一种新的操作系统设计

我一直在试图利用程序语言的设计原理，设计一种超越“Unix 哲学”的操作系统。这里是我的设想：

- 这种系统里面的程序间通信不使用无结构的字符串，而是使用带有类型和结构的数据。在这样的系统里面，Unix 和其它类似操作系统（比如 Windows）里的所谓“应用程序”的概念基本上完全消失。系统由一个个很小的“函数”组成，每个函数都可以调用另外一个函数，通过参数传递数据。每个函数都可以手动或者自动并发执行。用现在的系统术语打个比方，这就像是所有代码都是“库”代码，而不存在独立的“可执行文件”。

- 由于参数是数据结构而不是字符串，这避免了程序间通信繁琐的编码和解码过程。使得“进程间通信”变得轻而易举。任何函数都可以调用另一个函数来处理特定类型的数据，这使得像 “OLE 嵌入”这样的机制变得极其简单。

- 所有函数由同一种先进的高级程序语言写成，所以函数间的调用完全不需要“翻译”。不存在 SQL injection 之类由于把程序当成字符串而产生的错误。

- 由于这种语言不允许应用程序使用“指针运算”，应用程序不可能产生 segfault 一类的错误。为了防止不良用户手动在机器码里面加入指针运算，系统的执行的代码不是完全的机器代码，而必须通过进一步的验证和转换之后才会被硬件执行。这有点像 JVM，但它直接运行在硬件之上，所以必须有一些 JVM 没有的功能，比如把内存里的数据结构自动换出到硬盘上，需要的时候再换进内存。

- 由于没有指针运算，系统可以直接使用“实地址”模式进行内存管理，从而不再需要现代处理器提供的内存映射机制以及 TLB。内存的管理粒度是数据结构，而不是页面。这使得内存访问和管理效率大幅提高，而且简化了处理器的设计。据 Kent Dybvig 的经验，这样的系统的内存使用效率要比 Unix 类的系统高一个数量级。

- 系统使用与应用程序相同的高级语言写成，至于“系统调用”，不过是调用另外一个函数。由于只有这些“系统驱动函数”才有对设备的“引用”，又因为系统没有指针运算，所以用户函数不可能绕过系统函数而非法访问硬件。

- 系统没有 Unix 式的“命令行”，它的“shell”其实就是这种高级语言的 REPL。用户可以在终端用可视化的结构编辑方式输入各种函数调用，从而启动进程的运行。所以你不需要像 Unix 一样另外设计一种毛病语言来“粘接”应用程序。

- 所有的数据都作为“结构”，保存在一个分布式的数据共享空间。同样的那个系统语言可以被轻松地发送到远程机器，调用远程机器上的库代码，执行任意复杂的查询索引等动作，取回结果。这种方式可以高效的完成数据库的功能，然而却比数据库简单很多。所谓的“查询语言”（比如 SQL，Datalog，Gremlin，Cypher）其实是多此一举，它们远远不如普通的程序语言强大。说是可以让用户“不需要编程，只提出问题”，然而它们所谓的“优化”是非常局限甚至不可能实现的，带来的麻烦远比直接编程还要多。逻辑式编程语言（比如 Prolog）其实跟 SQL 是一样的问题，一旦遇到复杂点的查询就效率低下。所以系统不使用关系式数据库，不需要 SQL，不需要 NoSQL，不需要 Datalog。

- 由于数据全都是结构化的，所以没有普通操作系统的无结构“文件系统”。数据结构可能通过路径来访问，然而路径不是一个字符串或者字符串模式。系统不使用正则表达式，而是一种类似 NFA 的数据结构，对它们的拆分和组合操作不会出现像字符串那样的问题，比如把 /a/b/ 和 /c/d 串接在一起就变成错误的 /a/b//c/d。

- 所有的数据在合适的时候被自动同步到磁盘，并且进行容错处理，所以即使在机器掉电的情况，绝大部分的数据和进程能够在电源恢复后继续运行。

- 程序员和用户几乎完全不需要知道“数据库”或者“文件系统”的存在。程序假设自己拥有无穷大的空间，可以任意的构造数据。根据硬件的能力，一些手动的存盘操作也可能是有必要的。

- 为了减少数据的移动，系统或者用户可以根据数据的位置，选择： 1）迁移数据，或者 2）迁移处理数据的“进程”。程序员不需要使用 MapReduce，Hadoop 等就能进行大规模并行计算，然而表达能力却比它们强大很多，因为它们全都使用同一种程序语言写成。

我曾经以为我是第一个想到这个做法的人。可是调查之后发现，很多人早就已经做出了类似的系统。Lisp Machine 似乎是其中最接近的一个。Oberon 是另外一个。IBM System/38 是类似系统里面最老的一个。最近一些年出现的还有微软的 Singularity，另外还有人试图把 JVM 和 Erlang VM 直接放到硬件上执行。

所以这篇文章的标题其实是错的，这不是一种“新的操作系统设计”。它看起来是新的，只不过因为我们现在用的操作系统忘记了它们本该是什么样子。我也不该说它“超越了 Unix 哲学”，而应该说，所谓的 Unix 哲学其实是历史的倒退。