再包括在UNIX操作系统的发行版本中了，要想获得它们，就必须向AT&T支付一大笔费用。这种做法一直延续到FreeBSD和Linux项目实现以后）。

随软件发行而附带其源代码的观点其实是非常简单的：这样做了以后，软件用户们就不必非得和某个软件开发人员打交道了，而这位老兄还不一定支持用户对这个软件的某个具体想法。用户也不必非得等待软件漏洞补丁程序的发布了。尤其重要的是，在其他程序员环视的情况下编写出来的程序代码通常都会比闭门造车编写出来的代码质量更高。但是，自由软件的最大优势还是属于用户自己：如果需要某个原始程序中没有的功能，用户可以自己把它添加到程序中去，再体现到源代码中，其他人也可以因此分享到新的功能。

这一系列的思考最终形成了一个共识：发布一个UNIX风格的操作系统，对它的用户不再有任何许可证方面的限制。当然，在建立任何一个操作系统之前需要先建立软件工具。出于这些方面的原因，GNU计划诞生了。

请注意：GNU的意思是“GNU不是UNIX”（GNU是英文“GNU's Not UNIX”的单词字头缩写）——这种取名的方法是计算机老手们的一种幽默。如果读者理解不到这有什么可笑之处，也没什么关系，这只能证明读者还是属于大多数人那一类。

1.2.1 什么是GNU公共许可证

从GNU计划中产生的最重要的事物就是GNU公共许可证（GNU Public License，GPL）。这个许可证明确表示：按照这个许可证发行的软件是自由的，任何人都不能剥夺这种自由。获得某个软件再把它转卖给其他人是合法的，就是加价获利也没什么不可以；但在转卖过程中，卖方必须把完整的源代码及对它的任何增补都完整地转移给买方。因为这份经过转卖的软件依然遵守着GPL许可证制度，所以它还可以自由发行，允许再次转卖给其他人获利。这个许可证制度中最重要的部分就是其免责条款，即程序开发人员对他们编写的软件在事实使用中引起的损失将不承担任何责任。

1.2.2 “自由”软件的优势

从商业化的角度出发，如果认为GPL并不是一个好主意的话，就请想一想最近某些优秀自由软件所引发的热潮——它们证明了这种做法确实是行得通的。它们的成功主要基于两方面的原因：第一，正如我们前面提到的，在编程同行的注视之下，代码本身中的错误将比较容易被查出并迅速纠正；第二，在GPL制度下，程序开发人员发表代码的时候能够不再考虑法律诉讼方面的问题。如果没有这项保护措施，大概就没有多少人敢于发表自己编写的程序代码了。

这一机制要求我们回答这样一个问题：人们为什么会免费发布他们的工作成果？答案其实很简单：大多数的软件成果在刚刚完成的时候并不是功能齐备、光彩照人的。它们有可能是程序员为了解决某个恼人的特定问题而临时编写出来的。这类临时且又粗糙的代码当然没有什么值得卖给别人的价值。但是当把它们分享给遇到同样问题、有同样需求的人们时，这

些代码就将成为有用的工具。别的用户开始给这个程序添上他们所需要的功能，而这些增补又完善了原始的程序。一个项目成为集体智慧的结晶，直到达成最好的效果。最终，这个完善的程序可能包含了成百上千名程序员的贡献，每一个人都为它添了砖、加了瓦。事实上，第一个程序员编写的代码可能都没留下什么痕迹。

对通用许可证制度下的软件来说，它们的成功还有以下方面的原因：任何一位负责软件商业性开发的项目经理都知道这样一个事实，软件开发的真正成本并不体现在其开发阶段。确实需要花大钱的地方是在软件的柜台销售、市场规划、技术支持、编写文档、包装运输等方面。一个在周末晚上熬夜编出粗糙小程序解决某个问题的程序员恐怕既没有兴趣和时间，也没有足够的资金支持来把那个粗糙的东西发展为一个真正赚钱的好产品。

当Linus Torvalds发布Linux操作系统的时候，他选择了采用 GPL制度来发行。由于 Linux本身开放的特性，现在它已经拥有了数目可观的拥护者和分析员。这就使 Linux操作系统在功能上非常强大和丰富。根据 Torvalds本人的估计，从v.2.2.0版本内核开始，他亲自编写的代码在全部代码库中只占5%左右了。

已经有许多人利用 Linux发了财，因为任何人都可以拿到 Linux操作系统的内核（以及其他支持性程序），把它们重新包装后再加以转卖。但是只要这些人在其各自的软件发行包装中提供了完整的操作系统内核源代码，并且这些软件包都采用 GPL制度发行，那么所有的一切就都是合法的。这当然就意味着采用 GPL制度发行的软件都可以被他人以另外的名称转手赚钱。

还有一点需要说明的是，对某人来说，一份软件之所以比另外一份别人的东西更有价值，其主要原因就在于该软件的增值功能、技术支持渠道及其完备的记录档案。即使规模如 IBM这样的大公司也认同这一点：利润并不在产品本身，而在伴随产品的服务里面。而这么做的结果，就是他们从30年代到70年代赚了40多年的大钱。

1.3 NT和Linux操作系统的主要差异

大家都明白，Microsoft Windows NT和Linux操作系统之间的差异是无法在这样一个篇幅有限的小节里讨论清楚的。随着学习的深入，我们将逐一地比较出这两种操作系统之间的各种区别。但是在某些章节里，读者可能会发现我们并没有给出什么对比，那是因为两种操作系统在对应的那个方面并没有什么太大的不同。

在进入细节讨论之前，我们先花一点时间看看这两种操作系统在体系结构上有什么根本性的差异。

1.3.1 单用户、多用户、网络用户情况的比较

Windows NT是根据Microsoft的创始人比尔·盖茨的思路设计出来的，这个思路就是：一台计算机、一张桌子、一个用户。为了便于讨论，我们把这种情况称为“单用户”。依照这样的安排，两个人是无法在同一时间、同一计算机上并行运行（比如说）像 Microsoft Word这样的软件的。（当然，从另外的角度看，人们可能会认为如果有人打算共同运行规模像 Word这样巨型的程序，他的脑筋恐怕多少会有些问题！）

Linux沿用了UNIX操作系统的原理。UNIX最初是由贝尔实验室在70年代初期开发的，它运行在一台由整个部门共享使用的 PDP-7型计算机上。这就要求选用一种允许多个用户同时登

录到中央计算机的设计。在同一时刻，人们可以编辑文件、编译程序或者进行其他工作。中央计算机上的操作系统负责管理“共享”的细节，而每一个用户看起来都像是在独占着整个系统。这种“多用户”传统跨越80年代直到90年代，而其他种类的UNIX操作系统也基本如此。Linux操作系统出现的时间是在90年代的初期，因此它也支持多用户操作。

发展到今天，一个多用户设置情况最常见的应用就是要支持服务器——也就是那些用来运行大型程序供许多客户机共享的计算机系统。部门中每一个成员在办公桌上都可以有一台小一些的工作站，其功能足以应付日常的工作要求。当他们的工作需要使用到巨大的 CPU功能或者内存的时候，就可以在服务器计算机上执行操作。

“等一等”，也许有人会争论说：“Windows NT也允许人们把大计算量的工作分散到另一台计算机上！看看 SQL Server服务器吧！”其实，这只说对了一半。 Linux操作系统和Windows NT实际上都有能力通过网络提供诸如数据库之类的服务。我们可以把这种情况下的用户称之为“网络用户”，因为这些用户实际上从来没有真正登录到服务器主机上，他们只是把一些操作请求发送到服务器而已。服务器根据请求完成相应的工作，然后再通过网络把结果回传给用户。这种情况的关键之处在于应用软件必须经过专门的设计编写以完成这类服务器/客户机操作职能。在Linux环境中，用户可以在服务器上运行系统管理员允许他运行的任何程序而不必重新设计该程序。大多数用户认为能够在别的计算机上运行各种程序的能力是非常有好处的。

1.3.2 GUI图形界面与操作系统内核的彼此相对独立

在吸收Macintosh设计理念的基础上， Windows NT的开发人员把它的图形化用户界面（Graphical User Interface，GUI）与操作系统的核心部分结合为一个整体。二者相辅相成，缺一不可。把操作系统和用户界面紧密地结合在一起的好处是系统各组成部分的操作外观是一致的。虽然Microsoft并不像Apple那样严格地强调遵守什么规则，但是出于应用程序外观界面方面的考虑，大多数开发人员还是倾向于在同一个应用程序中保持同一种基本的视觉和操作模式。

Linux（和普遍意义上的 UNIX）操作系统都是把用户界面和操作系统分开的。像 X-Windows系统这类用户界面是作为一个用户级的应用程序运行的，这就使它更加稳定。如果图形化用户界面（对Linux和Windows NT两者来说，它们各自的GUI都相当复杂）出了问题，Linux操作系统的内核是不会随之瘫痪的。 X-Windows系统与NT的图形化用户界面的另外一个不同之处是它并不是一个完整的用户界面：它只定义了怎样在显示器的屏幕上绘制出基本的对象元素以及如何对它们进行操作控制。 X-Windows最突出的特点是它具备在网络中另外一个工作站的显示屏幕上绘制窗口的能力。这使得用户可以坐在主机 A的前面，登录到主机B上，再在主机B上运行一个应用程序，然后把程序输出回送到主机 A上。它能够让两个人登录到同一台计算机，再同时运行 Linux中相当于Microsoft Word那样的应用程序（比如 ApplixWare、WordPerfect或者StarOffice等等）。

在X-Windows核心部分的上层，还需要再运行一个窗口管理器程序来建立一个可供使用的环境。Linux操作系统本身带有好几个窗口管理器程序，还包括有对“标准的”通用桌面环境（Common Desktop Environment, CDE），因此各种UNIX系统看起来就都差不多（根据不同的情况，读者阅读这本书时的缺省窗口管理器程序可能是 Red Hat发行版本附带的GNOME环

境，也可能是独立发行的KDE环境）。在被设置为缺省桌面环境的时候， GNOME和KDE都提供了类似休闲性质的Windows用户的友好的操作环境。

Windows NT和Linux哪一种更好？为什么？这一切都取决于用户到底想要干些什么。由Windows NT提供的集成式环境使用起来比较方便，而且不那么复杂；但是它缺少了 X-Windows允许应用程序在网络中另外一台工作站的显示屏幕上绘制窗口的能力。虽然 Windows NT图形化用户界面的一致性比较好，但它本身无法被关闭；而 X-Windows却不必非得运行在服务器上才可以（运行它会消耗宝贵的内存）。

1.3.3 Windows中的“网络邻居”概念

在Windows NT环境下，“网络邻居”（Network Neighborhood）对用户来说是访问服务器主机的方便手段，用户可以通过它连接到远程硬盘和打印机。这样的安排要求用户知道应该去往哪一个服务器、连接哪一个共享硬盘。更有趣的是，用户可以与某些共享硬盘建立关联，采用DOS硬盘驱动器盘符的机制使它们看起来就像是额外的硬盘。这样做的结果是：客户机和服务器主机之间的区别对用户来说是一目了然的。

Linux操作系统使用的网络文件系统（Network File System，NFS）采用的是完全不同的网络连接方法。它没有采用与服务器主机的共享硬盘建立关联的办法，子目录是“挂装”在一起的。这个挂装的过程造成这样一个假象：好像存在着一个独立统一的子目录结构，它完全附属于面前的这台本地计算机。而事实上并非如此，这个独立统一的子目录结构中的某些部分其实是属于服务器主机的。这种安排最常见的一个例子就是挂装上的用户登录子目录（home directories）驻留在某个服务器主机上，客户机在开机引导时将（自动地）挂装上这些子目录。最终的结果是：/home子目录存在于客户机上，而/home/username子目录则存在于服务器主机上。

共享网络硬盘的最大好处是用户永远都不必知道服务器主机的具体名称或者对应的子目录路径，而用户们在这一方面的一无所知对系统管理员来说则是个福音！不会再有询问需要连接到哪个服务器主机的问题了，更进一步的好处是用户们不必知道需要在什么时候修改服务器主机的配置情况。在Linux系统中，系统管理员能够在不做任何通知或者重新辅导用户的前提下就对服务器主机的名称进行修改或者从客户机这一端对之进行调整。读者中如果有人经历过为新的服务器设置调整用户的事，就不可能不注意由此可能产生的影响。

打印工作也几乎是以同样的方式完成的。在Linux系统中，打印机都分配有不依赖于它们具体的主机名的名称（这在某个打印机没有使用 TCP/IP协议进行通信的情况下就更加重要）。客户机都指向某个打印服务器，而这个服务器的名称未经系统管理员的许可是不能随意改动的；有关的设置值也不可能在系统管理员不知情的情况下被修改。这个打印服务器再按不同的要求对各种打印请求进行重定向。 Linux统一的操作界面大大改进了用户在安装过程中可能搞得很混乱的打印机设置操作。它还意味着系统管理员不再需要把打印驱动程序分别安装到好几个位置了。

注意 如果打算通过Samba软件包使用Linux操作系统为Windows NT的客户机提供服务，那么系统管理员还是需要向用户们通报服务器主机的共享硬盘设置和打印机设置情况。

一个小故事：Linux解决了难题

在我们不久之前工作过的某个地方，我遇上了一个很奇怪的打印情况。有两台打印机只能通过AppleTalk进行打印。还有一台虽然支持TCP/IP协议，但是其Windows下的驱动程序并不很稳定，因此Macintosh机器的用户们没有办法使用它。采用Linux后解决了问题：使用Samba作为连接Windows的接口；使用Netatalk作为连接Macintosh机器的接口，所有用户机都统一指向某个服务器。在最顶层之下是对各种打印通信协议进行转换的脚本程序，它们用来完成以下艰巨任务：比如把基于Windows的请求转换为UNIX操作系统能够识别的打印请求，再把第一次转换后的打印请求送入一个AppleTalk转换程序中。最后的结局是所有操作平台上的打印工作都采用了统一的方式，每一个用户都可以访问所有的打印机。那可真称得上是一次系统配置的颠峰之作。

1.3.4 Windows中的注册表文件与文本文件的比较

我个人认为Windows NT中的注册表（Registry）文件可以称得上是系统配置数据库中的最高境界——成千上万的数据项，只有很少的一部分才有完整的注解，有的保存在服务器上而有的又保存在客户机上。

“什么？你是说你的注册表文件损坏了吗？”〈讽刺的笑声〉“好吧，是的，我们可以设法从昨天晚上的备份中恢复它，但是那样做Excel的启动就会不正常，技术员（他接这个电话就要收费50美元）说可以再重新安装一遍...”

如果读者还没有理解我的意思，让我来告诉你到底是怎么一回事：就是用最好的词来讲，也只能说Windows NT的注册表太难处理。虽然理论上它是一个好东西，但是我每次和注册表打交道，每次都弄得狼狈不堪。

Linux操作系统没有“注册表”文件。这既有好处，也有坏处。好处是各种配置文件通常都被保存为一系列文本文件（请想想注册表出现之前Windows中的那些.INI文件）。这意味着读者可以使用喜欢的文本编辑程序而不只局限于regedit这样的软件工具就可以对配置文件进行处理。大多数情况下，这还意味着读者能够随意在这些配置文件中加上一些注释，这样在六个月之后读者还可以想起自己为什么要做那样一个特殊的设置。Linux操作系统里的大多数配置文件都保存在/etc子目录或者它的某个下级子目录里。

没有注册表文件的坏处是配置文件没有统一标准的编写方法。每种应用程序或者服务器程序都有它自己的格式。如今，许多应用程序都捆绑了某些基于GUI的软件配置工具。这样用户就可以比较容易地进行一些基本的配置工作；如果需要进行更复杂的调整，可以人工编辑对应的配置文件。

在实际应用中，使用文本文件来保存配置信息通常是一个有效的办法。这些文件只要被设置好了以后，就很少需要再修改；一般，它们是一些比较容易阅读且容易理解的文本文件，在需要的时候很容易看明白。此外，可以很容易地编写一个脚本程序，用它读出相同的配置文件，并根据需要对其行为进行修改。这对服务器维护操作的自动化特别有帮助，而自动化的维护操作对一个拥有多个服务器主机的大站点来说是至关重要的。

1.3.5 域的概念

对一个Windows NT工作组来说，它们必须共存于同一个域（Domain）中。这就要求专门

有一个NT服务器系统被配置为一个主域名控制器（Primary Domain Controller，PDC）。域是Windows NT安全模型的基础。

Linux系统网络安全模型的基础是网络信息服务（Network Information Service，NIS）。NIS是一个简单的数据库（基于文本文件），它与客户工作站共享。每一个基本的NIS服务器（顺便提一句，这些服务器并不需要像一台PDC那样是一个专用的系统）都建立一个域。只要能够设置好自己的域名，任何一个想加入某个域的客户工作站都可以获得准许。对域名进行设置必须使用root（根）用户——在Linux操作系统中这相当于一个系统管理员用户。然而，加入某个域并不代表用户立刻就具有了原来所没有的权限，还必须由那个域的系统管理员把用户的登录信息添加到主NIS口令清单中去，域中的其他系统才能认出新用户来。

NIS里的域和NT里的域比较，二者的主要区别是NIS服务器本身并不像PDC那样进行身份验证。相反，每一台主机会从服务器中查找登录和口令信息并把它与用户输入的信息进行比较。对用户进行身份验证的工作是由各个应用程序来完成的。幸运的是对用户进行身份验证所必须的代码是很简单的。

另外一个重要的区别是：NIS能够当作一个普通用途的数据库，用它来保存需要与网络其他部分共享的任何类型的信息（这种情况通常包括NFS和电子邮件假名使用的挂装表）。唯一的限制是NIS数据库中的每个数据项只能有一个关键字；如果数据项的数量超过了20 000个，数据库的数据定位机制就会不稳定。当然，如果一个站点有20 000名用户，它就决不会把数据全部保存在单独一个NIS域中！

关于NT域和NIS最后一点说明是：它们都不是基本操作系统运行所必不可少的。只有在需要把一个多用户站点维护在一个适当的安全水平之上的时候，它们才是关键的。

如果要全面了解Linux操作系统本身，请从http://www.linux.org Linux操作系统站点开始。

1.4 小结

这一章讨论了三个重要的问题：Linux发行版本、GNU许可证（GPL）以及Windows NT和Linux在设计思想方面的主要区别。现在读者应该对后续章节将要讲述的内容有了一个更加清晰的认识。

Linux是一个功能非常强大的操作系统，有能力处理许多事情。也许我们可以说它最重要的特色就是它的可配置性——而配置过程本身又相当简单易行。读者可以从其基本系统开始根据需要把它定制成最复杂的样子。但是如果想要攻击它就没有那么容易了。在读者继续学习本书的其余部分以及在任何环境下使用Linux操作系统进行工作的时候，千万不要忘记这一点。如果读者自己有什么地方看不明白，觉得需要求助于比尔·盖茨大叔才能得到某个软件的功能，请先去万维网上查查，看看是不是已经存在能够解决读者具体问题的方案了。

Linux不仅仅是一个多能的有效的操作系统，它还给读者一个思考的机会。就笔者本人而言，已经坚持两年使自己的家里没有（微软）“窗口”的影子了。事实上，这本书最初的草稿就是用一个基于Linux操作系统的字处理软件写成的。