**操作系统设计读后感**

为了设计一个成功的操作系统，设计人员必须要有清晰的思路与目标。作者认为对于一个通用的操作系统需要留心四个基本要素，1)定义抽象概念; 2)提供基本操作; 3)确保隔离; 4)管理硬件。对于每一个抽象概念可以采用具体数据结构的形式实例化，基本操作则能够处理这些数据结构。对于用户而言，基本操作可以通过系统调用加以利用。由于多用户以及多进程的存在，操作系统必须确保它们之间能够”隔离“，然而”隔离”应该是有选择性的，因为用户可能希望能够共享某些数据和资源。至于管理硬件基本上是操作系统最基本的功能了。

设计操作系统为什么是困难的？作者提出了几点看法：1)操作系统过于庞大，各个子系统之间的相互作用往往导致意想不到的后果; 2)操作系统必须处理并发;3)操作系统必须能够处理好用户之间权限的问题; 4)操作系统必须能够正确的处理信息与资源的共享; 5)设计人员必须考虑未来可能出现的变化; 6)设计人员必须确保操作系统一定程度的通用性; 7)设计人员必须考虑操作系统的可移植性; 8)设计人员必须考虑操作系统向后兼容的问题。第一点是主要是由于我们需要操作系统能够提供更多的功能，所以它会相当庞大; 第二点是为了追求更快的速度; 第三点是出于安全的考虑; 第四点从用户角度而言是必须的; 第五、六、七点更像是为了市场化而准备的;至于第八点在某种程度上已经不仅仅局限在操作系统这里了，比如web开发行业让人诟病的浏览器兼容性问题，过去的许多设计可能已经过时了，但由于系统生态的问题，设计人员必须要实现向下兼容，典型的例子就是在windows下由于兼容dos不能用常规手段命名一个名字为COM1的文件夹。

编写一个操作系统并不容易，作者认为最好的起点是考虑操作系统提供的接口。关于接口设计的原则简而言之：简单、完备、高效。在此基础上，作者提出了范型，范型的存在很好的实现了体系结构一致性，同时也方便了操作系统调用接口的设计。对于系统调用的设计，通常较好的实现是具有处理一般情况的系统调用，之后由不同的库来对这个调用进行封装使得其能够处理特定的某些情况。如何实现一个操作系统？需要考虑的一些底层技术包括1)系统结构; 2)机制与策略的分离; 3)正交性; 4)命名;5)绑定的时机; 6)静态与动态结构的选择; 7)自底向上与自顶向下的实现以及一些其它的实用技术包括隐藏硬件、间接处理技术、可重用技术等。显然需要的技术远远不止这些，操作系统是复杂的，更多的细节还需要考虑。

当操作系统设计完成时，设计人员需要评估并尽可能的改进性能。时间与空间的权衡是改进性能的一般性方法，典型的是用更多的内存换取更快的运行速度。另一种技术是高速缓存，在cpu与内存之间使用高速缓存有效的解决了速度与价格之间的矛盾。由于程序运行存在局部性的特点，我们可以利用局部性来改进性能。还有便是优化常见的情况，对于常见的情况要使其尽可能的快，对于很少发生的情况使其正确就足够了。一般来讲优化的性能只需要到达一个合理的水平，不必为了很小的提升而花费大量的时间。

接下来是项目管理的部分，需要明白的是人与时间是不可交换的，不是人越多就能在越少的时间内完成，对于大型的项目，规划可能需要花费大量的时间。在团队中需要有一名才智超群的人员对设计进行控制，这样能够保持体系结构的一致性。最后是操作系统设计的趋势，包括虚拟化、多核、大型地址空间、网络连接、并行与分布式系统等，值得一提的是这里提到的预测在如今已经全部应验。