Craig Markwardt wrote:

> "R.Bauer" <R.Bauer@fz-juelich.de> writes:
>> Craig Markwardt wrote:
>>
>>> One thing that ROUTINE_NAMES() (**note) cannot do is *add* variables
>>> to another level. If the variable exists, then you can muck as much
>>> as you want with it, but if it doesn't exist, sorry.
>>
>> You can add variables to another level.
>> Try this!
>>
>>
>>
>> PRO DEEPSTOP, level
>>   IF level EQ 1 THEN BEGIN
>>   level = ROUTINE_NAMES(/LEVEL)
>>   varName = 'A'
>>   void = ROUTINE_NAMES(varName, STORE=(level+1), 8)
>>   ENDIF
>>   IF level EQ 2 THEN begin
>>   print,a
>>   STOP
>>   end
>>   deepstop, level + 1
>> END
>
> Ahhh, but I argue that your procedure works for the reasons I said
> before. The variable A already existed in the procedure because you
> used it in a statement ("print, a"). So you really were not *adding*
> the variable to the procedure.
>
> Try this one:
>
> PRO DEEPSTOP2, level
>   IF level EQ 1 THEN BEGIN
> ;   a = 0
>   level = ROUTINE_NAMES(/LEVEL)
>   varName = 'A'
>   void = ROUTINE_NAMES(varName, STORE=(level+1), 8)
>   help
>   ENDIF
>   IF level EQ 2 THEN begin
>   help

>   STOP
>   end
>   deepstop, level + 1
> END
>
> This procedure does not mention "A" explicitly anywhere, and I can't
> get beyond the first IF clause.
>
> IDL> deepstop2, 1
> % ROUTINE_NAMES: Variable is undefined: A.
> % Execution halted at: DEEPSTOP2   5 /dev/tty
> % $MAIN$
>
> But, if you uncomment the "a=0" line above, then you can get further.
> What I find is that the value of A is set at *both* levels!
>
> I am using an older version of IDL, 5.2. This tells me that the
> functionality of ROUTINE_NAMES continued to evolve between versions,
> and that you can't be guaranteed to be able to add new variables in
> older versions.
>

Dear Craig,

I did yesterday a bad mistake in my script deepstop.
Today early in the morning I recognized what's my script really does.

I have overwritten the variable level, which is the counter too.

Sorry.

Reimar