Linux 兼容内核
-提高 Linux 市场竞争力的一个手段

毛德操

虽然 Linux 在服务器、嵌入式系统、以及高性能计算领域都有不俗的表现，可是在桌面领域的市场占有率很低却是个不争的事实。尤其在中国，如果与 Windows 相比，桌面 Linux 的实际使用率几乎可以忽略不计。然而，对于一个现代操作系统而言，桌面应用是其主流；在桌面输了，那就是从总体上输了。

计算机技术的发展史上曾经出现过几次技术上更为先进的产品输给技术上较为落后(但在发展中)的产品的先例，而 Linux/Unix 和 Windows 在过去 20 年间的竞争恰恰是其中最大的一次。关键在于：当技术体现为某种大众化的产品时，就不再是单纯的技术问题了。

笔者不想在这篇短文中分析和总结之所以造成这种局面的原因和教训；更不想在这儿怨天尤人、非议微软的某些受到指责的行为；而只是针对已成之局提出提高 Linux 市场竞争力的一个手段。这所谓提高，也只是略有提高就好；桌面操作系统的市场那么大，哪怕一个百分点就已经不小了。

那么已成之局是什么呢？最主要的有这么几点：

1） 整整一代人的用户、甚至还包括下一代，都已经习惯于使用 Windows。这些人在学校里学的、在培训班里学的、从朋友/同事那儿学的都是 Windows，在家里用的、在单位用的、在网吧用的都是 Windows，因而它的人机界面和操作方式早已经成为事实上的标准。

2） Windows 的应用软件比 Linux 丰富得多。Windows 有着庞大的、高质量的第三方软件供应。无论是应用软件、中间件、还是系统软件，至少在数量上都是 Linux 所无可比拟的。事实上，就绝大多数的软件(甚至硬件)供应商的产品来看，其第一个版本几乎肯定是为 Windows 开发的，然后才有可能想到 Linux，实际上还往往置 Linux 于不顾。

3） 至少在表面上，Windows 的售后服务和支持比 Linux 的好。实际上，这种售后服务和支持绝大部分是针对第三方软件、由第三方软件供应商提供的；可是因为这些软件是在 Windows 上运行、而且只在 Windows 上运行，就给人一种印象，似乎 Windows 的用户得到了支持、而 Linux 的用户得不到，或者 Windows 是专业水平的、而 Linux 是业余水平的。

4） Windows 已经占据计算机应用的主流地位，运行着 Linux 的计算机必须能与运行着 Windows 的计算机互连互通，并“无缝”地、按 Windows 所定义的格式交换各种数据和文件(这一点对于电子商务和电子政务尤为重要)。

实际上，Windows 本身的用户界面和操作方式也在变。但是由微软来变则可(不可也得可)；而若是别的人来要求用户改变使用习惯，那就难了。

这些问题大家都看到了，但是应对的策略和意见则各有不同。

1）有一种意见和策略认为最主要的是上述的第二条、即 Linux 上的应用软件太少的问

题。既然是应用软件太少，就大力开发应用软件，并与第三方软件商合作、让它们提供 Linux 版的产品。久而久之，就应该可以使差距降下来。再说，就具体的用户而言，真正需要用到的软件还是很有限的，很可能只是“老三篇”(浏览器、Office、播放器)。反过来，Linux 也有不少优点，对用户应该也有些吸引力，更不必刻意去模仿 Windows。这样，慢慢就可以缩小差距，从而把用户吸引过来。国外特别是欧美、有不少人持此种意见。但是笔者对此甚感怀疑。据笔者观察，就桌面系统而言，过去十年内这种差距似乎反有扩大的趋势。问题在于，Windows 的应用软件也在发展、甚至更快地发展，新的应用又在不断涌现。而第三方软件商的投入与 Linux 市场竞争力的提高，就成了某种“先有鸡还是先有蛋”的问题。

2) 另一种意见和策略既重视第二条，也重视第一条和第四条，即用户已经习惯于使用 Windows 及其应用软件、以及与 Windows 系统交换数据和文件的问题，所以在开发 Linux 应用软件时很注意与 Windows“兼容”，实际上就是使应用软件在外观、功能、操作方式和效果等方面与 Windows 上的对应软件尽可能一致，让用惯了 Windows 的人可以毫无困难地转到 Linux 上来。在这一方面，OpenOffice 等软件的开发就很典型。持这种意见的人国外也有不少，国内则更多。说穿了，这就是走“仿制”Windows 应用软件的道路。然而，问题在于 Windows 的应用软件有那么多，又层出不穷，实在是仿不胜仿。

3) “仿制”既然不能很好地解决问题，那就来“嫁接”，就是设法在 Linux 上直接运行 Windows 应用软件的二进制映像。在这方面最重要的进展是开源软件 Wine 的开发。Wine 在 Linux 内核和 Windows 应用程序之间提供一个适配层。它的上方为 Windows 应用程序提供 Win32 API (实际上也包括 16 位界面)，为应用程序提供以动态连接库为主的 Windows 运行环境。而其下方，则把本来应该是对 Windows 内核的系统调用翻译/转化成对 Linux 内核的系统调用。这样，Windows 应用程序实质上是在一个模拟/仿真的 Windows 内核上运行，而 Wine 的主要功能就是在 Linux 内核外面实现对 Windows 内核的模拟/仿真。不过 Wine 的设计者不愿意别人认为 Wine 是个仿真器。这主要是因为“仿真器”这个词容易使人误解，以为是对 CPU 机器指令的仿真，那是效率非常低的，而 Wine 的效率确实比普通意义上的仿真要高得多。在过去几年中，Wine 一直在稳步地发展，过不多久就会有个较新的版本发布。有几个公司，如 Linspire、Xandros 等，也在推出基于 Wine 的产品。一般是同时有两套版本，一套是开源的，另一套则包括一些不公开源码的优化和改进。

4) 还有一种策略是让用户既可以使用 Windows，也可以使用 Linux。最简单的办法就是所谓双引导，即在同一台机器上装上两个系统、而在引导时加以选择。这样，有些事情适合在 Linux 上做，就引导 Linux，反之则引导 Windows。可是这样很不方便，更好的办法就是 Vmware 所采用的虚拟机技术，就是在 Linux 上再装一个 Windows 系统，让 Windows 作为 Linux 的一个应用来运行；或者也可以反过来，在 Windows 上再装一个 Linux 系统，让 Linux 作为 Windows 的一个应用来运行。这种策略最大的问题就是不能摆脱 Windows 操作系统，因而对于提高 Linux 市场竞争力的作用也就不大。

在这四种不同的意见和策略中，笔者以为最有效、最有前途的是第三种。说它有效和有前途，是因为至少从理论上说可以把绝大多数的第三方软件变成与操作系统无关。对于第三方软件的购买和使用者来说，本来是一旦买了/用了这种软件就好像定了终身，从此只能用

Windows 了，而现在却还可以有选择，还可以摆脱 Windows，这对于 Linux 当然是意义重大。

然而，在实践中，情况却并不如想象中那样美妙。主要有这么一些问题：

一、Wine 对 Windows 应用软件的支持面不如想象中那么大，一部分原因是 Wine 本身的开发有个过程，所以对 Win32 API 的支持一时还不那么全面。不过这并不是根本性的问题，随着事间的推移会不断改进。事实上，今年的版本比前年的版本就已经充实了不少。

二、虽然 Wine 不是仿真器，不像仿真器那么低效；但是既然要把原来是对 Windows 内核的系统调用转化成对 Linux 内核的系统调用，而两个内核的系统调用又存在着相当程度的差异，那么这种转化难免会使效率有所降低。这有点像是用一种高级语言来实现另一种高级语言，功能是达到了，性能却总不那么好。实际上，许多系统调用本质上是粒度相当大的“宏”操作，在这么大的粒度下很难高效地实现某些微妙的机制。甚至，有些系统调用还可能实际上无法很贴切地翻译成另一组系统调用，以至某些软件在运行时出现不同于 Windows 上的效果。与上一条相比，这一条是带有根本性的问题。

三、更重要的是，Wine 只是在用户空间为 Windows 应用软件营造了一个运行环境，却没有涉及内核、特别是没有涉及设备驱动程序。这又导致两个方面的问题。首先，许多外部设备厂商现在只为 Windows 提供设备驱动程序、一般是扩展名为.sys 的模块。如果不能把这些设备驱动模块安装到 Linux 内核中，就意味着 Linux 一时还不能支持这些外部设备。在这方面，开源项目 NdisWrapper 已经进行了努力、并取得了不小的成绩；但是它主要是针对一些网卡的驱动程序而开发的，还不够全面。其次，实际上更为重要的是，现在有些应用软件是与某些特定的设备驱动模块配套运行的；缺了内核中的特定设备驱动模块，相应的应用软件也就不能运行了。随着“数字家庭”等等应用和技术的发展，这样的倾向应该还会加剧。传统意义上的内核是公共的，设备驱动属于内核，因而也是公共的；而现在的内核却在朝着动态可定制，或者面向特定应用进行个性化配置的方向在发展。如果把内核比作一盆水，而传统意义上的应用软件像是浮在水面上的鸡蛋，那么现在的有些应用却像个哑铃了。就是说，它的一部分还在水面上、还像个鸡蛋，但是另一部分却在水底，当中有个柄，把两个球状物连成一个整体。显然，缺了内核中的设备驱动这一块，Wine 的作用就不完善。而且，二者必须有机地结合起来。

四、Wine 对 Windows 应用软件的支持面不如想象中那么大，还有一部分原因是存在着应用软件绕过 Win32 API 进入内核的可能。这往往发生在一些老的、特别是 16 位的软件，以及一些特殊软件。

由此看来，Wine 还不能很好地解决问题。正是出于这样的考虑，笔者才提出了“Linux 兼容内核”的设想和技术路线。其目的是：将 Linux 的内核加以扩充，使其既支持 Linux 本身的应用和设备驱动，又支持 Windows 的应用和设备驱动，从而成为一个“兼容内核”。要实现这样一个内核，就得在 Linux 的内核中增加这么几个成分：

- 一个符合 Windows 设备驱动程序的特征和要求的框架，即 Windows 设备驱动框架，使得可以把多个 Windows 设备驱动模块装入内核，并使这些模块间的关系和运行条件跟它们在 Windows 内核中时相同。
- 一组由 Windows 内核导出(Export)函数界面(见 Windows DDK)定义的导出函数。对

于设备驱动程序而言，这些函数就相当于由内核提供的库函数。

- Windows 的系统调用界面。微软并没有公开它的系统调用界面，但是在“Windows NT/2000 Native API Reference”和其他资料中已经揭开了这个秘密。在 Linux 内核中实现 Windows 的系统调用界面，就相当于用汇编语言来实现另一种高级语言。这是因为，在内核里面，可以使用的“砖块”就不再是宏观的 Linux 系统调用，而是 Linux 的许多微观的内核函数了。

一旦有了这么一个兼容内核，从原理上说只要把 Windows 的那套 DLL(动态连接库)和有关的服务进程软件搬过来，就可以直接运行所有的 Windows 应用软件了。但是我们当然不能那样干，因为这牵涉到微软的知识产权(第三方提供的 DLL 当然可以)。所以，有了兼容内核以后，Wine 是仍旧需要的，只是再不需要把 Windows 系统调用翻译/转换成 Linux 系统调用了，因而将是一个优化了的 Wine。这样一个 Wine，实际上就是 Windows 的那些“系统 DLL”(以及有关服务进程和一些工具软件)的开源版。

这样，兼容内核的 Windows 系统调用界面，加上优化了的 Wine，这二者的结合在用户空间为 Windows 应用软件营造出一个运行环境；而设备驱动界面、即 DDK 所定义的内核导出函数界面，以及设备驱动框架，这二者的结合在系统空间、即内核中为 Windows 设备驱动程序也营造出一个运行环境。就是说，“哑铃”的两头都照顾到了。

于是，Windows 应用所“看到”的就是一个 Windows 环境，无论是在用户空间还是系统空间都一样，而且这样的环境是由“汇编语言”构筑的，性能上应该与 Windows 本身所提供的不相上下。这意味着将来 Windows 软件一般都可以在兼容内核上运行，而且运行时的效果和性能都可以跟在 Windows 上运行时相仿。既然如此，对于用户来说，改用 Linux 操作系统就是很现实可行的事了。