Subject: What does an optimal scientific programming language/environment need?
Posted by grunes on Fri, 19 Sep 2003 18:29:12 GMT
View Forum Message <> Reply to Message

I'm working on creating an optimal scientific programming language and environment. My hope is that people who use current environments have specific things they love about it, that need to be included. For now I'm trying to combine the best concepts from FORTRAN, BASIC, C, APL, IDL, PV-WAVE, and possibly MATLAB.

I have been an applications programmer in support of scientific research for 23 years, in radar and optical remote sensing, data compression, and statistical image processing, and have also produced operational software for ship-borne, airborne, and orbital platforms, so I have some ideas of my own. But I want other people's input before I begin, because I expect this to be a lot of work.

What I have in mind so far is:

1. Free or cheap software. Moderate price documentation ($20-50?).

2. Deliverable free and non-free executable products without special permission or royalty. Unfortunately, this seems to be the single most important criteria of modern scientific/engineering applications development, from the sponsor's point of view.

3. Very rapid developement and testing. Requires extreme conciseness, support of arrays, complex number, linear algebra, finite element and numerical integration of functions and differential equations, and little need for type and shape declarations. Can easily switch on automatic detection of subscript checking, memory reference checking, argument mismatches, fixed and floating point errors. FORTRAN style adjustable array bounds (e.g., a(-3:4, 5:7)).

4. Very rapid learning. Most of the language and environment must be summarizeable in a very few pages. Design must be consistent. Documentation unambiguous. For those who like lots of words, examples or homework problems, I could supply a larger tutorial manual.

5. 100% upwards compatibility with earlier versions of the same language/environment, and between platforms.

6. Close enough to standard mathematical notation to be mostly debuggable by inspection. But must have ASCII transliteration so people can use there own editors if they don't like mine.

7. Can add the declaration statments that make efficiency possible.

8. A compiled mode that really is as fast as FORTRAN or C, if you add those declarations. Compiler would produce 2nd level code for compilation by g77 and gcc.

9. The interpreter and Executables must be rock solid stable. That means very few heap dynamic memory allocation operations (malloc, calloc, alloc, new, allocate). Support dyanmic stack allocation, and a garbage collected heap area.

10. Screen and file graphics (plots, diagrams, images) must be very easy to produce. Default style products must be publication quality.

11. Can call and be called from FORTRAN and C programs.

12. Must support two primary development platforms at this time: Lintel and Wintel (PCs under Linux and Windows). (Nothing else is economically viable any more.) For the moment, I will ignore multiple CPU support, networking, and signal processing chips.

13. Be able to associate an area of virtual memory with an entire file. Should handle raw bytesream files, as well as TIFF scientific data sets. Since the indicated platforms only support 1-2 GB of user virtual memory space, I will have to create a special FILE type that allows 64 bit address manipulation.

14. Multi-precision calculations (e.g., 2, 4, 8, 16 byte and greater floating point numbers), and exact (rational) calculations must be easy to do.

15. No bugs. (I'll try.)

Please add your own criteria!
