Subject: Re: REDUCE
Posted by Craig Markwardt on Mon, 02 Apr 2001 20:17:13 GMT
View Forum Message <> Reply to Message

JD Smith <jdsmith@astro.cornell.edu> writes:
> The thing I don't like about REDUCE:
>
> It is horribly ugly.
>
> The reason it is horribly ugly is all those damn types. If you followed
> Ronn Kling's book, you'd know he recommends handling multiple types
> like:
>
> switch(type){
> case IDL_TYP_INT: myvar=(short *) foo; stuff1; stuff2; stuff3; break;
> case IDL_TYP_LONG: myvar=(int *) foo; stuff1; stuff2; stuff3; break;
> case IDL_TYP_FLOAT: myvar=(float *) foo; stuff1; stuff2; stuff3;
> break;
> ...
> ...
> }
>
> That is, just replicate things over and over again for the various
> types. REDUCE works natively in 9 types. Luckily, I didn't have to
> copy everything over nine times as above, but in essence that's what I
> did. I just used a host of clever C pre-processor directives to
> indirect the type replication.

[Sorry if some of this appeared on Friday; somehow I don't think it
got sent.]

Hey JD--

REDUCE looks very cool. I think this is exactly the kind of thing
we'd like in the core of IDL.

As for your question about ugliness, I think it is unavoidable. You
were hoping for some way, at *runtime* to do some polymorphic
operations on the data. Unless you want to get into self-modifying
code, I think it's impossible. To see this, you only have to realize
that there are different machine language opcodes for different data
types such as float, int32, byte, and so on.

Since you want you operation to occur on any data type, at some
fundamental level all of these opcodes have to be exercised, thus it
has to occur at compile time, not runtime. You are doing it now with
macros. You could presumably do it using templates in C++. Them's
the breaks.

To go slightly off the main topic, you did mention the code to handle the casting of data from one type to the other. This is quite difficult to get right using the C interface to the IDL runtime. This is actually a classic case where using the IDL language as a *wrapper* is good. This is really quite straightforward: you do all the type checking, casting, and other checking in an IDL script, which then passes the resulting data to an internal routine. The internal routine is thus freed from performing lots of complicated checking itself and can get down to the business of crunching the numbers. WRITE_GIF is a good example of this, despite the internal routine ENCODE_GIF being disabled because of the GIF licensing malarkey.

While this doesn't answer your fundamental question, it may help to remove some of the complexity from the C code.

Craig

--

---------- ----------
Craig B. Markwardt, Ph.D. EMAIL: craigmnet@cow.physics.wisc.edu
Astrophysics, IDL, Finance, Derivatives | Remove "net" for better response
---------- ----------