Subject: Re: Interesting Rant
Posted by Jeff Hester on Sat, 18 Nov 2006 21:42:44 GMT
View Forum Message <> Reply to Message

There is an interesting generational aspect to this discussion. I grew up during a fairly narrow window of time when data analysis (and science more generally) was becoming more and more computerized, but before specialized software had become available. During that time one *had* to just write the code yourself, whether it was data I/O, array manipulations, numerical code, or even primitives that allowed you to talk to an image display device.

In that environment you had to be cognizant about everything from how information was stored at the bit level to the way memory was utilized to things like the efficiency differences between stepping through an array with *array++ as opposed to array[i], or the tradeoffs between efficiency and flexibility in data storage formats. (I can't count the number of times as a graduate student or postdoc that I was handed a tape and told, "Get the data off of it," without so much as a suggestion as to what the format might be.) You thought in terms of, "What would I like to do to/with my data, and how can I do it?" And when you found that a door was shut, you didn't think twice before starting to look around for the nearest window. Data processing and analysis meant getting your hands dirty. Period. Those of you who came of age in such an environment know what I mean.

As new tools came along, we were happy to learn to use them. It was kind of a relief to not have to write *everything* yourself, and there is no way that one person can keep up with a whole world full of programmers. But at the same time, we carry that early mindset with us. When we use a tool we can't help but build a mental picture of what that tool must be doing internally. We've got a pretty good guess about what is there inside the black box, and how it might go wrong.

It came as a shock to me when I realized the extent to which this mindset has vanished. Don't get me wrong. There are a lot of very good programmers coming up through the ranks. But the majority of people who sit down in front of a data set these day begin with the the question, "What convenient tools has someone written for me, and which might be best to apply to these data?" In other words, their thinking about problems tends to be heavily shaped by the parameters of their software environments. A lot of those tools are exceedingly powerful and allow you to do great stuff, often very efficiently. But a cage is still a cage, no matter how gilded.

Which arrives at the issue at hand. While I have developed a number of "packages" in IDL, for the most part every data set that comes across my desk is unique. Probably 75% of the code that I write is for the

immediate purpose, and may not see use again after that day, much less after the end of that project. I include image and other data processing, numerical modeling, and data visualization in that category. I tend to cut, paste, hack, modify, stick together with duct tape, and in general do whatever I need to in an effort to tease information from a data set. Some of the code that I write is admittedly kind of ugly... but it shows me a lot of really interesting things in my data.

This is the kind of application that IDL was built for ("Interactive Data Language..."), and IDL remains at least for me a more productive environment than any other I have run across. Is it the best programming language out there? No. Does it produce the prettiest plots with the spiffiest fonts? No. Do I use other tools? Sure. But when it comes time to get down and dirty with your data, IDL's combination of power and on-the-fly flexibility is hard to beat.

So when students complain, "I can't do that in Maple/Matlab/Whatever", I chuckle, then point out that quite often those who are limited only by their imagination and creativity tend to win out over those who are limited by the flexibility of their software. I then point them to an IDL tutorial.

BUT... as for personal pet peeves, it is the inefficiency of loops, hands down. (I will now resist the temptation of telling stories about compiling C code and linking it into Forth kernels to do real-time instrument control or doing image processing on a PDP 11/55....)

Surviving to be a dinosaur sure beats hell out of the alternative... ;-)

Cheers,
Jeff Hester

Professor and Dinosaur in Residence
School of Earth and Space Exploration
Arizona State University
