Subject: Re: Common or not common
Posted by Liam Gumley on Fri, 14 Nov 1997 08:00:00 GMT
View Forum Message <> Reply to Message

morisset@iagusp.usp.br wrote:

> I read already lot of times David (and others) speaking
> about "never use common" (except to save values used in multiple
> call of the _same_ routine). I cannot understand how I could
> develop my codes without any common. I have a lot of routines
> that have to deal with a lot of variables (of various type).
>
> I can't imagine passing 25 or more parameters to the routines.
> And with the previous version of IDL, the parameters where
> limited in number.
>
> And what happens when I decide to add one other parameter to
> one routine? I have to check all the call to the routine
> to be sure that the new parameter is passed
> (I can't wait for IDL to do it itself ;-) ?
>
> I use parameters when I want to deal with procedure or
> function I want to apply to different variables. But when
> in every routines/functions of the program
> the spectrum, lambdas, corrections of
> different types and a lot of other things,
> have always the same name, it's easier and safer
> to pass them via a common, no?

Two words:
Structures,
Pointers.

Structures can contain anything that you would normally put in a common block. In the simplest case, you maintain one global structure that contains all the information you need (see any of David's example widget programs). You define it once it your main program, or in an included file.

A pointer (or handle if you're still using IDL 4.0) to then passed to any procedures or functions that need items from the structure (the pointer points to the structure). Then whenever you make changes to the structure (e.g. adding a new variable), the changes are automatically available to any routines that use the structure. If you want to change the size or type of an item in the structure, then instead of putting the item itself in the structure, put a *pointer* to the item in the structure.

Trust me, it works. David wouldn't keep saying it if it didn't! Please look at any of David's example widget programs: you will definitely learn

something. I recall reading quite a while ago (maybe it was in one of
David's manuals) that the first step to becoming a proficient in IDL is
learning how to *think* like an IDL programmer, not a FORTRAN programmer.

Cheers,
Liam.
PS: David I'll take that case of beer whenever you're ready.
