A graphical depiction of the steps in building a C++ executable, enhanced for classic Win32

devblogs.microsoft.com/oldnewthing/20240530-00

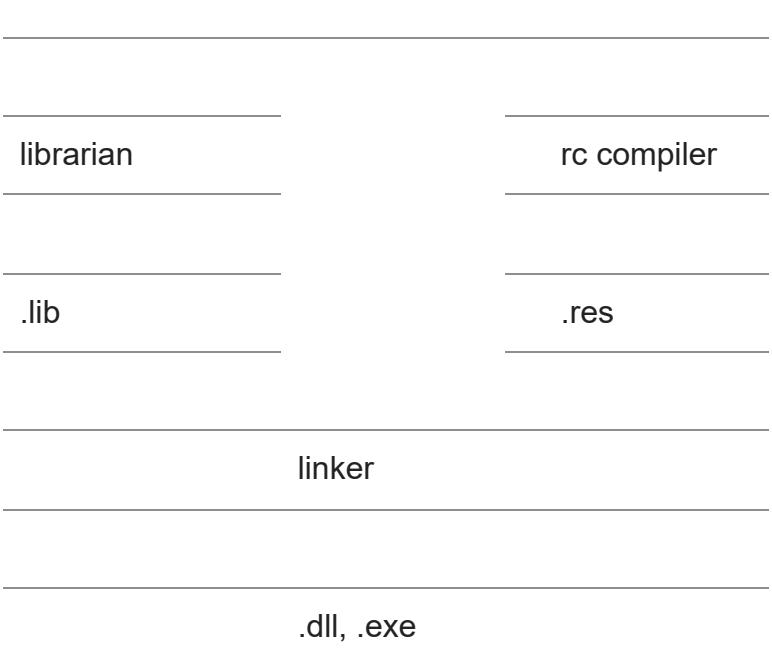
May 30, 2024

Raymond Chen

Last time, we generated a diagram showing the basics of how a C++ executable is built. The diagram applies to Windows programs too, but it is often the case that some of the parts that go into the C++ build are themselves build outputs of other tools. And there are usually other parts of a Windows program that aren't covered by the C++ build workflow because they produce things beyond just a C++ program.

Let's add some of them into our diagram, at least the ones most often seen in classic Win32 apps.

.idl		
MIDL compiler		
.h, .cpp		
	C++ compiler	
	.obj	.rc, .ico, .bmp



The original diagram is still there in the bottom left corner, where it consumes C++ source files (.h, .cpp) and eventually produces a module (.dll, .exe). The added steps are things that either produce C++ source files from other sources or add additional content to the resulting module.

If there is an .idl file, it is processed by the MIDL compiler, which produces .h and .c files, which you consume in your project code.

You may have other code generators as part of your project. If you have a yacc grammar or a Lottie animation, you'll also have a compiler compiler or an animation generator as a code generation step, to take the raw materials and turn them into code and resources that are consumed as inputs by other boxes.

Once we have generated all the C++ source files, we can go through the process we saw last time: Compile the C++ source files to object files, and possibly generate a .lib.

Meanwhile, for classic Win32 resources, the resource compiler takes the .rc file and any supporting files (like .ico and .bmp) and produces a .res file.

After all the .obj, .lib, and .res files have been produced, we feed them all to the linker, which produces the resulting module, a .dll or .exe.

The essence of the process hasn't changed. There's just more rigmarole before the compilation (to produce header files and code), alongside the compilation (to produce a resource file), and after the compilation (to incorporate the resources into the final binary).

Next time, we'll add more rigamarole around the central diagram by adding packaging and XAML.