Subject: Re: Ellipsis in IDL?
Posted by Jeff Guerber on Thu, 22 Jul 2004 21:46:34 GMT
View Forum Message <> Reply to Message

On Wed, 21 Jul 2004, Michael Wallace wrote:

> How do you define a procedure to take N number of arguments when you
> don't know what N is before the procedure call? For those of you who
> have worked with C, what I'm after is something similar to the ellipsis
> (...) which allows N many arguments to be specified for functions such
> as printf.
>
> In IDL, the print command is obvious example of what I'm trying to do.
> The signature of print is:
>
> print [, Expr1, Expr2, ... , ExprN]
>
> So, how can I write a procedure to take N many arguments?

I don't know how print works internally, but remember that you can have fewer actual arguments (in the call) than dummy arguments (in the procedure definition). It's a bit cumbersome, but one thing you can do is define your procedure with more arguments than you expect to need:

```
pro ManyArgs, arg1, arg2, arg3, arg4, arg5, arg6, arg7, arg8, arg9, arg10, $
   arg11, arg12, arg13, arg14, arg15, arg16, arg17, arg18, arg19, arg20, $
   arg21, arg22, arg23, arg24, arg25, arg26, arg27, arg28, arg29, arg30
```

Then as you process each one, check whether the caller used it with statements like:

```
  if (n_elements(arg23) NE 0) then begin
     ...process arg23...
  endif
```

If you want to loop over all the arguments, one obvious possibility would be to use a case statement inside the loop, to check n_elements of the appropriate argument. (I said it was a bit cumbersome!)

I'll bet even print has some limit on the number of arguments it can take. Hope this helps,

Jeff Guerber