## Calendar Scheduler

日历管理是计算的难题之一。许多人设计了它，但是到目前为止，还没有完全令人满意的解决方案出现。日历计划程序为许多人维护一致的会议计划。这些日程安排至少记录每个会议的时间，持续时间和参与者。某些会议可能包括日历日程表未维护其日程表的人员。可以在任何时候（直到会议发生的时刻）添加或删除会议，并且可以添加或删除会议的参与者。可以在所有（或足够）会议参与者方便的任何时间安排会议，除了某些会议可能需要按特定顺序进行。调度器可以维护关于其所服务的人的度偏好的信息

[历史](http://www.cs.cmu.edu/~ModProb/CShist.html):

这是一个长期存在的问题，通常很难解决。现有的产品能够记录简单的调度决策并共享数据库，但是远远不能满足个人喜好。

很明显，只要有纸和笔，这个问题就已经有纸和笔的解决方案了，在此之前无疑还有其他解决方案。在计算机领域，有单独的计算器大小的机器可以代替日历笔记本（例如……？有很多这样的机器），以及能够接管的多用户系统上的许多调度程序。一些时间选择和一致性检查任务

<<描述当前产品及其缺点xc8

该问题已用于重点讨论需求和规范。Axel van Lamsweerde提供了这些讨论的结果，作为扩展的问题陈述[vanLamsweerde92，vanLamsweerde93]。

[设计注意事项](http://www.cs.cmu.edu/~ModProb/CSdsgn.html)

挑战来自两个方面：日历的多方，分布式，异构，异步性质；以及适应个人喜好的需求，其中有些是私人的或表达不清的。

此问题可能面临相当大的硬件和环境约束。例如，个人电子笔记本还不能自由通信，因此不可能假设所有感兴趣的日历都可以立即或同时访问。

用户的期望也是考虑替代方案的一个因素；更改会议时完全重新安排每个人的时间可能是不可接受的：添加和删除会议时必须保持一定的稳定性。通过考虑一段时间对于一个参与者或一组参与者“方便”意味着什么，这个问题可能会变得更加复杂。用户可以在允许的时间表上施加什么样的约束？

影响体系结构的一些关键考虑因素包括：

个人灵活性：可以表达和容纳一组个人偏好的丰富程度？

异质性：不同的个人日历表示处理得如何？

优先级和条件：当可以提供有关优先级的强度程度时，系统如何解决冲突？

易用性：一个人定义和操纵一组会议要参加会议有多容易？如何合并来自多台计算机的信息？例会处理得如何？可以定义法定人数吗？

最优性： 如果有时间表，系统会找到时间表吗？该体系结构是否可以支持应急策略，例如在缺少完整信息的情况下？

[解决方案](http://www.cs.cmu.edu/~ModProb/CSsoln.html)：

罗布·艾伦（Rob Allen）提出了一个简单的问题并组织了讨论。Axel van Lamsweerde提供了与问题说明链接的扩展规范。

## Compiler

编译器将编程语言转换为机器语言。它们还与其他编程工具（例如交互式编辑器和调试器）进行交互。

编译器将编程语言中的源代码转换为可以与其他目标代码链接并在计算机上执行的目标代码。

[历史](http://www.cs.cmu.edu/~ModProb/CMPhist.html)

编译器是最古老的，容易理解的非凡软件系统。编译器是介绍多模块软件组织的本科课程的首选示例，但是高性能增量式分布式编译器继续给设计带来挑战。

[设计注意事项](http://www.cs.cmu.edu/~ModProb/CMPdsgn.html)

单的编译器可以作为类练习。但是，生产编译器必须响应有关性能和可用性的问题。

架构必须*响应其环境的使用情况*。例如，学生编译器必须支持小型程序的快速周转，但不必太在乎代码的质量。但是，对于生产编译器，代码速度可能至关重要。

该体系结构必须与其*相关的软件开发环境*兼容。例如，这可以是批处理或交互式的。

[解决方案](http://www.cs.cmu.edu/~ModProb/CMPsoln.html)

许多编译器设计教科书提供了解决方案。Seshadri [Seshadri88]显示了如何创建并行版本。Perry和Wolf [PerryWolf92]以及Garlan和Shaw [GarlanShaw93]从体系结构的角度研究了其中一些解决方案。

[贡献者](http://www.cs.cmu.edu/~ModProb/CMPcont.html)

Alex Wolf，Dewayne Perry和Bill Griswold指出，如果没有编译器示例，此集合将是不足的。