

Giving C++ a Boost

Contributed by Andreas Haberstroh

Saturday, 25 August 2007

Last Updated Wednesday, 27 February 2008

C++ Extensions, Platform Independence and a build environment, what more could I ask for?

I'm a software developer, through and through. I enjoy the process of developing software, and I keep my software skills in shape by developing recreationally and professionally. Much of my professional development has been platform specific, but not necessarily contained to one platform; Windows, Netware, Unix, Linux and sometimes firmware. In some cases, I've developed cross-compatible code that worked on a wide variety of platforms.

As any developer will agree, cross-platform development is the biggest pain in the compiler that you can come across. The first place that pain starts is with the make process. Your platform may have make, nmake and gmake. The make files are not necessarily cross platform. Compiler support is just as fun. Some compilers don't support certain language requirements. One compiler supports one thing, and another supports something else. Then, there is word size, and endian'ness. The list goes on and on.

My recreational development usually revolves around network protocol related software; DHCP, DNS, LDAP. So, my recreational development requirements are pretty straightforward: a platform independent build environment and C++ library. Gosh, that isn't too much to ask for, is it?

There are a number of platform independent build tools that support a common make file. The one I finally settled on settled on was Perforce's Jam. The application had a healthy mailing list activity and seemed to be supported by third party developers. The make file (actually, called a jamfile) format was similar to traditional make, where there are rules, targets, variables, etc. Yes, there were quirks in the format, however I could live with those for the trade offs I received by using Jam.

Then, I started to write my own abstraction classes. Unfortunately, I didn't have a robust build environment to test out my code. I have Windows, Linux and FreeBSD at my office, all of which are Intel based processors. I spent a lot of time working out the kinks and bugs in the platform independent code.

While working on a solution for a unified threading class that would work in Windows and Posix, I discovered a website called Boost.org. Their stated goal was to extend the C++ Standard Library, and I must say, they out did themselves. There were many libraries already published, and more were being submitted and tested. There were many proposed libraries, and one that caught my eye was the Boost.Thread library. Pretty soon, I threw out my platform independent code and started using Boost. An added bonus to this decision to use Boost was, they based their build environment on a modified version of Jam. This was great, since a lot of my target projects were already using Jam to build.

There are many libraries in Boost that I use on a regular basis:

- All the types for cross platform development
- Threads
- Smart pointers
- Pointer containers
- String algorithms

As a matter of fact, Boost supports all the TR1 functionality as presented by open-std.org. If you're unsure of what TR1 is, take a look at <http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2005/n1836.pdf>.

Yes, there are things that are sorely missed in the library as well. Until recently, there wasn't a platform independent method for socket support. In the last year, Asynchronous Socket IO (ASIO) library was released into the wild. And, like all software packages, it is still under development and has what seems to be a stable following of supporters.

So, in conclusion, yes, I am cheerleading for the Boost project. Enough said, I think.