I know programmers who started using C++ solely to get the ability to use “//” as a comment delimiter. C++ definitely owes a lot of its popularity to the fact that any legal ANSI-C program was also a C++ program that pretty much did the same thing. – Ed.

I find C++ interesting. No, not because my compiler can spit out an incoherent set of errors if I fail to include all of the right headers to appease the angry STL gods. And, no, not because its population of practitioners has reached a steady state and is now beginning a slow decline.

I find it interesting because there’s a lesson to be learned about how it conquered an existing community. The tactic it took was deceptively simple yet it’s one that technologists, especially the “system architects,” rarely learn.

To understand what happened, we need to fire up the way-back machine. Before P2P, before spam, before the Web, before the Internet was even close to being mainstream, we need to go back to a time when the Macintosh was still running on those old-school 68000 Motorola chips.

C++ was born in a world that was clearly on its way to being dominated by C. In the late 1980s, C had become the language of choice for

many computer science graduates. It was just respectable enough to be taught at the collegiate level, and fast enough to be usable for that degrading domain of problems known as “the real world.”

The only real competition that C had faced was from such powerful threats as Pascal, Basic, FORTRAN, and Cobol. Pascal briefly flirted with fame but flamed out. Basic won its market share, but could never shake the stink of its backwater roots, undeserved as it may be.

With that, we were left with the only two real contenders. FORTRAN was for the slide ruler crowd and Cobol was, well, it was Cobol. C found the then-perfect balance between respectable programming language and reasonable business tool. From there, it took over the development scene.

Now, of course I’m aware that there are many other languages out there. There was Ada, but it had all the sex appeal of an 800-page requirements document from the U.S. Department of Defense. The rest of the plausible contenders—Modula, CLU, Smalltalk, Prolog—couldn’t find their tipping point because they overlooked the needs of their core audience: the undergrad student. The language couldn’t have fit into their dorm-room PCs let alone their brains.

Across the industry, nothing was as entrenched as FORTRAN, Cobol, and Basic over such a large swath of development arenas.

In any given system that used these classic programming languages, one could achieve a semi-plausible détente between most of them. Interoperability between any of them was never perfect, but it was certainly doable. Depending on your operating system, FORTRAN could call into Cobol or even Basic into FORTRAN. Link-level compatibility was possible.

The power of this détente did more than mitigate the debates among the communities. It meant that the high-priced business consultants writing in Cobol had half a chance at repurposing a statistical package written in FORTRAN. Never pretty, but most real-world integration efforts rarely are. If anything, the groups got to leverage each other through the level playing field that was the linker in most environments.

Let’s not forget that the Mac had an early academic love affair with the Pascal community that resulted in its Pascal calling convention and predilection toward Pascal strings. Fortunately, the Mac was cured of that silliness. Despite the awkward nature, C and FORTRAN still inter-operated with the Mac.