内部名字是MFT中文件的索引。目录的任务是在外部名字和内部名字之间提供映射，如图13-4所示。

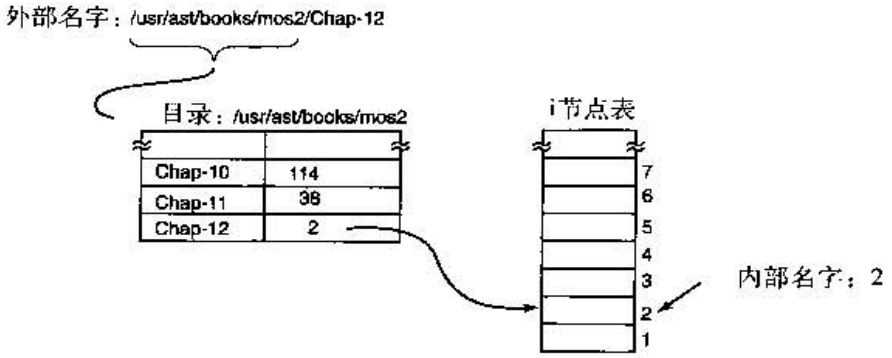

图13-4 目录用来将外部名字映射到内部名字上

在许多情况下（例如上面给出的文件名的例子），内部名字是一个无符号整数，用作进入一个内部表格的索引。表格-索引名字的其他例子还有UNIX中的文件描述符和Windows Vista中的对象句柄。注意这些都没有任何外部表示，它们严格地被系统和运行的进程所使用。一般而言，对于当系统重新启动时就会丢失的暂时的名字，使用表格索引是一个很好的主意。

操作系统经常支持多个名字空间，既在内部又在外部。例如，在第11章我们了解了Windows Vista支持的三个外部名字空间：文件名、对象名和注册表名（并且还有我们没有考虑的活动目录名）。此外，还存在着使用无符号整数的数不清的内部名字空间，例如对象句柄、MFT项等。尽管外部名字空间中的名字都是Unicode字符串，但是在注册表中查寻一个文件名是不可以的，正如在对象表中使用MFT索引是不可以的。在一个良好的设计中，相当多的考虑花在了需要多少个名字空间，每个名字空间中名字的语法是什么，怎样分辨它们，是否存在抽象的和相对的名字，如此等等。

13.3.5 绑定的时机

正如我们刚刚看到的，操作系统使用多种类型的名字来引用对象。有时在名字和对象之间的映射是固定的，但是有时不是。在后一种情况下，何时将名字与对象绑定可能是很重要的。一般而言，早期绑定（early binding）是简单的，但是不灵活，而晚期绑定（late binding）则比较复杂，但是通常更加灵活。

为了阐明绑定时机的概念，让我们看一看某些现实世界的例子。早期绑定的一个例子是某些高等学校允许父母在婴儿出生时登记入学，并且预付当前的学费。以后当学生长大到18岁时，学费已经全部付清，无论此刻学费有多么高。

在制造业中，预先定购零部件并且维持零部件的库存量是早期绑定。相反，即时制造要求供货商能够立刻提供零部件，不需要事先通知。这就是晚期绑定。

程序设计语言对于变量通常支持多种绑定时机。编译器将全局变量绑定到特殊的虚拟地址，这是早期绑定的例子。过程的局部变量在过程被调用的时刻（在栈中）分配一个虚拟地址，这是中间绑定。存放在堆中的变量（这些变量由C中的malloc或Java中的new分配）仅仅在它们实际被使用的时候才分配虚拟地址，这便是晚期绑定。

操作系统对大多数数据结构通常使用早期绑定，但是偶尔为了灵活性也使用晚期绑定。内存分配是一个相关的案例。在缺乏地址重定位硬件的机器上，早期的多道程序设计系统不得不在某个内存地址装载一个程序，并且对其重定位以便在此处运行。如果它曾经被交换出去，那么它就必须装回到相同的内存地址，否则就会出错。相反，页式虚拟内存是晚期绑定的一种形式。在页面被访问并且实际装入内存之前，与一个给定的虚拟地址相对应的实际物理地址是不知道的。

晚期绑定的另一个例子是GUI中窗口的放置。在早期图形系统中，程序员必须为屏幕上的所有图像设定绝对屏幕坐标，与此相对照，在现代GUI中，软件使用相对于窗口原点的坐标，但是在窗口被放置在屏幕上之前该坐标是不确定的，并且以后，它甚至是可能改变的。

13.3.6 静态与动态结构

操作系统设计人员经常被迫在静态与动态数据结构之间进行选择。静态结构总是简单易懂，更加容易编程并且用起来更快；动态结构则更加灵活。一个显而易见的例子是进程表。早期的系统只是分配一个固定的数组，存放每个进程结构。如果进程表由256项组成，那么在任意时刻只能存在256个进程。试图创建第257个进程将会失败，因为缺乏表空间。类似的考虑对于打开的文件表（每个用户的和系统范