## 7.2 文件共享与保护

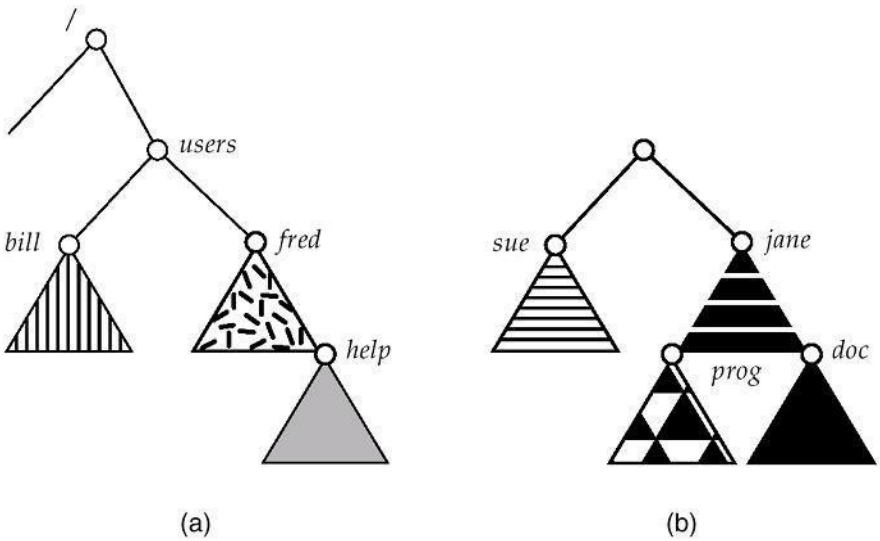
### 1. 文件系统的分区与安装

如同文件使用前必须要打开，文件系统在被系统上的进程使用之前必须安装（mount）。具体地说，目录结构可以建立在多个卷上，这些卷必须被安装以使它们在文件系统命名空间中可用。

安装步骤相对简单。操作系统需要知道设备名称和文件系统的安装位置（称为安装点）。通常，**安装点（mount point）**为空目录。例如，在 UNIX 中，包括用户主目录的文件系统可安装在/home。这样，在访问该文件系统中的目录结构时，只需要在目录名前加上/home，如/home/jane 即可。将文件系统安装在/usr 可使路径名/usr/jane 指向同一目录。

然后，操作系统验证设备是否包含一个有效文件系统。验证是这样进行的：通过设备驱动程序读入设备目录，验证目录是否具有期望的格式。最后，操作系统在其目录结构中记录如下信息：一个文件系统已安装在给定安装点上。这种方案允许操作系统遍历其目录结构，并根据需要可在文件系统之间进行切换。

为了说明文件系统的安装，考虑如图 7.11 所示的文件系统，其中三角形表示所感兴趣的目录子树。图 7.11 (a)表示一个已有文件系统，而图 7.11 (b)表示一个未安装的驻留在/device/dsk 上的文件系统。这时，只有现有文件系统上的文件可被访问。图 10.13 表示把/device/dsk 卷安装到/usrs 后的文件系统的情况。如果该卷被卸载，那么文件系统就又恢复




到如图 7.11 所示的情况。

图 **7.11** File system (a) 已有文件系统 (b) 未安装的卷

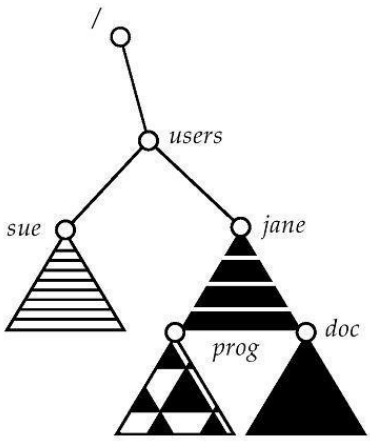


图 **7.12** 安装点

系统利用语义可以清楚地表达功能。例如，系统可能不允许在包含文件的目录上进行安装，或者使所安装的文件系统在目录中可用，并且使目录中已存的文件不可见，直到文件系统被卸载。文件系统的卸载会终止文件系统的使用，并且允许访问目录中原来的文件。另一个例子是，有的系统可允许对同一文件系统在不同的安装点上进行多次重复安装，或者只允许对一个文件系统安装一次。

Microsoft Windows 操作系统系列（95，98，NT，small 2000，XP）维护一个扩展的两层目录结构，并用驱动器字母表示设备和卷。卷具有通常的图结构。特定文件的路径的形式如同 driver-letter:\path\to\file。最近的 Windows 版本允许文件系统安装在目录树的任意位置，这和 UNIX 一样。Windows 系统自动发现所有设备，然后在启动时安装所有定位到的文件系统。有的系统，如 UNIX，安装命令是显式的。系统配置文件包括一系列设备和安装点，以便在启动时自动安装，也可手动进行其他安装。

磁盘布局因操作系统而异。一个磁盘可以分成多个分区，或者一个卷可以横跨多个磁盘上的多个分区。这里，讨论前一种情况，而后一种情况作为RAID的一种形式更为合适，将在下一章中讨论。

分区可以是**“生的”（或原始的，raw）**，即没有文件系统，或者“**熟的”（cooked）**即含有文件系统。**“生”磁盘（raw disk）**用于没有合适文件系统的地方。UNIX交换空间可以使用生分区，因为它不使用文件系统而是使用其自己的磁盘格式。同样，有的数据库使用生磁盘，格式化它来满足其特定需求。生磁盘也可以用于存储RAID磁盘系统所需要的信息，如用以表示哪些块已经镜像和哪些块已改变且需要镜像的位图。类似地，生磁盘可包括一个微型数据库，以存储RAID配置信息，如哪些磁盘属于RAID集合。生磁盘的使用将在下一章中进一步讨论。

引导信息能保存在各个分区中。同样，它有自己的格式，因为在引导时系统并没有文件系统设备驱动程序，所以并不能解释文件系统格式。因此，引导信息通常为一组有序块，并作为镜像文件读入内存。该镜像文件按预先指定的位置如第一个字节开始执行。引导信息除了包括如何启动一个特定操作系统外，还可以有其他指令。例如，PC和其他系统可以双引导。可以把多个操作系统装在这样的系统上。系统如何知道引导哪个？一个启动加载器能够知道位于引导区的多个文件系统和多个操作系统。一旦装入，它可以引导位于磁盘上的一个操作系统。磁盘可以有多个分区，每个分区包含不同类型的文件系统和不同的操作系统。

**根分区（root partition）**包括操作系统内核或其他系统文件，在引导时装入内存。其他卷根据不同操作系统可以在引导时自动装入或在此之后手动装入。作为成功装入操作的一部分，操作系统会验证设备上的文件系统确实有效。操作系统通过设备驱动程序读入设备目录并验证目录是否有合适的格式。如果为无效格式，那么检验分区一致性，并根据需要自动和手动地加以纠正。最后，操作系统在其位于内存的装入表中注明该文件系统已装入和该文件系统的类型。装入功能的细节因操作系统而异。基于MS Windows的系统将卷装入在独立名称空间中，名称用字母和冒号表示。例如，操作系统为了记录一个文件系统已装在f:上，会在对应f:的设备结构的一个域中加上一个指向该文件系统的指针。当一个进程给定设备字母时，操作系统会查找到合适文件系统的指针，并遍历设备上的目录结构以查找给定的文件和目录。Windows的后续版本可以在已有目录结构的任何一个点上安装文件系统。

UNIX 可以将文件系统装在任何目录上。这可以通过在位于内存的相应目录的**索引节点（inode）**上加上一个标记来实现。该标记表示此目录是安装点。一个域指向安装表上的一个条目，以表示哪个设备安装在哪里。该安装表条目包括一个指向位于设备上的文件系统的超级块的指针。这种方案使操作系统可以遍历其目录结构，并根据需要无缝切换文件系统。

## 2.文件共享

面向用户的操作系统必须满足共享文件的需要,本节将研究文件共享的一系列问题。首先是多用户共享文件时可能产生的问题。其次，一旦允许多用户共享文件，需要将共享扩展到多个文件系统如远程文件系统。最后，需要考虑对共享文件的冲突操作需要采取什么措施。例如，如果多个用户对一个文件进行写，那么所有写都被允许吗？或者操作系统应该如何保护用户操作不受其他用户的影响?

### （1）多用户

当一个操作系统支持多个用户时，文件共享、文件名称和文件保护问题就尤其突出了。对于允许用户共享文件的目录结构，系统必须控制文件共享。系统可缺省地允许一个用户访问其他用户的文件，也可要求一个用户明确授予文件访问权限。

为了实现共享和保护，多用户系统必须要比单用户系统维护更多的文件和目录属性。虽然过去有许多方法，但是现在绝大多数系统都采用了文件（或目录）拥有者（或用户）和组的概念。拥有者是目录最高控制权的用户，可以改变属性和授权访问。组属性定义对文件拥有相同权限用户子集。例如，在 UNIX 系统中，一个文件的拥有者可对文件执行所有操作，文件组的成员只能执行这些操作的子集，而所有其他用户可能只能执行另一操作子集。组成员和其他用户对文件可以进行的操作取决于文件拥有者的定义。有关更多权限属性的细节，参见下一节。

一个文件或目录的拥有者 ID 和组 ID 与其他文件属性一起保存。当用户请求文件操作时，用户 ID 可与拥有者属性相比较，以确定请求者是不是文件拥有者。同样，可比较组 ID。比较结果表示可使用哪些权限。这样系统再用这些权限来检查所请求的操作，以决定是允许还是拒绝。许多系统有多个局部文件系统，包括单个磁盘的分区或多个磁盘的多个分区。在这种情况下，只要文件系统已安装，那么 ID 检查和权限匹配就简单了。

### （2）远程文件系统

网络的出现允许在远程计算机之间进行通信。网络允许在校园范围内或全世界范围内进行资源共享。一个重要共享资源是文件形式的数据。

随着网络和文件技术的发展，远程文件共享方式也不断改变。采用的第一种实现方式，用户通过程序如 ftp，可实现在机器之间进行文件的人工传输。采用第二种实现方式 DFS（分布式文件系统），远程目录可从本机上直接访问。采用第三种方法 WWW（这有点回到了第一种），可用浏览器获取对远程文件的访问，其单个操作（基本上是 FTP 的包装）用于传输文件。

ftp 可用于匿名访问和验证访问。匿名访问允许用户在没有远程系统帐号的情况下传输文件 。WWW 几乎总是采用匿名文件交换。DFS 在访问远程文件的机器和提供文件的机器之间提供了更加紧密的结合。这种结合增加了复杂性，将在本节加以讨论。

### 客户机-服务器模型

远程文件系统允许一台计算机安装一台或多台远程机器上的一个或多个文件系统。在这种情

况下，包含文件的机器称为服务器，需要访问文件的机器称为客户机。对于网络机器，客户机-服务器关系是常见的。通常，服务器声明一个资源可为客户机所用，并精确地说明是哪种资源（此时为哪些文件）和哪些客户。根据客户机-服务器关系的实现，一台服务器可服务多个客户机，而一台客户机可使用多个服务器。

服务器通常标明目录或卷的哪些文件可用。客户机标识更为困难。客户机可通过其网络名称或其他标识符如 IP 地址来指定，但是这些可能被欺骗（或模仿）。未经验证的客户机可能欺骗服务器以使其认为该客户是验证过的，这样它就可获得访问。更为安全的解决方案是客户通过加密密钥向服务器进行安全验证。然而，安全也带来了许多挑战，包括确保客户机和服务器的兼容性（它们必须使用相同加密算法）和安全密钥交换（被截的密钥可允许未经验证客户进行访问）。这些问题本身太过困难，所以绝大多数情况下使用不太安全的验证。

对于 UNIX 及其网络文件系统（NFS），验证缺省是通过客户网络信息进行的。采用这种方案，用户 ID 在客户机和服务器上要匹配。否则，服务器不能确定对文件的访问权限。考虑这样一个例子，用户 ID 在客户机上为 1000，而在服务器上为 2000。来自客户机并试图访问服务器上特定文件的请求可能不会得到正确处理，这是因为服务器认为用户 ID1000 访问文件，而不是用户 ID2000。根据不正确的验证信息，访问可以被允许或拒绝。服务器必须相信客户提供正确的用户 ID。NFS 协议允许多对多关系。即，许多服务器可为多个客户提供文件。事实上，一个机器不但可以对某些 NFS 客户来说是服务器，还可以是其他 NFS 服务器的客户机。

一旦安装了远程文件系统，那么文件操作请求会代表用户通过网络按照 DFS 协议发送到服务器。通常，一个文件打开请求与其请求的用户 ID 一起发送。然后，服务器应用标准访问检查以确定该用户是否有权限按所请求的模式访问文件。请求可能被允许或拒绝。如果允许，那么文件句柄就返回给客户应用程序，这样该程序就可执行读、写和其他文件操作。

当访问完成时，客户会关闭文件。操作系统可采用与本地文件系统安装相同的语义，也可采用不同的语义。

## 分布式信息系统

为了便于管理客户机-服务器服务，**分布式信息系统**，也称为**分布式命名服务**，用来提供用于远程计算所需的信息的统一访问。**DNS（域名系统）**为整个 Internet（包括 WWW）提供了主机名称到网络地址的转换。在 DNS 发明和广泛使用之前，包含同样信息的文件是通过 email 或 ftp 在网络机器之间进行交流的。这种方法不可扩展。

其他分布式信息系统为分布式应用提供了用户名称/口令/用户 ID/组 ID。UNIX 系统有很多分布式信息方法。Sun Microsystem 引了入黄页（后来改名为**网络信息服务（NIS，Network Information System）**），业界绝大多数都采用了它。它将用户名、主机名、打印机信息等加以集中管理。然而，它使用了不安全的验证方法，如发送未加密的用户密码，用 IP 地址来标识主机 。SUN 的 NIS+是 NIS 的更为安全的升级，但是也更为复杂，且并未得到广泛使用。

对 Microsoft 网络的**公共 Internet 文件系统（common internet file system, CIFS）**，网络信息与用户验证信息（用户名和密码）一起，用以创建**网络登录**，这可以被服务器用来确定是否允许或拒绝对所请求文件系统的访问。要使验证有效，用户名必须在机器之间匹配（如 NFS）。Microsoft 采用两个分布式命名结构为用户提供单一名称空间。旧的技术命名是**域（Domain）**，从 Windows XP 和 Windows 2000 之后采用了称为**活动目录（Active**

Directory）的新技术。一旦建立，分布式命名工具可供客户机和服务器用来验证用户。

现在，业界正在采用**轻量级目录存取协议(Lightweight Directory Access Protocol，LDAP)**作为安全的分布式命名机制。事实上，活动目录是基于 LDAP 的。Sun Microsystem 在操作系统中包含了 LDAP 来用于用户验证和获取系统范围内的信息，如打印机等。一个分布 LDAP 目录可用于存储一个企业内的所有计算机的所有用户和资源。这种结果是**单一密码签入（secure single sign-on）**：用户只需要输入一次验证信息，就可访问企业内的所有计算机。通过将分布于每个系统上的各种文件信息和不同分布信息服务集中起来，也减轻了系统管理的工作负担。

### 故障模式

本地文件系统可能因各种原因而出错，如包含文件系统的磁盘出错、目录结构或其他磁盘管理信息（总称为**元数据（metadata）**）的损坏、磁盘控制器故障、电缆故障，或主机适配器故障。用户或系统管理员的错误也可能导致文件丢失，或整个目录或分区被删除。许多这类错误都会导致主机关闭、显示错误条件或需要人工干预以修补。

远程文件系统具有更多的故障模式。由于网络系统的复杂性和远程机器间所需的交互，所以会存在更多会影响远程文件系统的正确操作问题。在网络情况下，两主机间的网络可能被中断。这可能是由于硬件故障或配置错误，或有关主机的网络实现出现问题。虽然有的网络有内置的弹性，包括在主机之间有多个路径，但是还有很多网络没有这种功能。任何一个故障都会中断 DFS 命令流。

考虑一个客户在使用远程文件系统。它打开了源自远程主机的文件，在许多动作中，它可能执行目录查找以打开文件、读写文件数据和关闭文件。现在，假设网络断开、服务器故障、或服务器定期关机等，突然地，远程文件系统不可访问。这种情况很常见，所以客户系统不应该将它作为本地文件系统故障一样来处理。但是，系统应该终止对故障服务器的所有操作，或者延迟操作直到服务器再次可用为止。这种故障语义是由远程文件系统协议所定义和实现的。终止所有操作会导致用户丢失数据和耐性，因此，绝大多数 DFS 协议强制或允许延迟对远程主机的文件系统操作，以寄希望于远程主机会再次可用。

要恢复这种故障，在客户机和服务器之间可能需要一定的状态信息。如果服务器和客户机都维持它们当前活动和打开的文件，它们就能无缝地从故障中恢复过来。如果服务器故障，那么它必须知道哪些文件系统已输出，哪些已经被远程安装了，哪些文件被打开了。 NFS 采用一种简单方法，以实现无状态 DFS。简单地说，它假定除非已经安装了远程文件，并打开了文件，否则客户不会请求有关文件读写。NFS 协议携带所有需要的信息，以定位适当的文件并执行所请求的文件操作。同样，它并不跟踪哪个客户机安装了远程文件系统，而是假定客户机所请求的操作是合法的。这种无状态方法使 NFS 具有弹性并容易实现，但是它并不安全。例如，在没有进行必要的安装请求和许可检查时，伪造的读写请求会被 NFS 服务器允许。这些问题在行业标准的 NFS 4 中提出，其中的 NFS 是有状态的，以提高其安全性、性能和功能。

### 3. 文件保护

当信息保存在计算机系统中，需要保护其安全，使之不受物理损坏（可靠性）和非法访问（保护）。

可靠性通常是由文件备份来提供的。许多计算机都有系统程序，自动（或通过计算机操作员

干预）并定期地（一天或一周或一月一次）把可能被突然损坏的文件复制到磁带上。文件系统可能在以下情况下损坏：硬件问题（如读写错误），电源过高或故障、磁头损坏、灰尘、温度不适和故意破坏等。文件可能被无意删除。文件系统软件错误也会引起文件内容丢失。

保护有多种方法。对于小的、单用户系统，可以通过移走软盘和将它们锁在抽屉里或文件柜里提供保护。然而，对于多用户系统，则需要其他的机制。

### （1）访问类型

文件保护的需要是允许访问的直接结果。如果系统不允许对其他用户的文件进行访问，也就不需要保护了。因此，通过禁止访问可以提供完全保护。另外，可通过不加保护以提供自由访问。这两种方法都太极端，不适用于普通使用。人们所需要的是**控制访问**（controlled access）。

通过限制可进行的文件访问类型，保护机制可提供控制访问。是否允许访问的决定因素有好几个，其中之一就是所请求的访问类型。以下几种类型的操作都可以加以控制：

- **读**：从文件中读。
- **写**：对文件进行写或重写。
- **执行**：将文件装入内存并执行它。
- **添加**：将新信息添加到文件结尾部分。
- **删除**：删除文件，使其空间用于其他目的。
- **列表清单**：列出文件名称及其属性。

其他操作，如文件的重命名、复制、编辑，也可加以控制。然而对于许多系统，这些高层功能可以用系统程序调用低层系统调用来加以实现。保护可以只在低层提供。例如，复制文件可利用一系列读请求来简单地实现。这样，具有读访问的用户就可对文件进行复制、打印等。

目前，提出了许多保护机制。每种机制都有其优缺点，适用于特定的应用。小计算机系统（只为少数几个研究成员使用的）不需要提供大型企业级计算机（用于研究、商务其他人事）一样的访问类型。

### （2）访问控制

解决保护问题最为常用的方法是根据用户身份进行控制。不同用户可能对同一文件或目录需要有不同类型的访问。实现基于身份访问的最为普通的方法是为每个文件和目录增加一个**访问控制列表**（access-control list, ACL），以给定每个用户名及其所允许的访问类型。当一个用户请求访问一个特定文件时，操作系统检查该文件的访问控制列表。如果该用户属于可访问的，那么就允许访问。否则，会出现保护违约，且用户进程不允许访问文件。

这种方法的优点是可以使用复杂的访问方法。访问控制列表的主要问题是其长度。如果允许每个用户都能读文件，那么必须列出所有具有读访问权限的用户。这种技术有两个不好的结果：

- 创建这样的列表可能比较麻烦且很可能没有用处，尤其是在事先不知道系统的用户列表时。
- 原来固定大小的目录条目，现在必须是可变大小，这会导致更为复杂的空间管理。

这些问题可以通过使用精简的访问列表来解决。

为精简访问列表，许多系统为每个文件采用了三种用户类型：

- **拥有者**：创建文件的用户为拥有者。

- **组**：一组需要共享文件且需要类似访问的用户形成了组或工作组。
- **其他**：系统内的所有其他用户。

现在最为常用的方法是将访问控制列表与更为常用的用户、组和其他成员访问控制方案（前面述及的）一起组合使用。例如，Solaris 2.6 及后来版本一般使用三种访问类型，但在需要更细粒度的访问控制时可以允许增加访问控制列表。

作为一个例子，考虑一个用户 Sara 在写一本书。她雇了三个研究生（Jim、Dawn 和 Jill）来帮忙。该书的文本保存在名为 book 的目录中。与该目录相关的保护如下：

- Sara 应该能对其中文件执行所有操作。
- Jim、Dawn 和 Jill 应该只能对其中文件进行读和写，而不允许删除文件。
- 所有其他用户应用能对其中文件读但不能写。（Sara 希望尽可能多的用户能读到该书，以便能收到合适反馈）

为了实现这种保护，必须创建一个新组，称其为 text，并具有三个成员 Jim、Dawn 和 Jill。组 text 的名称必须与目录 book 相关联，且其访问权限必须按以上所描述的策略进行设置。

现在，假定有一个访问者，Sara 希望允许其暂时访问第一章。该访问者不能增加到组 text 中，因为这样会授予其访问所有章节。由于文件只能在一个组，故不能向第一章增加另一个组。采用增加访问控制列表功能，访问者可增加到第一章的访问控制列表。

为了使该方案正常工作，许可和访问权限必须紧密控制。这种控制可以通过多种方式完成。例如，在 UNIX 系统中，只有管理员或超级用户可以创建和修改组。因此，这种控制是由人机交互来完成的。在 VMS 系统中，文件拥有者可创建和修改其列表。

采用更为有限的保护分类只需要三个域就可定义保护。每个域通常为一组位，其中每位允许和拒绝相关访问。例如，UNIX 系统定义了三个域以分别用于文件拥有者、组和其他用户。每个域为三个位：rwx，其中 r 控制读访问，w 控制写访问，而 x 控制执行。一个单独的域用来保存的文件拥有者，文件的组，以及所有其他用户。采用这种方法，每个文件需要 9 位来记录保护信息。因此，对上面的例子，book 的保护域为如下：对于拥有者 Sara，所有三个位均已设置；对组 text，r 和 w 位设置；而对其他用户，只有 r 位设置了。

组合方法的困难之一是用户接口。用户必须能区分一个文件是否有可选的 ACL 许可。在 Solaris 例子中，普通许可之后的“+”表示有可选 ACL 许可。如

19 –rw-r--r--+ 1 jim staff 130 May 25 22:13 file1

一组独立命令 setfacl 和 getfacl 用来管理 ACL。

Windows XP 用户通常采用 GUI 管理访问控制列表。图 7.13 说明了 Windows XP 的 NTFS 文件系统上的文件许可窗口。在此例子中，用户“guest”被特别拒绝对文件 10.tex 的访问。

图 **7.13** Windows XP 访问控制列表管理

另一困难是当许可和 ACL 冲突时谁占先。例如，如果 Joe 在一个文件的组中，该组具有读权限，但该文件有一个 ACL 允许 Joe 读和写，那么 Joe 能写吗？Solaris 允许 ACL 许可占先（因为它们更为细致且缺省并不指派）。这遵守一个通常准则：特殊操作应该占先。

### （3）其他保护方式

保护问题的另一解决方案是为每个文件加上密码。正如对计算机系统的访问通常有密码控制一样，对文件的访问也可用密码控制。如果随机选择密码且经常修改，那么这种方案可有效地用于限制文件为少数知道密码的用户所访问。然而，这种方案有多个缺点。第一，用户需要记住的密码的数量过大，以致这种方案不可行。第二，如果所有文件只使用一个密码，那么它一旦被发现，所有文件就可被访问。有的系统（如 TOPS-20）允许用户为目录而不是文件关联密码，以解决这个问题。IBM VM/CMS 操作系统允许一个分区有三个密码，以分别用于读、写和多次访问。

有些单用户操作系统，如 MS-DOS 和早于 Mac OS X 版本的 Macintosh 操作系统提供很少的文件保护。由于这些系统现已联网以共享文件和进行通信，所以必须向这些操作系统增加必要的保护机制。在现有的操作系统上增加功能要比在新操作系统上设计功能要难。而且，这种更新通常效果欠佳，且不可能无缝。

对于多层目录结构，不仅需要保护单个文件，而且还需要保护子目录内的文件，即需要提供一种机制来进行目录保护。保护目录的必要操作不同于文件操作。需要控制在一个目录中创建和删除文件。另外，可能需要控制一个用户能确定一个目录内是否有一个文件存在。有时，关于文件存在和名称的知识本身就很重要。因此，列出目录内容必须是个保护操作。类似地，如果一个路径名表示一个目录内的一个文件，那么用户必须允许访问其目录和该文件。对于支持文件有多个路径名的系统（采用无环图结构目录和图结构目录），根据所使用的路径名的不同，一个用户可能对同一个文件具有不同的访问权限。

#### UNIX 系统中的许可

在 UNIX 系统中，目录保护类似于文件保护。即，每个子目录都有三个相关域：拥有者、组和其他，每个域都有三个位 rwx。因此，如果一个子目录的相应域的 r 位已设置，那么一个用户可列出其内容。类似地，如果一个子目录(foo)的相应域的 x 位已设置，那么用户可改变其当前目录为该目录（foo）。

图 7.14 显示了在 UNIX 环境下一个目录的列表。第一个域表示文件或目录的权限。第一个字母 d 表示子目录。图 7.14 还列出了文件链接数、拥有者名称、组名称、文件字节数、上次修改时间和文件名称（具有可选扩展部分）

| | | | | | |
|---|---|---|---|---|---|
| -rw-rw-r- | 1 pbg | staff | 31200 | Sep 3 08:30 | intro.ps |
| drwx----- | 5 pbg | staff | 512 | Jul 8 09:33 | private/ |
| drwxrwxr-x | 2 pbg | staff | 512 | Jul 8 09:35 | doc/ |
| drwxrwx-- | 2 pbg | student | 512 | Aug 3 14:13 | student-proj/ |
| -rw-r--r- | 1 pbg | staff | 9423 | Feb 24 1999 | program.c |
| -rwxr-xr-x | 1 pbg | staff | 20471 | Feb 24 2000 | program |
| drwx-x-x | 4 pbg | faculty | 512 | Jul 31 10:31 | lib/ |
| drwx----- | 3 pbg | staff | 1024 | Aug 29 06:52 | mail/ |
| drwxrwxrwx | 3 pbg | staff | 512 | Jul 8 09:35 | test/ |

图 **7.14** 目录列表示例