Subject: Error handling by build-in IDL routines
Posted by Frank Holland on Mon, 15 Mar 1999 08:00:00 GMT
View Forum Message <> Reply to Message

Hi!

I have a question about error handling by build-in IDL routines.
Consider the following example:

```
fritz = 'the cat'
print, median(fritz)
```

IDL replies with:

```
% MEDIAN: Expression must be an array in this context: FRITZ.
% Execution halted at:  $MAIN$
```

My question:
How does the function MEDIAN knows the name of the parameter (i.e.
FRITZ) I passed into it? How can I implement this functionality into my
own IDL routines?

Thanks for any suggestions,

Frank

Subject: Re: Error handling by build-in IDL routines
Posted by William Daffer on Sat, 20 Mar 1999 08:00:00 GMT
View Forum Message <> Reply to Message

"R.Bauer" <R.Bauer@fz-juelich.de> writes:

```
>  Vapuser wrote:
>
>>  "R.Bauer" <R.Bauer@fz-juelich.de> writes:
>>
>>>  Try this!
>>>
>>>  PRO t2,test
>>>
>>>     HELP,/recall,output=output
>>>     for_test=(STR_SEP(output[1],','))[1]
>>>     varsize=SIZE(routine_names(for_test,fetch=-1),/type)
>>>     VarValue = Routine_Names(for_test, FETCH=-1 )
>>>
>>>     IF test EQ varvalue THEN IF varsize NE 4 THEN $
>>>      MESSAGE,'Expression must be of type FLOAT:'+for_test,/info
>>>
```

```
>>>  END
>>>
>>>  dd='dummy'
>>>  t2,dd
>>>
>>>  % T2: Expression must be of type FLOAT:dd
>>>
>>>
>>>  R.Bauer
>>
>>    I don't think this work in a procedure.
>>
>>    If anyone out there in RSI land is listening...
>>
>>    It would be nice to have a function like the Perl package Carp.pm,
>>  which reports errors from the line number of the invocation of Carp's
>>  calling routine. So, say you have a perl routine foo which reports
>>  some error by calling carp. The linenumber given in the error message
>>  emitted by Carp is the line at which foo is called, not the line at
>>  which Carp is called. That way, you can write error handling code that
>>  doesn't have to keep track of the stack, and depend on the output from
>>  help.
>
>  help,call=call reports this.
>
>  another example for working in a procedure is:
>  called by t3 :
>
>  IDL> t3
>  % T2: Expression must be of type FLOAT:doof
>
>
>  both are seperate files.
>
>
>  PRO t3
>
>    doof=1
>    t2,doof
>
>  END
>
>  -----------
>
>  PRO t2,test
>
>    IF N_PARAMS() LT 1 THEN BEGIN
>       MESSAGE,'Try: t2,test',/info
>       RETURN
>    ENDIF
>
>    HELP,call=call
>    ; T2 <C:\t2.pro(   8)>
>    ;  T3 <C:\t3.pro(   4)>
>    ;  $MAIN$
>
>    help_of_interest=within_brackets(call[1],brackets=['<','('])
>    IF help_of_interest EQ '' THEN BEGIN
>
>      HELP,/recall,output=output
>      for_test=(STR_SEP(output[1],','))[1]
>
>      varsize=SIZE(routine_names(for_test,fetch=-1),/type)
>      VarValue = Routine_Names(for_test, FETCH=-1 )
>
>      IF test EQ varvalue THEN IF varsize NE 4 THEN $
>      MESSAGE,'Expression must be of type FLOAT:'+for_test,/info
>
>      ENDIF ELSE BEGIN
>      txt=get_file(help_of_interest)
>      line=within_brackets(call[1],brackets=['(',')'])
>     ;4
>      cmd=txt[line-1]
>      for_test=(STR_SEP(cmd,','))[1]
>      varsize=SIZE(routine_names(for_test,fetch=-1),/type)
>      VarValue = Routine_Names(for_test, FETCH=-1 )
>
>      IF test EQ varvalue THEN IF varsize NE 4 THEN $
>      MESSAGE,'Expression must be of type FLOAT:'+for_test,/info
>    ENDELSE
>
>  END
>
>
>
```

Thanks! This looks pretty useful!

whd

--
My mail address has been mangled by my mailer. Send replies to...
daffer@primenet.com

--
Outside of a dog, a book is man's best friend

Inside of a dog, it's too dark to read.
Groucho Marx.