Subject: Suppressing diagnostic messages en masse
Posted by MarioIncandenza on Fri, 03 Feb 2006 00:45:42 GMT
View Forum Message <> Reply to Message

Hey great minds,

I am engaged in semi-permanent debugging of some pretty long scripts,
which in the course of developing I have peppered with PRINT statements
to achieve various diagnostic ends.

I don't think the script is "final," whatever that means, but it's time
to do some production runs, where the script will be invoked 10,000
times, instead of the usual 1.

I can do a search-and-replace to comment out the diagnostics for this
purpose, but what would really be nice is if I could somehow put a
keyword flag into the routine that, when set, would suppress|promote
the diagnostic messages. Sounds like the sort of thing that might
exist. Anyone know how to do this?

Subject: Re: Suppressing diagnostic messages en masse
Posted by Craig Markwardt on Sun, 05 Feb 2006 19:09:04 GMT
View Forum Message <> Reply to Message

"Ed Hyer" <ejhyer@gmail.com> writes:
> Hey great minds,
>
> I am engaged in semi-permanent debugging of some pretty long scripts,
> which in the course of developing I have peppered with PRINT statements
> to achieve various diagnostic ends.
>
> I don't think the script is "final," whatever that means, but it's time
> to do some production runs, where the script will be invoked 10,000
> times, instead of the usual 1.
>
> I can do a search-and-replace to comment out the diagnostics for this
> purpose, but what would really be nice is if I could somehow put a
> keyword flag into the routine that, when set, would suppress|promote
> the diagnostic messages. Sounds like the sort of thing that might
> exist. Anyone know how to do this?

I have had a need for this before. There are a lot of cases where I
need to keep a log of diagnostic output, but not print it unless there
is an error, or to save it to a file at the end.

I developed a small routine called PRINTLOG which is a drop-in
replacement for PRINT. You work it like this,

```
  PRINTLOG, LOG=LOG, "X = ", X
  PRINTLOG, LOG=LOG, "Y = ", Y
```
and the result is that the LOG variable slowly builds up a transcript of whatever you print to it. Unfortunately you will need to retool your subroutines to accept a LOG keyword, but once you do that, you can maintain one continuous log while traversing up and down the subroutine call tree. Like this,
```
  MYPROC1, LOG=LOG, X, Y, Z
    ... within MYPROC1 ...
    PRINTLOG, LOG=LOG, "X = ", X
    MYPROC2, LOG=LOG, L, B
       ... within MYPROC2 ...
       PRINTLOG, LOG=LOG, "L = ", L
       PRINTLOG, LOG=LOG, "B = ", B
```

The final trick is that you can globally turn the console output on or off by setting,
```
  PRINTLOG, DEFAULT_PRINT=0
```
and later re-enable it with
```
  PRINTLOG, DEFAULT_PRINT=1
```

Hope that helps!
Craig

See the following web page for PRINTLOG,
  http://cow.physics.wisc.edu/~craigm/idl/

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