NICK STOUGHTON

update on standards: undue influence?

USENIX Standards Liaison

nick@usenix.org

REGULAR READERS OF THIS COLUMN probably know by now that I am a participant in a number of committees that run under the auspices of the ISO/IEC Joint Technical Committee on Information Technology, in the programming languages and their run-time environments area, known by insiders by the catchy title "ISO/IEC JTC 1/SC 22." I sit in the Working Groups for POSIX, the Linux Standard Base, the C programming language, and the C++ programming language, as well as the special working group on language vulnerabilities and the top-level committee for administering all of these sub working groups.

Several years ago, that top-level committee gave me the unenviable task of handling all of the coordination and liaison required between POSIX and the programming languages that reference it. For example, there's a big overlap between parts of the C standard and POSIX. My job is to make sure that both sides know what the other is thinking when considering a change to something in that shared space.

Now, with C it's easy. We all know where the overlap is and how to handle the questions. But C++ is a different kettle of fish. POSIX isn't written in terms of C++. Although POSIX and C both specify the `fopen()` call, the equivalent isn't so obvious in C++.

But C++ is, as I've reported before, going through a revision. There are *lots* of new features going into the language. And for some of those features it is easier to draw parallels with POSIX (for example, multithreading). POSIX has long had a powerful set of thread APIs. C++ is adding some. Can we at least align the two so that C++ threads can be built on top of POSIX threads?

The POSIX working group saw some serious problems with implementing the proposals for C++ multithreading. The "Nick Stoughton" robot was wound up and pointed at the C++ committee with a message to make sure that its members realized there was a problem.

I did what I was asked to do.

The C++ working group was convinced that there was a problem, and its members collectively voted to change their proposal to remove the contentious thread cancellation wording. I thought about standing on an aircraft carrier deck with a "Mission Accomplished" banner flying behind me.

At the same time, the POSIX working group decided that it would be a good idea to study creating a C++ language binding. Just as there is an Ada and a Fortran binding to POSIX, why relegate C++ to using the C interfaces, forsaking the strong type checking and object orientation and all that good stuff? A study group was formed, and a substantial number of people from the C++ working group (C++, *not* POSIX!) joined up with enthusiasm. This was their language, and their platform of choice. What could be better than such a marriage! The group did the necessary work to prove that there was a viable body to produce a standard sometime in the future (probably around 2012). They did the paperwork with the IEEE (one of the three participating bodies in the POSIX world) and formed the 1003.27 working group.

The 1003.27 working group then read through the table of contents of the current working draft of the C++ language revision document, looking for places where a POSIX language binding might touch on something that is already in the C++ draft. If it is already in the draft, and the ink isn't dry on the draft, can we influence the draft to make sure it

- provides the hooks needed for whatever POSIX extensions might be needed in the future?
- doesn't contain anything truly problematic with respect to POSIX?

Remember, the vast majority of these people in 1003.27 (the POSIX-C++ binding) are also members of the C++ working group. The draft of C++ they were reviewing was one that they had helped to write. I was a member of that group, and we came up with a relatively short list of issues.

Naturally, the messenger picked to walk into the C++ group with this list was yours truly.

There are only a few people in leadership roles in C++, and in general they are all deeply committed to doing a good job for the language. However, they do have their own agendas, and by and large, POSIX isn't a big part of that agenda. So being requested by a sizable minority of their working group to change or, worse, remove some of the wonderful new (untested and unimplemented) features of their draft certainly took them by surprise. And they have worked hard at one by one shooting down all of the requests made by 1003.27. They haven't succeeded yet, but when you consider that the simplest request ("Please could you reserve the namespaces ::posix and ::std::posix for our use") generated about 90 emails in a week, you can see how hard they are fighting.

Building a standard is all about achieving consensus. But what is consensus? When do we know we have reached it? When everyone is exhausted talking about it, does the last voice win? Is it unanimity? In general, in all of the groups I have worked in, when we realize we have a contentious issue, the best thing to do is to omit the issue from the standard, however painful that might be to some. We have done this in POSIX. We have done it in C. We have done it in the LSB. So why is C++ so different that if one loud voice says, "I want this feature in my language," we have to have it, even if 45% or 50% of the working group don't feel so strongly and another 45% or 50% feel more strongly the other way?

Every formal process I have studied has a means for reversing a previously made decision. Robert's Rules of Order has a substantial section on it. Voting something in at one meeting never stops us voting it out at the next! Within SC 22, this is one of the golden rules: "This is the decision we've made until we decide to change it."

Saying, in C++, "we voted at the last meeting to add the `system_error` object, so we can't remove it now" doesn't fit that model.

When the message to change comes from an officer of your parent committee, you cannot simply ignore it. And if that officer happens to have a sizable contingent of your own working group agreeing with the message, you cannot ignore it. Complaining that POSIX is having an undue influence on the purity of the language is specious and pusillanimous. Certainly it is true that C++ runs on platforms other than POSIX. But POSIX is the only international standard platform on which the international standard language is going to be implemented.

And, in my role as POSIX liaison, I'm going to continue to rattle the bars at the C++ meetings. They can try to silence me, but they can never succeed! Pray for fewer "features"!

USER FRIENDLY by Illiad
ZIKPT, TIME FOR AN EXPERIMENT! FIRE BOTH OUR SUCCESS RAY AND THE DYSLEXIA RAY AT A RANDOM EARTHLING AT THE SAME TIME.
BY YOUR COMMAND, YOUR PUTRITUDE.
COPYRIGHT©2003 ILLIAD HTTP://WWW.USERFRIENDLY.ORG/
KA-ZZOWWNT!
ANY RESULTS?
YOU COULD SAY THAT. THE TARGET JUST GOT APPROVED FOR A PATENT.
WOOHOO!