Retro Programming for the Modern(ish) Software Developer

by Mark Dalrymple. markd@borkware.com

You've used IDEs, programmed in typesafe languages with highly featured rich standard libraries, and huge amount of complexity.

You've seen (or grew up with) the older computer systems that do way less, but look like they're a huge amount of fun. Nothing gets in the way between you and being able to exploit all the hardware, and being able to push the hardware far beyond what the original creators envisioned.

I grew up with the Apple][, Apple //e, and early Mac. I learned to program from code listings from magazines (and debugging the errors I typed in), and had a huge amount of fun, enough to make me want to program for a living (having to choose between music and programming as I was leaving high school).

I want to share some of the fun, pain, glory, and enjoyment that comes from programming these older machines.

There's a bunch of different ways to program these older platforms.

You can go for the original experience, by finding or purchasing a retro system, such as a fully functioning Apple][, and program on that.

You can go for the original experience with conveniences, by purchasing a retro system, and add-on hardware that lets you use an SD card as floopy or a hard drive. That way you don't need to source blank diskettes, and things generally get a lot more convenient by being able to download and use disk images.

You can also go for the basic emulator route, which is free (or very low price depending on the emulator package you're using). There are also super emulators (my term) for programs that will give you all sorts of telemetry about the machine as it runs (virtually) - examine RAM while running, take snapshots of memory, see visualizations of anything useful to a developer.

I'm aiming for folks going the original-hardware or simple emulator route, to give the flavor of what it was like back in the day to program these beasties (especially as a hobbyist / amateur programming), along with the pain points involved. (debugger? what's that? compiling, what's that? type safety? what's that?) If you decide to write serious software for these platforms (whether to sell, or give away for bragging rights), you'll probably want more powerful tools to help you out.

I'm planning on targeting the Apple][platform (since that's what I grew up on, from sixth grade through 10th grade until we got a 512K Mac for the household), and the TRS-80 Model 100 (a recent acquisition, but it's a delightful, if slow, portable computer)