

[TOC](#) [Previous](#) [Next](#)

The C API

[C Compilers](#) [Data Structures](#) [Functions](#) [Example](#)

We can write run-anywhere MySQL-enabled web applications in PHP or Java. We can write small MySQL-enabled, run-anywhere standalones in Perl. We can write standalone MySQL-enabled Windows applications in Microsoft Access, and standalone Windows or Linux MySQL-enabled applications in a variety of .NET languages including Visual C#, Visual Basic and Visual C++, bypassing the C API if we wish.

With some of these options, the compiled code itself will run more slowly than straight C compiled to native machine code, but that difference is usually tiny compared with the time an application spends communicating with its database.

And after all, C could have been the original inspiration for the ironic slogan "hard to write, hard to read." Development time is often a order of magnitude longer than with other languages for which there is a MySQL API. Not for nothing has C been characterised as high-level assembly language. Then, there's generalisability. Have you tried to port a C program to another C compiler lately?

Do we still need to know how to write to the MySQL C API? Yes. There is no other way to build your own customisations into MySQL. Many MySQL-enabled C applications are still running, and need to be maintained. Some computation-intensive modules and programs need to be compiled to optimised native code for specific classes of machine. The C API is the foundation for other MySQL APIs, so understanding the C API deepens our understanding of the others. And C++ is a viable development platform which is here to stay.

To read the rest of this and other chapters, [buy a copy of the book](#)

[TOC](#) [Previous](#) [Next](#)
