NICK STOUGHTON

update on standards

Nick is the USENIX Standards Liaison and represents the Association in the POSIX, ISO C, and LSB working groups. He is the ISO organizational representative to the Austin Group, a member of INCITS committees J11 and CT22, and the Specification Authority subgroup leader for the LSB.

- *nick@usenix.org*

At the end of last year, the USENIX Board of Directors asked me to prepare a report for *;login:* on the activities in the world of formal standards in 2004.

And a very busy year it's been.

For two years or so, since the Austin Group revised the POSIX standard completely, the majority of activities have been centered on maintenance. Maintenance is a good thing; the fact that since 2001 we have addressed hundreds of problem reports against the 3700+ pages of POSIX, as well as publishing two Technical Corrigenda, shows that the standard is being used. People are reading it carefully and asking hard questions of the form "Did you really mean to say this?" Sometimes the answer is "yes," sometimes it's "no," but every question is carefully reviewed , and if necessary new text is written to clarify the meaning.

Having the standard freely available online (see http://www.unix.org/single_unix_specification/) helps enormously in spreading the use of POSIX: no system implementer or application developer can excuse nonconformance with "I couldn't afford the standard." Additionally, the relative ease of licensing the text means that open source implementations have started to adopt the actual standard text for their man pages; both FreeBSD and the Linux Man Page project have licensed it for this purpose. Increasingly, open source solutions set POSIX conformance as a deliberate objective.

POSIX is about to embark on its next major revision. This year will see the development of material suitable for inclusion in POSIX as Open Group specifications, IEEE specifications, or ISO Technical Reports. Then, during 2006, this new material will be worked into the core document and published, probably in 2007. Work here includes fixing some of the more controversial items from the maintenance process, adding new interfaces widely used in GNU utilities but missing from POSIX, and handling convergence with other ISO standards (see the LSB discussion below).

The Linux Standard Base

POSIX is all well and good for people working in software at the source level and looking for portability. However, there is still a substantial market for closed source applications. Although Linux has undergone a meteoric rise in popularity over recent years, the lack of proprietary software is still seen as a stumbling block for enterprises that equate "proprietary" with "commercial." The Free Standards Group (FSG) has been developing a very different sort of standard to help address this problem: the Linux Standard Base (LSB). The LSB is an Application Binary Interface (ABI) standard, whereas POSIX is an Application Programming Interface (API) standard.

The LSB ABI allows an application developer to build an application in such a way that the binary will run in the same way on any conforming implementation. You, as an application developer, can feel sure that your conforming application will run on any conforming distribution. The ABI deals with issues such as versions of libraries, the value of certain macros, and so on. Much of the detail of what a particular interface does is left to some underlying specification such as POSIX.

Although the LSB is heavily based on POSIX, with most of its interfaces (excluding C++) coming from that source, there are a few notable differences between some implementations of these interfaces. In particular, POSIX may specify one behavior, but glibc or some GNU utility has implemented a subtly different behavior. POSIX is in a position to dictate how a conforming implementation should behave. The LSB, on the other hand, is an ABI standard, describing how something has been implemented rather than how something should be implemented.

A good example is the open system call. POSIX states clearly that open "shall fail if . . . [ENXIO:] The named file is a character special or block special file, and the device associated with this special file does not exist." However, the Linux kernel returns ENODEV in this case, rather than ENXIO. If you are writing an application that cares about the value of errno (i.e., tries to perform some sort of error recovery for certain values of errno), then you'll need to test for both values for this case.

One of the goals of the next POSIX revision is to see whether any of these differences could be resolved by careful wording in POSIX. The LSB is also looking to see how many of these differences it can remove. Clearly, this also requires support of the upstream maintainers, the folks who write and support glibc, the Linux kernel, the various GNU utilities, etc. If code changes are required to achieve conformance to POSIX, how many existing applications will be broken? Developing standards can sometimes be a veritable tightrope walk between competing requirements.

Moving Toward Standardization

At the beginning of 2004, the LSB was almost ready to release version 2.0, a major overhaul of version 1.3. Version 2.0 contains several new libraries, notably, the standard C++ library. Many Independent Software Vendors (ISVs) had asked for C++ support in the LSB; it was one of the most common reasons people said that previous versions of the LSB were not useful.

One of the criteria for inclusion in the LSB is "best practice." You have to remember that standards are not typically at the leading edge of technology development. Good standards document existing practice, finding a common ground for multiple competing similar interfaces. It turns out that the C++ ABI is not yet particularly stable. GCC has had an unfortunate habit of changing the ABI for C++ with almost every release. When the LSB started the work of documenting the C++ ABI, GCC was at version 3.2. Midway through the LSB development, along came version 3.3, with a subtly altered ABI. Almost all the vendors said, "We are going to base our next product on GCC 3.3," so the LSB followed suit and upgraded the ABI specification to be based on GCC 3.3. Guess what: just as the LSB was about to release, GCC came out with 3.4, a new ABI.

To complicate matters further, the ISO (International Organization for Standards) had decided in 2003 that they wanted some standard "in the Linux space" to help keep up their relevance. The FSG was engaged in all of the discussions concerning this, and they proposed that the LSB was an appropriate answer.

Having the LSB become an International Standard like POSIX and ISO-C would help cement the position of Linux as an acceptable enterprise-level operating system. The ISO's Publicly Available Specification (PAS) process offers a way to transform standards developed by other groups into full-scale International Standards: first, the submitting organization is vetted and voted on by the members (countries), and then the document is submitted and goes through a six-month ballot to become an International Standard.

The FSG was accepted as a PAS submitter on October 31, 2003. They then had twelve months to submit a document. The plan was to submit 2.0 when it was published, which at that time was expected to be around February 2004.

Two factors then kicked in simultaneously: first, the C++ ABI instability issue, with distribution vendors arguing as to which version represented "best practice." The word "best" was originally supposed to mean "most widely used." In other words, if nine distributions were using GCC 3.3 and one was using GCC 3.4, then 3.3 was the "best." Of course, we all know there's another meaning to "best"—the motivation for the changes in the ABI from 3.3 to 3.4 was that 3.3 was buggy! The other thing that acted to delay an early release of LSB 2.0 was evaluating it in terms of "This is going to ISO: is it as good as it deserves to be?"

The answer to that last question landed directly on my own shoulders. I had to say to the LSB workgroup, "You know, this isn't a truly wonderful piece of work when you look at it in the light of an ISO standard." Thus started six months or more of hard slog to review and fix some of the inconsistencies, ambiguities, and other potential problems so that the FSG could submit it before 10/31/2004.

We made it on both fronts. Version 2.0 was published at the end of August, with the C++ material still based on GCC 3.3, but separated into a distinct module. That module was excluded from the material submitted to ISO but remained a required part of the FSG's certification program for LSB 2.0. To date, there have been 11 different products (distributions) certified against 2.0.

Since the C++ module was based on an outdated, buggy version of the GCC ABI, a roadmap was made for upgrading to GCC 3.4's ABI during the first quarter of 2005. It looks as though this version of the LSB (which will be called LSB 3.0, since the ABI itself has changed), will indeed have been published by the time you're reading this.

Meanwhile, the six-month ISO ballot is in progress. It started November 10, 2004, and finishes May 10, 2005. Votes here are by country. The U.S. has a vote, as do 21 other member countries (see http://www.jtc1.org under "Membership"). The procedure is somewhat complex: each member country is responsible for putting together its national body recommendation, but there are no overall rules as to how this recommendation is made. The U.S. does it one way, the U.K. another, and Australia yet another. On May 10, each national body sends its vote to the ISO secretariat in Zurich.

A country can vote yes, yes with comments, no (which must include substantive reasons), or abstain. The project editor has to address any no vote's comments, and doing so may change that no to a yes.

After the ballot has closed and the project editor has called and held a ballot resolution meeting to produce a "disposition of comments" report, the final votes are counted. If more than two-thirds of the votes are yes and fewer than one-quarter no, the document becomes a numbered International Standard. POSIX is ISO/IEC 9945, C is ISO/IEC 9899, and the LSB is destined to become ISO/IEC 23360.

As the LSB gathers momentum, we are starting to see public attacks on it from some organizations whose revenues could be directly affected by the widespread adoption of Linux at the enterprise level. This is proof positive that we are doing The Right Thing (TM).