Subject: Re: What does an optimal scientific programming language/environment need?
Posted by donotreply on Fri, 03 Oct 2003 00:08:57 GMT
View Forum Message <> Reply to Message

In article <k6Meb.443$ye2.217564282@newssvr11.news.prodigy.com>,
unixmonster@hotmail.com says...
>
>
> bv wrote:
>
>> grunes wrote:
>>>
>>> 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.
>>
>> Before you embark on what is bound to be a long and winding road you
>> might want to consider a recent quote by "DB" from sci.math.num-analysis
>> ng which would invariably apply to whatever you might come up with.
>>
>> "To get any chance of succeeding new programming languages should
>> from the beginning provide a huge advantage compensating the loss of
>> decades of expertises contained in the already available libraries, in
>> the trained people, as well as in the compiler technology. Now to make
>> the situation worse, the many functional languages compete with each
>> others."
>>
>> --
>> Dr.B.Voh
>> ---
>> Applied Algorithms http://sdynamix.com
>
> I would prefer to see APL extended with operator overloading and with
> defined primitive numeric types - so that one could model things like
> Grassmann algebras, moving frames etc. and maintain the concise syntax.
>
> I see little point in inventing another syntax.
>
> The most useful math machine I have is my TI-92+, because I can take it
> anywhere and it has a "good enough" symbol manipulation capability. I use
> it mainly for doing calculations in 6-d space. The syntax is based on
> "Derive" and I find it quite acceptable.

DERIVE has been my favorite computer algebra and ad-hoc calculation
language for a long time. The fact that it is now sold by Texas Instruments

through their education department belies its power. It's LISP-based
(although the LISP is almost entirely hidden) and, in its current
incarnation, quite programmable. However, there's little that's procedural
about its programming (not unexpected, given its LISP roots); instead, one
writes a number of functions that reference each other.

DERIVE and its ancestor, MuMath, has actually been around for a LONG time
-- IIRC, since the late 1970s. By the standards of most anything found in
the computer world, it's remarkably bug-free. It also allows symbolic
results to be output in Fortran syntax.

Highly recommended.