Subject: Re: subverting IDL builtin variables !FORMYOWNPURPOSES
Posted by SonicKenking on Wed, 23 Feb 2011 12:42:18 GMT
View Forum Message <> Reply to Message

> I ended up using common blocks for my logging framework, but system
> variables would work just as well.
>
> See MG_LOG and MGffLogger in the dist_tools for the way I did it:
>
> http://docs.idldev.com/dist_tools/
>
> By the way, you don't have to subvert a pre-existing system variable,
> you can create your own with DEFSYSV.
>
> Mike
> --www.michaelgalloy.com
> Research Mathematician
> Tech-X Corporation

Hi Mike,

The MG_LOG is a really nice tool and inspiring. I did some modifications to it and think it could be useful to others.

One limitation I felt for MG_LOG is that it accepts only a single string message to print. I'd like to have something similar to the IDL built-in PRINT, which accepts any number of arguments and also does the logging as MG_LOG.

This is one tricky thing I always struggled, i.e. you cannot get the full power of using variable length parameters without using "Execute". But Execute is something I try to avoid as much as possible, since it does not run on an IDL VM. Mike suggested before that this can be solved by using DLM or embeded C programs. But I have no idea how to do it. I would appreciate if anyone can come up with a universal wrapper for this and is willing to share with us. :)

Anyway, I ended up re-using PRINTF for the logging and sneak the modified logging program in the position of LUN. So it is something like:

PRINTF, myLun(), a, b, c, etc

Where myLun() is a similar routine to MG_LOG.

They are similar in that they are both a wrapper that manages an underlying logging object. They both control whether the message/ variables are actually printed out based on a logging level (1-5).

The differences are:
1. myLun() returns a lun for either the terminal (-1 or -2), or a
file, or /dev/null for suppressing the message to be printed.
2. myLun() accepts a level parameter indicating the level of this
message, e.g. myLun(5) for debugging level.
3. myLun() accepts a filename for logging in a file and once the file
is set, it is persistent until the logging is explicitly redirected to
another file/terminal.
4. The underlying objects can manage multiple output files. An new
file will be opened if a new filename is passed to myLun().

Some of the limitations are:
1. It cannot keep the output to both terminal and files at the same
without using "Journal", which is another thing I try to avoid.
2. Currently it only controls text printing. But I'd like to have
similar mechanisms but work for plottings.

I'll post the code if anyone is interested.

Yang
