在许多系统中，进程可以请求操作系统在一定的时间间隔之后向它报警。警报通常是信号、中断、消息或者类似的东西。需要这类报警的一个应用是网络，当一个数据包在一定时间间隔之内没有被确认时，该数据包必须重发。另一个应用是计算机辅助教学，如果学生在一定时间内没有响应，就告诉他答案。

如果时钟驱动程序拥有足够的时钟，它就可以为每个请求设置一个单独的时钟。如果不是这样的情况，它就必须用一个物理时钟来模拟多个虚拟时钟。一种方法是维护一张表，将所有未完成的定时器的信号时刻记入表中，还要维护一个变量给出下一个信号的时刻。每当日时间更新时，时钟驱动程序进行检查以了解最近的信号是否已经发生。如果是的话，则在表中搜索下一个要发生的信号的时刻。

如果预期有许多信号，那么通过在一个链表中把所有未完成的时钟请求按时间排序链接在一起，这样来模拟多个时钟则更为有效，如图5-34所示。链表中的每个表项指出在前一个信号之后等待多少时钟滴答引发下一个信号。在本例中，等待处理的信号对应的时钟滴答分别是4203、4207、4213、4215和4216。

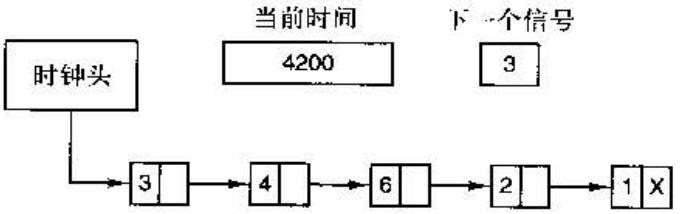

图5-34 用单个时钟模拟多个定时器

在图5-34中，经过3个时钟滴答发生下一个中断。每一次滴答时，下一个信号减1，当它变为0时，就引发与链表中第一个表项相对应的信号，并将这一表项从链表中删除，然后将下一个信号设置为现在处于链表头的表项的取值，在本例中是4。

注意在时钟中断期间，时钟驱动程序要做几件事情——将实际时间增1，将时间片减1并检查它是否为0，对CPU记账，以及将报警计数器减1。然而，因为这些操作在每一秒之中要重复许多次，所以每个操作都必须仔细地安排以加快速度。

操作系统的组成部分也需要设置定时器，这些定时器被称为监视定时器（watchdog timer）[⊖]。例如，为了避免磨损介质和磁头，软盘在不使用时是不旋转的。当数据需要从软盘读出时，电机必须首先启动。只有当软盘以全速旋转时，I/O才可以开始。当一个进程试图从一个空闲的软盘读取数据时，软盘驱动程序启动电机然后设置一个监视定时器以便在足够长的时间间隔之后引发一个中断（因为不存在来自软盘本身的达到速度的中断）。

时钟驱动程序用来处理监视定时器的机制和用于用户信号的机制是相同的。惟一的区别是当一个定时器时间到时，时钟驱动程序将调用一个由调用者提供的过程，而不是引发一个信号。这个过程是调用者代码的一部分。被调用的过程可以做任何需要做的工作，甚至可以引发一个中断，但是在内核之中中断通常是不方便的并且信号也不存在。这就是为什么要提供监视定时器机制。值得注意的是，只有当时钟驱动程序与被调用的过程处于相同的地址空间时，监视定时器机制才起作用。

时钟最后要做的事情是剖析（profiling）。某些操作系统提供了一种机制，通过该机制用户程序可以让系统构造它的程序计数器的一个直方图，这样它就可以了解时间花在了什么地方。当剖析是可能的事情时，在每一时钟滴答驱动程序都要检查当前进程是否正在被进行剖析，如果是，则计算对应于当前程序计数器的区间（bin）[⊜]号（一段地址范围），然后将该区间的值加1。这一机制也可用来对系统本身进行剖析。

5.5.3 软定时器

大多数计算机拥有辅助可编程时钟，可以设置它以程序需要的任何速率引发定时器中断。该定时器是主系统定时器以外的，而主系统定时器的功能已经在上面讲述了。只要中断频率比较低，将这个辅助定时器用于应用程序特定的目的就不存在任何问题。但是当应用程序特定的定时器的频率非常高时，麻烦就来了。下面我们将简要描述一个基于软件的定时器模式，它在许多情况下性能良好，甚至在相当高的频率

⊖ watchdog timer也经常译为看门狗定时器。——译者注

⊜ 直方图（histogram）用于描述随机变量取值分布的情况，虽然在中文术语中有一个“图”字，但并不是必须用图形来表示。它将随机变量（对于本例而言是程序计数器的取值）的值空间（对于本例而言是进程的地址空间）划分成若干个小的区间，每个小区间就是一个bin。通过计算随机变量的取值落在每个小区间内的次数就可以得到直方图。如果用图形表示直方图的话则表现为一系列高度不等的柱状图形。——译者注