This is an announcement that the Portable Tools Library (previously Portable Windows Library) had it's first code cut between Christmas and New Years 1992.

That is 20 years ago, and it is still going strong!

There were a few major events in computing during the 1991-1992 period that inspired Craig Southeren and myself to start a project that subsequently so dominated our professional careers. Not necessarily in order of importance:

- Windows 3.1
- Microsoft C++ compiler version 7.0
- Linux
- GNU C++ compiler version 2.x
- XFree86
- MacOS System 7
- Codewarrior C++ compiler (actually late 1993, but ...)

Seeing all these pieces and wanting to "write cool applications" we also did not want to limit ourselves to one of the three camps. In those days the concept of "portability" was "which version of Unix are you using?", and the rise of autoconf. The Windows people really did not want anything to do with the Unix people, and vice versa. And the Mac people looked down on both the others. Attitudes that still exist to this day, I might add.

So, after many a Wild Turkey fuelled discussion, we decided we needed a portable library to abstract the different operating systems so we could "write once, port to all".

I had been working with Object Oriented systems for a couple of years at that time, and was thoroughly sold on the methodology. The "pure" OO languages such as Smalltalk or Actor didn't really do it for me, even though, like Java today, their big selling point was portability. Both Craig an myself had spent a lot of our careers up to that time rather close to the solder, so these slow, bloated, garbage collected, high level systems didn't appeal. Memory and CPU was not as abundant as it is today!

As long time C code cutters, the idea of a language that was C "with bits on" was very attractive to us. Though over the years I personally have had a love/hate relationship with the C++ language. Many incredible things have been done, but sometimes the arcane syntax, especially for template, and the inexplicable error messages from the compiler have made me long for alternatives! They came, but too late, by that time we had a huge investment in written code.

So, the basic concepts for the library were formed: object oriented, natively compiled, truly portable, with both low level OS services and high level GUI abstractions.

The first things written were the "container" classes. You can't do very much in programming

until you have arrays lists, trees, dictionaries etc. Bear in mind this was before STL was common, let alone portable! There was an early design decision there that turned out to be ... wrong ... though in concept it is similar to STL. That was, every container, be it a list, sorted list, tree, whatever, should be able to be iterated in the same way. The mistake was I chose an integer to do it with, rather than an iterator class. This produces some performance and a ***lot*** of multi-threading issues, and was eventually abandoned. We have iterators like STL now.

Then came abstractions for files, directories, sockets, threads, sound, video etc. On top of these classes to handle standard protocols like POP3, HTTP etc were written, to support some applications we were doing at the time. And these new classes were able to be written once, identically, for all supported platforms. The concept worked.

Abstractions for windows, menus, dialogs etc were also done. Microsoft Windows led the way, but Unix was a mess. We had partial implementations for straight X, then rewritten to Motif, and I think another widget set that I can't remember. Macintosh had some work done but it was always the poor cousin due to lack of hardware and resources. There are some tiny vestiges of that in functions like PString::ToPascal().

As the millennium headed to it's conclusion, the applications we were writing tended to be "server" based and not "user oriented". The GUI code continued to lag while many more classes were added to do protocol stuff. Including the first cut at doing this new thing "Voice over IP" using this protocol H.323. Then, how did *that* take over our lives! I'll leave that story for it's 20th anniversary.

It took a few more years, but eventually calling the system the Portable *Windows* library was becoming silly. No one, including us, used the widows abstractions. Things like Qt and wxWidgets were available they were concentrating on the GUI side while we were doing OS/protocol stuff. So it was renamed to the Portable Tools Library. You still see references to "pwlib" in various places.

Here we are, couple of decades later, wondering if OPAL should be switched to boost or something, and rejecting the idea because, fundamentaly, PTLib really does a lot of things well, at least in my mind, But then I *am* biased! :-)

Finally, and I am sure I am speaking for Craig too, I would like to thank all of the contributors to PWLib, PTLib, OpenH323 and OPAL over the years. It has been a long road, and simply would not have been possible without your support and tolerance.

Robert Jongbloed