Foreword

Over the years, there have been many efforts to try to apply standards to the UNIX system and systems sharing its traits. Some of these standards attempted to standardize source-level programming interfaces, such as the Portable Operating System Interface (POSIX) and the Single UNIX Specification (SUS). Others attempted to develop a standard reference implementation trying to unify the operating system utilized by multiple companies—one such example was the Open Software Foundation's OSF/1 operating system.

Unfortunately, past attempts have not been particularly successful in avoiding fragmentation of the marketplace for UNIX systems and providing a single healthy ecosystem upon which software vendors could rely. So why should we believe that Linux Standard Base can succeed where past efforts have not, at least with respect to the needs of independent software vendors (ISVs)?

In order to answer this question, it's necessary to review the history of past standardization efforts for the UNIX system. While they were certainly not all failures, they did not result in the nirvana that ISVs have long been seeking.

Those standards which focused on source-level compatibility, such as POSIX, were critical in the development of a vibrant Open Source community, both before and after that term was coined. Indeed, there is no doubt that Linux owes much of its success to the existence of the POSIX standards. Unfortunately, source-level standards do not provide compatibility at the binary

level, and the requirement to support multiple binary images significantly increases the software vendor's support costs. One of the reasons why Java was so enthusiastically adopted by so many people was its promise to allow programmers to *Write (Compile) Once, Run Anywhere*.

One obvious solution which would apparently allow ISVs to ship a single binary image that would work everywhere is to dictate or otherwise standardize on the same operating system or runtime environment. The OSF/1 operating system was one such attempt, but it was a commercial failure. It was adopted by only two companies, neither of which exists today: Kendall Square Research and Digital Equipment Corporation.

At first glance, it would seem that arranging to have multiple competing OS vendors use exactly the same set of software to comprise their runtime environments would solve the compatibility problems that the ISV community has been struggling with. And technically, this would be true. However, there are business, social, and political issues that must be solved.

For example, many decisions that must be made when choosing which features or patches to include in a release are not simple technical questions but involve complicated tradeoffs between support costs, risks of potentially introducing bugs to an otherwise stable platform, versus the benefits of potential increase in sales in particular markets. How these decisions are made by a hardware vendor or a Linux distribution company may be very different depending on their support structure, willingness to take risks, and market focus. So, if one company has 70% of the market, should it have 70% of the decision making power? If not, why would it be willing to surrender the control of its destiny to a dozen of its competitors that together share 30% of the market?

Another problem with mandating a specific implementation is that it is extremely difficult to upgrade the OS software to newer versions without risking breaking application portability. Even security fixes may introduce interoperability problems.

A position midway between a source-level interface standard, such as POSIX, and mandating a specific runtime implementation, such as OSF/1, is to specify an Application Binary Interface (ABI). The Linux Standard Base falls into this camp. The advantage of this approach is that it specifies the

minimum necessary to assure true application portability—namely, the binary interfaces. The provider of an LSB Runtime Environment may choose any implementation, so long as it provides the necessary binary interfaces.

In other words, the LSB obeys the old adage, "Do the simplest thing possible, but no simpler." The LSB standardizes what is necessary for binary application compatibility, but does not overconstrain the runtime environment. Indeed, it would be possible for a non-Linux system (such as Solaris or Net/Free/Open BSD) to provide a certified LSB Runtime Environment.

Will the Linux Standard Base succeed? Time will tell, but I believe the current work and momentum bodes well. Recently, the 2.0 version of the Linux Standard Base was released. This version includes full POSIX threads and C++ support. These are the last remaining pieces that will allow software vendors to cost-effectively develop applications that can be used on any LSB Certified Runtime Environment.

When I first started working on the Linux Standard Base, my dream was that someday, ISVs would make software such as personal accounting programs or tax preparation software available under Linux. I believe we are almost at that point. After all, Linux desktop numbers have been growing rapidly, to the point where (depending on which study you believe) Linux is either overtaking or about to overtake another platform for which ISVs have considered it profitable to create such products. And with the Linux Standard Base defining a single platform that ISVs can target, I believe that we will finally start seeing such products—and the LSB will be a large contributor towards this success.

Theodore Ts'o
September, 2004