情况，即有一个警报。然而，由于大多数时间不存在警报待完成，并且由于删除一个现有的警报代价高昂，所以区分这两种情况是一个好主意。

做这件事情的一种方法是在进程表中保留一个位，表明是否有一个警报待完成。如果这一位为0，就好办了（只是添加一个新的定时器队列项而无须检查）。如果该位为1，则必须检查定时器队列。

13.5 项目管理

程序员是天生的乐观主义者。他们中的大多数认为编写程序的方式就是急切地奔向键盘并且开始击键，不久以后完全调试好的程序就完成了。对于非常大型的程序，事实并非如此。在下面几节，关于管理大型软件项目，特别是大型操作系统项目，我们有一些看法要陈述。

13.5.1 人月神话

经典著作《人月神话》的作者Fred Brooks是OS/360的设计者之一，他后来转向了学术界。在这部经典著作中，Fred Brooks讨论了建造大型操作系统为什么如此艰难的问题（Brooks, 1975, 1995）。当大多数程序员看到他声称程序员在大型项目中每年只能产出1000行调试好的代码时，他们怀疑Brooks教授是否生活在外层空间，或许是在臭虫星（Planet Bug——此处Bug为双关语）上。毕竟，他们中的大多数在熬夜的时候一个晚上就可以产出1000行程序。这怎么可能是任何一个IQ大于50的人一年的产出呢？

Brooks指出的是，具有几百名程序员的大型项目完全不同于小型项目，并且从小型项目获得的结果并不能放大到大型项目。在一个大型项目中，甚至在编码开始之前，大量的时间就消耗在规划如何将工作划分成模块、仔细地说明模块及其接口，以及试图设想模块将怎样互相作用这样的事情上。然后，模块必须独立地编码和调试。最后，模块必须集成起来并且必须将系统作为一个整体来测试。通常的情况是，每个模块单独测试时工作得十分完美，但是当所有部分集成在一起时，系统立刻崩溃。Brooks将工作量估计如下：

- 1/3规划
- 1/6编码
- 1/4模块测试
- 1/4系统测试

换言之，编写代码是容易的部分，困难的部分是断定应该有哪些模块并且使模块A与模块B正确地交互。在由一名程序员编写的小型程序中，留待处理的所有部分都是简单的部分。

Brooks的书的标题来自他的断言，即人与时间是不可互换的。不存在“人月”这样的单位。如果一个项目需要15个人花2年时间构建，很难想像360个人能够在1个月内构建它，甚至让60个人在6个月内做出它或许也是不可能的。

产生这一效应有三个原因。第一，工作不可能完全并行化。直到完成规划并且确定了需要哪些模块以及它们的接口，甚至都不能开始编码。对于一个2年的项目，仅仅规划可能就要花费8个月。

第二，为了完全利用数目众多的程序员，工作必须划分成数目众多的模块，这样每个人才能有事情做。由于每个模块可能潜在地与每个其他模块相互作用，需要将模块-模块相互作用的数目看成随着模块数目的平方而增长，也就是说，随着程序员数目的平方而增长。这一复杂性很快就会失去控制。对于大型项目而言，人与月之间的权衡远不是线性的，对63个软件项目精细的测量证实了这一点（Boehm, 1981）。

第三，调试工作是高度序列化的。对于一个问题，安排10名调试人员并不会加快10倍发现程序错误。事实上，10名调试人员或许比一名调试人员还要慢，因为他们在相互沟通上要浪费太多的时间。

对于人员与时间的权衡，Brooks将他的经验总结在Brooks定律中：

对于一个延期的软件项目，增加人力将使它更加延期。

增加人员的问题在于他们必须在项目中获得培训，模块必须重新划分以便与现在可用的更多数目的程序员相匹配，需要开许多会议来协调各方面的努力等。Abdel-Hamid和Madnick（1991）用实验方法证实了这一定律。用稍稍不敬的方法重述Brooks定律就是：

无论分配多少妇女从事这一工作，生一个孩子都需要9个月。

13.5.2 团队结构

商业操作系统是大型的软件项目，总是需要大型的人员团队。人员的质量极为重要。几十年来人们